2006-03

Recognition of in-ear microphone speech data using multi-layer neural networks

Bulbuller, Gokhan.

Monterey California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

RECOGNITION OF IN-EAR MICROPHONE SPEECH DATA USING MULTI-LAYER NEURAL NETWORKS

by

Gokhan Bulbuller

March 2006

Thesis Advisor:	Monique P. Fargues
Co Advisor:	Ravi Vaidyanathan

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** March 2006	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Recognition of In-Ear Microphone Speech Data Using Multi-Layer Neural Networks			**5. FUNDING NUMBERS**
6. AUTHOR(S) Gokhan Bulbuller			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Speech collected through a microphone placed in front of the mouth has been the primary source of data collection for speech recognition. There are only a few speech recognition studies using speech collected from the human ear canal. In this study, a speech recognition system is presented, specifically an isolated word recognizer which uses speech collected from the external auditory canals of the subjects via an in-ear microphone. Currently, the vocabulary is limited to seven words that can be used as control commands for a wide variety of applications. The speech segmentation task is achieved by using the short-time signal energy parameter and the short-time energy-entropy feature (EEF), and by incorporating some heuristic assumptions. Multi-layer feedforward neural networks with two-layer and three-layer network configurations are selected for the word recognition task and use real cepstrum (RC) and mel-frequency cepstral coefficients (MFCCs) extracted from each segmented utterance as characteristic features for the word recognizer. Results show that the neural network configurations investigated are viable choices for this specific recognition task as the average recognition rates obtained with the MFCCs as input features for the two-layer and three-layer networks are 94.731% and 94.61% respectively on the data investigated. Average recognition rates obtained using the RCs as features on the same network configurations are 86.252% and 86.7% respectively.

14. SUBJECT TERMS Speech Recognition, Isolated Word Recognition, In-Ear Microphone, External Auditory Canal, Multi-Layer Neural Networks, Real Cepstrum, Mel-Frequency Cepstral Coefficients, End Point Detection, Short-Time Energy, Energy-Entropy Feature	**15. NUMBER OF PAGES** 187
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

RECOGNITION OF IN-EAR MICROPHONE SPEECH DATA USING MULTI-LAYER NEURAL NETWORKS

Gokhan BULBULLER
Lieutenant Junior Grade, Turkish Navy
B.S., Turkish Naval Academy, 2000

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2006

Author: Gokhan Bulbuller

Approved by: Monique P. Fargues
Thesis Advisor

Ravi Vaidyanathan
Co-Advisor

Jeffrey B. Knorr
Chairman, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Speech collected through a microphone placed in front of the mouth has been the primary source of data collection for speech recognition. There are only a few speech recognition studies using speech collected from the human ear canal. In this study, a speech recognition system is presented, specifically an isolated word recognizer which uses speech collected from the external auditory canals of the subjects via an in-ear microphone. Currently, the vocabulary is limited to seven words that can be used as control commands for a wide variety of applications. The speech segmentation task is achieved by using the short-time signal energy parameter and the short-time energy-entropy feature (EEF), and by incorporating some heuristic assumptions. Multi-layer feedforward neural networks with two-layer and three-layer network configurations are selected for the word recognition task and use real cepstrum (RC) and mel-frequency cepstral coefficients (MFCCs) extracted from each segmented utterance as characteristic features for the word recognizer. Results show that the neural network configurations investigated are viable choices for this specific recognition task as the average recognition rates obtained with the MFCCs as input features for the two-layer and three-layer networks are 94.731% and 94.61% respectively on the data investigated. Average recognition rates obtained using the RCs as features on the same network configurations are 86.252% and 86.7% respectively.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION....1
A. OBJECTIVE....2
B. RELATED RESEARCH....3
C. THESIS ORGANIZATION....4

II. BACKGROUND....5
A. INTRODUCTION....5
B. EQUIPMENT USED....6
C. DATA DESCRIPTION....8
1. Phonetic Classification of the Data....11
D. SPECTRAL CHARACTERISTICS OF THE DATA....12
E. SUMMARY....18

III. END POINT DETECTION....19
A. END POINT DETECTION....19
B. PROBLEMS ENCOUNTERED WITH END POINT DETECTION....21
C. END POINT DETECTION MEASURES....26
1. Short-Time Energy Measure....26
2. Teager's Energy Algorithm....30
3. Short-Time Zero-Crossing Rate (ZCR)....33
4. Energy-Entropy Feature....36
D. THE END POINT DETECTION ALGORITHM....40
1. Preprocessing stage....40
2. Framing....41
3. Threshold Mechanism....42
4. The End Point Detection Algorithm....43
E. THE EFFECT OF THE FILTER ON END POINT DETECTION....52
F. SUMMARY....56

IV. FEATURE EXTRACTION....57
A. REAL CEPSTRUM....57
B. MEL-FREQUENCY CEPSTRAL COEFFICIENTS....61
C. SUMMARY....66

V. RECOGNITION RESULTS....67
A. INTRODUCTION....67
B. MULTI-LAYER NEURAL NETWORKS....70
C. THE BACKPROPAGATION ALGORITHM....73
D. CONJUGATE GRADIENT ALGORITHM....78
E. LEVENBERG-MARQUARDT ALGORITHM....80
F. IMPLEMENTATION....82
G. RECOGNITION RESULTS....87
1. Network Configurations Considered....87

2. Computational Time Issues....88
3. Results....89
H. SUMMARY....107
VI. CONCLUSIONS AND RECOMMENDATIONS....109
A. CONCLUSIONS....109
B. RECOMMENDATIONS....110
APPENDIX A. IN-EAR MICROPHONE SPECIFICATIONS....111
APPENDIX B. MATLAB SOURCE CODE....113
1. DESCRIPTION....113
a. End Point Detection....113
b. Feature Extraction....114
c. Recognition....114
2. MATLAB CODE LISTING....114
LIST OF REFERENCES....159
INITIAL DISTRIBUTION LIST....163

LIST OF FIGURES

Figure 2.1. The complete set-up used for the recordings (ear microphone, A/D Converter, DAQ Card, and notebook with interface program). 7
Figure 2.2. In-ear microphone used for recordings. 7
Figure 2.3. Speech waveform (top plot) and associated spectrogram (bottom plot) of the word “down” (down_8_18_04_29_05.txt). 13
Figure 2.4. Speech waveform (top plot) and associated spectrogram (bottom plot) of the word “up” (down_8_36_04_29_05.txt). 14
Figure 2.5. Speech waveform (top plot) and associated spectrogram (bottom plot) of the word “left” (left_3_18_04_25_05.txt). 14
Figure 2.6. Speech waveform (top plot) and associated spectrogram (bottom plot) of the word “pan” (pan_13_35_05_04_05.txt). 15
Figure 2.7. Speech waveform (top plot) and associated spectrogram (bottom plot) of the word “kill” (kill_3_20_04_25_05.txt). 15
Figure 2.8. Speech waveform (top plot) and associated spectrogram (bottom plot) of the word “move” (move_8_6_04_29_05.txt). 16
Figure 2.9. Speech waveform (top plot) and associated spectrogram (bottom plot) of the word “right” (right_3_25_04_25_05.txt). 16
Figure 2.10. Speech waveform (top plot) and associated spectrogram (bottom plot) of the word “right” collected through the in-ear microphone placed in front of the mouth (right_outside_13_10_05_04_05.txt). 17
Figure 3.1. Block diagram of a speech recognizer using an explicit end point detector. 21
Figure 3.2. Typical silence waveform (top plot) and related power spectral density (bottom plot) for the recordings with the ear microphone (first 600 *ms* of “left_3_18_04_25_05.txt”). 22
Figure 3.3. Typical silence waveform (top plot) and related power spectral density (bottom plot) for the recordings with the microphone placed in front of the mount (first 400 *ms* of “right_outside_19_17_08_22_05.txt”). 23
Figure 3.4. Example of utterance contaminated by mechanical noises; speech waveform (up_1_12_04_13_05.txt) (top plot), absolute magnitude energy contour (bottom plot). Vertical dotted lines indicate the utterance boundaries. 25
Figure 3.5. Example of speech contaminated by speaker generated artifacts; speech waveform (pan_7_4_04_29_05.txt) (top plot), absolute magnitude energy contour (bottom plot). Vertical dotted lines indicate the utterance boundaries. 25
Figure 3.6. Absolute magnitude energy contour (top plot) and squared energy contour (bottom plot) of the word “left” (left_3_18_04_25_05.txt). Vertical dotted lines indicate the end points of the utterance. 28

