Performance Evaluation of Speech Recognition System Using Conventional and Hybrid Features and Hidden Markov Model Classifier

By

Hussien Elharati

A dissertation submitted to the College of Engineering and Science of Florida Institute of Technology in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering

Melbourne, Florida
December, 2019
We the undersigned committee hereby approve the attached thesis, “Performance Evaluation of Speech Recognition System Using Conventional and Hybrid Features and Hidden Markov Model Classifier”

By

Hussien Elharati

Hereby indicates its unanimous approval.

___________________________
Philip Bernhard, Ph.D.
Associate Professor and Head
Department of Computer Engineering and Sciences
Abstract

Performance Evaluation of Speech Recognition System Using Conventional and Hybrid Features and Hidden Markov Model Classifier

by

Hussien Elharati

Major Advisor

Këpuska, Veton Z. Ph.D.

Extracting the auditory information from speech signal is considered a computationally demanding task. However, past researches in mathematics, acoustics, and speech technology have provided many methods for signal processing and modeling. Although, all methods have their strengths and weaknesses, but they remain a serious attempt towards speech recognition system.

Using multivariate statistical machine learning (Hidden Markov Model), this work investigates the performance of selected conventional and new hybrid feature extraction algorithms in both clean and noisy environments. The resultant conventional features include MFCC, LPCC, PLP, and RASTA-PLP, while the new hybrid features include LPR, MLP, MLR, and MPR. The whole speech system was designed using MATLAB software, and evaluated using isolated-word human voice corpus (TIDIGITS). This data set are consists of eleven words (zero to nine and the letter O), sampled at 8-kHz and digitalized with a resolution of 16 bit, recorded from
208 different adult speakers (men & women), each person uttered each word two times.

Giving a dependency in multi-dimensions through transition probabilities organized in a Markov mesh, HMMs Pattern matching technique considers the observations statistically dependent on neighboring observations as shown; In training session HMM, generates several reference models and stored in for later use. With a statistical model in hand, we can perform several important tasks related to speech recognition. In testing session, statistical models were applied to find the highest probability that helps to generate the decision in order to recognize the unknown word. Consequently, training models are derived in to evaluate the behavior of the proposal speech recognition system based on WER scale, and all the results are compared with some ready published models.

The results showed that the acoustic signals extracted using LPC and LPR algorithms are given the best recognition rate at 99.9949% and 99.9733% in quite condition, while in noisy condition, RASTA-PLP algorithm was provides the best recognition rate by 98.9999%, 98.7945%, 94.7672, and 93.9809% at 30, 20, 10, 5db respectively. As far as the validity of the commonly used models is concerned, the comparison to the measurements reveals that the applicability of those models for the studied environment is still debatable. The main technical contribution of this research is a way of estimating the parameters of new four hybrid feature extraction algorithms comparing with conventional features. So, this research can serve as a useful reference for the engineers to design ASR applications.
# Table of Contents

Table of Contents .................................................................................................................. v

List of Figures ......................................................................................................................... x

List of Tables .......................................................................................................................... xiii

Acronyms & Abbreviations ..................................................................................................... xv

Acknowledgement .................................................................................................................. xvii

Dedication ............................................................................................................................... xviii

Chapter 1 Introduction ........................................................................................................... 1
  1.1 Motivation ......................................................................................................................... 1
  1.2 Objectives ......................................................................................................................... 2
  1.3 Dissertation Outline ....................................................................................................... 4

Chapter 2 Literature Review ................................................................................................. 5
  2.1 Front-End ......................................................................................................................... 7
  2.2 Back-End ......................................................................................................................... 11
  2.3 Performance Evaluation ................................................................................................. 15

Chapter 3 Speech Signal ....................................................................................................... 18
  3.1 Introduction ..................................................................................................................... 18
  3.2 Speech Production .......................................................................................................... 19
3.2.1 Source-filter Models of Speech Production .............................................. 20
3.3 Speech Perception ....................................................................................... 22
3.3.1 Spectral Representation ........................................................................ 23
3.3.1.1 Three states representation .............................................................. 23
3.3.1.2 Parameterization of the Spectral ...................................................... 24
3.3.2 Phonetics and Phonemics ...................................................................... 24

Chapter 4 Signal Processing for Speech Recognition ...................................... 26
4.1 Introduction ................................................................................................. 26
4.1.1 Pre-emphasis .......................................................................................... 27
4.1.2 Signal-to-Noise Ratio Estimation ......................................................... 28
4.1.3 Frame Blocking and windowing ............................................................. 30

Chapter 5 Acoustic features for speech recognition ....................................... 35
5.1 Introduction ................................................................................................. 35
5.2 Conventional Features ................................................................................ 36
5.2.1 Mel-scale Frequency Cepstral Coefficients (MFCC) .............................. 36
5.2.1.1 Fast Fourier Transform .................................................................. 37
5.2.1.2 Spectral magnitudes ...................................................................... 40
5.2.1.3 Mel-frequency filterbank ............................................................... 40
5.2.1.4 Logarithm of Filter energies .......................................................... 42
5.2.1.5 Discrete Cosine Transform ............................................................. 43
5.2.1.6 Liftering or Cepstral Weighting ...................................................... 43
5.2.1.7 Dynamic feature ........................................................................... 44
5.2.2 Linear Prediction Coding Coefficients (LPCC) ...................................... 45
5.2.2.1 Autocorrelation analysis ................................................................. 46
5.2.2.2 LPC analysis .............................................................................. 48
5.2.2.3 Cepstrum Analysis ..................................................................... 50
5.2.3 Perceptual Linear Prediction (PLP) .................................................. 51
5.2.3.1 Frequency band analysis ............................................................. 52
5.2.3.2 Equal-loudness curve ................................................................. 55
5.2.3.3 Intensity-loudness power law ....................................................... 55
5.2.4 Relative Spectral Perceptual Linear Prediction (RASTA-PLP) .......... 55
5.3 Hybrid Feature Extraction ................................................................. 57

Chapter 6 Acoustic Modeling for speech recognition .............................. 60
6.1 Introduction ..................................................................................... 60
6.2 Hidden Markov Model (HMM) Classifier ......................................... 60
6.2.1 Initialization .................................................................................. 62
6.2.2 Evaluation ..................................................................................... 63
6.2.2.1 Forward algorithm ..................................................................... 64
6.2.2.2 Backward algorithm .................................................................. 66
6.2.3 Learning (Parameters Estimation) .................................................. 69
6.2.3.1 Multivariate Gaussian distribution .............................................. 70
6.2.3.2 Baum-Welch algorithm ............................................................. 73
6.2.4 Decoding ....................................................................................... 75

Chapter 7 Speech Recognition Performance Analysis ............................ 79
7.1 Introduction ..................................................................................... 79
7.2 Toolboxes ....................................................................................... 80
Appendix K. Front-End Feature Extraction Results ................................. 135

Appendix L. Back-End Training, and Decoding Results .......................... 136

Appendix M. Publications ..................................................................... 138

References ............................................................................................ 139
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Speech recognizer system overall architecture</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>Markov chain with transition probabilities</td>
<td>13</td>
</tr>
<tr>
<td>2.3</td>
<td>Viterbi trellis computation for HMM</td>
<td>15</td>
</tr>
<tr>
<td>3.1</td>
<td>Human vocal mechanism</td>
<td>19</td>
</tr>
<tr>
<td>3.2</td>
<td>Source/Filter model of speech production</td>
<td>21</td>
</tr>
<tr>
<td>3.3</td>
<td>Speech production discrete-time model</td>
<td>21</td>
</tr>
<tr>
<td>3.4</td>
<td>Spectrogram using Welch’s method and speech amplitude</td>
<td>23</td>
</tr>
<tr>
<td>3.5</td>
<td>Three state representation</td>
<td>24</td>
</tr>
<tr>
<td>4.1</td>
<td>Speech signal preprocessing steps</td>
<td>26</td>
</tr>
<tr>
<td>4.2</td>
<td>Original Vs pre-emphasized speech signal</td>
<td>27</td>
</tr>
<tr>
<td>4.3</td>
<td>Word seven, corrupted by different SNR values</td>
<td>30</td>
</tr>
<tr>
<td>4.4</td>
<td>Hamming window</td>
<td>32</td>
</tr>
<tr>
<td>4.5</td>
<td>Hamming window applying on speech frame</td>
<td>33</td>
</tr>
<tr>
<td>5.1</td>
<td>MFCC Feature extraction block diagram</td>
<td>37</td>
</tr>
<tr>
<td>5.2</td>
<td>FFT windowed speech signal</td>
<td>38</td>
</tr>
<tr>
<td>5.3</td>
<td>Power spectrums of speech signal ‘She had your dark suit in greasy wash water all year’. Top: Amplitude waveform, bottom: spectrogram. Magnitude spectrum was calculated in frames of 256 samples, and DFT was evaluated at 256 points</td>
<td>39</td>
</tr>
</tbody>
</table>
Figure 5.4: Mel-Scale function ................................................................. 41
Figure 5.5: Frequency response of 26 Mel-filter bank.............................. 42
Figure 5.6: LPCC feature extraction block diagram ............................... 45
Figure 5.7: Autocorrelation of 20ms portion of speech signal .................. 48
Figure 5.8: Linear productive filter ........................................................ 50
Figure 5.9: Original speech signal with power spectrum of the speech frame
and the linear frequency cepstral coefficients, which calculated from
autocorrelation coefficients ................................................................. 51
Figure 5.10: PLP analysis block diagram ............................................... 52
Figure 5.11: Bark Scale Filter Bank (Sampling Frequency = 8 kHz) ........... 54
Figure 5.12: Filtered spectrum of 18 filter in The Bark-Scale ................... 54
Figure 5.13: RASTA-PLP block diagram ............................................. 56
Figure 5.14: Hybrid feature extraction analysis block diagram ................. 59
Figure 6.1: HMM elements .................................................................. 61
Figure 6.2: Forward algorithm procedure ............................................. 64
Figure 6.3: Forward probability, alfa scaled ......................................... 66
Figure 6.4: Backward process .............................................................. 66
Figure 6.5: Backward probability, beta scaled ....................................... 68
Figure 6.6: Alfa and beta probabilities .................................................. 69
Figure 6.7: Multiple mixture gaussian .................................................. 70
Figure 6.8: Viterbi trellis computation .................................................. 78
Figure 7.1: Isolated word HMM recognizer block diagram. .................................82
Figure 7.2: Recognition classification rates based on MFCC. ...............................89
Figure 7.3: Recognition classification rates based on LPC. .................................90
Figure 7.4: Recognition classification rates based on PLP. .................................91
Figure 7.5: Recognition classification rates based on RASTA-PLP. .......................92
Figure 7.6: Conventional methods recognition classification rates. .......................94
Figure 7.7: Recognition classification rates of LPR. .............................................98
Figure 7.8: Recognition classification rates used MLR hybrid algorithm. ...............99
Figure 7.9: Recognition classification rates used MPR hybrid algorithm. .............100
Figure 7.10: Recognition classification rates used MLP hybrid algorithm. .........101
Figure 7.11: Recognition classification rates of hybrid methods. .......................102
Figure 8.1: Overall recognition classification rates of conventional and hybrid methods. .................................................................105
List of Tables

Table 3.1: TIDITS database in IPA format and English pronunciation..................25
Table 7.1: Experiment conditions. .................................................................85
Table 7.2: Common and major configurations of conventional methods

experiments. .................................................................88
Table 7.3: Recognition rates of MFCC front-ends in various state numbers and
different SNR levels using static + delta + delta-delta” coefficients. ...............89
Table 7.4: Recognition rates of LPC front-ends in various state numbers and
different SNR levels using static + delta + delta-delta” coefficients. ..........90
Table 7.5: Recognition rates of PLP front-ends in various state numbers and
different SNR levels using static + delta + delta-delta” coefficients ..........91
Table 7.6: Recognition rates of RASTA-PLP front-ends in various state numbers
and different SNR levels using static+delta+delta-delta” coefficients ..........92
Table 7.7: Overall recognition rates in various state numbers and different SNR
levels using conventional feature methods. .................................................93
Table 7.8: Common and major configurations of hybrid methods experiments......96
Table 7.9: Recognition rates of LPR front-ends in various state numbers and
different SNR levels using[13-LPC+13-PLP+13-RASTA-PLP] coefficients.97
Table 7.10: Recognition rates of MLR front-ends in various state numbers and different SNR levels using [13-MFCC+13-LPC+13-RASTA-PLP] coefficients .................................................................98

Table 7.11: Recognition rates of MPR front-ends in various state numbers and different SNR levels using [13-MFCC+13-PLP+13-RASTA-PLP] coefficients .................................................................99

Table 7.12: Recognition rates of MLP front-ends in various state numbers and different SNR levels using [13-MFCC+13-LPC+13-PLP] coefficients .................................................................100

Table 7.13: Recognition rates in various state numbers and different SNR levels of hybrid feature methods ..................................................................................................................102

Table 8.1: Overall recognition rates in various state numbers and different SNR levels of conventional and hybrid feature methods. .........................................................104
### Acronyms & Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR</td>
<td>Automatic Speech Recognition</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial neural networks</td>
</tr>
<tr>
<td>DCT</td>
<td>Discrete Cosine Transform</td>
</tr>
<tr>
<td>DFT</td>
<td>Discrete Fourier Transforms</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>DFT</td>
<td>Discrete Fourier Transform</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processing</td>
</tr>
<tr>
<td>DTW</td>
<td>Dynamic Time Warping</td>
</tr>
<tr>
<td>EPD</td>
<td>End Point Detection</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>GMM</td>
<td>Gaussian Mixture Model</td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Markov Model</td>
</tr>
<tr>
<td>INS</td>
<td>Insertion</td>
</tr>
<tr>
<td>INV</td>
<td>In Vocabulary</td>
</tr>
<tr>
<td>LPCC</td>
<td>Linear Predictive Coding Coefficients</td>
</tr>
<tr>
<td>LPR</td>
<td>LPC + PLP + RASTA-PLP</td>
</tr>
<tr>
<td>MATLAB</td>
<td>Matrix Laboratory</td>
</tr>
<tr>
<td>MFCC</td>
<td>Mel-Frequency Cepstral Coefficients</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
</tr>
<tr>
<td>MHz</td>
<td>Mega Hertz</td>
</tr>
<tr>
<td>MLR</td>
<td>MFCC + LPC + RASTA-PLP</td>
</tr>
<tr>
<td>MLP</td>
<td>MFCC + LPC + PLP</td>
</tr>
<tr>
<td>MPR</td>
<td>MFCC + PLP + RASTA-PLP</td>
</tr>
<tr>
<td>NIST</td>
<td><em>National Institute of Standards and Technology</em></td>
</tr>
<tr>
<td>NLP</td>
<td><em>Natural Language Processing</em></td>
</tr>
<tr>
<td>OOV</td>
<td>Out Of Vocabulary</td>
</tr>
<tr>
<td>PLP</td>
<td>Perceptual Linear Predictive</td>
</tr>
<tr>
<td>RASTA-PLP</td>
<td>Relative Spectral Perceptual Linear Predictive</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal-To-Noise Ratio</td>
</tr>
<tr>
<td>SR</td>
<td>Speech Recognition</td>
</tr>
<tr>
<td>SUB</td>
<td>Substitution</td>
</tr>
<tr>
<td>TI</td>
<td>Texas Instruments, Inc.</td>
</tr>
<tr>
<td>TIDIGITS</td>
<td>Texas Instruments for Digits</td>
</tr>
<tr>
<td>VAD</td>
<td>Voice Activity Detector</td>
</tr>
<tr>
<td>VQ</td>
<td>vector quantization</td>
</tr>
<tr>
<td>WER</td>
<td>Word Error Rate</td>
</tr>
<tr>
<td>WUU</td>
<td>Wake-Up-Word</td>
</tr>
<tr>
<td>ZCR</td>
<td>Zero Crossing Rate</td>
</tr>
</tbody>
</table>
Acknowledgement

I am very thankful to “ALLAH” almighty God, I thank Him for His countless bounties and generous blessings upon us. I thank Him for providing me the opportunity and granting me the capability to successfully complete this research work in a good situation.

I would also like to express my deepest appreciation and sincere gratitude to my advisor, Dr. Viton Kepuska for the professional guidance that has enabled me to accomplish this research. In addition to being an excellent advisor, I admire his way of teaching and his deep knowledge in his field. I am thankful for his inspiring guidance, invaluable constructive criticism and friendly advice during this work. He provided me with all the required experimental data and the technical information. It was an honor to have such a knowledgeable advisor.

I am thankful to my parents for their endless love, support, prayers and patience during the time I studied abroad. I would like to appreciate my wife, for being such a wonderful, loving and understanding wife. Thanks for giving me space and time to dedicate to my studies, my success is your success. I also would like to express my warmest thanks and gratitude to my children for their support and patience. I express my gratitude to my committee members Dr. Muzaffar Shaikh, Dr. Susan Earles, and Dr. Ivica Kostanic, for their guidance.

Finally, I would like to thank all my friends, fellow graduate students, the faculty and staff at Florida Tech and the department of Electrical and Computer Engineering I have made over the years.
Dedication

To:

My father, asking my God to accept him in his Mercy, and put him in paradise,

My mother for their love, prayers, endless support and encouragement

My wife for her love, support, encouragement, patience and understanding

My children Mohamed, Yasin, Nourelhuda, and Hamza for their love

My brothers, sisters, and friends for their love and support
Chapter 1
Introduction

1.1 Motivation

After nearly seventy years of research, as a result of the rapid development of hardware and software technology, ASR has reached a relatively high level of innovation, and became more and more expedient as an alternative human-to-machine interface [1]. Consequently, ASR spread heavily in several application areas.

Speech recognition is applied in almost all fields of life such as mobile applications, weather forecasting, agriculture, healthcare, military, database querying, automatic voice translation, command and control, training air traffic controllers, telephone directory assistance, office dictation devices, robotics, video games, transcription, etc. [2]. Moreover, transferring data to a computer through spoken language is much faster than that of transferring data through hand writing or using keyboard.

The Development of ASR system has been impeded by several issues. The most important issue is that how to select the most reliable and robustness feature extraction method, that can works in front-end ASR system in adverse environment [3]. The other issue is how to select the suitable templet or classifier technique, that can works in back-end ASR system to classify the feature vectors in order to generate the final decision with respect the statistical models [4].
In recent years, there are some serious researches related to front-end code optimization algorithms, such as; Linear Prediction Cepstral Coefficients (LPCC), Perceptual Linear Prediction Coefficients (PLP), Mel Frequency Cepstral Coefficients (MFCC), and Relative Spectral Perceptual Linear Prediction Coefficients (RASTA-PLP), which are used to extract the features from speech sound in ASR system. In addition, there are another code optimization algorithm that works in back-end part such as; Hidden Markov model (HMMs), Artificial Neural Networks (ANN), vector quantization (VQ), and dynamic time warping (DTW) [5].

In order to improve the performance of speech recognition system, Researches have outlined some successful optimization code related, but unfortunately, they did not give a full representation or comparison between these feature extraction techniques, and furthermore, there is still no such a framework that generally summarizes the difference between them. Thus, the main motivation behind this research work is to evaluate the performance of conventional features comparing with new hybrid features.

1.2 Objectives

The ultimate goal of this research work is to design an efficient speech recognition system (front-end and back-end), which will help to improve the performance of ASR system based on recognition accuracy of recognizing speech.

The suggestion front-end system is designed to extract the conventional and hybrid features using different algorithms. Conventional features are derived using MFCC, LPC, PLP, and RASTA-PLP algorithms, while hybrid features are derived by mixing the previous features in order to find a new features, MLP, MLR, MPR, and LPR.
MLP, MLR, MPR, and LPR are considered a new hybrid feature extraction algorithms, each kind of them has an output vector consists of 39 features (13 from each three conventional features). Where, MLP is a combination of (MFCC + LPCC + PLP), MLR is a combination of (MFCC + LPCC + RASTA-PLP), MPR is a combination of (MFCC + PLP + RASTA-PLP), and LPR is a combination of (LPCC+PLP+RASTA-PLP).

The second step, is to design a back-end HMM recognizer, which is used to generate a statistical reference models related to the proposal features. The related models help us to select the wining features in order to provide the final recognition decision. HMM recognizer, which is used in our work is considered the most successful statistical classifier in speech recognition system. Their success is due to both their rich mathematical structure, which engenders a theoretical basis for many domains, and also for the Baum-Welch algorithm, which is an efficient training algorithm that allows to estimating the numeric values of the model parameters from training data [6].

The third step, is to find out the most robustness algorithm between the proposal feature extraction algorithms. Therefore, the system has been evaluated, in adverse conditions in order to investigate the effectiveness of the acoustic speech signal.

To achieve our goal, a small vocabulary isolated word (TIDIGITS) corpus was used to evaluate the performance of the prospected front-end algorithms. This corpora of speech, was originally designed and collected at Texas Instruments, Inc. and hosted by University of Pennsylvania, which consists of eleven words include (zero, to nine and the letter O), recorded from 95 adult subjects as training data (38 male, 57 females), and 113 adult subjects as testing data (56 male, 57 females).
sampling by 8-kHz sampling frequency rate, where each subject repeated the word two times.

1.3 Dissertation Outline

The rest of this dissertation is as follows: Chapter 2 gives survey of the relevant work on feature extraction techniques, pattern classification, and Performance evaluation of the automatic speech recognition system. Chapter 3 provides a brief description of the Speech signal for speech recognition. Chapter 4 explains the signal processing in speech recognition. Overview of the acoustic Feature Extraction algorithms for speech recognition are given in Chapter 5. Statistical modeling is presented in Chapter 6. Chapter 7 presents the Speech Recognition Performance Analysis. Finally, summary, conclusions, and future work are drawn in Chapter 8.
Chapter 2
Literature Review

Automatic Speech Recognition system ASR can be roughly divided into two main categories; front-end approach, and back-end based techniques. Front-end part is responsible for extracts the physical characteristics from input speech signal and transform it into a stream of feature vectors, while back-end aimed to match this features with respect the reference models in order to generate the recognition result [7]. The following Figure 2.1, illustrates a typical speech recognition system which is used in our research.