Figure 3.7. RMS energy contour (top plot) and logarithmic energy contour (bottom plot) of the word “left” (left_3_18_04_25_05.txt). Vertical dotted lines indicate the end points of the utterance.....29
Figure 3.8. Speech waveform of one of the recordings of the word “left” (left_3_18_04_25_05.txt) (top plot), and Teager’s energy plot of the utterance (bottom plot). Vertical dotted lines indicate the end points of the utterance.....33
Figure 3.9. ZCR plot (bottom plot) of a noise-free utterance (kill_3_20_04_25_05.txt) (top plot). Vertical dotted lines on the plots indicate the actual end points of the utterance.....35
Figure 3.10. ZCR plot (bottom plot) of a noisy utterance (up_1_1_04_13_05.txt) (top plot). Vertical dotted lines on the plots indicate the actual end points of the utterance.....35
Figure 3.11. Waveform of one of the utterances of the word “right” (right_3_25_04_25_05.txt) (top plot), and corresponding entropy curve (bottom plot). Vertical dotted lines indicate the edge points of the utterance.....37
Figure 3.12. Absolute magnitude energy contour (top plot) and energy-entropy curve (bottom plot) of the waveform shown in Figure 3.11 (right_3_25_04_25_05.txt). Vertical dotted lines indicate the end points of the utterance.....39
Figure 3.13. Flowchart for the initial speech start point estimation using the short-time absolute magnitude energy quantity. 48
Figure 3.14. (Continued from Figure 3.13) Flowchart for the initial speech end point estimation using the short-time absolute magnitude energy quantity.....49
Figure 3.15. (Continued from Figures 3.13 and 3.14) Flowchart for the last part of the energy-based end point detection algorithm that addresses the miss and misdetection cases.....50
Figure 3.16. (Continued from Figures 3.13 through 3.15) Flowchart for determining the final speech edge points using the energy-entropy feature (EEF). 51
Figure 3.17. Speech waveform of one of the utterances of the word “pan” (pan_15_14_05_12_05.txt). The vertical dotted lines indicate the actual end points.....53
Figure 3.18. Speech waveform of the IIR filtered utterance (pan_15_14_05_12_05.txt) (top plot), and the corresponding absolute magnitude energy contour (bottom plot). Vertical dotted lines indicate the detected end points. 54
Figure 3.19. Speech waveform of the FIR filtered utterance (pan_15_14_05_12_05.txt) (top plot), and the corresponding absolute magnitude energy contour (bottom plot). Vertical dotted lines indicate the detected endpoints. 55
Figure 4.1. Computation of the real cepstrum using the DTFT (After: [Deller, Proakis, Hansen, 1993]).....59
Figure 4.2. Computation of the real cepstrum using the DFT (After: [Deller, Proakis, Hansen, 1993]).....59

Figure 4.3. Real Cepstrum Coefficients of one of the segmented utterances of the word “left” (left_11_1_05_02_05.txt). Only the first 14 coefficients are plotted. 61
Figure 4.4. The mel scale (From: [Deller, Proakis, Hansen, 1993]). 62
Figure 4.5. Conceptual triangular filters for extracting the MFCCs (From: [Deng, O’Shaughnessy, 2003]).....63
Figure 4.6. The MFCC computation as a block diagram (After: [Zhu, Alwan, 2003]). 65
Figure 4.7. Mel-frequency Cepstral Coefficients (MFCC) of one of the segmented utterances of the word “up” (up_8_1_04_29_05.txt). 15 MFCCs, including the first coefficient, are plotted. 66
Figure 5.1. A simplified model of a biological neuron (From: [Deller, Proakis, Hansen, 1992]).....67
Figure 5.2. Artificial neuron model (After: [Deller, Proakis, Hansen, 1992]).....68
Figure 5.3. Mathematical model of a single artificial neuron with multiple inputs (After: [Hagan, Demuth, Beale, 1996]). 71
Figure 5.4. Feedforward two-layer neural network (After: [Hagan, Demuth, Beale, 1996]).....72
Figure 5.5. Backpropagation of sensitivities in a feedforward two-layer neural network (After: [Duda, Hart, Stork, 2001]). 77
Figure 5.6. Two-layer feedforward neural network architecture implemented; $(150-7)$ configuration.....83
Figure 5.7. Three-layer feedforward neural network architecture implemented; $(60-40-7)$ configuration.....84
Figure 5.8. Average Classification Rates and 95% Confidence Intervals; Testing data; $(50-7)$ network; 14 MFCCs as input features; 80 experiments. 100
Figure 5.9. Average Classification Rates and 95% Confidence Intervals; Testing Set; $(100-7)$ network; 14 MFCCs as input features; 80 experiments. 100
Figure 5.10. Average Classification Rates and 95% Confidence Intervals; Testing Set; $(150-7)$ network; 14 MFCCs as input features; 80 experiments. 101
Figure 5.11. Average Classification Rates and 95% Confidence Intervals; Testing Set; $(150-7)$ network; 14 RCs as input features; 80 experiments. 101
Figure 5.12. Average Classification Rates and 95% Confidence Intervals; Testing Set; $(30-20-7)$ network; 14 MFCCs as input features; 80 experiments. 102
Figure 5.13. Average Classification Rates and 95% Confidence Intervals; Testing Set; $(40-20-7)$ network; 14 MFCCs as input features; 80 experiments. 102
Figure 5.14. Average Classification Rates and 95% Confidence Intervals; Testing Set; $(50-30-7)$ network; 14 MFCCs as input features; 80 experiments.....103
Figure 5.15. Average Classification Rates and 95% Confidence Intervals; Testing Set; $(60-40-7)$ network; 14 MFCCs as input features; 80 experiments. 103
Figure 5.16. Average Classification Rates and 95% Confidence Intervals; Testing Set; $(60-40-7)$ network; 14 RCs as input features; 80 experiments. 104

Figure 5.17. Average Classification Rates and 95% Confidence Intervals; Testing Set; $(40-20-7)$ network; LM method; 14 MFCCs as input features; 80 experiments. 104
Figure 5.18. Speech waveform (top plot) and associated spectrogram (bottom plot) of "kill_noise" (kill_noise_16_35_08_25_05.txt).....107
Figure A.1. Specifications of the in-ear microphone used for recordings (From: www.knowlesacoustics.com, last accessed on February 15, 2006).....111
Figure A.2. (Continued from Figure A.1.) Specifications of the in-ear microphone used for recordings (From: www.knowlesacoustics.com, last accessed on February 15, 2006).....112

LIST OF TABLES

Table 2.1. Quantitative information about the data set used for the speech recognition task. Subject numbers with an asterisk (*) next to them indicate female speakers. 10

Table 2.2. Phonetic classification of the vocabulary words. 12

Table 5.1. Class numbers and target vectors associated with the vocabulary words. 84

Table 5.2. Multi-layer network structures considered in the study. 86

Table 5.3. Average recognition results obtained for the different multi-layer neural network configurations considered in this study; 80 experiments. 88

Table 5.4. Average recognition rates for Training data; $(50-7)$ network configuration; MFCCs as input features; 80 experiments. 90

Table 5.5. Average recognition rates for Testing data; $(50-7)$ network configuration; MFCCs as input features; 80 experiments. 90

Table 5.6. Average recognition rates for the words "*kill_noise*" and "*right_outside*;" $(50-7)$ network configuration; MFCCs as input features; 80 experiments 90

Table 5.7. Average recognition rates for Training data; $(100-7)$ network configuration; MFCCs as input features; 80 experiments. 91

Table 5.8. Average recognition rates for Testing data; $(100-7)$ network configuration; MFCCs as input features; 80 experiments. 91

Table 5.9. Average recognition rates for the words "*kill_noise*" and "*right_outside*;" $(100-7)$ network configuration; MFCCs as input features; 80 experiments. 91

Table 5.10. Average recognition rates for Training data; $(150-7)$ network configuration; MFCCs as input features; 80 experiments. 92

Table 5.11. Average recognition rates for Testing data; $(150-7)$ network configuration; MFCCs as input features; 80 experiments. 92

Table 5.12. Average recognition rates for the words "*kill_noise*" and "*right_outside*;" $(150-7)$ network configuration; MFCCs as input features; 80 experiments. 92

Table 5.13. Average recognition rates for Training data; $(150-7)$ network configuration; RCs as input features; 80 experiments. 93

Table 5.14. Average recognition rates for Testing data; $(150-7)$ network configuration; RCs as input features; 80 experiments. 93

Table 5.15. Average recognition rates for the words "*kill_noise*" and "*right_outside*;" $(150-7)$ network configuration; RCs as input features; 80 experiments 93

Table 5.16. Average recognition rates for Training data; $(30-20-7)$ network configuration; MFCCs as input features; 80 experiments. 94
Table 5.17. Average recognition rates for Testing data; $(30-20-7)$ network configuration; MFCCs as input features; 80 experiments. 94
Table 5.18. Average recognition rates for the words "*kill_noise*" and "*right_outside*;" $(30-20-7)$ network configuration; MFCCs as input features; 80 experiments. 94
Table 5.19. Average recognition rates for Training data; $(40-20-7)$ network configuration; MFCCs as input features; 80 experiments. 95
Table 5.20. Average recognition rates for Testing data; $(40-20-7)$ network configuration; MFCCs as input features; 80 experiments. 95
Table 5.21. Average recognition rates for the words "*kill_noise*" and "*right_outside*;" $(40-20-7)$ network configuration; MFCCs as input features; 80 experiments. 95
Table 5.22. Average recognition rates for Training data; $(50-30-7)$ network configuration; MFCCs as input features; 80 experiments. 96
Table 5.23. Average recognition rates for Testing data; $(50-30-7)$ network configuration; MFCCs as input features; 80 experiments. 96
Table 5.24. Average recognition rates for the words "*kill_noise*" and "*right_outside*;" $(50-30-7)$ network configuration; MFCCs as input features; 80 experiments. 96
Table 5.25. Average recognition rates for Training data; $(60-40-7)$ network configuration; MFCCs as input features; 80 experiments. 97
Table 5.26. Average recognition rates for Testing data; $(60-40-7)$ network configuration; MFCCs as input features; 80 experiments. 97
Table 5.27. Average recognition rates for the words "*kill_noise*" and "*right_outside*;" $(60-40-7)$ network configuration; MFCCs as input feature; 80 experiments. 97
Table 5.28. Average recognition rates for Training data; $(60-40-7)$ network configuration; RCs as input features; 80 experiments. 98
Table 5.29. Average recognition rates for Testing data; $(60-40-7)$ network configuration; RCs as input features; 80 experiments. 98
Table 5.30. Average recognition rates for the words "*kill_noise*" and "*right_outside*;" $(60-40-7)$ network configuration; RCs as input feature; 80 experiments. 98
Table 5.31. Average recognition rates for Training data; $(40-20-7)$ network configuration; LM algorithm; MFCCs as input features; 80 experiments.....99

Table 5.32. Average recognition rates for Testing data; $(40-20-7)$ network configuration; LM algorithm; MFCCs as input features; 80 experiments.....99
Table 5.33. Average recognition rates for the words "*kill_noise*" and "*right_outside;*" $(40-20-7)$ network configuration; LM algorithm; RCs as input features; 80 experiments. 99
Table 5.34. Average Classification Rates and 95% Confidence Intervals for the Testing sets of all configurations considered. 105

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

This thesis is dedicated to my mother, Aysan Bulbuller, without whom I would not be where I am today. Her dedication, love, and support were the keys to my success.

I would like to express my sincere gratitude to my thesis advisor Professor Monique P. Fargues for her invaluable guidance, help, and encouragement throughout this thesis work. I am also thankful to her as the instructor who made the biggest contribution to my current knowledge and by giving me the chance to work with her on this thesis.

I am also grateful to my big sisters Melek and Makbule, and my big brothers Rifki, Burhan, and Sukru for their constant support and encouragement.

I would like to express my love and thanks to our newborn baby, Alara, for the joy and enthusiasm she has brought into our lives.

Last, but surely not the least, I would also like to thank my wife Ozge for her love, understanding, support, endless encouragement, and patience throughout my studies at NPS.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Speech is the most natural way of communication for humans. The aim of speech recognition is to create machines that are capable of processing spoken information, and there have been quite remarkable advances and many successful applications in speech recognition, especially with computing technology advances beginning in the 1980s.

Speech signals collected through a microphone placed in front of the mouth have usually been the primary source of data in speech recognition applications. The problem associated with such collection is that any ambient noise is also picked up via the microphone at the same time. As a result, there has been interest in collecting speech from other locations, which would provide some isolation from surrounding noise. For this purpose, speech collection from the external auditory canal (or the ear canal) via an ear-insert microphone has also been considered. The external auditory canal, when isolated properly with an ear-insert microphone, can provide intelligible speech even in severe noise conditions while intelligibility of the same speech collected through a microphone placed in front of the mouth decreases significantly. However, speech collected from the external auditory canal has not found many applications in speech recognition, except in a few recent studies [Westerlund, Dahl, Claesson, 2002a].

The main objective of this thesis is to implement a basic isolated word recognition system which operates on the utterances collected from the external auditory canals of the speakers via an ear-insert microphone. This study is an extension of an on-going research initiated by Newton [Newton, 2006], who collected the required speech data for the recognition implementations considered here.

The speech data were collected from the external auditory canals of 20 native adult American English speakers, including sixteen males and four females, via an ear-insert microphone. All the recordings were done in an office environment. The vocabulary used in this study consisted of seven words: *up*, *down*, *left*, *right*, *kill*, *move*, and *pan*. In order to investigate the recognizer performance in the presence of noise and changes in microphone placement, the subjects also uttered the word "*kill*" in a simulated

noisy environment via the in-ear microphone (denoted as "*kill_noise*") and the word "*right*" through a microphone placed in front of their mouth (denoted as "*right_outside*"). The total data size used for the study is 8,228 isolated utterances including "*kill_noise*" and "*right_outside.*"

In order to implement the speech recognizer, recorded utterances were first isolated from the silence sections by an end point detection algorithm that follows a bandpass filter employed as the preprocessor. Note that the end point detection task is usually quite difficult in real-life conditions, except for high signal-to-noise ratio environments. However, it is the crucial part of any isolated word recognizer because detecting accurate utterance boundaries significantly reduces recognition errors. Different measures (parameters and features) used so far in literature for end point detectors were investigated, and the end point detection algorithm implemented uses both the short-time absolute magnitude energy parameter and the short-time energy-entropy feature (EEF), as well as by incorporating some heuristic assumptions on the thresholds required for this task.

Next, fourteen real cepstrum (RC) coefficients and fourteen mel-frequency cepstral coefficients (MFCCs) were extracted separately for every segmented utterance and used as input features to a classifier. Feedforward back-propagation neural networks (BPNNs) were selected as the classifier types in this study, and ten different two- and three-layer neural network configurations were considered for word recognition.

Recognition results showed that the performance of the two-layer and three-layer networks increased as the numbers of hidden neurons increased. Best overall average recognition rates were obtained for a $(150-7)$ two-layer network configuration (94.731%) and for a $(60-40-7)$ three-layer network (94.61%) with fourteen MFCCs used as input features. Implementation results also showed that the conjugate gradient algorithm was more accurate and reliable than the Levenberg-Marquardt algorithm for the network complexities and data size considered in this study. Finally, we observed that the classifier performances degraded significantly under severe noise conditions, i.e.,

when tested on "*kill_noise*," showing that some type of noise cancellation scheme should be incorporated into the recognizer preprocessing stage.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Speech is the most natural way of communication for humans. The aim of speech recognition is to create machines that are capable of receiving speech from humans (or some spoken commands) and taking action upon this spoken information [Deller, Proakis, Hansen, 1993]. Although it was once thought to be a straightforward problem, many decades of research has revealed the fact that speech recognition is a rather difficult task to achieve, with several dimensions of difficulty due to the nonstationary nature of speech, the vocabulary size, speaker dependency issues, etc. [Deller, Proakis, Hansen, 1993]. However, there have been quite remarkable advances and many successful applications in speech recognition field, especially with the advances in computing technology beginning in the 1980s.

Until recently, speech signals collected through a microphone placed in front of the mouth have been the primary source of data in speech recognition applications. The problem associated with this type of speech collection is that the ambient noise is also picked up via the microphone at the same time. As a result, there has always been a search for another location on the human body rather than the mouth which could provide for some isolation from environmental noise as the performance of speech recognizers deteriorates under increasing noise levels. For this purpose, speech collection from the external auditory canal (or the ear canal) via an ear-insert microphone has also been considered before [Black, 1957]. The external auditory canal, when isolated properly with an ear-insert microphone, can provide intelligible speech even in severe noise conditions where the intelligibility of speech collected through a microphone placed in front of the mouth decreases significantly. However, speech collected from the external auditory canal has not found many applications in speech recognition, except in a few recent studies [Westerlund, Dahl, Claesson, 2002a].