![Figure 2.1: Speech recognizer system overall architecture.](image-url)
ASR technology has been progressed greatly over the past few years. Since the 1930’s, the first study of speech recognition began when researchers were investigating the science of speech perception in Bell Laboratories. The earliest attempts to implement automatic speech recognition was began in the 1950’s, whereas the first significant communicate between human and machine were made up in 1952, when David Biddulph, and Balashek, investigated the basic ideas of acoustic phonetics [8].

A great deal of a basic ideas in speech recognition field were developed and published in the 1960s, and 1970s, includes a linear predictive coding (LPC) [9, 10], fast Fourier transform (FFT), cepstral analysis [11, 12], as well as dynamic time warping (DTW), and Hidden Markov Models (HMMs)[6, 13].

In 1970s, the approach of pattern recognition was shifted from template-based to statistical modelling method, when isolated speech recognition was introduced [14]. In 1980s, and 1990s, the researches starting to focus on connected word instead of isolated word speech recognition problem, and start move their interests to the Large Vocabulary (LVCSR), instead of isolated speech [15].

In recent years, ASR has reached very high level of performance, and accuracy with minimum word error rate, due to using common speech corpora, new ideas in acoustic modeling, and language modelling with statistical grammars [16]. In addition, the improvement of speech algorithms, beside the increasing of the computer processing speed, and storage memory helped the researches to improve the performance of ASR system [17].

Since 1990, the scientists proposed some metric tools to weight the accuracy of speech recognition, or machine translation system, such as word error rate (WER), and word recognition accuracy (WRA), Which considered the most commonly
methods used in ASR system [17]. These two terms are used to measure the recognition accuracy based on the central idea of differential weighting of different ASR errors. Thus, many researches conducting some experiments based on accuracy evaluation to understand the behavior of the feature extraction algorithms.

In this chapter, the history of an ASR system and the related works are briefly discussed. Sections 2.1, to 2.3 highlight some popular feature extraction methods, template and classifier techniques, ASR Performance Evaluation, and provide an overview on recent researches related to ASR system.

2.1 Front-End

The main purpose of front-end part in ASR system is to generate the feature vectors, which is a sequence of vectors that carrying a good representation of the input speech signal. Feature extraction is a technique that used to extract some physical characteristics from the input speech signal, which are then used to classify and decoding the unknown word in testing session in respect of reference models [18]. According to Martens, in [19], several feature extraction techniques are available, include; Mel-Frequency Cepstral Coefficient (MFCC), Linear Predictive Coding (LPC), Perceptual Linear Prediction (PLP), and Relative Spectral Perceptual Linear Prediction Coefficients (RASTA-PLP).

Mel-frequency Cepstral Coefficients or MFCC, is considered as the most popular feature extraction techniques that used in front-end ASR system. This algorithm was introduced by Davis, and Mermelstein, in 1980's, [20]. The most steps that involved to extracting the MFCC features are slightly different from researcher to another. Ursin, in [18] investigated that the most common processes related to MFCC feature extraction analysis, are includes, sampling with a preferred sampling rate, pre-emphasis, followed by windowing, and frequency warping. Ursin in his
previous paper research, prefers to add the first and the second delta to the static feature coefficients, to improve the performance of the ASR system. Ursin also adds the cepstral lifting to re-scale the cepstral coefficients to hold similar magnitudes.

Huang, et al. in [21] mentioned that the advantage of MFCC in log-energy calculation is that the filter energies. This technique is working well against distortion and spectral errors. Huang considered that frequency domain features are more precise than time domain features, and demonstrated MFCC as a representation of the real cepstral of a windowed short-time signal using (FFT). Chetouani, et al. in [22] considered MFCC as the most dominant methods that used to extract spectral features. He suggested this method in speech recognition task because MFCC using Mel scale based on human ear scale.

G. Zorić, in [23] indicated that MFCC is extracting the parameters from speech signal similar to parameters that are used by humans for hearing speech. Also G. Zorić, in [24] mentioned that de-emphasize process, which are commonly used in speech recognition systems take into consideration specific characteristics of the human auditory system. In addition, Z. Razak, N. J. Ibrahim, Z. M. Yusoff, M. Y. I. Idris, and E. M., in [22] Confirmed that MFCC features are extracted by transforming the speech signal into a cepstrum, that can be separated by band pass linear filters.

Kou, W. Shang, I. Lane, and J. Chong, in [25] explained that MFCCs analysis is a process applied on the frame sequence, started with Fast Fourier Transform (FFT) to obtain certain parameters, and then undergoes Mel-scale perceptual weighting and de-correlation to generate sequence feature vectors. In the same manner, Martens, in [19] reported that MFCC features are computed by performing a Fourier transform analysis on windowing signal, converting the power-spectrum to
a Mel-frequency spectrum, and finally, taking the logarithm of that spectrum and computing its inverse Fourier transform.

Linear Productive Coefficients LPC, is one of the earliest algorithm that can works at low bit-rate, and represents an attempt to mimic the human speech [26]. LPC algorithm, which introduced in late 1960s, and early 1970s, by (Bishnu, S. Atal & Manfred, R. Schroeder) is considered as a very useful method in front-end based on a natural modulization of the vocal tract [22]. U.H. Yapanal, and J.H.L. Hansen, in [27] declared that the assumption behind discovering LPC model is that the speech signal are modeled by an excitation either noise-like or quasi-periodic. In addition Huang, et al. in [21] descried LPC as a very powerful method for speech analysis, therefore, it is known as Auto-Regressive modeling or LPC analysis.

The researcher Felber, in [28] reported that LPC is widely used because it is fast, simple, useful, and has a capability to extract and store time varying formant information, where the loudness is boosted. In addition, the author mentioned that the coefficients produced using LPC analysis describe digital filter, and produce a sound like the original speech.

According to Rabiner, and Juang, in [1], LPC Coefficients are generated according to several processes include; pre-emphasis speech signal, divide the pre-emphasized signal into frames, apply windowing on each frame to minimize the signal discontinuities, autocorrelation process, LPC analysis, convert LPC parameter into cepstral coefficients, and finally, parameter weighting.

Perceptual Linear Prediction PLP is a feature extraction algorithm, which was originally proposed by Hynek Hermansky in 1990 as a way of warping spectra, to minimize the differences between speakers while preserving the important speech information [29]. This technique is used to deriving more auditory-like spectrum
based on linear perceptual (LP) analysis of speech. This kind of feature extraction is reached by making some estimations of the psychophysical attributes of human hearing process [30]. Hermansky, in [29] reported that PLP technique estimates the human ear process more precisely than many other feature extraction methods.

The main reason behind discovered and examined PLP algorithm, is to design and establishing an auditory system processes that much likely the human auditory system. Rabiner, and Juang, in [1] assumed that better human auditory system, is to design speech recognition system that can truly recognize the contents of the speech and the meanings. In addition, Ursin, in [18] reported that PLP dealing with equal-loudness curve, critical-band resolution curves, and intensity-loudness power-law relation. In the same context, Madisetti, and Williams, in [31] specified some sense attributes include; threshold of hearing, pitch, differential threshold, critical bands, auditory filters, and masked threshold. They found that PLP analysis is more stable and reliable for human hearing than LP analysis.

Hermansky, in [29] reported that PLP is working perfectly in noisy environment in automatic speech recognition system. Florian H’onig, and his colleagues, in [32] mentioned that PLP features are more reliable in automatic speech recognition system, when there is a mismatch between training and testing data.

Another popular speech feature representation is known as Relative Spectral Perceptual Linear Predictive filtering (RASTA-PLP). This speech analysis technique is an improvement of the traditional PLP method, which introduced by H. Hermansky, and N. Morgan, in 1994. Hermansky, in [33] reported that RASTA is slightly different of PLP technique. RASTA applies a special band-pass filter in the log-spectral domain in order to smooth over short-term noise variations, and to remove any constant offset in the speech channel such as telephone line. Vilas
thakare, and his graduate student Urmila Shrawankar in [34] investigated that the difference between PLP and RASTA_PLP is to apply a special band-pass filter to the energy in each frequency sub-band, which aimed to remove any offset resulting from spectral coloration, and to smooth over short-term noise variations.

Vilas and Urmila in [35] state that, many researchers have proposed many hybrid feature techniques in order to improve the recognition performance. They indicates that the accuracy of the speech recognition system could be increased by using the combination of features instead of using a single feature. They strongly suggest that more new hybrid features need to be developed for achieving better performance in robust speech recognition.

2.2 Back-End

Pattern classification techniques are the main processes that involved in back-end part of ASR system. Pattern is a group of data points in an n-dimensional feature space, while classification is the procedure for discriminating that clustered data from other data sources in the feature space. Pattern classification is much broader topic in automatic speech recognition system, due to using tools from signal processing, statistics, probability, computational geometry, and machine learning. Thus, it is considered as a central importance to artificial intelligence.

Huang, et al. in [16] classified pattern classification and recognition as the most challenging topic in speech processing. He explained the pattern classification and recognition as a process used to evaluate the matching between two speech patterns. The first pattern represents the unknown speech word, while the second is the reference or the model stored in the memory. Felber, in [28] reported that the most advanced automatic speech recognition system can search in a huge database of all similar words in library in order to find the best match to an unknown speech
input. Pattern classification and recognition system are consisting of some elements include; speech analysis, training, matching, models, and decision logic. All that important back-end processes can be used in isolated, connected, and continuous word in speech recognition system [31].

In a broad sense, signal models can be categorized into two classes; namely deterministic models and statistical models [36]. In the first class, there are some straight forward techniques like Dynamic Time Wrapping (DTW), and Vector Quantization (VQ), that provides a non-parametric template-based model, which aims to match the incoming speech to the distinct templates. The second class, the statistical model, such as Hidden Markov Model (HMMs) and Artificial Neural Network (ANN), which are considered a better account for non-linear variability in speech waveform, and they are generating the speech models based on some probability distributions.

Although, these techniques have been proven successfully in several image and signal processing fields, but HMM is still the most successful technique in speech recognition system, which is used to model the speech signal as a ‘maximum likelihood classifier’. This statistical method is evaluated by computing the probability of a sequence words given a sequence of acoustic observations, and then generate a discrete time random process consisting of two sequences of random variables hidden states and observations [37].

The basic idea behind the HMM starts in 1900s, when Andrei Markov presented a Markov chains theory. In the late of 1960s and early 1970s, Baum and his colleagues presented the Hidden Markov Model as an extension to the first-order stochastic Markov process [38]. In 1970s, the first implementation of HMM was occurred when Baker, at Carnegie Mellon University and Jelinek, at IBM
implements the first HMM paradigm in speech processing applications. The system was consisting from a set of hidden states, observation, transition probabilities, emission probabilities, and initial state probabilities. Figure 2.2 shows the simple 6-state Markov chain with its transition probabilities [1].

Unfortunately, there is no real method to determine the accurate number of necessary states to encode the related characteristics of the observation data into the model. Therefore, the relationship between the number of states and the performance of ASR system is critical. Some researchers suggest that the number of states be at least equal to the number of phonemes contained in the word. In most ASR applications, the experimental results suggesting the proper number of states [39]. Selecting too small number of states would result in poor classification due to the lacking capability in modeling properly. Also, large states would increase the computational and memory cost.

![Markov chain with transition probabilities](image)

**Figure 2.2: Markov chain with transition probabilities.**
Rabiner, in [36] reported that HMM is a very important technique that used in speech recognition task for two reasons; firstly, HMM models are very rich in mathematical structure, and therefore, theoretical basis can be formed for use in a wide range of applications. Secondly, HMM models working perfectly in practice for several useful applications.

The main issue of HMM statistical model is that it needs a priori assumptions, which are susceptible to be inaccurate and decrease the system performance [36]. In addition, Hidden Markov Model faced with three conical problems include; evaluation, decoding, and training problems [21, 36]. HMM focused on how the system models matching the observation sequence, how the decoding attempts to cover the hidden state of the model to find the “correct” state sequence, and how training process attempting to optimize the model parameters to observe training data. In our research, a range of based-statistical solutions were designed using Viterbi and Baum-Welch algorithms to build a good model for real phenomena.

Viterbi algorithm is a dynamic programming used to compute the probability of observations giving the model $P(O \mid \lambda)$. As shown in Figure 2.3, the probability can achieved by finding the best scoring path in a directed graph with weighted arcs, and summing up all possible state sequences through a Trellis structure in a HMM [1]. In other words, Viterbi Algorithm can be used to determine such an optimal state sequence as a dynamic programming method in HMM.

Baum and his colleagues established an expectation-maximization algorithm named interactive Baum-Welch or Forward- Backward algorithm. This algorithm is used to re-estimating iteratively the model parameter $\lambda (A,B,\pi)$, in an effort to learn and encode the characteristics of the observation sequence into the model, such the model should identify a similar observation sequence in future [6, 40]. Rabiner, And
David, at [36] said that Baum-Welch algorithm maximizes the likelihood function of a given model $\lambda$ every single iteration, and re-estimates the HMM parameters to a closer value of the “global”.

![Figure 2.3: Viterbi trellis computation for HMM.](image)

**2.3 Performance Evaluation**

In the past few years, many researchers have conducting different front-end related experiments to understand the behavior of the feature extraction algorithms in adverse conditions. Empirically, in ASR system, the database is separated into two sets. The first set, which is the most available data is a training set. This set is utilized for parameter estimation of the acoustic models. The second set, which is the remaining of the data is a testing set, which utilized to evaluate the ASR performance. In the matter of fact, selecting training data that is very likely to testing data in both quality and environments, will help the system to better matching and recognizing the testing data.
In this section, using the techniques that were discussed in this literature review, we will try to highlight some related researches in speech recognition task, and provide a simple comparison between them, where, the main criterion for this comparison is related to the accuracy and recognition rates.

M. Ostendorf, in [41], who worked on Segment-Based Stochastic Modeling for Speech Recognition, found that LPC feature extraction provides 62% to 96% in different state sequences using vector quantization as a classifier. Another researcher called Milner, in [30], who worked on ‘a comparison of front-end configurations for Robust Speech Recognition’ said that MFCC feature extraction provides a high recognition rate (33% to 45%) comparing with PLP, which provides a recognition rate (30% to 40%) in an unconstrained monophony test. Ahadi, et al. in [42] presents a work named “An Efficient Front-End for Automatic Speech Recognition”, he found that MFCC provides 98.86% recognition rate using clean data in both training and testing process, and provides (28% to 78%) recognition rate, when using noisy data based on HMM statistical classifier.

Hasan, et al. in [43] found that MFCC provides (57% to 100%) recognition rate, when using vector quantization as statistical classifier. Another work from ahsanul kabir, and sheikh mohammad ahsan, in [44] they found that MFCC provides (70% to 85%) recognition rate using Vector quantization. Hönig, et al., in [32] Summarized that MFCC features has a recognition performance that is slightly larger to PLP and thus to LPC.

Another attempt was conducted by cheng, in [45] on both MFCC and LPCC as feature extraction and Dynamic Time Warping as a recognizer. This experiment has been shown that LPCC reached 95.52%, MFCC reached 96.30%, and the combination of them reached 97.12% recognition accuracy based on different.
speakers. Jackson, in [46] working on “Automatic Speech Recognition, Human Computer Interface for Kinyarwanda Language”, founds that MFCC provides 92% recognition rate using HMM as statistical classifier. R.K. Aggarwal, and M. Dave, in [47] evaluates the recognition performances of PLP and MFCC features in different environments. They report that the recognition performance of the system is working well in clean environment, and it can be easily get above 90% recognition rate. However, when the system is trained in a clean environment and tested in a noisy environment ≈15dB SNR, the performances of both features type drop by 40-57%. The researcher suggests that feature extraction algorithm is perform poorly in different environments.

Through all these information, it is showing clear to us that the main objective of ASR researches is to allow the computer to recognize in real-time with 100% accuracy all words that are intelligibly spoken by any person, independent of vocabulary size, noise, speaker characteristics, and accent. Now days, if the automatic speech system is trained to learn an individual speaker's voice, then much larger vocabularies are possible, and accuracy can be greater than 90%.

As a result of improving the quality of speech recognition systems, we are expecting a plenty of voice applications in the future to enable humans interact easily with machines by voice. In recent years, ASR recognizers reached a very high level of recognition rate, which makes it very useful to use in researches and speech applications. The Limitations include; lack of robustness to speech variations, vocabulary size, sensitivity to background noise, and lengthy training procedure, as well, more natural quality of speech is still required for speech synthesizers.
Chapter 3
Speech Signal

3.1 Introduction

Speech is the most natural form of human-human communications. Speech signal can be classified into voiced, unvoiced, and silence regions, where voiced region represents by the periodic signal of vocal folds, unvoiced region represents by random signal, and silence region denotes to no-excitation. Related to sound and acoustics, the frequency response of a human ear is commonly given as 20 Hz to 20 kHz, with the high frequency capability diminishing with age [48].

Human ear is most sensitive to a frequency spectrum ranging from 500 Hz to 4kHz, which roughly corresponds to the speech bandwidth carried along analog telephone lines [48]. In addition to this inherent sensitivity, the human ear is capable of a wide dynamic range of hearing intensity, which is practically measured at approximately 130 dB from the threshold of hearing to the threshold of pain.

Physiology, individuals using relatively the same anatomy to produce the sound using some human vocal-mechanism. As shown in Figure 3.1, the most important parts of human vocal-mechanism are vocal tract and nasal cavity, which limited by velum, lips, tongue, and jaw. These important parts are used to vibrate and forming the air expelled from the lungs through the trachea to produce different sounds [49].

When the speaker’s brain made an idea, the source word sequence (W) is performed and turns the source into the speech signal through vocal system, and then transfer it via air to the listener. As soon as the acoustical signal is perceiving by the
human auditory system, the listener’s brain starts processing this waveform and understand its content. These common processes in speech processing have been modeled by speech generator and speech processing & decoding components in human system [21]. Thus, automatic speech recognition system considered as the inverse process of human’s speech production and speech perception system.

![Diagram of the human vocal mechanism](image)

**Figure 3.1: Human vocal mechanism.**

### 3.2 Speech Production

The most simplistic description of how human utter sounds in speech can be characterized by the control of air generated by the lungs. However, the production of speech is described as a two-level process, where in the first stage the sound is initiated and in the second stage it is filtered [50].
The air expelled from the human lungs was modulated to produce the sound using several parts of vocal organs, such as lungs, nose, larynx, and other parts of the human mouth. The rest of vocal organs such as vocal tract, vocal cords, nasal cavity, tongue, and lips are used to modify the characteristic of the speech signal to produce the sound.

The vocal cords are expressed as a vibration model, where the pitches of the speech are changing. When the vocal cords close, the result is voiced sound, and when it open, the unvoiced sounds result. The vocal tract model is created as a non-uniform sound tube with differing cross-sectional areas, and the transmission of sound waves inside the sound tube is expressed by a digital filter. In general, resonance system can be considered as a filter used to shape the spectrum of the source of the sound [51].

3.2.1 Source-filter Models of Speech Production

Source filter model of speech production still in the heart of many speech analysis methods and drives thinking in speech perception research. The view put forward in the source filter model is that speech sounds are produced by the action of a filter, the vocal tract, on a sound source, either the glottis or some other constriction within the vocal tract. There are two acoustic sources in speech production corresponding to voiced and unvoiced speech sounds.

Source-filter model describes speech production as a two-stage process involving the generation of a sound source, which then shaped or filtered by the resonant properties of the vocal tract. Excitation signal and vocal tract are modeled as signal and filter respectively using all-pole filter as shown in Figure 3.2
Typical discrete-time model, explains the most important parts that are involved in speech production system. As shown in Figure 3.3, discrete-time speech production model consists of several parts include; impulse train (DT), which acts like the lungs, glottal filter (G(z)), which regarded as the vocal cords in the human vocal mechanism, gain (b0), which using to provide a louder speech, vocal tract filter (V(z)), which is modeled over the vocal tract and the nasal cavity, and finally, the lip radiation filter (R(z)), which modeled by human lip to produce different sounds [50].
Speech production transfer function $H(z)$ is taken as a combination between $G(z)$, $V(z)$, and $R(z)$, in order to model voiced sound [52] using the next formula:

$$H(z) = G(z)V(z)R(z) = \frac{G}{1 - \sum_{i=1}^{p} a_i z^{-i}} \quad (3-1)$$

Where,
- $G$: Glottal filter.
- $P$: Filter's order.

In order to synthesize the speech samples $x(n)$, the next formula can be derived [52].

$$x[n] = \sum_{i=1}^{p} a_i x(n - i) + Ge[n] \quad (3-2)$$

### 3.3 Speech Perception

Speech perception is the process by which the sound is heard, interpreted, and comprehended. Speech perception refers to set of operations that transform an auditory signal into mental representations. Processing speech signal in the human brain is still the core challenge for cognitive science and neuroscience. Thus, various models have been developed to help understand the methods behind perceiving different components of speech. At the physical level, the speech in the acoustic signal is exist in both time and frequency domains, and representing in three different ways, include; spectral representation, three-state representation, and parameterization of the spectral [1].
3.3.1 Spectral Representation

Spectral representation is considered as the most popular methods used to represent the speech signal over the time. Figure 3.4 shows the speech signal in time domain and sound spectrogram in frequency domain, based on Welch’s method. The dark blue in this spectrogram representation represents the silent parts of the speech waveform, while the lighter red represents the intensity of speech waveform [53].

![Spectrogram using Welch’s method and speech amplitude.](image)

3.3.1.1 Three states representation

As shown in Figure 3.5, the speech signal is divided into three states representation include; Silence (S), unvoiced (U), and voiced (V). Silence means no speech, unvoiced means vocal cords are not vibrating, while voiced means vocal
cords are tensed and vibrating periodically over a short-time period between 5 to 100ms [1].