The Hidden Markov Models (HMMs) have become the primary tool for speech recognition since the 1970s. As the interest in artificial neural networks (ANNs) increased with the reinvention of the backpropagation algorithm for multi-layer neural networks in the 1980s, researchers have started to consider the ANN as an alternative to

the HMM approach in speech recognition due to two broad reasons: speech recognition can basically be viewed as a pattern classification problem, and ANNs can perform complex classification tasks. Particularly, some advantages of the ANN over the HMM made them attractive for speech recognition, advantages such as flexible architecture, highly parallel and regular structure, robustness to the limited training data, ability to accommodate discriminant learning, and no need for assumptions about the statistical distribution of the inputs [Morgan, Bourlard, 1995]. As a result, ANNs have been successfully applied to speech recognition, especially to the phoneme, digit, and isolated word recognition problems, for the last two decades.

Features that are efficient parametric representations of the speech are extracted and used in a speech recognizer. Common features used include the linear predictive coding coefficients (LPCCs), real cepstrum (RC) coefficients, and the mel-frequency cepstral coefficients (MFCCs). The LPCCs and RCs were the most popular choices for speech recognizers up until the 1980s [Deller, Proakis, Hansen, 1993]. However, MFCCs have become the most popular features for speech recognition nowadays after a study by Davis and Mermelstein in 1980 [Davis, Mermelstein, 1980] showed the superior performance of the MFCCs versus the well-known LPCCs and RCs. The power of the MFCCs comes from the fact that their extraction approximates the human perception.

Words must be extracted before they can be recognized. Accurate segmentation is essential as studies have shown that more than half of the recognition errors are due to word boundary detection errors [Zhang, Zhu, Hao, 1997]. The Segmentation task is achieved by an end point detection algorithm which isolates speech utterances from the background noise sections, and this phase remains a challenging problem in real-life conditions where wide ranges of signal-to-noise ratios may be encountered.

A. OBJECTIVE

The main objective of this thesis is to implement an isolated word recognition system, which operates on utterances collected from the external auditory canals of the speakers via an ear-insert microphone.

This study is part of an on-going research project and uses data collected earlier by Newton [Newton, 2006]. The database includes repetitions of seven words collected in

an office environment from the external auditory canals of 20 native adult American English speakers.

The study had three main phases. First, the speech segmentation phase, where recorded utterances were first isolated from the background noise sections. This task was accomplished with a short-time absolute magnitude energy parameter and short-time energy-entropy feature (EEF). Second, the feature extraction phase, where we considered both RC coefficients and MFCCs extracted separately from every segmented utterance to be used as input features to the recognizer. Third, the classification phase, where we investigated several feedforward backpropagation neural network configurations. Ten different two- and three-layer neural network configurations were implemented and their word recognition performances compared.

B. RELATED RESEARCH

As discussed earlier, only a few speech recognition research studies that directly use the speech data collected from the ear canal have been reported in the literature, although speech collection from the ear canal through an ear-insert microphone is not a new concept in itself. The most recent in-ear microphone based study for hands-free communications applications has been reported recently by Westerlund [Westerlund, Dahl, Claesson, 2002a], [Westerlund, Dahl, Claesson, 2002b], [Westerlund, Dahl, Claesson, 2005]. In this study, the authors designed a speech recognition system for speech collected in a severely noisy environment from an in-ear microphone, showed that the recognizer accuracy was superior to that obtained with speech collected outside the mouth in the same noisy environment, and that performances further increased when noise reduction schemes were introduced.

Another recent study by Vaidyanathan showed that the air flows caused by the tongue movements in the external auditory canal could be collected via an in-ear microphone, and mapped to respective tongue movements with 97.7% accuracy. This study showed that the control commands collected via an in-ear microphone could be used in a man-machine interface for the operation of either a robot or a device for a handicapped person [Vaidyanathan, 2004a], [Vaidyanathan, 2004b].

C. THESIS ORGANIZATION

This thesis consists of six chapters and two appendices. Chapter II presents the equipment used for the recordings, how recordings were conducted, and background information about the speech data. Chapter III discusses the end point detection task, and the problems and difficulties associated with this step. Several different measures investigated for the end point detection scheme are also presented and discussed within Chapter III. Chapter IV presents the spectral-based RC and MFCC features selected as input features to the classifier. Chapter V discusses the concept of multi-layer feedforward neural networks, and presents recognition results. Finally, conclusions and suggestions for future studies are presented in Chapter VI.

Appendix A presents the specifications background information about the speech data used for recognition of the in-ear microphone used to collect the speech data. Appendix B contains the main Matlab codes implemented for the speech recognizer.

II. BACKGROUND

This chapter first presents a brief introduction to the concept of collecting the speech from the ear canals of the subjects. Next, it presents the equipment used and procedures followed for the recordings. Finally, it discusses the phonetic and spectral characteristics of the specific vocabulary words selected for the study, including “up,” “down,” “left,” “right,” “kill,” “pan,” and “move.”

A. INTRODUCTION

Collecting speech signals from locations other than the speaker’s mouth is not a new concept. Throat and bone conduction microphones can be commonly found in specialized applications [O’Neill, 1958], [Graciarena, Franco, Sonmez, Bratt, 2003], [Shahina, Yegnanarayana, 2005]. Ear canal microphones have also been considered when dealing with severe noise conditions. Speech collection through the external auditory canal may be achieved by means of a small microphone placed into the ear canal.

One of the earliest studies which investigates the intelligibility of the speech collected from the external auditory canal and the use of an ear-insert microphone was conducted by Black in 1957 [Black, 1957]. This early study showed that an ear-insert microphone could be used as an alternative for speech collection, and that the intelligibility of the speech recorded from the external auditory canal was superior to that collected via a microphone placed in front of the mouth in decreasing signal-to-noise ratios. Nowadays, speech collection through the ear canal is an ongoing research topic in biomedical applications [Rafaely, Furst, 1996] such as hearing aids, hearing screening tests, and speech enhancement for hands-free communication.

With the advances in computers and electronics in the last two decades, the reduced size and increased performance of the ear microphones accelerated the research on the speech collected from the external auditory canal. For speech enhancement purposes in hands-free communication, a microphone placed into the external auditory canal was used recently by Westerlund in his research, [Westerlund, Dahl, Claesson, 2002a], and [Westerlund, Dahl, Claesson, 2005]. In [Westerlund, Dahl, Claesson, 2002b], authors designed a speech recognition system for speech collected in a severely noisy

environment from an in-ear microphone, showed that the recognizer accuracy was superior to that obtained with speech collected outside the mouth in the same noisy environment, and that performances further increased when introducing noise reduction schemes.

Thus, previous research results show that collecting speech data via an in-ear microphone may be a viable alternative for speech processing applications, specifically for speech recognition. The current study investigates the implementation of a basic speech recognition system using in-ear microphone speech data, with the long-term goal of investigating the system robustness to environmental noise distortions, as a noise immune recognizer would be highly desirable in numerous applications.

B. EQUIPMENT USED

The speech data used for this research were recorded in an office environment at the Naval Postgraduate School (NPS) in an earlier study by Newton [Newton, 2006].

The complete system used for the recordings consisted of four main devices: an ear microphone, an analog-to-digital converter (A/D), a data acquisition (DAQ) card by National Instruments, and a notebook PC. The complete system is shown in Figure 2.1.

The in-ear microphone is mounted in a foam plug, which serves two purposes, enabling the microphone to stand still in the ear canal, and shielding the external auditory canal from the ambient noise by closing the ear canal entrance at the pinna (outer ear) side. The in-ear microphone used for the recordings is encased in foam and placed into the ear of a subject, as shown in Figure 2.2. Appendix A contains the in-ear microphone specifications.

The analog speech signals coming from the in-ear microphone are digitized using an analog-to-digital (A/D) converter before being sent to the computer for recording. The A/D converter produces speech signals sampled at $8\,kHz$ with a 16-bit quantization scheme.

Figure 2.1. The complete set-up used for the recordings (ear microphone, A/D Converter, DAQ Card, and notebook with interface program).

Figure 2.2. In-ear microphone used for recordings.

Next, digitized speech signals are transferred to a notebook PC via an interface cable which connects to the PCMCIA slot, or wireless connection slot, of the notebook via a National Instruments DAQ card. A Labview-based interface program is used during the recording sessions to control recordings, mark recordings as "good" or "bad," and save them to the computer.