![Amplitude vs Time graph](image)

**Figure 3.5: Three state representation.**

### 3.3.1.2 Parameterization of the Spectral

Parameterization of the Spectral is set based on the model of speech production, which described in previous section (3.2). In this representation, the human vocal tract is described as a tube excited by air, and expresses the transfer function of the energy (formants) that hold the most acoustic energy. While this Parameterization is highly effective, it is more difficult to estimate the formant frequencies in low-level speech and express the formants for silent and unvoiced speech [53].

### 3.3.2 Phonetics and Phonemics

Phonetics is a science related to the sounds in human languages, where concerns to how the ear, nerves, and brain can process the speech signal. Phonetics science studies the production and the perception of the sound, consists from some subcategories, include; acoustic phonetics, articulatory phonetics, and auditory phonetics [54].
Phoneme is a set of sounds produced in a language, and distinguishable by native speakers of that language from other sounds in that language. Each language has a set of phonemes which described using several notations.

Phoneme is a smallest unit of speech that makes the connection between the sound and the meaning [54]. As shown in Table 3.1, International Phonetic Alphabet (IPA) is one of this notation, which developed by International Phonetic Association to describe the sound of all human languages. ARPA-bet is another phonetic alphabet notation, which developed by Advanced Research Projects Agency (ARBA) in (1971–1976). This code was designed for American English based on ASCII symbols, and has been used in several speech synthesizers, thus, it is more convenient than IPA to use on computers [54].

<table>
<thead>
<tr>
<th>Digits</th>
<th>Digits in English</th>
<th>IPA Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Zero</td>
<td>/zē-tō/</td>
</tr>
<tr>
<td>1</td>
<td>One</td>
<td>/ˈwʌn/</td>
</tr>
<tr>
<td>2</td>
<td>Two</td>
<td>/tū/</td>
</tr>
<tr>
<td>3</td>
<td>Three</td>
<td>/ˈTHRē/</td>
</tr>
<tr>
<td>4</td>
<td>Four</td>
<td>/ˈfɔːr/</td>
</tr>
<tr>
<td>5</td>
<td>Five</td>
<td>/ˈfīv/</td>
</tr>
<tr>
<td>6</td>
<td>Six</td>
<td>/ˈsiks/</td>
</tr>
<tr>
<td>7</td>
<td>Seven</td>
<td>/ˈseven/</td>
</tr>
<tr>
<td>8</td>
<td>Eight</td>
<td>/ˈēt/</td>
</tr>
<tr>
<td>9</td>
<td>Nine</td>
<td>/ˈnin/</td>
</tr>
<tr>
<td>0</td>
<td>oh</td>
<td>/ˈō/</td>
</tr>
</tbody>
</table>
Chapter 4
Signal Processing for Speech Recognition

4.1 Introduction

Human vocal system has a mechanical limitation that prohibit the rapid changes to the sound. Therefore, in short time of speech the signal doesn't change much, and their spectral characteristics relatively remain stationary in a sufficiently short period interval. For this reason, speech signal is divided into several short time frames. Each frame overlaps its previous frame to smoothing the transition from frame to frame [55]. If frame size is much shorter, the sample doesn’t enough to estimate the spectral, and if the frame is too longer then the signal changes too much throughout the frame (too much information).

![Speech signal preprocessing steps](image)

**Figure 4.1:** Speech signal preprocessing steps.

Each frame has multiplied by window function to keep the continuity of the first and the last points in the frame. The same process was repeated for all subsequent frames; thus, several unique coefficients are calculated and combined to produce a set of feature vectors. Figure 4.1, shows the input speech signal $y[n]$, which pre-emphasized to flatten speech spectrum, corrupted by adding some white noise to mimic the real conditions, separated into a sequence of uncorrelated frames, and
covered by window function to reduce signal discontinuity at the beginning and the end of the frames [56].

4.1.1 Pre-emphasis

In order to compensate the high-frequency part of the speech signal that was suppressed during the human sound production mechanism, a pre-emphasis process is chosen. This tricky process is very necessary in speech recognition task before spectral analysis. Pre-emphasis process often represented by first order High-pass filter (FIR), in order to flatten speech spectrum, and compensate the unwanted high frequency part of the speech signal [1]. Figure 4.2, illustrates the difference between original and pre-emphasized signal, and clarifying how the lower frequency components are toned down compared with higher frequency.

![Figure 4.2: Original Vs pre-emphasized speech signal.](image-url)
A simple form of pre-emphasis process can be implemented in time domain and z-domain using the below formula, and the MATLAB script code:

\[ x[n] = y[n] - a \times x[n - 1] \]  

(3-3)

\[ H(z) = 1 - aZ^{-1} \]  

(3-4)

Where,

- \( y(n) \): Input speech signal.
- \( x(n) \): Pre-emphasis speech signal.
- \( a \): Pre-emphasis parameter (0.9 to 1.0).

\[
\text{speech} \ (2: \text{end}) = \text{speech}_\text{raw} \ (2: \text{end}) - \text{pre_emp} \times \text{speech}_\text{raw} \ (1: \text{end}-1); \\
\]

Where,

- \( \text{speech}_\text{raw} \): input speech signal.
- \( \text{speech} \): Pre-emphasis speech signal.
- \( \text{pre_emp} \): Pre-emphasis parameter coefficient equal 0.975.

### 4.1.2 Signal-to-Noise Ratio Estimation

Signal-to-noise ratio (SNR or S/N) is a measurement, which used to compare between desired signal level and background noise level. It is defined as the ratio of signal power to the noise power, which means that the larger signal to noise ratio is the better the quality of the speech, and it given as:

\[ SNR = \frac{P_{signal}}{P_{noise}} \]  

(3-5)

Where,

- \( P \): the average power of both signal and noise power.
Based on signal processing analysis, some signals have a very wide dynamic range. Thus, they are often expressed using different scale named logarithmic decibel scale (dB). This scale makes it easier to deal with the big numbers as:

\[
P_{\text{signal,db}} = 10 \log_{10}(P_{\text{signal}})
\]

\[
P_{\text{noise,db}} = 10 \log_{10}(P_{\text{noise}})
\]

In order to express the general form of signal-to-noise ratio, SNR term is considered as the better expressed in logarithmic terms using dB scale, where the formula is given as:

\[
\text{SNR}_{db} = 10 \log_{10}\left(\frac{P_{\text{signal}}}{P_{\text{noise}}}\right)
\]

(3-5)

Since pre-emphasis process is completed, speech signal x[n] has been digitally disturbed and corrupted by white noise to mimic an adverse environment. In this process, different values of SNRs levels has been added started from 5dB to 30dB. Figure 4.3, shows the white noise, which added to the original speech signal (word Seven) using the next MATLAB script code.

```
speech_raw2 = v_addnoise (speech,fs,SNR);
```

where,

- `speech_raw2`: -Corrupted speech signal.
- `speech`: -Pre-emphasis signal.
- `fs`: -sampling rate = 8 kHz.
- `SNR`: -Signal-to-Noise Ratio value.
Figure 4.3: Word seven, corrupted by different SNR values.

4.1.3 Frame Blocking and windowing

Speech is a time varying acoustic signal due to alteration in the vocal tract. However, in speech recognition, usually assume that the statistical properties of the speech signal changing slowly over time. This assumption makes it possible to process signals in very short time intervals. Thus, short-time speech frame and
overlapping is the best way to capture the speech parameters, which are extracted frame-by-frame [57].

In order to collect a sufficient physical characteristics, input speech signal is segmented into a number of frames of 10~40ms, and then overlapping with short frame of 1/3~1/2 of the frame size. Practically, there are some causes of doing frame processing. Firstly, properties of the signal are easier to be seen in small frames such as the energy level of a speech signal. Secondly, most of the signal analyses available in frequency domain such as short-time Fourier transform, lastly, all the application is in real-time system.

As shown in Figure 4.4, hamming function is used to minimize the nearest side lobe in speech processing [58]. In short-time analysis, hamming window function is applied onto the frames, where the mathematical solution is given by:

\[ w(k) = 0.54 - 0.46 \cos \left( \frac{2\pi n}{N-1} \right), \quad 0 \leq n \leq (N-1) \]  

(3-6)

Where,

N: -Represents the length of window.
n: - Sequence of samples.

After calculating the hamming window \([w(k)]\), the next step is to multiply each frame by a windowing function to cover the entire speech sequence using the next formula:

\[ x_t[n] = x[k] w[k-p] \]  

(3-7)

Where,

x[k]: -the speech sequence.
x_t: -windowed speech frame at time t.
As a matter of fact, the proper frame duration is actually depending on the change of the rate of vocal tract shape. Therefore, Frames of 25ms and 60% overlapping was chosen in our research. Thus, the first (200) samples in the utterance are starting from sample (0 to 200) and the next (200) samples will starts from sample (80 to 280) etc. In short time intervals, input speech signal $x[n]$ was divided into $K$ samples short frame buffer. Each frame is overlapped with short frame to ensure the smoothing transition of estimated parameters from frame to another as shown in Figure 4.5. In our research, frame size, frame shift, and frames number are calculated using next MATLAB script codes:

```matlab
frame_size = round(fs*frame_size_sec);
```
\text{frame\_shift} = \text{round}(fs \times \text{frame\_shift\_sec});

\text{frame\_no} = \text{floor}(1 + (\text{len} - \text{frame\_size})/\text{frame\_shift});

Where,

\begin{itemize}
  \item \text{fs}: \text{-sampling rate}
  \item \text{frame\_size\_sec}: \text{-Size of frame in seconds.}
  \item \text{frame\_size}: \text{-Number of samples in frame.}
  \item \text{frame\_shift\_sec}: \text{-Size of frame shift in seconds.}
  \item \text{frame\_shift}: \text{-Size of frame shift.}
  \item \text{frame\_no}: \text{-No. of frames in each utterance.}
\end{itemize}

\textbf{Figure 4.5:} Hamming window applying on speech frame.
Hamming window curve size of 25ms, 8kHz sampling frequency, and 10ms frame shift was picked up and apply on the frames using the next MATLAB script code.

```matlab
win = createHammingWindow(frame_size);
```

Where,

- `frame_size`: number of samples in one frame.
- `Win`: output frame window.

According to the initial condition configuration values of frame length, shift length, and sampling rate, we get;

- Block length in each frame = frame size in sec * sample rate = 0.025 * 8000 = 200 samples
- Shift length = shift size in sec * sample rate = 0.010 * 8000 = 80 samples frame shift
- Overlapping = overlapping size in sec * sample rate = 0.015 * 8000 = 120 = 60.0% overlapping
Chapter 5
Acoustic features for speech recognition

5.1 Introduction

In order to extract the features, the physical characteristics of specific frame, such as; power, pitch, and vocal tract configurations etc., are converted into some types of parametric representation at a lower information rate.

Feature extraction is the most important process in ASR system, because it is responsible for converting acoustic signal into a sequence of acoustic features, these features are normally consisting of static and dynamic coefficients. Dynamic coefficients can be devoted by adding the first and second derivative approximation into static feature parameters [48]. In our work, four different sets of features algorithms were designed in order to evaluate the proposal ASR system as follows:

- Mel-scale Frequency Cepstral Coefficients (MFCC).
- Linear Predictive Coding Coefficients (LPCC).
- Perceptual Linear Prediction Coefficients (PLP).
- Relative Spectral Perceptual Linear Prediction Coefficients (RASTA-PLP).

All of these feature extraction methods have been selected partly for the following reasons; their calculations lead to a source-filter separation, have an analytically tractable model, and finally, the experience shows that these parameter coefficients work fine in recognition applications [48].

During front-end phase, feature vectors were generated based on TIDIGIT data base. These features can either be only “static” with the energy, or “static with the energy + delta + delta-delta”. Static vector means that the feature vectors are

35
contains p number of coefficients, where p is the order, plus an additional zeroth order term. While “static + delta + delta-delta” vectors have both the 1st and 2nd order dynamic features appended to the “static” feature vectors.

In our research, conventional front-ends generated 39 parameter coefficients in the form of “[12-static + 1-energy] + [13-delta] + [13-delta-delta]”, while hybrid front-ends generated 39 parameter coefficients 13-static coefficients from each three conventional methods. In order to imitate the reality, training data set was corrupted by adding different SNR background noise.

5.2 Conventional Features

5.2.1 Mel-scale Frequency Cepstral Coefficients (MFCC)

MFCC algorithm, which is widely used in automatic speech and speaker recognition is the most dominant method used to extract spectral features. This algorithm was first introduced to represent the short-term power spectrum of the sound, which established based on a nonlinear Mel-scale of frequency [47].

In order to compute the MFCC features, and obtain the certain parameters spectral magnitude, the first step is to calculate the power spectrum of each frame using Fast Fourier Transform, and then finding the periodogram using Mel-Frequency Filter Bank to determine the energy levels of frequency regions, which contains a lot of information about the speech.

Because the human ear doesn’t have a linear scale in term of hearing then the next step is to take a log of filter bank energies, then DCT of the log filter bank energies is calculated in order to decorrelate the overlapped filter bank energies for better classification. The final step is to compute the cepstral lifting by keeping 2
to 13 DCT coefficients and discard the rest [22]. Figure 5.1 represents a proposed diagram for MFCC computation.

![Diagram](image)

**Figure 5.1:** MFCC Feature extraction block diagram.

### 5.2.1.1 Fast Fourier Transform

Fast Fourier transform (FFT) algorithm is a fast implementation of the DFT, which converts N-samples of frames into frequency spectrum. After windowed process is completed, the next step is to apply discrete Fourier Transform on each frame in order to transfer time domain samples into frequency domain. FFT reduces the computation time that required to compute a discrete Fourier transform, and improves the performance by a factor of 100 or more.
DFT method requires $N^2$ operations, while FFT algorithm requires $N \log_2 N$ operations, so it is widely used to deal with data in speech recognition. As shown in Figure 5.2, given the windowed speech signal $x(n)$, the direct computation of the DFT windows speech signal can be expressed as the next equation:

$$X(k) = \sum_{n=0}^{N-1} x[n]e^{-j2\pi nk/N}, \quad 0 \leq k < N$$ \hspace{1cm} (4-1)

Where,

- $J$: Imaginary unit.
- $N$: Number of FFT points (larger than the frame size).
- $X(k)$: Complex numbers.

![FFT of Windowed Speech Signal](image)

*Figure 5.2: FFT windowed speech signal.*
The number of FFT points per frame should approximately equal or greater than the size of that frame. When using 8 kHz sampling rate, a 256-points FFT were needed to transform 200 samples into frequency spectrum as shown in the next script code:

```matlab
PowerSpec = abs(fft(win_speech,NFFT)); % applyFFT()
```

Where,

- `win_speech`: windows speech frame.
- `Nfft`: -n-point Fourier transform (255 points).

As soon as FFT is taken of the windowed speech, a series of time discrete spectral frames is obtained. The result is often referred to as spectrum or periodogram, as shown in Figure 5.3.

Figure 5.3: Power spectrums of speech signal ‘She had your dark suit in greasy wash water all year’. Top: Amplitude waveform, bottom: spectrogram. Magnitude spectrum was calculated in frames of 256 samples, and DFT was evaluated at 256 points.
5.2.1.2 Spectral magnitudes

Because ASR system deals only with the real part of speech [58]. Therefore, the image parts of spectrum signal are ignored, and the magnitude of the complex value $X(k)$ is given as:

$$|X(k)| = \sqrt{\text{Real}(X(k))^2 + \text{image}(X(k))^2} \quad (4-2)$$

5.2.1.3 Mel-frequency filterbank

Due to the concept said that human hearing is less sensitive at frequencies above 1000 Hz [59]. Therefore, the spectrum is warped using a logarithmic Mel-scale to emphasize the low frequency over the high frequency, as shown in Figure 5.4.

To achieve the goal, Mel filterbank spectrum used a set of overlapping triangular bandpass filter under a non-linear frequency scale. The center frequencies of these filters are linear equally-spaced below 1 kHz. The relationship between the Mel-scale and linear frequency scale is given as:

$$F_{\text{mel}} = 2595 \cdot \log_{10}(1 + \frac{F_{Hz}}{700}) \quad (4-3)$$

$$F_{Hz} = 700 \cdot \left(10^{\frac{F_{\text{mel}}}{2595}} - 1\right) \quad (4-4)$$

Where,

$F_{\text{mel}}$: -Mel-scale filterbank.

$F_{Hz}$: -Central frequency of each Mel-scale.
Mel filterbank, is used to emphasize center frequencies correspond to Mel center frequencies using the following MATLAB script code.

```matlab
mel_spectrum = applyMelFilters(freq_spectrum, melFilters)';
```

Triangular Mel scale filterbank handles the warping between frequency in Hz and frequency in Mel-scale. Figure 5.5, shows the linear frequency scale between $f_L(mel)$ and $f_H(Mel)$ of a triangular bandpass filter.
5.2.1.4 Logarithm of Filter energies

After passing through the filterbanks, the log-energy at the output of each filter is calculated and expressing as:
\[
S(m) = \log_{10}\left[\sum_{k=0}^{N-1}|X(k)|^2 \cdot H_m(k)\right]
\]  

(4-7)

The logarithmic scale calculates the energy from spectrum using the following MATLAB script code.

```matlab
log_mel_spectrum = log10(mel_spectrum);
```

### 5.2.1.5 Discrete Cosine Transform

Since Log power spectrum is a real and symmetric, therefore, discrete cosine transformation (DCT) is used instead of (DFT). This method leads to using diagonal covariance matrices instead of full covariance matrices. thus, complexity and computational cost can be reduced when modeling the features [60]. Significant reduction in computational cost can be achieved as:

\[
\text{cep}_x(n;m) = \sum_{k=0}^{N-1} \alpha_k \cdot \log f_{\text{mel}_k} \cdot \cos\left(\frac{\pi(2n+1)k}{2N}\right), n = 0, 1, 2, \ldots N - 1
\]

(4-5)

By discarding higher order coefficients, DCT gathers most of the information from its lower order coefficients. As a result, the first few coefficients \(k\) are collected as a feature of a specific frame, which is typically ranges between 8 and 13 using the following MATLAB script code.

```matlab
temp_mfcc = dct(log_mel_spectrum);
```

### 5.2.1.6 Lifting or Cepstral Weighting

After decorrelating the cepstral coefficients, a lifting algorithm is applied on the log magnitude spectrum to minimize the sensitivity. This process has been completed by excluding the first two coefficients, and collecting the rest, which is normally equal 13 as features. All of these coefficients are collected by calculating the real cepstrum of a window-weighted frame of speech as:
\[ \dot{C}(k) = C(k) \left[ 1 + \frac{K}{2} \sin \left( \frac{k\pi}{K} \right) \right], \quad 0 \leq k < K \quad (4-6) \]

Lifter process, utilized to minimize the sensitivities by lessening the higher and lower cepstral coefficients using the following MATLAB script codes.

```matlab
mfcc = temp_mfcc(3:14);
n=(1:cep_order)';
lifefer_weighting=1+(lifter/2)*sin(pi*n/lifter);
mfcc(:,fr)=lifer_weighting.*mfcc(:,fr);
```

Where,
- `cep_order`: -MFCC coefficients.
- `lifter`: -final cepstral coefficients equal 22.
- `fr`: -frame number.

### 5.2.1.7 Dynamic feature

In order to represent the dynamic characteristic. Time derivative approximations are obtained by finding the first and second derivative of the cepstral coefficients. These extra coefficients called delta and acceleration coefficients, which derived and computed using a standard linear regression formula as:

\[ d_t = \frac{\sum_{\theta=1}^{\theta} \theta (c_{t+\theta} - c_{t-\theta})}{2 \sum_{\theta=1}^{\theta} \theta^2} \quad (4-8) \]

Where,
- `dt`: -Delta coefficient at frame (t).
- `ct`: -Static coefficients.
- `\Theta`: -Window size.
First-order delta coefficients and second-order delta coefficients are added to the feature vectors using the next MATLAB script codes.

delta_weight = ones (1,5);
d_fea=(0.01/0.010) * Delta([mfcc;logpow],delta_weight);
dd_fea=(0.01/0.01) * Delta(d_fea,delta_weight);

5.2.2 Linear Prediction Coding Coefficients (LPCC)

Figure 5.6: LPCC feature extraction block diagram.
In order to represent the short-time spectrum, LPC algorithm is selected to estimate the parameters of speech signals. In our work, LPC coefficients were computed using autocorrelation and Levinson-Durbin recursion methods. Next, LPC parameters are converted into L(Ω) cepstral parameters using cepstrum analysis [61]. Figure 5.6, illustrates the common process that involving to obtaining LPCC features.

Autocorrelation method was used instead of covariance and lattice methods in order to resolve the Yule Walker Equations. This is due to autocorrelation domain has recently proved to be successful for robust speech recognition [62].

5.2.2.1 Autocorrelation analysis

Autocorrelation technique, is almost exclusively method used to find the correlation between the signal and itself [57]. In other words, each windowed set of speech samples is autocorrelated to give a set of (p + 1) coefficients. The long-term autocorrelation of a power signal is defined as:

$$ R(n) = \sum_{n=0}^{N_w-1-m} s_w(n) s_w(n + m) \quad 0 \leq m \leq p \quad (4-9) $$

Where,

- $N_w$: Length of the used window
- $S_w$: Windowed segment
- $P$: Order of the LPC analysis

Autocorrelation method is corresponding to resolve a basic matrix equation, which giving as:

$$ r \approx Rx_a \quad (4-9) $$
Where,

\( R \): Autocorrelation matrix of the speech signal

\( r \): Autocorrelation vector of the speech signal

\( a \): Vector of the LPC coefficients.

Levinson-Durbin recursion algorithm, is used to solve a set of linear autocorrelation equations. The recursion finds the solution of all prediction coefficients of order less than \( p \). Where \( R[p] \) is a correlation function of windowed speech frame.

\[
\begin{bmatrix}
R[0] & R[1] & \cdots & \cdots & R[p-1] \\
R[1] & R[0] & \cdots & \cdots & \cdot \\
\cdot & \cdot & \cdot & \cdots & \cdot \\
R[p-1] & \cdots & \cdots & R[1] & R[0]
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
\cdot \\
a_p
\end{bmatrix} =
\begin{bmatrix}
R[1] \\
R[2] \\
\cdot \\
R[p]
\end{bmatrix}
\] 

To find the pitch and the repeating patterns, each frame of windowed signal is auto correlated in order to find the correlation between the signal and a delayed version itself. This will be done using the following MATLAB script code.

```matlab
[rs,eta] = xcorr(win_speech,cep_order,'biased');
```

Where,

\( \text{win\_speech} \): -signal vector

\( \text{rs} \): -autocorrelation vector

\( \text{eta} \): -vector of lag indices between (-eta\_max to eta\_max)

Figure 5.7 represents the power signal, which correlated using long-term autocorrelation function in order to resolve basic matrix equations.
5.2.2.2 LPC analysis

The basic idea of LPC is to approximate the current speech sample as a linear combination of past samples. This module gets windowed data from the window module. Windowing data representing the spectral envelope of a digital signal of speech in compressed form using the information of a linear predictive model. LPC analysis converts each frame of \((p+1)\) autocorrelation coefficients into LPC parameters, where Levinson-Durbin’s recursive procedure is one of these algorithms [22], and can achieved as:
\[ E_0 = r(0) \] (4-12)

\[ k_i = \frac{r(i) - \sum_{j=1}^{i-1} \alpha_{i-1}^{j-1} r(|i - j|)}{E^{i-1}} \quad 1 \leq i \leq p \] (4-13)

\[ \alpha_i^{(i)} = k_i \] (4-14)

\[ \alpha_i^{(i)} = \alpha_{i-1}^{(i-1)} - k_i \alpha_{i-j}^{(i-1)} \quad 1 \leq j \leq i - 1 \] (4-15)

\[ E^{(i)} = (1 - k_i^2)E^{i-1} \] (4-16)

\[ a_m = \alpha_m^{(p)} \] (4-17)

Where,

P: - Order of the LPC analysis

Ki: - The reflection or BARCOR coefficients

aj: - LPC coefficients

Durbin’s method solves the LPC matrix equations in which the recursion finds the solution of all prediction coefficients of order less than p. LPC solve the equations in order to provide both polynomial coefficients and reflection coefficients, where polynomial coefficients are used to compute and plot the LPC spectrum, as shown in Figure 5.8. Durbin recursion can be derived using the following MATLAB script code.

\[ [a(1:cep_order), lag] = durbin(rs(cep_order+1:2*cep_order+1), cep_order); \]

Where,

cep_order: - lpc order (default is 12).

lag: - vector of lag.
5.2.2.3 Cepstrum Analysis

The sufficient numbers of cepstral analysis can be derived from the infinite numbers of LPC coefficients. Using next formulas, the first 12 to 20 cepstrum coefficients can be calculated, where the default value is 12, as shown in Figure 5.9,

\[ c_m = a_m + \sum_{k=1}^{m-1} \frac{k}{m} \cdot c_k \cdot a_{m-k} \quad 1 \leq m \leq p \quad (4-19) \]

\[ c_m = \sum_{k=m-p}^{m-1} \frac{k}{m} \cdot c_k \cdot a_{m-k} \quad m > p \quad (4-20) \]
5.2.3 Perceptual Linear Prediction (PLP)

In order to calculate several spectral characteristics from speech signal, PLP algorithm was established to emulate the human auditory system as shown in Figure 5.10. Cepstral analysis in PLP is Somewhat similar to LPCC analysis, but, there are three main concepts behind PLP analysis, include; critical band frequency, equal-loudness curve, and intensity-loudness power law [29].
5.2.3.1 Frequency band analysis

Based on non-linear frequency scale filter bank, Bark scale was used instead of Mel-filter bank in PLP analysis. The mathematical relationship between the Bark scale and the linear frequency scale is given as:

$$\Omega(\omega) = 6 \ln \left( \frac{\omega}{1200\pi} + \left( \frac{\omega}{1200\pi} \right)^2 + 1 \right)^{0.5}$$  \hspace{1cm} (4-22)
Filter bank is a system divides the input speech signal into a set of analysis signals, which corresponds to a different region in the spectrum. Typically, the regions in the spectrum given by the analysis signals from approximately 20 Hz to 20 kHz.

In order to simulate the frequency response of the ear using critical band curve $\Psi(\Omega)$, all loudness of the tones lying within a critical bandwidth was integrated into one loudness of equivalent tone. The critical band curve of Bark-scale is given as:

$$
\Psi(\Omega) = \begin{cases} 
0 & \text{for} \quad \Omega < -1.3 \\
10^{2.5(\Omega+0.5)} & \text{for} \quad -1.3 \leq \Omega < 0.5 \\
1 & \text{for} \quad -0.5 \leq \Omega < 2.5 \\
10^{-(\Omega-0.5)} & \text{for} \quad 0.5 \leq \Omega < 2.5 \\
0 & \text{for} \quad \Omega > 2.5 
\end{cases}
$$ (4-23)

In our work, spectral resolution of $P(\omega)$ has been reduced using masking critical band curve $\Psi(\Omega)$ equations to cover the frequency range. The output of $\Psi(\Omega)$ convolution is then down-sampled by sampling it in 1-Bark intervals at integer points between 1 to 15. Figure 5.11 illustrates the Bark-scale filter bank on linear frequency scale sampled by 8 kHz sampling frequency by using the following MATLAB script code:

```matlab
[aspectrum wts] = audspec(psignal, fs);
```

The Nyquist frequency is defined as 5 kHz, which is equal to 16.9 Barks. The transformation formula shows that the first and the last filter is located at 0 and 16.9 Barks respectively. There will be a total of 18 filters with step sizes of 0.994 Barks.
The shape of the 1st filter at 0 Bark is identical to the 2nd filter, while the shape of the last filter at 16.9 Barks is the same as the 17th filter. Figure 5.12, illustrates the filtered spectrum of 18 filter in the Bark-scale. This filter is defined as the sum of the product of each FFT point on the speech power spectrum and the filter weights.

![Bark-scale filter mode](image1)

**Figure 5.11:** Bark Scale Filter Bank (Sampling Frequency = 8 kHz).

![Bark scale](image2)

**Figure 5.12:** Filtered spectrum of 18 filter in The Bark-Scale.
5.2.3.2 Equal-loudness curve

Human ear is more sensitive to frequencies between 500 Hz and 4 kHz, and doesn’t hear all frequencies with equal sensitivity. Therefore, to estimate the sensitivity of human hearing at several frequencies, the sampled convolution output re-emphasized by an approximation of the equal-loudness curve.

\[ E(\omega) = \frac{(\omega^2 + 56.8 \times 10^6)\omega^4}{(\omega^2 + 6.3 \times 10^6)(\omega^2 + 0.38 \times 10^9)} \]  
(4-24)

5.2.3.3 Intensity-loudness power law

Human hearing has a nonlinear relationship between the perceived loudness and sound intensity. Therefore, PLP model approximated this relationship as cubic root using the following formula:

\[ L(\Omega) = I(\Omega)^{0.33} \]  
(4-25)

Where,

L(\Omega): -Output of the equal-loudness pre-emphasis operation.

Intensity-loudness compression can be derived in PLP analysis using the following function:

\texttt{postspectrum} = \texttt{postaud (aspectrum, fs/2)};

5.2.4 Relative Spectral Perceptual Linear Prediction (RASTA-PLP)

RASTA-PLP is a feature extraction algorithm invented by Hermansky in 1994. This algorithm was achieved to extract more robust features to linear spectral distortions [33]. RASTA-PLP is like PLP algorithm, except it using band pass filter to filter the time trajectory in each spectral component, as shown in Figure 5.13.
To extract the features, RASTA-PLP algorithm passes through a series of steps, which discussed as follows:
I. Compute the critical-band power spectrum as in PLP.

II. Transform spectral amplitude through a compressing static nonlinear transformation.

III. The time trajectory of each transformed spectral component was filtered by bandpass IIR filter, using the next formula.

\[ H(z) = 0.1 \frac{2 + z^{-1} - z^{-3} - 2z^{-4}}{z^{-4}(1 - 0.98z^{-1})} \]  

(4-30)

IV. Transform the filtered speech via expanding static nonlinear transformations.

V. Multiply the filtered speech by the equal loudness curve and raise it to the 0.33 power to simulate the power law of hearing as in PLP.

VI. Compute an all-pole model of the spectrum, as in PLP.

5.3 Hybrid Feature Extraction

A new hybrid algorithms were developed using a combination of some conventional feature extraction methods include; MFCC, LPC, PLP, and RASTA-PLP. These algorithms were evaluated and designed in order to provide a 52-hybrid parameter coefficients, 13-parameters from each conventional method. Each time, three different kinds of features were selected to carry on 39-parameter hybrid features and placed together in one vector, as shown in Figure 5.14.

The hybrid system is completely designed using MATLAB program, verified, and tested in adverse environments. The performance evaluation of the new hybrid algorithms is evaluated, comparing with the conventional methods, and discussing in detail in the next sections. The new hybrid feature extraction algorithms are as follows:

In order to recognize the new hybrid features behavior, the pitch information over all speech frames for training and testing data set are extracted during feature extraction time in front-end stage. Furthermore, the output features are fed into back-end stage in order to classify and decode the data using Hidden Markov Model technique, which will be explain in detail in chapter 6.
Figure 5.14: Hybrid feature extraction analysis block diagram.
Chapter 6
Acoustic Modeling for speech recognition

6.1 Introduction

Acoustic modeling, is a part of statistical inference foundation that embodies a set of assumptions concerning the generation of the observed data. Speech data is described as a set of probability distributions of speech using one of distribution functions, such as Gaussian, Laplacian, and Gamma probability density functions.

Statistical modeling, which generated during training process are consists of likelihood probability of states sequences. Speech engine takes the probability of impersonality words and statistically compares them to other words in statistical models. These models are developed using data captured from many hundreds of samples of native speakers of the language. When a match is found, the corresponding word is recognized and as will a final decision.

Compared with other decoding analysis, such as ANN, VQ, and DTW. HMM classifier considered the most powerful statistical tool that can used in speech recognition and speaker identification systems [63], due to the ability of model non-linearly aligning speech and estimating the model parameters [36, 64].

6.2 Hidden Markov Model (HMM) Classifier

HMM is a finite-state machines with a set of hidden states, observations, transition probability matrix, emission probability matrix, and initial state probability matrix. As shown in Figure 6.1, HMM technique is characterized by a set of parameters $\lambda$, and defined by the following set of elements:
• Hidden states set $Q = [q_i]$ , $i = 1, \ldots, N$.

• Observation set $O = [o_k]$ , $k = 1, \ldots, M$.

• Transition probability matrix $A = \{a_{ij} = P(q_i at t + 1 \mid q_i at t) \}$, $t = 1, \ldots, T$.

• Emission probabilities matrix $B = \{b_{lk} = b_{l(o_k)} = P(o_k \mid q_l) \}$.

• Initial state probabilities distribution vector $\pi = [\pi_l = P(q_l at t = 1)]$.

Figure 6.1: HMM elements.

Statistical model HMM, is a technique used as a recognize gesture by computing likelihood computation of the observation sequence given the model. The system followed by chosen the state sequence giving the observation sequence and the model, and adjusting the probability to maximize the $P(O \mid \lambda)$. Practically, HMM has three canonical issues must be solved, which listed as follows [30]:

61
1. **Evaluation or scoring problem**, Given the observation sequence \((O)\) and the model parameters \((\lambda)\), what is the probability of the observation sequence given the model \(P(O|\lambda)\).

2. **Decoding problem**, given the observation sequence \((O)\) and the model parameters \((\lambda)\), how can the state sequence \([q_t]\) be chosen.

3. **Learning or training problem**, given an output sequence, How the probability can adjust to maximize \(P(O | \lambda)\).

Among the three problems, evaluation problem can be solved using forward and backward iterative algorithms, while learning and decoding problems are solving using Baum-welch and Viterbi algorithm respectively. Baum-Welch or Maximum Likelihood Estimation algorithm are deals with teaching or training process to estimate and re-estimate the parameters, while Viterbi Algorithm is used to provide the best path by sequentially considering each observation sequence.

### 6.2.1 Initialization

Before training the system using training data set, the system should be initialized using the next MATLAB script code as shown:

```matlab
start_initializing('...training_list.mat',dim,model_mat);  
```

Where,

- `training_list.mat`: -list of training data.
- `model_mat`: -model matrix.

The first step of the initialization process is to generate the transition probability matrix \(A\), and the initial probability matrix \(\pi\). In our research, matrix \(A\)
is initialized with equal probability for each state, while the initial state distribution vector \( \pi_i \) is initialized with probability to be in state one at the beginning, and it is also assumed that \( i \) is equal to five states in this case, as shown in equation (4-25, 4-26).

\[
A_{ij} = \begin{bmatrix}
0.5 & 0.5 & 0 & 0 & 0 \\
0 & 0.5 & 0.5 & 0 & 0 \\
0 & 0 & 0.5 & 0.5 & 0 \\
0 & 0 & 0 & 0.5 & 0.5 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]  
(4-25)

Where,

\( i \): -current state.

\( j \): -next state.

\[
\pi_i = [1 \ 0 \ 0 \ 0 \ 0]
\]  
(4-26)

Where,

\( i \): -Number of states, \( 1 \leq i \leq N \).

Meanwhile, feature vectors are normally not frequency distribution. In this case, Gaussian distribution function is used to describe the observations, and then store the data in matrix B, which described by mean and variance.

### 6.2.2 Evaluation

Scoring is the first canonical problem in HMM, which lies in how to evaluate the probability of a particular output sequence given the model. However, this probability can be computed using forward and backward algorithms, and then the computations of the possible state sequence paths are stored in a matrix.
in our work, forward and backward, Gaussian distribution, and Baum-welch processes were applied on the feature vectors in each HMM states in order to find the probabilities of state and observation sequence.

6.2.2.1 Forward algorithm

Forward algorithm, is a direct computation probability of the partial observation sequence \(O_t\) up to time \(t\), and the system is at state \(i\) at time \(t\) given the model \(P(O|\lambda)\). As shown in Figure 6.2, forward variable \(\alpha_t(i)\) is produced by summing over all possible states sequence at \(i\) state, which defined as:

\[
\alpha_t(i) = P(O_1, O_2, O_3, ..., O_t|\lambda)
\]

(5-1)

![Figure 6.2: Forward algorithm procedure.](image)

Forward probability \(\alpha_{t+1}(i)\), can obtained by summing all the forward variable over all \(N\) state sequences, starting at the first state and being at time \(t\) in state \(j\), and
then multiply it by transition probability \((a_{ij})\) and emission probability \(b_j(O_{t+1})\), as shown in the following procedures [65].

1. **Initialization**

   \[ \text{Set } t = 1; \]

   \[ \alpha_t(i) = \pi_i b_j(o_i), \quad 1 \leq i \leq N \]

2. **Recursion**

   \[ \alpha_{t+1}(j) = b_j(o_{t+1}) \sum_{i=1}^{N} \alpha_t(i)a_{ij}, \quad 1 \leq j \leq N \]

3. **Update time**

   Set \( t = t + 1; \)

   If \( t < T \) go to step 2

   Else, go to step 4

4. **Termination**

   \[ P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_{t}(i) \]

Forward probability procedure is accomplished by calculating the forward variables (alfa scaled). Alfa scaled, which is the direct computation probability can be computed at any time \( t \), where \( 1 \leq t \leq T \geq \), as shown in Figure 6.3.
Figure 6.3: Forward probability, alfa scaled.

6.2.2.2 Backward algorithm

Figure 6.4: Backward process.
Like forward algorithm, backward $\beta(i)$ was calculated inductively, as shown in Figure 6.4. Backward Recursion is a conditional probability of the observation sequence, which started at the last state and being at time $t+1$ to time $T$ given the state $i$ and model $\lambda$ at time $t$, where given as:

$$\beta(i) = P(o(t + 1), o(t + 2), \ldots, o(T) \mid \lambda) \quad (5-2)$$

Backward probability $P(O\mid \lambda)$ can be evaluated by calculating the backward variables $\beta(i)$, and the probability is calculated by going backward along the observation sequence [65], as shown in the following procedures:

1. **Initialization**
   
   Set $t = T - 1$;
   
   $\beta_t(i) = 1, \quad 1 \leq i \leq N$

2. **Recursion**
   
   $$\beta_i(i) = \sum_{j=1}^{N} \beta_{t+1}(i)a_{ij}b_j(o_{t+1}), \quad 1 \leq i \leq N$$

3. **Update time**
   
   Set $t = t - 1$;
   
   if $t \geq 0$ go to step 2;
   
   Else, terminate the program.
In like manner of forward probability, beta scaling $\beta_t(i)$ are calculated in order to find the backward probabilities. In this procedure, Probability at time $t$ and in state $i$ given the model, having generated the partial observation sequence from $t+1$ observation until observation number $T$, $O_{t+1}O_{t+1}...O_T$, as shown in Figure 6.5.

Practically, forward recursion process itself is not enough to compute $P(O \setminus \lambda)$. This is due to the complexity of precision range when calculating with multiplications of probabilities. Thus, the combination of them makes a scaling of both $\alpha$ and $\beta$ very necessary, which giving as:

As shown in Figure 6.6, the scaling factor of both forward and backward

$$P(O \setminus \lambda) = \sum_{i=1}^{N} P(O, q_i = \lambda) = \sum_{i=1}^{N} \alpha_i(i) \beta_i(i)$$ (5-3)

variable is dependent only of the time $t$ and independent of the state $i$.  

68
6.2.3 Learning (Parameters Estimation)

Learning or training process estimated the model parameters that best describe the process. This procedure aims to encode the characteristics of the observation sequence in order to recognize a similar observation sequence in future [65]. This problem can be formed as.

\[ \hat{\lambda}^* = \arg \max_{\lambda} P(O | \lambda) \]  

Getting new model (\( \lambda^* \)) from all possible (\( \lambda \)) given the observations is considered the most difficult problems in training process. This is due to no way to analytically finding the model parameters in a closed form. In training process model parameters mean, variance, and transition probability matrix are re-estimated to maximize the \( P(O | \lambda) \), using the following MATLAB script code.
start_training ('training_list.mat', dim);

### 6.2.3.1 Multivariate Gaussian distribution

Gaussian or normal distribution function, is the most common and easiest process that used in statistics to analyze continuous distribution data. This process described by two parameters, mean $\mu$ (location) and variance $\sigma^2$ (dispersion). If the possible values of the observation $o_t$ are normally distributed. Then, the observation likelihood function $b_j(ot)$ is represented as a Gaussian with mean $\mu_j$ and variance $\sigma_j^2$, as shown in Figure 6.7.

![Figure 6.7: Multiple mixture gaussian.](image)

If the observation was a single real-valued 1D vector instead of 39D vectors, and we had learned a Gaussian over the distribution of values of this feature, then we can compute the likelihood of any given observation $O_t$, using the next formula:
\[ b_t(O_t) = p(O|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp \left( -\frac{(O_t - \mu_j)^2}{2\sigma_j^2} \right) \]  

(5-8)

Where,

\( O \) and \( \mu \): -1-by-\( d \) vectors.

\( \sigma \): -\( d \)-by-\( d \) symmetric positive definite matrix.

In single Gaussian distribution, which characterized by mean and variance, the maximum likelihood parameters that estimates from the observation data is gaining as:

\[ \mu_i = \frac{1}{T} \sum_{t=1}^{T} O_t \]  

(5-8)

\[ \sigma_i^2 = \frac{1}{T} \sum_{t=1}^{T} (O_t - \mu_i)^2 \]  

(5-9)

Since we don’t know which observation was produced by which state, then what we want to do is to assign each observation vector \( o_t \) to every possible state \( i \), prorated by the probability the HMM was in state \( i \) at time \( t \), which giving as:

\[ \bar{\mu_i} = \frac{\sum_{i=1}^{T} \zeta_i(i) o_t}{\sum_{i=1}^{T} \zeta_i(i)} \]  

(5-13)
\[ \sigma^2_i = \frac{\sum_{i=1}^{T} \zeta_i (o_i - \mu_i)^2}{\sum_{i=1}^{T} \zeta_i} \]

(5-14)

Where,
\( \zeta_i \): the probability of being in state \( i \) at time \( t \)

Instead of a single mean and variance, vector of means \( \mu \), and covariance matrix \( \Sigma \) are used, and the equation can be expressed as:

\[ p(O|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2} \Sigma^{1/2}} \exp \left( -\frac{1}{2} (O - \mu)^T \Sigma^{-1} (O - \mu) \right) \]

(5-5)

Where,
\( d \): - Number of dimensions
\( x \): - Input vector
\( \mu \): - Mean vector, E(O).
\( \Sigma \): - \( d \times d \) symmetric covariance matrix, \( \Sigma = E[(O - \mu)(O - \mu)^T] \)

In our work, (pdf) is parameterized by mean vector and covariance matrix. So, the \( i-j \)th element of \( \Sigma \) is \( \sigma^2_{ij} = E[(O_i - \mu_i)(O_j - \mu_j)] \), and the argument to the exponential \( \frac{1}{2} (O - \mu)^T \Sigma^{-1} (O - \mu) \) is referred to as a quadratic form, which helps to determine the relationship between two components. If \( O_j \) is large when \( O_i \) is large, then \((O_j - \mu_j)(O_i - \mu_i)\) will tend to be positive. But, if \( O_j \) is small when \( O_j \) is large, then \((O_j - \mu_j)(O_i - \mu_i)\) will tend to be negative.
6.2.3.2 Baum-Welch algorithm

Baum-Welch, is the recommended algorithm that used to learn the speech recognition system. This is due to numerically stable every iteration, converges to a local optima, and it has linear convergence that used to maximize the likelihood of a model $\lambda^*$[36, 63]. Thus, Baum-Welch algorithm is selected to re-estimates the model parameters to a closer value of the global maximum. In each training iteration, Baum-welch trying to learn and store the characteristics of the observation sequence that best describe the process.