Every recording is saved into a text file via the interface program loaded in the notebook by the session controller person. Each text file is named by the word being uttered, the respective number of the subject (speaker), the repetition number of the word by the same subject, and the recording date (*month-day-year* format with two digits). Therefore, the convention used for saving the individual recordings is as follows:

Word_subject #_repetition #_month_day_year

For example, "down_1_10_04_13_05.txt" indicates the tenth repetition of the word "down" by the first subject on April, 13, 2005. The noisy version of the word "kill" is denoted as "*kill_noise,*" and the word "right" collected through a microphone placed in front of the lips is denoted as "*right_outside.*" From now on, this convention will be used to denote a specific recording of a vocabulary word.

All recordings belonging to a given subject are saved into one folder. Next, each word file is split into individual trials in separate folders for each subject to allow for easy access and manipulation. Finally, each trial is cropped to isolate the speech section using an "end point detection" algorithm to be presented in the next chapter.

The following sections provide more details about the data set, vocabulary words, and their characteristics.

C. DATA DESCRIPTION

Twenty native adult American English speakers, including sixteen males and four females, were invited to speak seven different words, and recordings by each subject were conducted in separate sessions. The vocabulary used in this study consisted of seven words: *up*, *down*, *left*, *right*, *kill*, *move*, and *pan*. The data were collected via the in-ear microphone discussed earlier. In addition, subjects were asked to utter the word "*kill*" in a simulated noisy environment through the in-ear microphone, where the noise was generated by playing loud music in the office. Finally, each speaker was also asked to

utter the word "*right*" through a microphone placed in front of the mount of the speaker to investigate signal characteristic changes resulting from differences in the microphone location during recording.

Even though the final number of repetitions per each word by each subject may vary somewhat, due to the removal of a few "bad" trials in each recording session, each speaker uttered each word and a noisy version of the word "kill" forty nine times on average. The average number of recordings for the word "right" collected via the microphone placed in front of the mouth is nineteen. Therefore, the total size of the database, including all usable (i.e., good) repetitions of all words for all speakers, and including "kill_noise" and "right_outside," is 8,228.

The quantitative information about the data is given in Table 2.1. Each speaker is denoted by an integer number from 1 to 20 in the table. Female speakers are identified with an asterisk (*) next to their identification number. This table lists the number of repetitions per word for each speaker, the average number of repetitions per word, the total number of repetitions per word, and the total data size (i.e., the total number of recordings).

Subject Number	The Number of Repetitions per Word								
	up	**down**	**left**	**right**	**kill**	**pan**	**move**	**kill_noise**	**right_outside**
1	44	49	49	49	49	49	43	45	19
2	48	49	49	50	49	49	49	49	19
3	50	44	49	49	48	48	39	49	19
4*	49	49	49	49	49	49	49	49	19
5	52	48	49	51	48	46	49	51	20
6	51	51	50	49	49	50	50	52	19
7	51	46	52	48	49	40	45	48	21
8	52	49	49	49	53	52	50	49	19
9*	46	50	49	50	49	50	52	49	19
10	49	49	50	49	49	49	49	49	24
11*	49	48	50	48	50	50	50	50	19
12*	49	49	48	49	50	48	51	51	19
13	52	51	52	50	44	50	50	51	21
14	48	49	49	50	49	51	49	48	19
15	49	50	50	49	49	48	49	30	18
16	54	51	49	50	48	50	50	50	20
17	49	49	49	49	49	49	49	49	20
18	46	49	49	51	49	48	45	47	19
19	62	49	49	49	49	49	49	49	19
20	48	48	49	52	49	49	50	49	19
Average # of Repetitions per Word	49.9	48.85	49.45	49.5	48.9	48.7	48.35	48.2	19.55
Total # of Repetitions per Word	998	977	989	990	978	974	967	964	391
Overall Data Size	**8228**								

Table 2.1. Quantitative information about the data set used for the speech recognition task. Subject numbers with an asterisk (*) next to them indicate female speakers.

1. Phonetic Classification of the Data

Speech is produced by the movements of the articulators constituting the human vocal tract, i.e., the movement of the organs such as vocal folds, larynx, pharynx, tongue, lips, and teeth, with the force of air through the lungs. Since speech is time-varying (or non-stationary) due to the rapid changes of the vocal tract during speech production, speech is usually segmented into smaller units or sounds that carries acoustical information [Deller, Proakis, Hansen, 1993]. These actual sound units that have certain acoustic and articulatory properties are called *phones*. Phones are realizations of *phonemes*, which are the basic theoretical linguistic units that comprise the word [Deng, O'Shaughnessy, 2003]. Therefore, phonemes are the smallest meaningful unit in a language, and the phones are the actual sounds of these phonemes uttered by a speaker.

The *phonetic classification* is the process of grouping the phonemes based on their properties related to their waveforms, frequency characteristics, manner of articulation, place of articulation, type of excitation, and the stationary characteristic of the phoneme [Deller, Proakis, Hansen, 1993]. The most general classification scheme groups phonemes into two broad categories, voiced speech which does not restrict the airflow through the vocal tract, and unvoiced speech which restricts the airflow at some point and look like noise [Deng, O'Shaughnessy, 2003]. A more specific classification based on the properties mentioned above include vowels, diphthongs, fricatives, affricates, nasals, liquids, glides, and stops (or, plosives).

The knowledge of the phonetic context of the vocabulary words will be especially important for analyzing the implementation and performance of the end point detection algorithm that will be discussed in the next chapter. As will be explained later, it may be a very challenging task to isolate the speech segment within the data file when the speech starts and/or ends with a weak fricative and/or a stop consonant (plosive), especially when the recording includes unwanted distortions.

Although the vocabulary size is small, it is very rich in phonetic context such as different types of vowels, fricatives, stops (plosives), nasals, liquids, and diphthongs. Next, the phonemes present in each word will be discussed. The definitions and the

phoneme tables in both [Deller, Proakis, Hansen, 1993] and [Deng, O'Shaughnessy, 2003] are used for the phonetic classification of the vocabulary words.

The word "*up*" has the vowel /A/ and the unvoiced oral stop /p/ following the vowel sound. The word "*down*" starts with a voiced stop /d/ and continues with a diphthong /aw/ and ends with a final nasal /n/. The word "*left*" consists of a liquid /l/, a vowel /@/, an unvoiced fricative /f/, and a final unvoiced stop /t/. The word "*right*" has a liquid /r/, a diphthong /ay/, and a final unvoiced stop /t/. The word "*kill*" has an unvoiced plosive /k/, a vowel /i/, and a liquid /l/. The word "*pan*" starts with an unvoiced plosive /p/ and continues with the vowel /E/ and ends with a nasal /n/. The word "*move*" has a nasal /m/, a vowel /u/, and a voiced fricative /v/.

The phonetic classification of the vocabulary words is summarized in Table 2.2.

Vocabulary Words	**Phonemes**					
	vowel	**diphthong**	**liquid**	**nasal**	**fricative**	**stop (plosive)**
up	/ A /					/ p /
down		/ aw /		/ n /		/ d /
left	/ @ /		/ l /		/ f /	/ t /
right		/ ay /	/ r /			/ t /
kill	/ i /		/ l /			/ k /
pan	/ E /			/ n /		/ p /
move	/ u /			/ m /	/ v /	

Table 2.2. Phonetic classification of the vocabulary words.

D. SPECTRAL CHARACTERISTICS OF THE DATA

In order to extract the spectral characteristics of the vocabulary words, their short-time Fourier Transform (STFT) magnitudes or spectrograms are investigated since they best express the time-varying nature of the speech signals and combine both the time-domain and frequency-domain information into a single, consistent, and integrated framework [Deng, O'Shaughnessy, 2003]. The recorded utterances are sampled at $8\ kHz$. Therefore, the frequency axis on the spectrograms ranges up to $4\ kHz$. The time axis, on the other hand, goes up to the end of the recordings.

An example of both the time-domain waveform and the respective spectrogram for each vocabulary word collected via the in-ear microphone is illustrated in Figure 2.3

through Figure 2.9. The time-domain waveform and associated spectrogram of the word "right" collected via a microphone placed in front of the lips are shown in Figure 2.10.