Baum-Welch is an efficient procedure, which used to evaluate the expected number of transition from state $i$ ($\gamma(i)$), and the expected number of transition from state $i$ to $j$ ($\xi(i)$). According to Bayes' theorem, all what we need for the purpose of Baum-Welch algorithm are the following:

- Probability $\gamma(i)$ of being in state $i$ at time $t$ given the observed sequence $O$, using the next formula:

$$
\gamma_t(i) = \frac{P(O, q_t = i \mid \lambda)}{\sum_{j=1}^{N} P(O, q_t = j \mid \lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \quad (5-5)
$$

- Conditional probability $\xi(i)$ of being in state $i$ at time $t$, and being in state $j$ at time $t+1$, given the observed sequence $O$ and model parameters $\lambda$, which derived as:

$$
\xi_t(i, j) = P(q_t = i, q_{t+1} = j \mid O, \lambda) = \hat{\alpha}_t(i)a_i b_j(a_{t+1})\hat{\beta}_{t+1}(j) \quad (5-6)
$$

- Lastly, the connection between $\gamma_t(i)$ and $\xi_t(i)$ is giving as:
\[ \gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j) \]  

(5-7)

In order to re-estimate the model parameters, the expected number of transitions is used in the next sections to find the new transition probability matrix, mean, and variance.

**Re-estimate the state transition probability matrix**

To help maximize the probability of the model, \( \zeta \) and \( \gamma \) variables are calculated for each word in training time in order to adjust the model parameters \( \lambda^* \), which giving as:

\[ \xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \hat{\beta}_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \hat{\beta}_{t+1}(j)} \]  

(5-10)

\[ \gamma_t(i) = \frac{\hat{\alpha}_t(i) \hat{\beta}_t(i)}{\sum_{j=1}^{N} \hat{\alpha}_t(i) \hat{\beta}_t(i)} \]  

(5-11)

The average estimation of state transition probability is giving as:

\[ a_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \]  

(5-12)
Re-estimate the mean

A new mean is calculated, and replaced with the old one. This process will help the system to enhance the parameters in order to use in the next iteration of the process, which giving as:

\[ \mu_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) o_t}{\sum_{t=1}^{T} \gamma_t(j,k)} \]  

(5-14)

Re-estimate the covariance

A new covariance is calculated every iteration. The new iteration also be used in the next iteration of the process to enhance the system parameters, which giving as:

\[ \Sigma_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)(o_t - \mu_j)(o_t - \mu_j)^T}{\sum_{t=1}^{T} \gamma_t(j,k)} \]  

(5-14)

Each training time we got a custom model for each word. At the end of training session, the final model will be used further in decoding process.

6.2.4 Decoding

Decoding process is the last step toward recognizing the word. This process is aimed to find the state sequence that most likely to have Produced an observation sequence. Viterbi algorithm considered as a one of the best solution of this issue. This method is used to find the optimal scoring path of state sequence for the given
observation sequence [66], the following MATLAB script code is used for this purpose.

```matlab
start_recognition('testing_list.mat', dim);
```

Unlike the Forward algorithm, Viterbi used the maximization instead of summing over all possible state sequences at the recursion and termination steps. This method keeps track of the arguments that maximize \( \delta_t(i) \) in matrix \( \psi \), which is used to recover the optimal state sequence at the backtracking step, which defined as:

\[
\delta_t(i) = \max(P(q(1), q(2), \ldots, q(t-1); o(1), o(2), \ldots, o(t) | \lambda)
\]  \hspace{1cm} (5-14)

In our research, Alternative Viterbi algorithm, the Lazy Viterbi algorithm, has been used because it’s much faster than the original Viterbi decoder. Although those procedures are the same, Viterbi algorithm includes multiplication with probabilities, and the Alternative Viterbi algorithm does not. The following steps shows the major alternative Viterbi procedure:

1. **Initialization**

   Set \( t = 2 \);

   \[
   \begin{align*}
   \delta_1(i) &= \pi_i + h_i(o_1), \quad 1 \leq i \leq N \\
   \psi_1(i) &= 0, \quad 1 \leq i \leq N
   \end{align*}
   \]

2. **Induction**

   \[
   \begin{align*}
   \delta_t(j) &= h_j(o_t) \max_{i \in N} [\delta_{t-1}(i) + a_{ij}], \quad 1 \leq j \leq N \\
   \psi_t(j) &= \arg \max_{i \in N} [\tilde{\delta}_{t-1}(i) a_{ij}], \quad 1 \leq j \leq N
   \end{align*}
   \]
3. Update time

Set \( t = t + 1 \);

if \( t \leq T \) go to step 2;
else, go to step 4

4. Termination

\[
P^* = \max_{1 \leq i \leq N} [\delta_T (i)]
\]

\[
q^* = \arg \max_{1 \leq i \leq N} [\delta_T (i)]
\]

5. Optimal state sequence backtracking:

a. Initialization

Set \( t = T - 1 \);

b. Backtracking

\[
q_t^* = \psi_{t+1} (q_{t+1}^*)
\]

c. Update time

Set \( t = t - 1 \);

if \( t \geq 1 \) go to step b;
else, go to step 4.

As shown in Figure 6.8, state sequence backtracking found out the optimal state sequence using \( \psi_t \). The backtracking begins in the most likely end state, which moves towards the start of the observations by selecting the state in the \( \psi_t \) (i) that at time t-1 refers to current state.
Testing process is completed in such matter that the utterance is compared with each model, and score is defined for each comparison. With respect to the models, which generated in training session there is one model generated for each word. Consequently, Viterbi trellis computation helps the system to provide the decision and decode the unknown words.

Figure 6.8: Viterbi trellis computation.
7.1 Introduction

This chapter mainly discussed about how to evaluate the performance of the automatic speech recognition system using the proposal conventional and hybrid features. In previous chapters, all the required materials have been illustrated and discussed include; signal processing, feature extraction methods, as well as the underlying assumptions of Hidden Markov Model (HMM) classifier techniques. Based on these background knowledge, all the materials are placed into practice in order to design a small vocabulary isolated word system (ASR), which is introduced in this research.

After designing the required system, the project will proceed forward to find out the most robustness features. To achieve our goal, four popular conventional methods were selected, designed, applied, and tested in order to develop a new hybrid feature algorithms. Four this purpose eleven distinct sets of acoustic signals (numbers: one to nine and the letter O) are used to train and test the ASR system. The performance evaluation of the proposed system is measured by finding the maximum word recognition rate at the recognizer at the back-end part of the system.

The acoustic dataset, which used in our research is an English native speech corpus named TIDIGITS. This speech corpora was originally designed and collected at Texas Instruments, Inc. (TI) in 1982 in a quiet acoustic enclosure, which was recorded using an Electro-Voice RE-16 Dynamic Cardioid microphone.
TIDIGITS are used for the purpose of evaluating the performance of isolated-digit ASR System. It contains of 208 informants (94 men and 114 women), each person producing twenty two digit sequences, where data are grouped into two sets, training set and testing set. The sequences can include eleven different digits, starting from "zero" to "nine", plus letter "oh". This data has been sampled at 8-kHz, digitalized with a resolution of 16-bits, and saved in a wave file format. The performance evaluation results and its analysis are presented in section 7.4.

7.2 Toolboxes

ASR system is mainly related to signal processing, classification, and recognition tasks. Therefore, matrix-based MATLAB language, which is a very rich library in scientific computing has been chosen in this research to design the proposal system.

To reach our goal, some useful MATLAB’s library toolboxes were used to help build the system, speed up the process, and for their reliability. The first toolbox that is used is called (Voicebox), which comprised of set of functions that help handle and process the speech signal, extracting the relevant features, and classify these features to generate the related reference models. This toolbox covers many of speech processing tools, such as pre-emphasis, blocking, signal to noise ratio (SNR), Fourier transform (FFT), and other feature extraction tools.

The second toolbox is a “Library for Support Vector Machines” (LIBSVM). This integrated functions are considered as the foundation of vector classification, regression, and distribution estimation. ‘LIBSVM’ toolbox also considered a very helpful library in Machine learning technology because it contains; Gaussian distribution, Baum-Welch, Viterbi decoding algorithm tools, and much more. All of
these Available, allowable, and public DSP system toolboxes are used for design, implement, and validate the proposed ASR system.

## 7.3 System Design

To carry out the goal of this research, ASR system based on feature extraction and HMM techniques were divided into two main modules, and programmed using MATLAB platform.

The first module of the ASR system is front-end subsystem, where the main function is to extract the important information from the speech signal and save them at feature files, while the second module is the back-end subsystem, whose function is to create the reference acoustic models in training phase, and tried to find the unknown input speech given in the testing phase. In training and testing time, under adverse conditions, each module was tested and verified using TIDIDITS speech corpora in which provided a perfect results.

Front-end, which is the first part of the proposed system has been designed and adjusted to extract the features using conventional and hybrid feature extraction methods as shown below:

- **Conventional methods:**
  - Mel-scale Frequency Cepstral Coefficients (MFCC).
  - Linear Predictive Coding Coefficients (LPCC).
  - Perceptual Linear Prediction Coefficients (PLP).
  - Relative Spectral-PLP Coefficients (RASTA-PLP).

- **Hybrid methods:**

On the other hand, Hidden Markov Model (HMM) has been selected to play vital role in back-end part of the system. This statistical classifier is responsible to generate the reference models, and as well to classify and decode the testing data. The entire recognition system has been evaluated includes feature extraction, training, reference models, utterance testing, and result analysis. Figure 7.1 illustrated the proposal isolated word ASR system that is used in our research.

Figure 7.1: Isolated word HMM recognizer block diagram.
After finished up building the system, the main approach here is to identify most reliable and robustness front-end feature extraction algorithms. Therefore, several experimental setups were carried out under clean and different background noise environments. These experiments were conducted using the same dataset corpus (TIDIGITS) in training and testing phase.

### 7.4 Experimental results and Performance Analysis

Having a complete ASR system designed, it is now necessary to test and evaluate the features. The experiments that will be considered in our research are related to how to generate, classify, and matching different types of features in respect to HMM statistical models, and TIDIGIT corpus to find out the most robustness features.

Proposal conventional algorithms include; MFCC, LPCC, PLP, and RASTA-PLP, while the new hybrid methods include; LPR, MLR, MPR, and MLP. Front-end part of the system automatically stores the related results, which is the training and testing feature vector arrays in specific files. Each single saved file must has a specific name and extension includes; (.mfcc, .lpcc, .plp, .rasta, .lpr, .mlr, .mpr, .mlp).

After generating the features, back-end part reads training files respectively in order to learn the system. During training time, five through twelve sequence states have been selected to find out the proper system model size. Each state applies twelve Gaussian distribution components to best clustering the data.

The system’s states number should correspond roughly to the average number of distinct sounds phonemes in each word, and as mentioned in chapter 6.0, the model parameters λ were estimated and initialized during the first iteration to produce the following:
• *Initial state probabilities distribution vector* $\pi$.
• *Transition probability matrix* $A$
• *Emission probabilities matrix* $B$.

In the first iteration, using forward-backward algorithm, the problem of estimating the parameter values of hidden Markov word models is addressed by finding the maximum-likelihood estimation of the parameters. Giving $N$ number of an observation sequences of the word, the training of the model $\lambda$ has been generated by adjusting the parameters in the next iterations to best represent the word. The adjustment is an estimation of the parameters for the model $\lambda$ that maximizes $P(O|\lambda)$, which is the solutions of the first and third HMM problems [36]. During the first iteration, all the model parameters $\lambda$ ($\pi$, $A$, and $B$) are initiated by fixed values in order to start the system.

Using the iterative procedure (Baum-Welch algorithm), the model parameters $\lambda$ were re-estimated several times inside the system to maximize the values of the paths. This process repeated many times hoping to converge an optimal values for the model parameters $\mu$. So, the system has been trained and modeled.

At the end of the training time, eleven models ($\lambda_1$ throw $\lambda_{11}$) have been generated, where each one represents to the statistical model of the specific digit in the system. As a result, eleven models were obtained using several feature extraction algorithms, which considered by varying the feature extractor method under HMM configuration parameters as shown in the following MATLAB script codes:

```matlab
model_no=11; % No. of models
dim=39; % Observation vectors
no_iter=5; % re-estimation
state_no=6; % Number of states
num_mixture=4; % Mixture model
```
The following Table 7.1, shows the summary of the experiment configurations and conditions in detail;

<table>
<thead>
<tr>
<th>Type of the system</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional front-ends</td>
<td>MFCC, LPCC, PLP, and RASTA-PLP.</td>
</tr>
<tr>
<td>Hybrid front-ends</td>
<td>LPR, MLR, MPR, and MLP.</td>
</tr>
<tr>
<td>Conventional front-ends Parameter coefficients</td>
<td>39 parameter coefficients, 13 [12-static + 1-power] + 13-delta + 13-delta-delta</td>
</tr>
<tr>
<td>Hybrid front-ends parameter coefficients</td>
<td>39 parameter coefficients, 13 static coefficients from each three conventional</td>
</tr>
<tr>
<td>Back-end</td>
<td>HMM.</td>
</tr>
<tr>
<td>Number of speakers</td>
<td>208 adult speakers, [38 males, 57 females] training and [56 males, 57 females] testing.</td>
</tr>
<tr>
<td>Training set &amp; Test set</td>
<td>Eleven numeric digits [0-9 and letter O], Each digit is spoken two times.</td>
</tr>
<tr>
<td>Number of states</td>
<td>5 through 12.</td>
</tr>
<tr>
<td>Gaussian mixture models</td>
<td>2 through 12.</td>
</tr>
<tr>
<td>Noise type</td>
<td>White noise.</td>
</tr>
<tr>
<td>SNR range</td>
<td>5dB, 10dB, 20dB, 30dB, clean.</td>
</tr>
<tr>
<td>Platform</td>
<td>PC.</td>
</tr>
<tr>
<td>Programming Language</td>
<td>MATLAB.</td>
</tr>
</tbody>
</table>

During decoding session, Viterbi algorithm computes recursively the probabilities of the most probable path. The resultant probabilities help the system
to find the state sequence that is most likely to have generated that observation sequence. In order to find out the recognition results and choose the winner digit, HMM reads saved testing feature files to compare them with the available statistical reference models, as shown in the next script MATLAB codes:

```matlab
generate_testing_list;
start_initializing('training_list.mat',dim,Model_mat);
for loop=1:iter_num;
    start_training('training_list.mat',dim);
    start_recognition('testing_list.mat',dim);
end
```

At the end of testing time, the recognition performance of the proposed front-end methods in percentage has been determined and recorded. This will have been done by comparing the output training models with the actual transcriptions of the test utterances and find the number of correct recognized utterances, which is given in the following equation:

\[
C = \frac{H}{N} \times 100\% \tag{5-14}
\]

Where,

- **C**: Percentage of correctly recognized utterances.
- **H**: Number of correctly recognized utterances.
- **N**: Total number of utterances of all test.
7.4.1 Experimental Results using Conventional Feature Extraction Algorithms

In order to investigate the performance of the proposal conventional front-end algorithms MFCC, LPC, PLP, and RASTA-PLP, several experimental setups in both clean and noisy environments were conducted using speaker-independent isolated word speech recognition system and TIDIGITS database. The purpose of this work is to compare the result with hybrid based feature extraction and attempt to find out the advantages and the drawbacks of them.

Speech signal is sampled at 8-KHz sampling rate, digitalized with a resolution of 16-bit. The speech signal is divided into frames of 200 samples (corresponding to 25ms) with an overlap of 120 samples (roughly corresponding to 60% overlapping) from frame to frame. In addition to that, Hamming window is applied to each frame in order to keep the continuity of the first and the last points in the frame, and of which 0.97 were as the pre-emphasis coefficients to flatten the spectrum.

To map the sound data from time domain to frequency domain, FFT instead of DFT is used to produce a 256-point FFT, and then 4 conventional methods have been applied individually in order to produce the parameter coefficients. As a result, twelve coefficients and the log-energy plus the first and second derivatives were generated, where each vector is composed of 39-real data. Error! Reference source not found., shows the common and the major configurations that used in these experiments:

In order to generate the feature, front-end converted the speech signal into parameterized sequence of feature vectors. This process is aimed to emphasize the characteristics of spoken words and suppress other irrelevant information. To
accomplish this work, 2,090 training files (95-speakers multiply by 22-words) and 2,486 testing files (113-speakers multiply by 22-words) were considered using different conventional feature extraction algorithms include; MFCC, LPCC, PLP, and RASTA-PLP. On the other hand, back-end classified the feature vectors by divide the data into five to twelve states in which 12-mixture Gaussians are used in each state. Using Baum-welch algorithm, eleven statistical models were generated, one model for each digit. So, the overall resulting data are collected and recorded.

Table 7.2: Common and major configurations of conventional methods experiments.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>indir='wav';</td>
<td>Wave folder</td>
</tr>
<tr>
<td>in_filter='.[Ww][Aa][Vv]';</td>
<td>Folder filter</td>
</tr>
<tr>
<td>frame_size_sec = 0.025;</td>
<td>Frame size 0.025 sec</td>
</tr>
<tr>
<td>frame_shift_sec= 0.010;</td>
<td>Frame shift 0.010 sec</td>
</tr>
<tr>
<td>pre_emp=0.975;</td>
<td>Pre-emphasis filter</td>
</tr>
<tr>
<td>bank_no=26;</td>
<td>Mel-spectrum</td>
</tr>
<tr>
<td>cep_order=12;</td>
<td>No. of static coefficients</td>
</tr>
<tr>
<td>outdir='mfcc_out';</td>
<td>MFCC output folder</td>
</tr>
<tr>
<td>out_extension ='.mfc';</td>
<td>MFCC File extension</td>
</tr>
<tr>
<td>outdir='lpc_out';</td>
<td>LPC Output folder</td>
</tr>
<tr>
<td>out_extension ='.lpc';</td>
<td>LPC File extension</td>
</tr>
<tr>
<td>outdir='plp_out';</td>
<td>PLP output folder</td>
</tr>
<tr>
<td>out_extension ='.plp';</td>
<td>PLP File extension</td>
</tr>
<tr>
<td>outdir='rastaplp_out';</td>
<td>RASTA-PLP output folder</td>
</tr>
<tr>
<td>out_extension ='.rplp';</td>
<td>RASTA-PLP file extension</td>
</tr>
</tbody>
</table>

The following tables, (Table 7.3 to Table 7.6) show the average classification confusion matrix results, which obtained on several front-end and back-end experiments under several SNR levels include; 5dB, 10dB, 20dB, 30dB, and clean environments. The resulting tables are followed by related plots, each plot illustrates the results graphically of the same method.
Table 7.3: Recognition rates of MFCC in various state numbers and different SNR levels.

<table>
<thead>
<tr>
<th>number of States</th>
<th>Clean speech</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SNR [dB]</td>
</tr>
<tr>
<td></td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>98.3105</td>
</tr>
<tr>
<td>6</td>
<td>98.3944</td>
</tr>
<tr>
<td>7</td>
<td>98.7651</td>
</tr>
<tr>
<td>8</td>
<td>99.5850</td>
</tr>
<tr>
<td>9</td>
<td>99.6955</td>
</tr>
<tr>
<td>10</td>
<td>99.9657</td>
</tr>
<tr>
<td>11</td>
<td>99.9652</td>
</tr>
<tr>
<td>12</td>
<td>99.9657</td>
</tr>
</tbody>
</table>

Figure 7.2: Recognition classification rates based on MFCC.
Table 7.4: Recognition rates of LPC in various state numbers and different SNR levels.

<table>
<thead>
<tr>
<th>number of States</th>
<th>Clean speech Word accuracy [%]</th>
<th>SNR [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>99.5748</td>
<td>98.1038</td>
</tr>
<tr>
<td>6</td>
<td>99.6941</td>
<td>98.5004</td>
</tr>
<tr>
<td>7</td>
<td>99.6955</td>
<td>98.6745</td>
</tr>
<tr>
<td>8</td>
<td>99.7157</td>
<td>98.9633</td>
</tr>
<tr>
<td>9</td>
<td>99.7958</td>
<td>98.9844</td>
</tr>
<tr>
<td>10</td>
<td>99.9949</td>
<td>98.8947</td>
</tr>
<tr>
<td>11</td>
<td>99.9900</td>
<td>98.5801</td>
</tr>
<tr>
<td>12</td>
<td>99.9934</td>
<td>99.5912</td>
</tr>
</tbody>
</table>

Figure 7.3: Recognition classification rates based on LPC.
Table 7.5: Recognition rates of PLP in various state numbers and different SNR levels.

<table>
<thead>
<tr>
<th>State number</th>
<th>Word accuracy [%]</th>
<th>SNR [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clean speech</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>98.6323</td>
<td>96.8222</td>
</tr>
<tr>
<td>6</td>
<td>98.6423</td>
<td>96.8564</td>
</tr>
<tr>
<td>7</td>
<td>98.9632</td>
<td>96.9453</td>
</tr>
<tr>
<td>8</td>
<td>98.9976</td>
<td>97.2067</td>
</tr>
<tr>
<td>9</td>
<td>99.1280</td>
<td>97.7046</td>
</tr>
<tr>
<td>10</td>
<td>99.9599</td>
<td>98.9994</td>
</tr>
<tr>
<td>11</td>
<td>99.9146</td>
<td>98.5049</td>
</tr>
<tr>
<td>12</td>
<td>99.9455</td>
<td>98.5083</td>
</tr>
</tbody>
</table>

Figure 7.4: Recognition classification rates based on PLP.
Table 7.6: Recognition rates of RASTA-PLP in various state numbers and different SNR levels.

<table>
<thead>
<tr>
<th>State number</th>
<th>Clean speech Word accuracy [%]</th>
<th>SNR [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>98.7060</td>
<td>97.1038</td>
</tr>
<tr>
<td>6</td>
<td>98.7970</td>
<td>97.7043</td>
</tr>
<tr>
<td>7</td>
<td>98.9904</td>
<td>97.9805</td>
</tr>
<tr>
<td>8</td>
<td>99.7000</td>
<td>97.9890</td>
</tr>
<tr>
<td>9</td>
<td>99.7544</td>
<td>97.9902</td>
</tr>
<tr>
<td>10</td>
<td>99.9445</td>
<td>98.9999</td>
</tr>
<tr>
<td>11</td>
<td>99.9245</td>
<td>98.6336</td>
</tr>
<tr>
<td>12</td>
<td>99.7537</td>
<td>98.6604</td>
</tr>
</tbody>
</table>

Figure 7.5: Recognition classification rates based on RASTA-PLP.
### 7.4.1.1 Performance analysis

Among the conventional feature extraction front-ends; MFCC, LPC, PLP, and RASTA-PLP, it is obvious that in clean conditions RASTA-PLP performs a minimum front-end ratio by 99.9045%. However, all conventional algorithms provided almost 100% accuracy in clean environment, as shown in the following confusion Table 7.7.

**Table 7.7: Overall recognition rates of conventional features in various state numbers and different SNR levels.**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Word accuracy [%]</th>
<th>Clean speech SNR [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFCC</td>
<td>99.9657</td>
<td>98.3080</td>
</tr>
<tr>
<td>LPC</td>
<td>99.9949</td>
<td>98.8947</td>
</tr>
<tr>
<td>PLP</td>
<td>99.9599</td>
<td>98.9994</td>
</tr>
<tr>
<td>RASTA-PLP</td>
<td>99.9445</td>
<td>98.9999</td>
</tr>
</tbody>
</table>

From all of the above plots (Figure 7.2 to Figure 7.5), it shows that in clean conditions LPC is the best performing by 99.9949%, then MFCC performs 99.9657%. In addition, PLP performs 99.9599% while RASTA-PLP performs 99.9445%. From the prior results, we conclude that LPC performs the maximum recognition rate by 0.0292%, 0.035%, and 0.0504% better than MFCC, PLP, and RASTA-PLP respectively.

In adverse conditions, the recognition rate is decreased due to the noise ratio, which added to the original testing data. In 30dB SNR, MFCC performs the minimum recognition rate by 98.3080%. Conventional MFCC performs approximately 0.6919%, 0.6914%, and 0.5867% less than RASTA-PLP, PLP, and LPC respectively. In 20db SNR, the recognition rate of LPC performs the minimum rate among other features by 98.5948%. As noted LPC provides approximately
0.1997%, 0.0650%, and 0.9749% less than RASTA-PLP, PLP, and MFCC respectively.

In 10db SNR, the minimum recognition rate occurs when using MFCC feature extraction Algorithm by 93.8798% recognition rate. It performs approximately 0.8874%, 0.1062%, and 0.0185% less than RASTA-PLP, PLP, and LPC respectively. In 5db SNR, MFCC also considered the worst algorithm among the others, which performs 91.3993% recognition rate. It performs 2.5816, 1.6789, and 1.0975 less than RASTA-PLP, LPC, and PLP respectively.

Among the conventional features MFCC, LPC, PLP, and RASTA-PLP. LPC algorithm performs slightly better than the others in clean environments. In noisy conditions RASTA-PLP produces some interesting results, it performs the best recognition rate due to the use of special band-pass filter in the log-spectral domain as shown in Figure 7.6.

![Performance analysis of Conventional feature extraction](image)

**Figure 7.6: Conventional methods recognition classification rates.**
The previous result suggests that it is definitely worthwhile to examine a new feature extraction algorithm in order to enhance the recognition rate. The new hybrid proposal features will obtained by selecting 3 conventional features, and take the first 13 coefficients from each vector in order to produce a new 39 coefficients in one array vector. In the next section (Section 7.4.2), a new set of recognition experiments will be conducted and the performance of each type of new hybrid features will be tested on a formal standardized speech data.

7.4.2 Experimental Results using Hybrid Feature Extraction Algorithms

In order to investigate the performance of different front-ends, another set of experiments were carried out. The front-end methods that will be considered in this setup of experiments are the hybrid feature extraction algorithms, which are explained in detail in Section 5.3. The features were generated by combining all together 39-static coefficients from three selected features in one array vector, 13-conventional feature coefficients are used to perform 39-features in one vector, which listed as; LPR, MLR, MPR, and MLP.

LPR was obtained by placing 39-coefficients as follows: (13-LPC, 13-PLP, and 13-RASTA-PLP) in one vector array. MLR also obtained by placing 39-coefficients of (13-MFCC, 13-LPC, and 13-RASTA-PLP) in one vector array. MPR was obtained by placing a 39-coefficients of (13-MFCC, 13-PLP, and 13-RASTA-PLP) in one vector array, and MLP was obtained by placing 39-coefficients of (13-MFCC, 13-LPC, and 13-PLP) in one vector array.

Several experiments setup have been conducted by using small vocabulary speaker independent isolated word TIDIGITS corpora. This corpus was used to training and testing the system to generate the acoustic models of 11-phonemes. The
experiment is repeatedly conducted several times by adding different white noise levels (30db, 20db, 10db, and 5db). A clean condition is also included, where both training and testing data is not corrupted with any artificial noise as shown in Table 7.8.

Table 7.8: Common and major configurations of hybrid methods experiments

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>indir='wav';</td>
<td>Wave folder</td>
</tr>
<tr>
<td>in_filter='.[Ww][Aa][Vv]';</td>
<td>Folder filter</td>
</tr>
<tr>
<td>frame_size_sec = 0.025;</td>
<td>Frame size 0.025 sec</td>
</tr>
<tr>
<td>frame_shift_sec= 0.010;</td>
<td>Frame shift 0.010 sec</td>
</tr>
<tr>
<td>pre_emp=0.975;</td>
<td>Pre-emphasis filter</td>
</tr>
<tr>
<td>bank_no=26;</td>
<td>Mel-spectrum</td>
</tr>
<tr>
<td>cep_order=12;</td>
<td>No. of static coefficients</td>
</tr>
<tr>
<td>outdir='lpr_out';</td>
<td>LPR output folder</td>
</tr>
<tr>
<td>out_extension ='.lpr';</td>
<td>LPR File extension</td>
</tr>
<tr>
<td>outdir=mlr_out';</td>
<td>MLR Output folder</td>
</tr>
<tr>
<td>out_extension ='.mlr';</td>
<td>MLR File extension</td>
</tr>
<tr>
<td>outdir=mpri_out';</td>
<td>MPR output folder</td>
</tr>
<tr>
<td>out_extension ='.mpr';</td>
<td>MPR File extension</td>
</tr>
<tr>
<td>outdir=mlp_out';</td>
<td>MLP output folder</td>
</tr>
<tr>
<td>out_extension ='.mlp';</td>
<td>MLP file extension</td>
</tr>
</tbody>
</table>
After generating the features, back-end part of the system begins to train, test, and evaluate them using Hidden Markov Model (HMM) classifier. HMM generates 11-reference models by divide the features into 5 to 12-state left-to-right, each state was modeled by the one to twelve-multivariate Gaussian Mixtures Densities. As a result, a transcription of the state sequence for each test utterance is produced, and the resultant transcriptions are compared with the actual states of the utterances.

The following tables, Table 7.9 to Table 7.12, show the confusion matrix of the average classification results of hybrid features MLP, MLR, MPR, and LPR. The resulting tables are followed by related plots, each plot illustrates the result graphically of the same method.

Table 7.9: Recognition rates of LPR in various state numbers and different SNR levels using [13-LPC+13-PLP+13-RASTA-PLP] coefficients.

<table>
<thead>
<tr>
<th>State number</th>
<th>Clean speech</th>
<th>SNR [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>98.6669</td>
<td>96.2351</td>
</tr>
<tr>
<td>6</td>
<td>98.6864</td>
<td>96.2534</td>
</tr>
<tr>
<td>7</td>
<td>98.9345</td>
<td>97.5880</td>
</tr>
<tr>
<td>8</td>
<td>99.6998</td>
<td>97.8534</td>
</tr>
<tr>
<td>9</td>
<td>99.7531</td>
<td>98.0554</td>
</tr>
<tr>
<td>10</td>
<td>99.9733</td>
<td>98.6994</td>
</tr>
<tr>
<td>11</td>
<td>99.9202</td>
<td>98.3245</td>
</tr>
<tr>
<td>12</td>
<td>99.9495</td>
<td>98.3251</td>
</tr>
</tbody>
</table>
Figure 7.7: Recognition classification rates of LPR.

Table 7.10: Recognition rates of MLR in various state numbers and different SNR levels using [13-MFCC+13-LPC+13-RASTA-PLP] coefficients.

<table>
<thead>
<tr>
<th>State number</th>
<th>Word accuracy [%]</th>
<th>Clean speech</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SNR [dB]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>98.0109</td>
<td>94.8654</td>
</tr>
<tr>
<td>6</td>
<td>98.6023</td>
<td>95.9004</td>
</tr>
<tr>
<td>7</td>
<td>98.8341</td>
<td>96.9987</td>
</tr>
<tr>
<td>8</td>
<td>98.9987</td>
<td>97.4580</td>
</tr>
<tr>
<td>9</td>
<td>98.9922</td>
<td>97.9745</td>
</tr>
<tr>
<td>10</td>
<td>99.3990</td>
<td>98.4065</td>
</tr>
<tr>
<td>11</td>
<td>99.1110</td>
<td>98.3000</td>
</tr>
<tr>
<td>12</td>
<td>99.1223</td>
<td>98.3065</td>
</tr>
</tbody>
</table>
Figure 7.8: Recognition classification rates used MLR hybrid algorithm.

Table 7.11: Recognition rates of MPR in various state numbers and different SNR levels using [13-MFCC+13-PLP+13-RASTA-PLP] coefficients

<table>
<thead>
<tr>
<th>State number</th>
<th>Word accuracy [%]</th>
<th>Clean speech</th>
<th>SNR [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>97.2311</td>
<td>93.2212</td>
<td>91.6747</td>
</tr>
<tr>
<td>6</td>
<td>98.4566</td>
<td>93.4544</td>
<td>91.9675</td>
</tr>
<tr>
<td>7</td>
<td>98.5833</td>
<td>93.6654</td>
<td>92.3433</td>
</tr>
<tr>
<td>8</td>
<td>98.7954</td>
<td>93.8766</td>
<td>92.9865</td>
</tr>
<tr>
<td>9</td>
<td>98.9999</td>
<td>94.2098</td>
<td>93.6870</td>
</tr>
<tr>
<td>10</td>
<td>99.1764</td>
<td>94.3990</td>
<td>94.1001</td>
</tr>
<tr>
<td>11</td>
<td>99.1299</td>
<td>94.5300</td>
<td>94.0511</td>
</tr>
<tr>
<td>12</td>
<td>99.1301</td>
<td>94.5362</td>
<td>94.0544</td>
</tr>
</tbody>
</table>

99
Figure 7.9: Recognition classification rates used MPR hybrid algorithm.

Table 7.12: Recognition rates of MLP in various state numbers and different SNR levels using [13-MFCC+13-LPC+13-PLP] coefficients

<table>
<thead>
<tr>
<th>State number</th>
<th>Word accuracy [%]</th>
<th>SNR [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clean speech</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>97.7965</td>
<td>95.9278</td>
</tr>
<tr>
<td>6</td>
<td>97.8955</td>
<td>95.9991</td>
</tr>
<tr>
<td>7</td>
<td>97.9345</td>
<td>96.5468</td>
</tr>
<tr>
<td>8</td>
<td>98.0900</td>
<td>96.6766</td>
</tr>
<tr>
<td>9</td>
<td>98.7419</td>
<td>96.8876</td>
</tr>
<tr>
<td>10</td>
<td><strong>98.9978</strong></td>
<td>97.8921</td>
</tr>
<tr>
<td>11</td>
<td>98.9807</td>
<td>97.9111</td>
</tr>
<tr>
<td>12</td>
<td>98.7921</td>
<td>97.9200</td>
</tr>
</tbody>
</table>
7.4.2.1 Performance Analysis

Error! Reference source not found. to Figure 7.10, show that the performance level increases as the value of SNR increases. However, the recognition rate reached approximately highest stability level when state numbers increased at all SNR levels. It has been remarkably verified by the experiments that Automatic Speech System performance can be improved by increasing the number of states to around ten, which are more sufficient to model the linguistic units of phonemes in our dataset and hence the recognition rate decreases as SNR increases as shown in Table 7.13.

From Figure 7.11, it can be shown that LPR has almost the best recognition performance at all experiment results. In clean conditions, all methods provide almost the same recognition rates, Moreover MLP is marginally the minimum front-end by 98.9978%, which performs approximately 0.9755%, 0.4012%, and 0.1786% worse than LPR, MLR,
and, MPR respectively. In noisy conditions 30db SNR, LPR performs the best recognition rate by 98.6994%. It provides approximately 0.2929%, 4.1632%, and 0.7794% better than MLR, MPR, and MLP respectively. In 20dB SNR, MLR performs better than the others by 0.7001-4.6456%. In 10dB SNR, MLP delayed by 4.006%, 7.5717%, and 7.6761% than MPR, MLR, and LPR respectively. In 5dB SNR, LPR performs well by approximately 1.3025%, 6.741%, and 7.5745% better than MLR, MPR, and MLP.

Table 7.13: Recognition rates in various state numbers and different SNR levels of hybrid feature methods.

<table>
<thead>
<tr>
<th>Feature Extraction</th>
<th>Word accuracy [%]</th>
<th>Clean speech</th>
<th>SNR [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>LPR</td>
<td>99.9733</td>
<td>98.6994</td>
<td>97.9999</td>
</tr>
<tr>
<td>MLR</td>
<td>99.3990</td>
<td>98.4065</td>
<td>98.7000</td>
</tr>
<tr>
<td>MPR</td>
<td>99.1764</td>
<td>94.5362</td>
<td>94.0544</td>
</tr>
<tr>
<td>MLP</td>
<td>98.9978</td>
<td>97.9200</td>
<td>94.0565</td>
</tr>
</tbody>
</table>

Figure 7.11: Recognition classification rates of hybrid methods.
Chapter 8
Conclusion and future work

8.1 Summary and conclusions

A typical automatic speech recognition system consists of two main modules, front-end and back-end. The main function of the back-end module is to generate related statistical models and to provide the recognition decisions based on the features extracted by front-end. Hence, feature extraction algorithm plays a leading role in the success of speech recognition technology. Researchers suggested many front-end algorithms in order to improve the accuracy of recognition decision.

In this research, some of commonly used feature extraction algorithms are selected include; Mel-Frequency Cepstral Coefficients (MFCC), Linear Predictive Coding Coefficients (LPCC), Perceptual Linear Predictive (PLP), and Relative Spectral Perceptual Linear Predictive (RASTA-PLP). These conventional methods are designed to be used as bases to develop a set of new hybrid features. The new hybrid features are generated using a new feature extraction algorithm. This algorithm is used to place 39 hybrid coefficients in one vector array and claimed to have better performance than the conventional methods.

Using MATLAB environment, both conventional and hybrid front-end algorithms are designed and tested using TIDIGITS corpora as an input database. TIDIGITS database consists of eleven digits (0 to nine and the letter O) are recorded from 208 adult speakers and corrupted by several background noise levels include; 5dB, 10dB, 20dB, and 30dB. Furthermore, a discrete Hidden Markov Model (HMM) was applied as statistical pronunciation models to identify the feature performance of proposed conventional and hybrid methods. In order to get the best clustering of
the data and to provide the best recognition rate, HMM states number are changed from 5 to 12 states and the Gaussian distribution components are extended to 12 in each state.

The primary contribution of this research is to develop a new hybrid front-ends features are successfully verified. The performance of the new hybrid feature extraction algorithms was examined and tested against the conventional front-ends. All the experiments are conducted in clean and in the presence of additive background noise, where the recognition results of whole ASR system was investigated as shown in the next Table 8.1.

Table 8.1: Overall recognition rates in various state numbers and different SNR levels of conventional and hybrid feature methods.

<table>
<thead>
<tr>
<th>Feature Extraction</th>
<th>Word accuracy [%]</th>
<th>SNR [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFCC</td>
<td>99.9657</td>
<td>98.3080</td>
</tr>
<tr>
<td>LPC</td>
<td>99.9949</td>
<td>98.8947</td>
</tr>
<tr>
<td>PLP</td>
<td>99.9599</td>
<td>98.9994</td>
</tr>
<tr>
<td>RASTA-PLP</td>
<td>99.9445</td>
<td>98.9999</td>
</tr>
<tr>
<td>LPR</td>
<td>99.9733</td>
<td>98.6994</td>
</tr>
<tr>
<td>MLR</td>
<td>99.3990</td>
<td>98.4065</td>
</tr>
<tr>
<td>MPR</td>
<td>99.1764</td>
<td>94.5362</td>
</tr>
<tr>
<td>MLP</td>
<td>98.9978</td>
<td>97.9200</td>
</tr>
</tbody>
</table>

Result in Figure 8.1, shows that hybrid feature extraction algorithms produced some interesting results. In clean environment, LPR, which considered the best hybrid features gives a better performance than MFCC, PLP and RASTA-PLP but worse than LPC. However, in noisy environments, LPR noticeably outperforms all
three conventional front-ends MFCC, LPC, and PLP. The recognition correctness of LPR is about 0.2996% less than RASTA-PLP, which considered the best among all conventional methods.

Figure 8.1: Overall recognition classification rates of conventional and hybrid methods.

MLR performs poorly in clean environment. Nevertheless, it shows its advantage over other front-ends in adverse environments. It outperforms MFCC conventional front-end in all the noisy situations (5-30dB SNR). It is even better than LPC in highly noisy conditions (20db SNR).
In this research work, we have designed a speech recognition system with a fair degree of accuracy. The system is designed to find the performance evaluation of some conventional and hybrid feature extraction algorithms by Using Hidden Markov model HMM classifier and TIDIGITS corpora.

The main emphasis was on isolated word speech recognition system ASR, which finds applications in industry and man-machine interfaces. However, there is scope for further research in this field and some of the future prospects are listed below.

- Add more speakers (male and female at different ages) into training data in order to improve the statistical model.
- Use another speech language such as Arabic, Spanish, or French.
- More researches on hybrid methods techniques and their effect on the classification for more robust results.
- Use larger vocabulary and continuous speech instead of isolated speech.
Appendix A. MFCC feature extraction MATLAB Program

1. MFCC Main program

```matlab
clear all;
clc;
indir='wav';  % wave folder
in_filter='\.[Ww][Aa][Vv]';  % folder filter
outdir='mfcc_out';  % output folder
out_ext='.mfc';  % MFCC file extension
frame_size_sec = 0.025;  % frame size
frame_shift_sec= 0.010;  % frame shift
bank_no=26;  % Mel-spectrum [20 - 40]
cep_order=12;  % MFCC coefficients
lifter=22;  % lifter to final cepstral coefficients
pre_emp=0.975;  % pre-emphasis filter
File_prepare(indir,in_filter,outdir,out_ext,frame_size_sec,...
    frame_shift_sec,bank_no,cep_order,lifter,pre_emp);
```

2. MFCC File prepare

```matlab
function File_prepare(indir,in_filter,outdir,out_ext,...
    frame_size_sec,frame_shift_sec,bank_no,cep_order,...
    lifter,pre_emp)
    % Prepare I/O folder %
    if indir(end)=='/'||indir(end)=='\\'end=indir(1:(end-1));
    end
    if outdir(end)=='/'||outdir(end)=='\\'endir=outdir(1:(end-1));
    end
    if exist(outdir) ~=7
        mkdir(outdir);
    end
    filelist=dir(indir);
    filelist_len=length(filelist);

    % read all wav files
    for k=3:filelist_len
        [pathstr,filenamek] = fileparts(filelist(k).name);
        if filelist(k).isdir  % if not folder! return to main function
```
File_prepare([indir filesep filenamek], in_filter, [outdir Filesep filenamek], out_ext, frame_size_sec, frame_shift_sec, bank_no,...
cep_order, lifter, pre_emp);
else
    if regexp(filelist(k).name, in_filter)
        %if file then build path
        infilename=fullfile(indir, filelist(k).name);
        outfilename=[outdir filesep filenamek out_ext];
        [speech_raw, fs]=audioread(infilename, 'native'); %Read wav file
        speech_raw=double(speech_raw);
        % MFCC Calculation
        feature_vector=wav2mfcc(speech_raw, fs, frame_size_sec, frame_shift_sec, bank_no, cep_order, lifter, pre_emp);
        [dim, frame_no]=size(feature_vector); %Save features 12B Header
        % Save MFCC feature vectors 
        fout=fopen(outfilename, 'w', 'b'); %'w'==Delete the contents of an existing file or create a new file, and open it for writing, 'b'=='big endian
        fwrite(fout, frame_no, 'int32'); %save frame
        sampPeriod=round(frame_shift_sec*1E7); % Time between each parameter vector, meaning how many parameter in one sec
        fwrite(fout, sampPeriod, 'int32'); % save sample period in file
        sampSize=dim*4;
        fwrite(fout, sampSize, 'int16'); % save sampSize in file
        parmKind=838; % parameter kind code: MFCC=6, _E=64, _D=256, _A=512, MFCC_E_D_A=6+64+256+512=838
        fwrite(fout, parmKind, 'int16');
        fwrite(fout, feature_vector, 'float32'); % write data
        fclose(fout);
    end
end