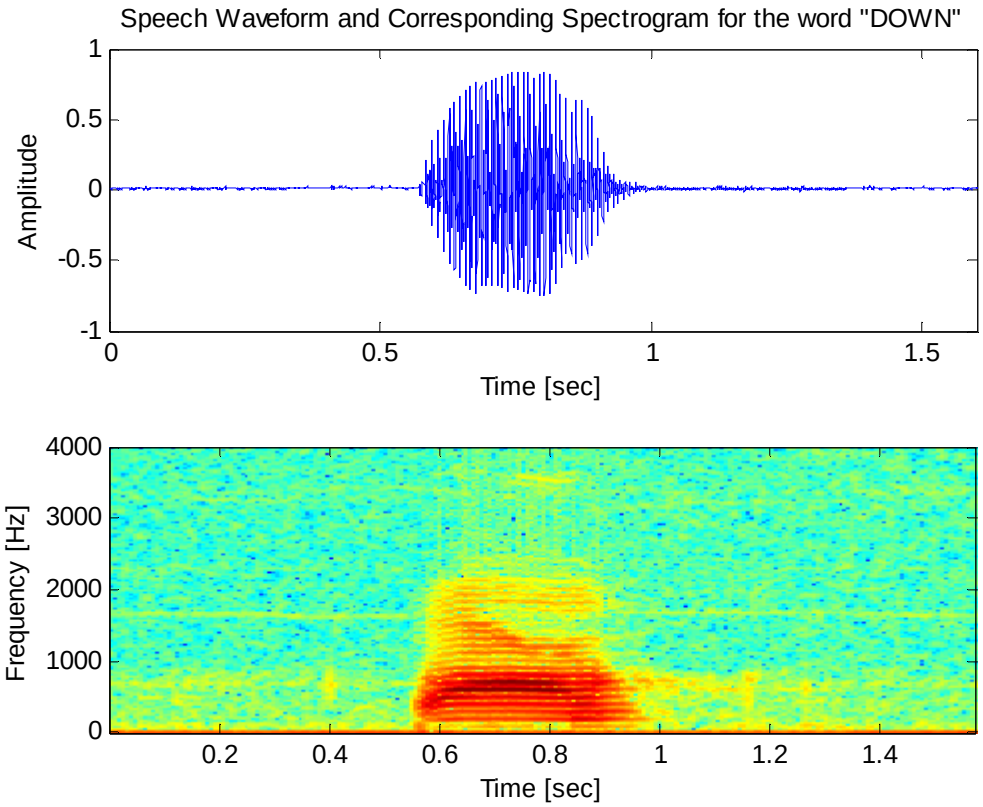

Figure 2.3. Speech waveform (top plot) and associated spectrogram (bottom plot) of the word "down" (down_8_18_04_29_05.txt).

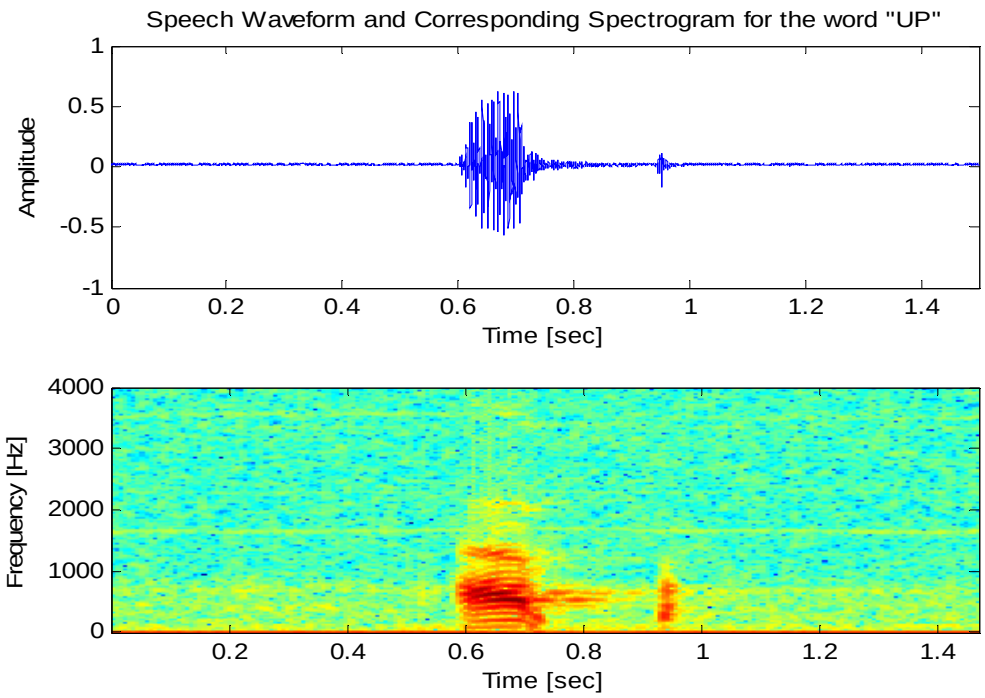

Figure 2.4. Speech waveform (top plot) and associated spectrogram (bottom plot) of the word “up” (down_8_36_04_29_05.txt).

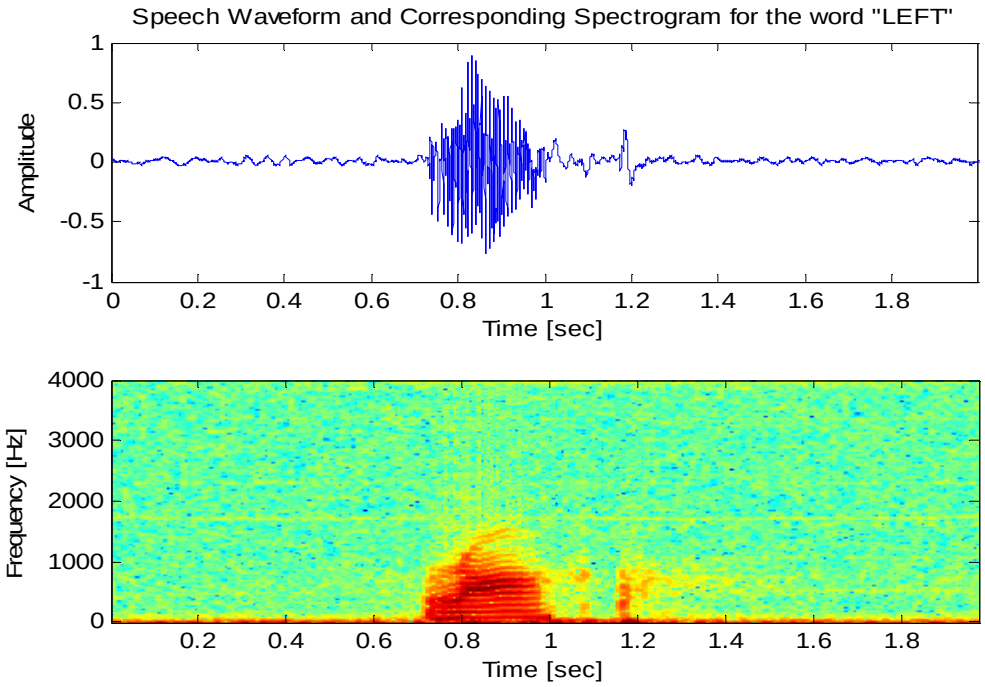

Figure 2.5. Speech waveform (top plot) and associated spectrogram (bottom plot) of the word “left” (left_3_18_04_25_05.txt).

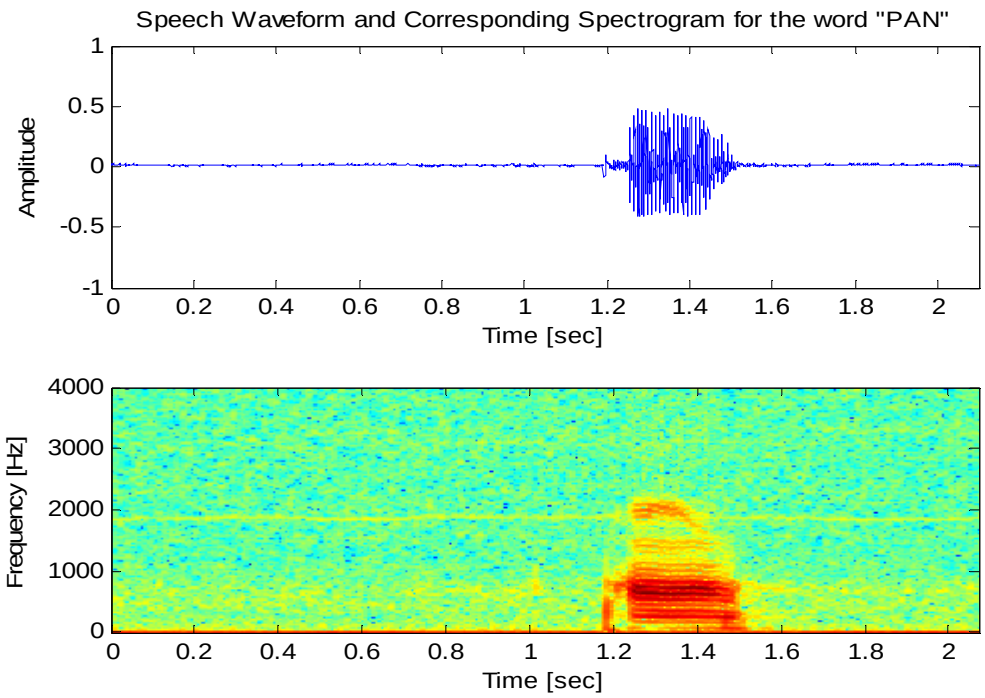

Figure 2.6. Speech waveform (top plot) and associated spectrogram (bottom plot) of the word “pan” (pan_13_35_05_04_05.txt).

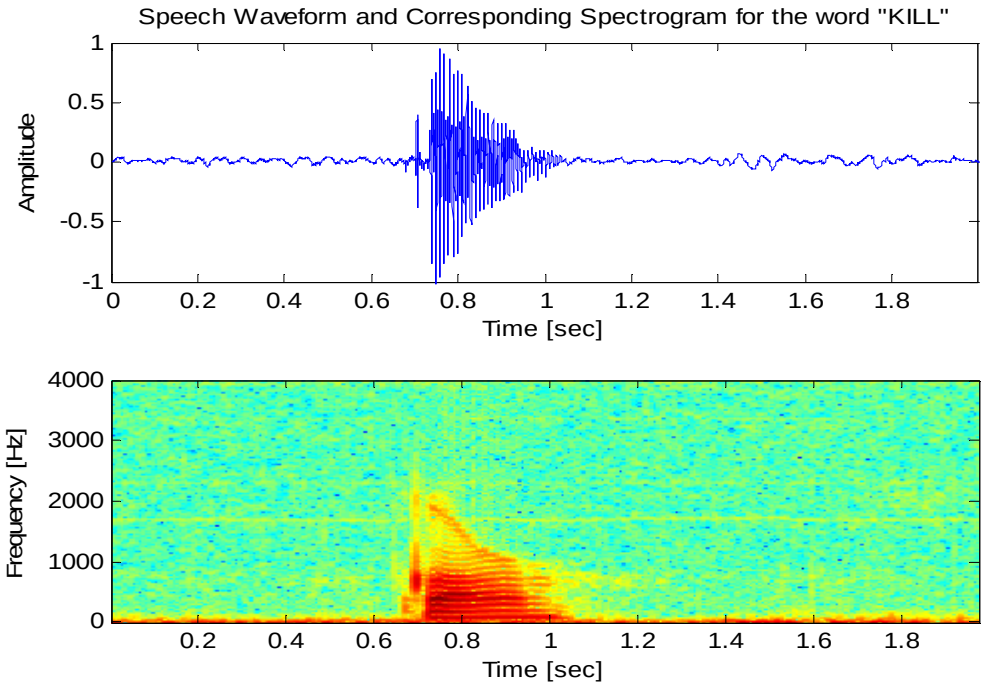

Figure 2.7. Speech waveform (top plot) and associated spectrogram (bottom plot) of the word “kill” (kill_3_20_04_25_05.txt).

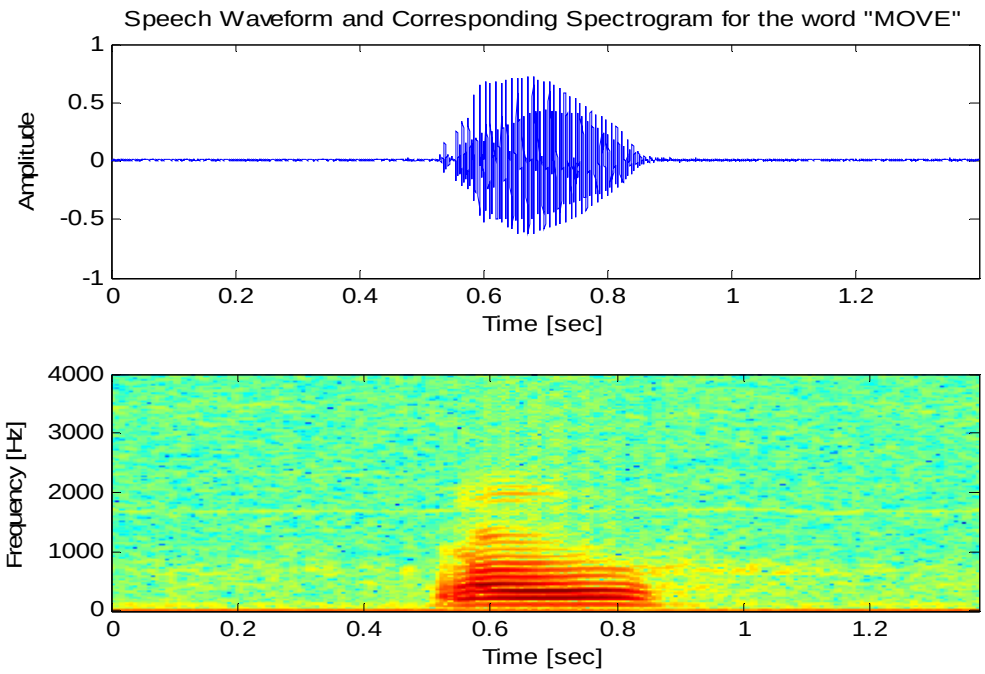

Figure 2.8. Speech waveform (top plot) and associated spectrogram (bottom plot) of the word “move” (move_8_6_04_29_05.txt).

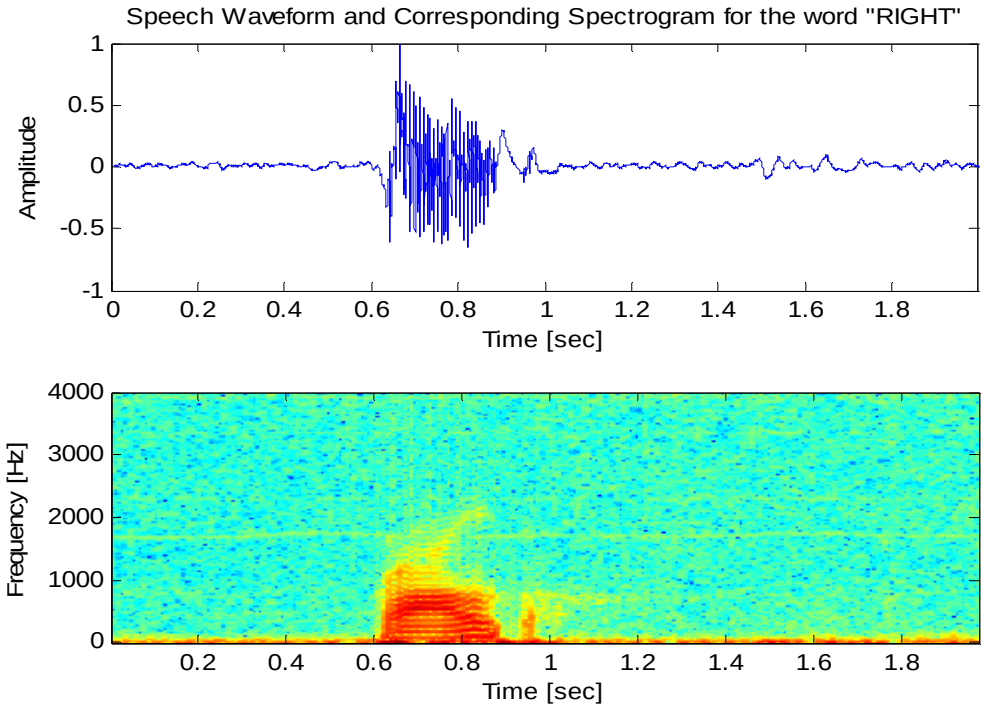

Figure 2.9. Speech waveform (top plot) and associated spectrogram (bottom plot) of the word “right” (right_3_25_04_25_05.txt).