3. MFCC Coefficients calculation

function feature_vector=wav2mfcc(speech_raw, fs, frame_size_sec, frame_shift_sec, bank_no, cep_order, lifter, pre_emp)
    len = length(speech_raw);
    % pre-emphasis
    preemp_speech = Premphasis_calc(speech_raw, pre_emp, len);
    % Framing 
    [frame_size, frame_shift, frame_no]=framing(fs, frame_size_sec, frame_shift_sec, len);
% create hamming window
win = createHammingWindow(frame_size);
% create Mel scale filter bank
[delta_mf,f,mfcc_tran] = createMelFilters(fs,bank_no,cep_order);
% lifter weighting
n=(1:cep_order)';
lifter_weighting=1+(lifter/2)*sin(pi*n/lifter);
% prepare matrices
logpow=-inf*ones(1,frame_no);
mfcc=zeros(cep_order,frame_no);
% FFT size %
NFFT = 2^(ceil(log(frame_size)/log(2)));
% main loop for frames
for fr=1:frame_no
    % grab a frame of speech from original signal
    speech_frame = preemph_speech((fr-1)*frame_shift+1:(fr-1)*frame_shift+frame_size);
    % apply hamming function to current frame
    windowed_speech = applyWindow(speech_frame,win);
    % apply FFT to windowed speech
    PowerSpec = applyFFT(windowed_speech,NFFT);
    % compute mel spectrum
    mel_power=applyMelFilters(bank_no,f,fs,NFFT,PowerSpec,..
        delta_mf);
    % inverse cosine transform
    mfcc(:,fr)=mfcc_tran*log(mel_power);
    % liftering
    mfcc(:,fr)=lifter_weighting.*mfcc(:,fr);
    % compute energy, log Power
    logpow(fr)=log(speech_frame'*speech_frame);
end
% Delta I+II calculation %
delta_win_weight = ones(1,5);
d_fea=(0.01/frame_size_sec)*Delta([mfcc;logpow],delta_win_weight);
% Delta Feat in unit of 10ms/dd_fea=(0.01/frame_size_sec)*Delta(d_fea,delta_win_weight);
% Observation vectors.
feature_vector=[mfcc;logpow;d_fea;dd_fea];
Appendix B. LPC feature extraction MATLAB Program

1. LPC Main program

```matlab
clear all;
clc;
indir='wav'; % wave folder
in_filter='\.\[Ww][Aa][Vv]\'; % folder filter
outdir='lpc_out'; % output folder
out_ext='.lpc'; % LPC file extension
frame_size_sec = 0.025; % frame size
frame_shift_sec = 0.010; % frame shift
cep_order=12; % LPC coefficients
pre_emp=0.975; % pre-emphasis filter
File_prepare(indir,in_filter,outdir,out_ext,frame_size_sec,...
frame_shift_sec,cep_order,pre_emp);
```

2. LPC File prepare

```matlab
Function File_prepare(indir,in_filter,outdir,out_ext,...
frame_size_sec,frame_shift_sec,cep_order,...
lifter,pre_emp)

% Prepare I/O folder %
if indir(end)=='\' || indir(end)=='\'indir=indir(1:(end-1));
end
if outdir(end)=='\' || outdir(end)=='\'outdir=outdir(1:(end-1));
end
if exist(outdir) ~=7
    mkdir(outdir);
end
filelist=dir(indir);
filelist_len=length(filelist);

% read all wav files
for k=3:filelist_len
    [pathstr,filenamek] = fileparts(filelist(k).name);
    if filelist(k).isdir % if not folder! return to main function
        File_prepare([indir\filenamek],in_filter,[outdir Filesep
filenamek],out_ext,frame_size_sec, frame_shift_sec,cep_order,...
cep_order, lifter,pre_emp);
```
else
if regexp(filelist(k).name, in_filter) %if file then build path
infilename=fullfile(indir, filelist(k).name);
outfilename=[outdir filesep filenamek out_ext];
speech_raw=doublesaudioread(infilename, 'native'); %Read wav file
speech_raw=double(speech_raw);
% LPC Calculation
feature_vector=wav2lpc(speech_raw, fs, frame_size_sec, frame_shift_sec, cep_order, pre_emp);
[dim frame_no]=size(feature_vector);
% Save LPC feature vectors
fout=fopen(outfilename, 'w', 'b'); %'w'==Delete the contents of an existing file or create a new file, and open it for writing, 'b'==big endian
fwrite(fout, frame_no, 'int32'); %save frame
sampPeriod=round(frame_shift_sec*1E7); % Time between each parameter vector, meaning how many parameter in one sec
fwrite(fout, sampPeriod, 'int32'); % save sample period in file
sampSize=dim*4;
fwrite(fout, sampSize, 'int16'); % save sampSize in file
parmKind=838; % parameter kind code: MFCC=6, _E=64, _D=256, _A=512, MFCC_E_D_A=6+64+256+512=838
fwrite(fout, parmKind, 'int16');
fwrite(fout, feature_vector, 'float32'); % write data
fclose(fout);
end
end
end

3. LPC Coefficients calculation

function feature_vector=wav2lpc(speech_raw, fs, frame_size_sec, ...
frame_shift_sec, cep_order, pre_emp)
len = length(speech_raw);
% pre-emphasis
preemp_speech = Premphasis_calc(speech_raw, pre_emp, len);
% Framing
[frame_size, frame_shift, frame_no]=framing(fs, frame_size_sec, frame_shift_sec, len);
% create hamming window
win = createHammingWindow(frame_size);
% lifter weighting
a = zeros(cep_order, 1);
gamma = zeros(cep_order, 1); % lpc Cepstral coefficients.
lpc = zeros(cep_order,frame_no); % Weighted cepstral sequence for frame t.
win_gamma = 1 + (cep_order/2)*sin(pi/cep_order*(1:cep_order)');
% Cepstral window function.
% prepare matrices
logpow=-inf*ones(1,frame_no);
% main loop for each frame
for fr = 1:frame_no
    % grab a frame of speech from original signal
    speech_frame = preemp_speech((fr-1)*frame_shift+1:(fr-1)*frame_shift+frame_size);
    % apply hamming function to current frame
    windowed_speech = applyWindow(speech_frame,win);
    % Short-term autocorrelation.
    [rs,eta] = xcorr(windowed_speech,cep_order,'biased');
    % LP analysis based on Levinson-Durbin recursion.
    [a(1:cep_order),lag]=durbin(rs(cep_order+1:2*cep_order+1),cep_order);
    % Cepstral coefficients.
    gamma(1) = a(1);
    for i = 2:cep_order
        gamma(i) = a(i) + (1:i-1)*(gamma(1:i-1).*a(i-1:-1:1))/i;
    end
    % Weighted cepstral sequence for frame t
    lpc(:,fr) = gamma.*win_gamma;
    % compute energy, log Power
    logpow(fr)=log(speech_frame'*speech_frame);
end
% Delta I+II calculation
delta_weight = ones(1,5);
d_fea=(0.01/frame_shift_sec)*Delta([lpc;logpow],delta_weight); % Delta Feature in unit of 10ms
dd_fea=(0.01/frame_shift_sec)*Delta(d_fea,delta_weight);
% Observation vectors.
feature_vector = [lpc;logpow;d_fea;dd_fea];
Appendix C. PLP feature extraction MATLAB Program

1. PLP Main program

```matlab
clear all;
clc;
indir='wav'; % wave folder
in_filter='\.[Ww][Aa][Vv]'; % folder filter
outdir='plp_out'; % output folder
out_ext='.plp'; % PLP file extension
frame_size_sec = 0.025; % frame size
frame_shift_sec = 0.010; % frame shift
cep_order=12; % PLP coefficients
pre_emp=0.975; % pre-emphasis filter
File_prepare(indir,in_filter,outdir,out_ext,frame_size_sec,...
frame_shift_sec,pre_emp,cep_order);
```

2. PLP File prepare

```matlab
Function File_prepare(indir,in_filter,outdir,out_ext,...
frame_size_sec,frame_shift_sec,bank_no,cep_order,...
lifter,pre_emp)

% Prepare I/O folder %
if indir(end)=='/\'|\indir(end)=='\\indir=indir(1:(end-1));
end
if outdir(end)=='/\'|\outdir(end)=='\\outdir=outdir(1:(end-1));
end
if exist(outdir) ~=7
    mkdir(outdir);
end
filelist=dir(indir);
filelist_len=length(filelist);

% read all wav files
for k=3:filelist_len
    [pathstr,filenamek] = fileparts(filelist(k).name);
    if filelist(k).isdir % if not folder! return to main function
        File_prepare([indir filesep filenamek],in_filter,[outdir Filesep
filenamek],out_ext,frame_size_sec,frame_shift_sec,bank_no,...
```
cep_order, lifter, pre_emp);
else
  if regexp(filelist(k).name, in_filter)
    infilename=fullfile(indir, filelist(k).name);
    outfilename=[outdir filesep filenamek out_ext];
    [speech_raw, fs]=audioread(infilename, 'native'); % Read wav file
    speech_raw = double(speech_raw);
    % PLP Calculation
    feature_vector = wav2plp(speech_raw, fs, frame_size_sec, 
                           frame_shift_sec, cep_order, pre_emp);
    [dim frame_no] = size(feature_vector); % Save features 12B Header
    % Save PLP feature vectors
    fout = fopen(outfilename, 'w', 'b'); % 'w' == Delete the contents of
    % an existing file or create a new file, and open it for
    % writing, 'b' == big endian
    fwrite(fout, frame_no, 'int32'); % save frame
    sampPeriod = round(frame_shift_sec*1E7); % Time between each
    % parameter vector, meaning how many parameter in one sec
    fwrite(fout, sampPeriod, 'int32'); % save sample period in file
    sampSize = dim*4;
    fwrite(fout, sampSize, 'int16'); % save sampSize in file
    parmKind = 838; % parameter kind code: MFCC=6, _E=64, _D=256,
    _A=512, MFCC_E_D_A=6+64+256+512=838
    fwrite(fout, parmKind, 'int16');
    fwrite(fout, feature_vector, 'float32'); % write data
  end
end
end

3. PLP Coefficients calculation

function [feature_vector] = wav2plp(speech_raw, fs, frame_size_sec, 
                                    frame_shift_sec, cep_order, pre_emp)

  len = length(speech_raw);
  % pre-emphasis
  preemp_speech = Premphasis_calc(speech_raw, pre_emp, len);
  % Framing %
  [frame_size, frame_shift, frame_no] = framming(fs, frame_size_sec, frame_shift_sec, len);
  % Hanning window
  win = hanning(frame_size);
  % FFT size %
NFFT = 2^(ceil(log(frame_size)/log(2)));  
% prepare matrices
logpow=-inf*ones(1,frame_no);
% main loop for each frame
for fr = 1:frame_no
    % compute power spectrum
    pspectrum = powspec(preemp_speech,fs,frame_size,frame_shift, win,NFFT);
    % next group to critical bands
    [aspectrum wts] = audspec(psppectrum, fs);
    % do final auditory compressions
    postspectrum = postaud(aspectrum, fs/2);
    % LPC analysis
    lpcas = dolpc(postspectrum, cep_order);
    % convert lpc to ceptra
    Plp_cep = lpc2cep(lpcas, cep_order);
    % apply lifter
    Plp_cep = lifter(plp_cep);
    % compute energy, log Power
    speech_frame = preemp_speech((fr-1)*frame_shift+1:(fr-1)*frame_shift+frame_size);
    logpow(fr)=log(speech_frame'*speech_frame);
end

% Delta I+II calculation %
delta_win_weight = ones(1,5);
% Delta Feat in unit of 10ms
d_fea=(0.01/frame_size_sec)*Delta([plp_cep;logpow],delta_win_weight);
dd_fea=(0.01/frame_size_sec)*Delta(d_fea,delta_win_weight);
% Observation vectors.
feature_vector=[plp_cep;logpow;d_fea;dd_fea];
Appendix D. RASTA-PLP feature extraction MATLAB Program

1. RASTA-PLP Main program

```matlab
clear all;
clsc;
indir='wav';                 % wave folder
in_filter='\.\.[Ww][Aa][Vv]'; % folder filter
outdir='rastaplp_out';       % output folder
out_ext='.rplp';             % RASTA-PLP file extension
frame_size_sec = 0.025;      % frame size
frame_shift_sec = 0.010;     % frame shift
cep_order=12;               % RASTA-PLP coefficients

File_prepare(indir,in_filter,outdir,out_ext,frame_size_sec,...
             frame_shift_sec,pre_emp,cep_order);
```

2. RASTA-PLP File prepare

```matlab
Function File_prepare(indir,in_filter,outdir,out_ext,...
                       frame_size_sec,frame_shift_sec,pre_emp,modelorder)

% Prepare I/O folder
if     indir(end)=='/' || indir(end)=='\'; indir=indir(1:(end-1));
end
if     outdir(end)=='/' || outdir(end)=='\'; outdir=outdir(1:(end-1));
end
if     exist(outdir) ~=7
    mkdir(outdir);
end
filelist=dir(indir);
filelist_len=length(filelist);

% read all wav files
for k=3:filelist_len
    [pathstr, filenamek] = fileparts(filelist(k).name);
    if filelist(k).isdir % if not folder! return to main function
        File_prepare([indir filesep filenamek],in_filter,[outdir Filesep
                    filenamek],out_ext,frame_size_sec, frame_shift_sec, bank_no,...
                    cep_order, lifter,pre_emp);
    else
        if regexp(filelist(k).name,in_filter) % if file then build path
            % Further code...
        end
    end
end
```

116
infilename=fullfile(indir, filelist(k).name);
outfilename=[outdir filesep filenamek out_ext];
[speech_raw,fs]=audioread(infilename, 'native');%Read wav file
speech_raw=double(speech_raw);

% RASTA-PLP Calculation
feature_vector=wav2rastapl(p(speech_raw,fs,frame_size_sec,
frame_shift_sec,cep_order,pre_emp);
[dim frame_no]=size(feature_vector);%Save features 12B Header
% Save rasta-PLP feature vectors %
fout=fopen(outfilename, 'w', 'b');%'w'==Delete the contents of
an existing file or create a new file, and open it for
writing,'b'==big endian
fwrite(fout, frame_no, 'int32'); %save frame
sampPeriod=round(frame_shift_sec*1E7); % Time between each
parameter vector, meaning how many parameter in one sec
fwrite(fout, sampPeriod, 'int32'); % save sample period in file
sampSize=dim*4;
fwrite(fout, sampSize, 'int16'); % save sampSize in file
parmKind=838;% parameter kind code: MFCC=6, _E=64, _D=256,
_A=512, MFCC_E_D_A=6+64+256+512=838
fwrite(fout, parmKind, 'int16');
fwrite(fout, feature_vector, 'float32');% write data
fclose(fout);
end
end

3. RASTA-PLP Coefficients calculation

function [feature_vector]=wav2plp(speech_raw,fs, frame_size_sec,
frame_Shift_sec, cep_order, pre_emp)

len = length(speech_raw);
% pre-emphasis
preemp_speech = Premphasis_calc(speech_raw,pre_emp,len);
% Framming %
[frame_size,frame_shift,frame_no]=framing(fs,frame_size_sec,frame_
shift_sec,len);
% hanning window
win = hanning(frame_size)';
% FFT size
NFFT = 2^ceil(log(frame_size)/log(2));
% prepare matrices
logpow=-inf*ones(1,frame_no);
% main loop for each frame
for fr = 1:frame_no
    % compute power spectrum
    pspectrum = powspec(preemp_speech,fs,frame_size,frame_shift,
        win,NFFT);
    % next group to critical bands
    [aspectrum wts] = audspec(psppectrum, fs);
    % put in log domain
    nl_aspectrum = log(aspectrum);
    % next do rasta filtering
    ras_nl_aspectrum = rastafilt(nl_aspectrum);
    % do inverse log
    aspectrum = exp(ras_nl_aspectrum);
    % do final auditory compressions
    postspectrum = postaud(aspectrum, fs/2);
    % LPC analysis
    lpcas = dolpc(postspectrum, cep_order);
    % convert lpc to cepstra
    rasta_cep = lpc2cep(lpcas, cep_order);
    % apply lifter
    rasta_cep = lifter(rasta_cep);
    % compute energy, log Power
    speech_frame = preemp_speech((fr-1)*frame_shift+1:(fr-1)*frame_shift+frame_size);
    logpow(fr)=log(speech_frame'*speech_frame);
end

% Delta I+II calculation %
delta_win_weight = ones(1,5);
% Delta Feat in unit of 10ms
d_fea=(0.01/frame_size_sec)*Delta([rasta_cep;logpow],delta_win_weight);
dd_fea=(0.01/frame_size_sec)*Delta(d_fea,delta_win_weight);
% Observation vectors.
feature_vector=[rasta_cep;logpow;d_fea;dd_fea];
Appendix E. Hybrid feature extraction LPR MATLAB Program

1. LPR Main program

```matlab
clear all;
clc;
indir='wav';
in_filter='.[Ww][Aa][Vv]';
outdir='lpr_out';
out_ext='.lpr';
frame_size_sec = 0.025;
frame_shift_sec= 0.010;
bank_no=26;
cep_order=13;
lifter=22;
pre_emp=0.975;
File_prepare(indir,in_filter,outdir,out_ext,frame_size_sec,frame_shift_sec, bank_no,cep_order,lifter,pre_emp);
```

2. LPR File prepare

```matlab
Function File_prepare(indir,in_filter,outdir,out_ext,...
                      frame_size_sec,frame_shift_sec,pre_emp,modeorder)
% Prepare I/O folder %
if indir(end)=='/'||indir(end)=='\\'indir=indir(1:(end-1));
end
if outdir(end)=='/'||outdir(end)=='\\'outdir=outdir(1:(end-1));
end
if exist(outdir) ~=7
    mkdir(outdir);
end
filelist=dir(indir);
filelist_len=length(filelist);

% read all wav files
for k=3:filelist_len
    [pathstr,filenamek] = fileparts(filelist(k).name);
    if filelist(k).isdir % if not folder! return to main function
        File_prepare([indir filesepfilenamek],in_filter,[outdir Files epfilenamek],out_ext,frame_size_sec, frame_shift_sec, bank_no,...
                   cep_order, lifter,pre_emp);
```
else
if regexp(filelist(k).name, in_filter) % if file then build path
    infilename=fullfile(indir, filelist(k).name);
    outfilename=[outdir filesep filenamek out_ext];
    [speech_raw, fs] = audioread(infilename, 'native'); % Read wav file
    speech_raw = double(speech_raw);
    % FFT length
    NFFT = 2^rceil(log(round(fs*frame_size_sec))/log(2));
    % LPR Calculation
    lpc_feature_vector = wav2lpc(speech_raw, fs, frame_size_sec, ... 
        frame_shift_sec, cep_order, pre_emp);
    plp_feature_vector = wav2plp(speech_raw, fs, frame_size_sec, ... 
        frame_shift_sec, cep_order, pre_emp, NFFT);
    rplp_feature_vector = wav2rastaplp(speech_raw, fs, ... 
        frame_size_sec, frame_shift_sec, cep_order, pre_emp, NFFT);
    feature_vector = [lpc_feature_vector; plp_feature_vector; ... 
        rplp_feature_vector];
    [dim frame_no] = size(feature_vector);
    % Save LPR feature vectors
    fout = fopen(outfilename, 'w', 'b'); % 'w' == Delete the contents of an existing file or create a new file, and open it for writing, 'b' == big endian
    fwrite(fout, frame_no, 'int32'); % Save frame
    sampPeriod = round(frame_shift_sec*1E7); % Time between each parameter vector, meaning how many parameter in one sec
    fwrite(fout, sampPeriod, 'int32'); % Save sample period in file
    sampSize = dim*4;
    fwrite(fout, sampSize, 'int16'); % Save sampSize in file
    parmKind = 838; % Parameter kind code: MFCC = 6, _E = 64, _D = 256, _A = 512, MFCC _E _D _A = 6 + 64 + 256 + 512 = 838
    fwrite(fout, parmKind, 'int16');
    fwrite(fout, feature_vector, 'float32'); % Write data
end
end
Appendix F. hybrid feature extraction MLR MATLAB Program

1. MLR Main program

```matlab
clear all;
clc;
indir='wav';           % wave folder
in_filter='\.\[Ww]\[Aa]\[Vv]\';  % folder filter
outdir='mlr_out';        % output folder
out_ext='.mlr';          % MLR file extension
frame_size_sec = 0.025;  % frame size
frame_shift_sec = 0.010; % frame shift
bank_no=26;              % Mel-spectrum [20 - 40]
cep_order=13;            % selected coefficients
lifter=22;               % lifter to final cepstral coefficient
pre_emp=0.975;           % pre-emphasis filter
File_prepare(indir,in_filter,outdir,out_ext,frame_size_sec,...
              frame_shift_sec,pre_emp/modelorder);
```

2. MLR File prepare

```matlab
Function
File_prepare(indir,in_filter,outdir,out_ext,frame_size_sec,...
              frame_shift_sec, bank_no,cep_order,lifter,pre_emp)
% Prepare I/O folder %
if indir(end)=='/'||indir(end)=='\'indir=indir(1:(end-1));
end
if outdir(end)=='/'||outdir(end)=='\'outdir=outdir(1:(end-1));
end
if exist(outdir) ~=7
    mkdir(outdir);
end
filelist=dir(indir);
filelist_len=length(filelist);

% read all wav files
for k=3:filelist_len
    [pathstr,filenamek] = fileparts(filelist(k).name);
    if filelist(k).isdir % if not folder! return to main function
        File_prepare((indir filesep filenamek),in_filter,[outdir Filesep
                    filenamek],out_ext,frame_size_sec, frame_shift_sec,bank_no,...
```
cep_order, lifter, pre_emp);
else
    if regexp(filelist(k).name, in_filter)
        infilename = fullfile(indir, filelist(k).name);
        outfilename = [outdir filesep filenamek out_ext];
        [speech_raw, fs] = audioread(infilename, 'native'); % Read wav file
        speech_raw = double(speech_raw);
        % FFT length
        NFFT = 2^(ceil(log(round(fs*frame_size_sec)/log(2))));

        % MLR Calculation
        mfcc_feature_vector = wav2mfcc(speech_raw, fs, frame_size_sec, frame_shift_sec, bank_no, cep_order, lifter, pre_emp);
        lpc_feature_vector = wav2lpc(speech_raw, fs, frame_size_sec, frame_shift_sec, cep_order, pre_emp);
        rplp_feature_vector = wav2rastaplp(speech_raw, fs, frame_size_sec, frame_shift_sec, cep_order, pre_emp, NFFT);
        feature_vector = [MFCC_feature_vector, lpc_feature_vector, ...
                         rplp_feature_vector];
        [dim frame_no] = size(feature_vector);
        % Save MLR feature vectors
        fout = fopen(outfilename, 'w', 'b'); % 'w' == Delete the contents of an existing file or create a new file, and open it for writing, 'b' == big endian
        fwrite(fout, frame_no, 'int32'); % save frame
        sampPeriod = round(frame_shift_sec*1E7); % Time between each parameter vector, meaning how many parameter in one sec
        fwrite(fout, sampPeriod, 'int32'); % save sample period in file
        sampSize = dim*4;
        fwrite(fout, sampSize, 'int16'); % save sampSize in file
        parmKind = 838; % parameter kind code: MFCC=6, _E=64, _D=256, _A=512, MFCC_E_D_A=6+64+256+512=838
        fwrite(fout, parmKind, 'int16');
        fwrite(fout, feature_vector, 'float32'); % write data
        fclose(fout);
    end
end
end
Appendix G. Hybrid feature extraction MPR MATLAB Program

1. MPR Main program

```matlab
clear all;
clc;
indir='wav';  % wave folder
in_filter='\.[Ww][Aa][Vv]';  % folder filter
outdir='mpr_out';  % output folder
out_ext='.mpr';  % MLR file extension
frame_size_sec = 0.025;  % frame size
frame_shift_sec = 0.010;  % frame shift
bank_no=26;  % Mel-spectrum [20 - 40]
cep_order=13;  % selected coefficients
lifter=22;  % lifter to final cepstral coefficient
pre_emp=0.975;  % pre-emphasis filter
File_prepare(indir,in_filter,outdir,out_ext,frame_size_sec,...
    frame_shift_sec,bank_no,cep_order,lifter,pre_emp);
```

2. MPR File prepare

```matlab
Function File_prepare(indir,in_filter,outdir,out_ext,..
    frame_size_sec,...
    frame_shift_sec,bank_no,cep_order,lifter,pre_emp)
% Prepare I/O folder %
if indir(end)=='/' || indir(end)=='\\'indir=indir(1:(end-1));
end
if outdir(end)=='' || outdir(end)=='\\'outdir=outdir(1:(end-1));
end
if exist(outdir) ~=7
    mkdir(outdir);
end
filelist=dir(indir);
filelist_len=length(filelist);
% read all wav files
for k=3:filelist_len
    [pathstr,filenamek] = fileparts(filelist(k).name);
    if filelist(k).isdir % if not folder! return to main function
        File_prepare([indir filesep filenamek],in_filter,[outdir Filesep
            filenamek],out_ext,frame_size_sec, frame_shift_sec,bank_no,...
            cep_order, lifter,pre_emp);
```
else
  if regexp(filelist(k).name, in_filter)
    infilename=fullfile(indir, filelist(k).name);
    outfilename=[outdir filesep filenamek out_ext];
    [speech_raw, fs] = audioread(infilename, 'native'); % Read wav file
    speech_raw = double(speech_raw);
    NFFT = 2^(ceil(log(round(fs*frame_size_sec))/log(2)));
    mfcc_feature_vector = wav2mfcc(speech_raw, fs, frame_size_sec, frame_shift_sec, bank_no, cep_order, lifter, pre_emp);
    plp_feature_vector = wav2plp(speech_raw, fs, frame_size_sec,...
                               frame_shift_sec, cep_order, pre_emp, NFFT);
    rplp_feature_vector = wav2rastaplp(speech_raw, fs,...
                                       frame_size_sec, frame_shift_sec, cep_order, pre_emp, NFFT);
    feature_vector = [mfcc_feature_vector, plp_feature_vector, ...
                      rplp_feature_vector];
    [dim frame_no] = size(feature_vector);
    fout = fopen(outfilename, 'w', 'b'); % 'w' == Delete the contents of an existing file or create a new file, and open it for writing, 'b' == big endian
    fwrite(fout, frame_no, 'int32'); % Save frame
    sampPeriod = round(frame_shift_sec*1E7); % Time between each parameter vector, meaning how many parameter in one sec
    fwrite(fout, sampPeriod, 'int32'); % Save sample period in file
    sampSize = dim*4;
    fwrite(fout, sampSize, 'int16'); % Save sampSize in file
    parmKind = 838; % Parameter kind code: MFCC=6, _E=64, _D=256, _A=512, MFCC_E_D_A=6+64+256+512=838
    fwrite(fout, parmKind, 'int16');
    fwrite(fout, feature_vector, 'float32'); % Write data
    fclose(fout);
  end
end
end
Appendix H. Hybrid feature extraction MLP MATLAB Program

1. MLP Main program

```matlab
clear all;
clc;
indir='wav';        % wave folder
in_filter='\.\[Ww]\[Aa]\[Vv]\';    % folder filter
outdir='mlp_out';   % output folder
out_ext='.lp';      % MLR file extension
frame_size_sec = 0.025; % frame size
frame_shift_sec= 0.010; % frame shift
bank_no=26;         % Mel-spectrum [20 - 40]
cep_order=13;       % selected coefficients
lifter=22;          % lifter to final cepstral coefficient
pre_emp=0.975;      % pre-emphasis filter
File_prepare(indir,in_filter,outdir,out_ext,frame_size_sec,...
              frame_shift_sec,bank_no,cep_order,lifter,pre_emp);
```

2. MLP File prepare

```matlab
Function
File_prepare(indir,in_filter,outdir,out_ext,frame_size_sec,...
              frame_shift_sec,bank_no,cep_order,lifter,pre_emp)

% Prepare I/O folder %
if indir(end)=='/'||indir(end)=='\\'indir=indir(1:(end-1));
end
if outdir(end)=='/'||outdir(end)=='\\'outdir=outdir(1:(end-1));
end
if exist(outdir) ~=7
    mkdir(outdir);
end
filelist=dir(indir);
filelist_len=length(filelist);

% read all wav files
for k=3:filelist_len
    [pathstr,filename] = fileparts(filelist(k).name);
    if filelist(k).isdir % if not folder! return to main function
        File_prepare([indir filesep filename],in_filter,[outdir Filesep filename],out_ext,frame_size_sec, frame_shift_sec,bank_no,...
```
cep_order, lifter, pre_emp);
else
    if regexp(filelist(k).name, in_filter) %if file then build path
        infilename = fullfile(indir, filelist(k).name);
        outfilename = [outdir filesep filenamek out_ext];
        [speech_raw, fs] = audioread(infilename, 'native'); %Read wav file
        speech_raw = double(speech_raw);
        % FFT length
        NFFT = 2^ceil(log(round(fs*frame_size_sec))/log(2));
        % MLP Calculation
        mfcc_feature_vector = wav2mfcc(speech_raw, fs, frame_size_sec, frame_shift_sec, bank_no, cep_order, lifter, pre_emp);
        lpc_feature_vector = wav2lpc(speech_raw, fs, frame_size_sec,...
                                   frame_shift_sec, cep_order, pre_emp);
        plp_feature_vector = wav2plp(speech_raw, fs, frame_size_sec,...
                                   frame_shift_sec, cep_order, pre_emp, NFFT);
        feature_vector = [MFCC_feature_vector, lpc_feature_vector,...
                          plp_feature_vector];
        [dim frame_no] = size(feature_vector);
        % Save MLP feature vectors
        fout = fopen(outfilename, 'w', 'b'); %'w' == Delete the contents of an existing file or create a new file, and open it for writing, 'b' == big endian
        fwrite(fout, frame_no, 'int32'); %save frame
        sampPeriod = round(frame_shift_sec*1E7); % Time between each parameter vector, meaning how many parameter in one sec
        fwrite(fout, sampPeriod, 'int32'); % save sample period in file
        sampSize = dim*4;
        fwrite(fout, sampSize, 'int16'); % save sampSize in file
        parmKind = 838; % parameter kind code: MFCC=6, _E=64, _D=256, _A=512, MFCC_E_D_A=6+64+256+512=838
        fwrite(fout, parmKind, 'int16');
        fwrite(fout, feature_vector, 'float32'); % write data
        fclose(fout);
    end
end
end
Appendix I. Hybrid features calculation MATLAB Program

1. MFCC Calculation

```matlab
function feature_vector=wav2mfcc(speech_raw,fs,frame_size_sec,...
frame_shift_sec,bank_no,cep_order,lifter,pre_emp)

len = length(speech_raw);
% pre-emphasis
preemp_speech = Premphasis_calc(speech_raw,pre_emp,len);
% Framming
[frame_size,frame_shift,frame_no]=framming(fs,frame_size_sec,frame_shift_sec,len);
% create hamming window
win = createHammingWindow(frame_size);
% create Mel scale filter bank
[delta_mf,f,mfcc_tran] = createMelFilters(fs,bank_no,cep_order);
% lifter weighting
n=(1:cep_order)';
lifter_weighting=1+(lifter/2)*sin(pi*n/lifter);
% prepare matrices
Mfcc_cep=zeros(cep_order,frame_no);
% fft size
NFFT = 2^(ceil(log(frame_size)/log(2)));
% main loop for each frame
for fr=1:frame_no
  % grab a frame of speech from original signal
  speech_frame = preemp_speech((fr-1)*frame_shift+1:(fr-1)*frame_shift+frame_size);
  % apply hamming function to current frame
  windowed_speech = applyWindow(speech_frame,win);
  % apply FFT to windowed speech
  PowerSpec = applyFFT(windowed_speech,NFFT);
  % compute mel spectrum
  mel_power = applyMelFilters(bank_no,f,fs,NFFT,PowerSpec,delta_mf);
  % inverse cosine transform
  Mfcc_cep(:,fr)=mfcc_tran*log(mel_power);
  %liftering
  Mfcc_cep(:,fr)=lifter_weighting.*mfcc_cep(:,fr);
end
% Observation vectors.
feature_vector=mfcc_cep;
```
2. PLP Calculation

```matlab
function [feature_vector]=wav2plp(speech_raw,fs,frame_size_sec,...
    frame_shift_sec, cep_order,pre_emp,NFFT)
len = length(speech_raw);
preemp_speech = Premphasis_calc(speech_raw,pre_emp,len);
[frame_size,frame_shift,frame_no]=framming(fs,frame_size_sec,frame_shift_sec,len);
win = hanning(frame_size)';
NFFT = 2^ceil(log(frame_size)/log(2));
% main loop for each frame
for fr = 1:frame_no
    pspectrum = powspec(preemp_speech,fs,frame_size,frame_shift,
        win,NFFT);
    [aspectrum wts] = audspec(pspectrum, fs);
    postpectrum = postaud(aspectrum, fs/2);
    lpcas = dolpc(postspectrum, cep_order);
    Plp_cep = lpc2cep(lpcas, cep_order);
    Plp_cep = lifter(plp_cep);
end
feature_vector=plp_cep;
```

3. LPC calculation

```matlab
function feature_vector=wav2lpc(speech_raw,fs,frame_size_sec,...
    frame_shift_sec, cep_order,pre_emp)
len = length(speech_raw);
preemp_speech = Premphasis_calc(speech_raw,pre_emp,len);
[frame_size,frame_shift,frame_no]=framming(fs,frame_size_sec,frame_shift_sec,len);
win = createHammingWindow(frame_size);
a = zeros(cep_order,1);
```
\[ \gamma = \text{zeros}(\text{cep}_\text{order},1); \]
%Weighted cepstral sequence for frame t.
\[ \text{Lpc}_\text{cep} = \text{zeros}(\text{cep}_\text{order},\text{frame}_\text{no}); \]
% Cepstral window function.
\[ \text{win}_\gamma = 1 + (\text{cep}_\text{order}/2) \times \sin(\pi/\text{cep}_\text{order} \times (1:\text{cep}_\text{order}')); \]
% main loop for each frame
\[ \text{for} \ fr = 1:\text{frame}_\text{no} \]
\[ % \text{grab a frame of speech from original signal} \]
\[ \text{speech}_\text{frame} = \text{preemp}_\text{speech}((fr-1)\times\text{frame}_\text{shift}+1:(fr-1)\times\text{frame}_\text{shift}+\text{frame}_\text{size}); \]
\[ % \text{apply hamming function to current frame} \]
\[ \text{windowed}_\text{speech} = \text{applyWindow}(\text{speech}_\text{frame},\text{win}); \]
% Short-Term autocorrelation.
\[ [\text{rs},\text{eta}] = \text{xcorr}(\text{windowed}_\text{speech},\text{cep}_\text{order},'\text{biased}'); \]
% LP analysis based on Levinson-Durbin recursion.
\[ [a(1:\text{cep}_\text{order}),\text{lag}] = \text{durbin}(\text{rs}(\text{cep}_\text{order}+1:2*\text{cep}_\text{order}+1),\text{cep}_\text{order}); \]
% LP analysis.
\[ % \text{Cepstral coefficients.} \]
\[ \gamma(1) = a(1); \]
\[ \text{for} \ i = 2:\text{cep}_\text{order} \]
\[ \gamma(i) = a(i) + (1:i-1) \times (\gamma(1:i-1) \times a(i-1:-1:1))/i; \]
\[ \text{end} \]
% Weighted cepstral sequence for frame t
\[ \text{Lpc}_\text{cep}(:,fr) = \gamma \times \text{win}_\gamma; \]
\[ \text{end} \]
% Observation vectors.
\[ \text{feature}_\text{vector} = \text{lpc}_\text{cep}; \]

4. RASTA-PLP calculation

\[ \text{function} \ [\text{feature}_\text{vector}] = \text{wav2plp}(\text{speech}_\text{raw},\text{fs}, \text{frame}_\text{size}_\text{sec}, \text{frame}_\text{shift}_\text{sec}, \text{cep}_\text{order},\text{pre_emp}) \]
\[ \text{len} = \text{length}(\text{speech}_\text{raw}); \]
% pre-emphasis
\[ \text{preemp}_\text{speech} = \text{Premphasis}_\text{calc}(\text{speech}_\text{raw},\text{pre_emp},\text{len}); \]
% Framming
\[ [\text{frame}_\text{size},\text{frame}_\text{shift},\text{frame}_\text{no}] = \text{framing}(\text{fs},\text{frame}_\text{size}_\text{sec},\text{frame}_\text{shift}_\text{sec},\text{len}); \]
% hanning window
\[ \text{win} = \text{hanning}(\text{frame}_\text{size}'); \]
% FFT size
\[ \text{NFFT} = 2^\text{ceil}(\log(\text{frame}_\text{size})/\log(2)); \]
% main loop for each frame
\[ \text{for} \ fr = 1:\text{frame}_\text{no} \]
% compute power spectrum

129
pspectrum = powspec(preemp_speech,fs,frame_size,frame_shift,win,NFFT);
% next group to critical bands
[aspectrum wts] = audspec(pspectrum, fs);
% put in log domain
nl_aspectrum = log(aspectrum);
% next do rasta filtering
ras_nl_aspectrum = rastafilt(nl_aspectrum);
% do inverse log
aspectrum = exp(ras_nl_aspectrum);
% do final auditory compressions
postspectrum = postaud(aspectrum, fs/2);
% LPC analysis
lpcas = dolpc(postspectrum, cep_order);
% convert lpc to cepstra
Rasta_cep = lpc2cep(lpcas, cep_order);
% apply lifter
Rasta_cep = lifter(rasta_cep);
end

% Observation vectors.
feature_vector=[rasta_cep];
Appendix J. Hidden Markov Model MATLAB Program

1. Main program

%%% V. Kepuska and Husssien A Elharati HMM program %%%
%%% based on Baum Welch training, Veturbi testing %%%
%%% and Multi Gauassion clusrting %%%
clear all;
clear;
addpath('VOICEBOX');  % add new path
dim=39;                % Number of observation dimensions
loop_num=5;            % Number of Baum-Welch re-estimation
num_state=5;           % Number of state
num_mixture=12;        % Number of Mixture Model in one state

% Number of Mixture Model for state
Model_mat=num_mixture*ones(1,num_state);
generate_training_list;  % gerateing training list
generate_testing_list;  % gerateing testing list
disp('start initializing data ...');
start_initializing('training_list.mat', dim,Model_mat);

for loop=1 :loop_num
    disp('start training data ...');
    start_training('training_list.mat', dim);
    disp('start testing data ...');
    start_recognition('testing_list.mat', dim);
end

2. Baum-Welch Training program

function hmm=baum(hmm, samples)
% Baum-welch training
% inputs:
%   hmm -- hmm model struct
%   samples -- speech sample structure
% output:
%   hmm -- hmm structure after training

mix = hmm.mix;   %gaussian mixture
N=length(mix);   % number of HMM states
K = length(samples);    % number of speech samples
SIZE = size(samples(1).data,2);    % order of speech parameter
% Calculate alpha, beta for multi observation
disp('calculate speech parameters...');

131
for k=1:K
    fprintf('%d',k);
    param(k) = getparam(hmm,samples(k).data);
end

% Reestimate transition probability matrix A : trans
disp('reestimate transition probability matrix A...')
for i = 1:(N-1)
    denom=0;
    for k=1:K
        tmp = param(k).ksai(:,i,:);
        denom=denom+sum(tmp(:) )
    end
    for j=i:i+1
        nom=0;
        for k=1:K
            tmp = param(k).ksai(:,i,j);
            nom=nom + sum(tmp(:) )
        end
        hmm.trans(i,j)=nom/denom;
    end
end
% re-estimate gaussian mixture
disp('reestimate gaussian mixture...')
for l=1:N
    for j=1:hmm.M(l)
        % Calculate mean and variance for each pdf
        nommean = zeros(1,SIZE);
        nomvar  = zeros(1,SIZE);
        denom  = 0;
        for k=1:K
            T = size( samples(k).data,1 );
            for t=1:T
                x = samples(k).data(t,:);
                nommean = nommean + param(k).gamma(t,l,j)*x;
                nomvar  = nomvar+param(k).gamma(t,l,j)*(x-
                    mix(l).mean(j,: ) ).^2;
                denom = denom + param(k).gamma(t,l,j);
            end
        end
        hmm.mix(l).mean(j,:) = nommean/denom;
        hmm.mix(l).var(j,:) = nomvar/denom;
        % Calculate the weights of each pdf
        nom=0;
        denom =0;
        for k=1:K
            ...
tmp=param(k).gamma(:,l,j);
nom=nom+sum(tmp(:));
tmp=param(k).gamma(:,l,:);
denom = denom + sum(tmp(:));
end
hmm.mix(l).weight(j) =nom/denom;
end

3. Viterbi decoding program

function [prob] = viterbi(hmm,O)
% Viterbi recognition and decoding
% INPUTS:
%   hmm -- hmm model struct
%   O -- input observation sequence, T*D
%   T is number of frames, D is order of speech parameter
% %
% OUTPUTS:
%   prob -- output probability
%   q -- state sequence

init = hmm.init;   % initial probability
trans = hmm.trans; % transition probability
mix = hmm.mix;     % gaussian mixture
N = hmm.N;         % number of HMM states
T = size(O,1);
% calculate log(init)
ind1 = find(init > 0);
ind0 = find(init <=0);
init(ind0) = -inf;
init(ind1) = log(init(ind1));
% calculate log(trans)
ind1 = find(trans>0);
ind0 = find(trans<=0);
trans(ind0) = -inf;
trans(ind1) = log(trans(ind1));
% initialization
delta = zeros(T,N);
fai = zeros(T,N);
q = zeros(T,1);
x=O(1,:);
% delta : cumulative probability of t-th time and state N
for i = 1:N
init(i);
delta(1,i) = init(i) + log(mixture(mix(i),x));
end
% t=2:T
for t=2:T
    for j=1:N
        [delta(t,j) fai(t,j)] = max(delta(t-1,:) + trans(:,j)');
        x=O(t,:);
        delta(t,j) = delta(t,j) + log(mixture(mix(j),x));
    end
end

% final probability and state
[prob q(T)] = max(delta(T,:));
% track back the best state sequence
for t=T-1:-1:1
    q(t)=fai(t+1,q(t+1));
end

4. Gaussian Mixture model program

function prob=mixture(mix,x)
%Calculate output probability
% INPUTS:
%  mix  --  gaussian mixture
%  x    --  input vector, SIZE*1
% OUTPUT:
%  prob  --  output probability

prob=0;
for j=1:mix.M
    m=mix.mean(j,:);
    v=diag (mix.var(j,:));
    w=mix.weight(j);
    tmp=w*(mvnpdf(x,m,v) );
    if tmp < -1e-50
        tmp =-1e-50;
    end
    prob=prob + tmp;
end
if prob ==0, prob=realmin; end

134
Appendix K. Front-End Feature Extraction Results
Appendix L. Back-End Training, and Decoding Results
training data 7
    calculate speech parameters...
    reestimate transition probability matrix $A$...
    reestimate gaussian mixture...

training data 8
    calculate speech parameters...
    reestimate transition probability matrix $A$...
    reestimate gaussian mixture...

training data 9
    calculate speech parameters...
    reestimate transition probability matrix $A$...
    reestimate gaussian mixture...

training data 10
    calculate speech parameters...
    reestimate transition probability matrix $A$...
    reestimate gaussian mixture...

training data 11
    calculate speech parameters...
    reestimate transition probability matrix $A$...
    reestimate gaussian mixture...
    start testing data ...

Recognition rate = 99.5809
Appendix M. Publications

Journal Papers:

- Robust Speech Recognition System Using Conventional and Hybrid Features of MFCC, LPCC, PLP, RASTA-PLP and Hidden Markov Model Classifier in Noisy Conditions

- Performance Evaluation of Conventional and Hybrid Feature Extractions Using Multivariate HMM Classifier
    [http://www.ijera.com](http://www.ijera.com)
References


[37] V. Zue, J. Glass, M. Philips, and S. Seneff, "Acoustic segmentation and phonetic classification in the SUMMIT