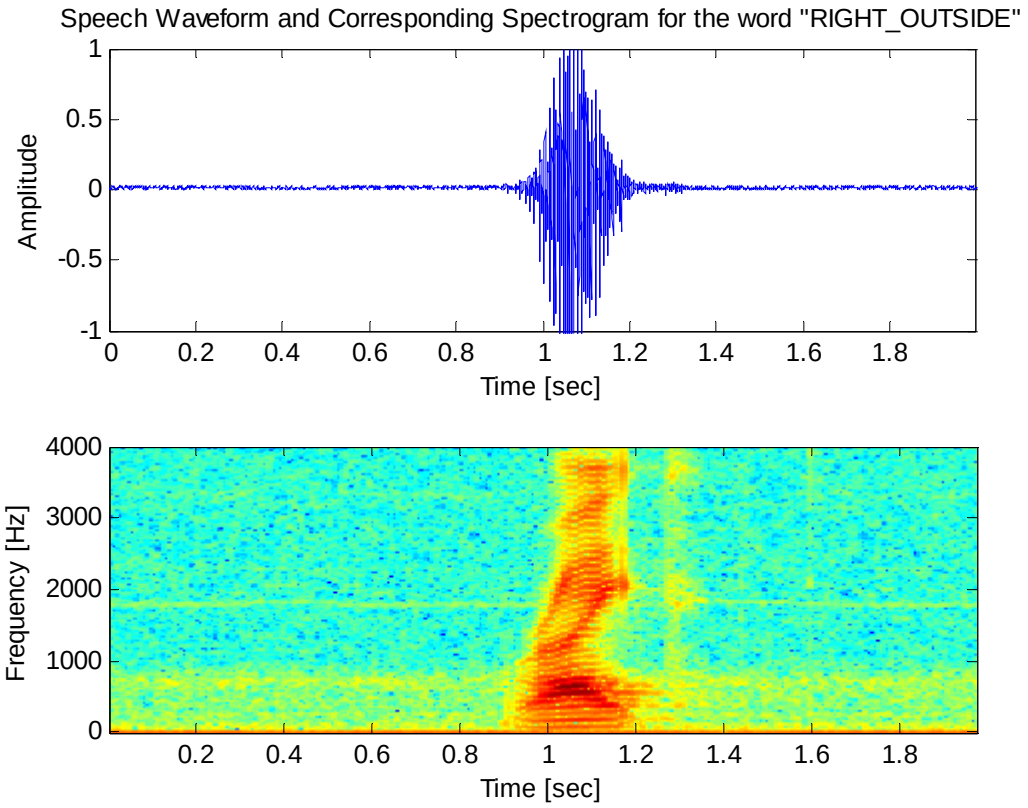

Figure 2.10. Speech waveform (top plot) and associated spectrogram (bottom plot) of the word "right" collected through the in-ear microphone placed in front of the mouth (right_outside_13_10_05_04_05.txt).

A few important general observations, which will be very important for the front-end processing of the data, can be made here:

- There is a constant very low-frequency disturbance or "hum" present in all recorded speech spectrograms. This hum occupies the frequencies between 0 and 100 *Hz*.
- The voiced portions of the speech waveforms, where most of the signal energy is concentrated, have frequency information mainly up to 2.3 *kHz*, and not much frequency content is left above 2.3 *kHz*.
- The frequency content of the word "*right*" collected via an outside microphone goes up to 3.5 *kHz*, and even up to 4 *kHz* in some trials.

Figures 2.9 and 2.10 show that the within-the-ear speech is dampened for frequencies above 2.1 *kHz*. The in-ear microphone placed into the ear canal picks up mainly the bone conducted speech and the sound conducted through the muscles and tissues covering the skull. However, the speech energy transmitted through the muscles

and tissues in the head is considered negligible. These muscles and tissues are also assumed to have an attenuation effect on higher frequencies, i.e., they have a low-pass filtering effect on the bone conducted speech [Westerlund, Dahl, Claesson, 2005]. As a result, the speech collected from the ear canal is expected to be the low-pass filtered version of the speech collected from the same microphone placed in front of the mouth, explaining why the frequency content of the same word collected from the different mediums differs from each other.

E. SUMMARY

This chapter presented the main idea behind collecting speech using an ear-insert microphone. It also discussed both phonetic and spectral properties of the vocabulary words selected for the data collection, together with the equipment used and how the recordings were conducted.

The following chapter presents the end point detection algorithm which is an essential and integral part of any speech recognizer.

III. END POINT DETECTION

This chapter discusses end point detection, which is one of the most important parts of any speech recognizer. The chapter first briefly presents the basic idea behind end point detection. Second, it presents the problems encountered in the end point detection for the data under consideration in this study. Third, it presents the different parameters (or features) employed for various end point detection algorithms commonly used nowadays which were considered in this study: the short-time energy, Teager's energy, zero-crossing rate, and energy-entropy schemes. Fourth, it discusses the end point detection algorithm implemented for the current study, which operates on the recordings that were filtered by an infinite impulse response (IIR) bandpass filter at the preprocessing stage prior to detection. Finally, this chapter concludes with a discussion about the effect of this IIR bandpass filter on the detection scheme.

A. END POINT DETECTION

End point detection is the process of finding the edge points of an uttered word or speech segment in the presence of background noise, i.e., finding the starting and ending points of the speech signal. End point detection is also equivalent to measuring the duration of an utterance in a certain interval of background noise [Taboada, Feijoo, Balsa, Hernandez, 1994].

Accurately detecting the speech signal end points is very important in speech recognition applications for two reasons [Lamel, Rabiner, Rosenberg, Wilpon, 1981]:

- The accuracy and reliability of the speech recognizer critically depend on the accurate detection of the boundaries of the uttered words.
- Accurate detection of the end points and successful removal of the background noise or silence from the speech portions reduce the subsequent computations significantly.

The first reason mentioned above can be understood easily if a recognizer using dynamic time warping (DTW) algorithm is considered, where an incoming end point detected speech signal is compared with some reference templates. In that case, the time alignment of the signals becomes a crucial issue. The study by Junqua et al., [Junqua, 1991], showed that more than half of the recognition errors were due to word boundary detection errors [Zhang, Zhu, Hao, Luo, 1997]. The second reason is self-explanatory

because the processing time and burden for a recognizer are greatly reduced when unnecessary parts of the recordings are removed via the end point detection algorithm prior to further processing.

End point detection may seem to be a trivial task to accomplish, but it is not the case, except in high signal-to-noise ratio environments, where discrimination between speech segments and background is very easy, and simple algorithms yield close to perfect results. In real-life applications, we are highly unlikely to find such high signal-to-noise levels; hence, speech signals with lower signal-to-noise ratios and different noise types make end point detection a challenging problem. As a result, end point detection has been an active research topic since the 1970s.

Up until the 1990s, a very limited amount of research on end point detection appears in the literature. Two studies that have been commonly referenced are by Rabiner, [Rabiner, Sambur, 1975], and Lamel, [Lamel, Rabiner, Rosenberg, Wilpon, 1981], which are mainly based on energy measures. There has been a noticeable increase on the amount of research conducted on word boundary detection after the 1990s, and many different techniques have been investigated such as spectral approaches, variable frame rate methods, and lately entropy-based approaches. However, these research studies are mostly data-specific, and yet, no globally accepted or widely used approach has been proposed.

Three types of end point detection schemes are currently available: explicit, implicit, and hybrid [Lamel, Rabiner, Rosenberg, Wilpon, 1981]. The main difference between explicit and implicit schemes is that there is a separate and independent end point detection stage prior to the recognizer in explicit techniques, whereas there is no separate endpoint detector in implicit approaches, i.e., it is accomplished by the recognition stage. The hybrid scheme essentially combines both explicit and implicit approaches. In [Lamel, Rabiner, Rosenberg, Wilpon, 1981], it is also indicated that a sophisticated explicit end point detection scheme always outperforms the other two approaches. Hence, an explicit end point detector is employed for the present study. The general block diagram of a speech recognizer using an explicit end point detector is shown in Figure 3.1.

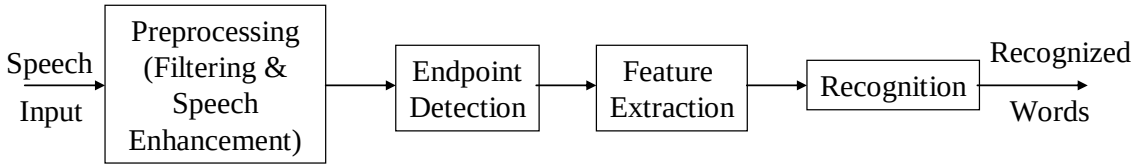

Figure 3.1. Block diagram of a speech recognizer using an explicit end point detector.

As noted earlier, end point detection algorithms that have been used or suggested so far are based on short-time energy, spectral-energy like Teager's energy, zero-crossing rate (ZCR), variable frame rate (VFR) methods using cepstral or time derivate features, entropy, or energy-entropy (EE) features. The end point detection algorithm used for this study relies on two of these schemes, namely short-time absolute magnitude energy and energy-entropy (EE). Some of the other schemes mentioned above were also considered during early stages of this study and are discussed later in this chapter.

B. PROBLEMS ENCOUNTERED WITH END POINT DETECTION

End point detection is very challenging in real-life applications, as noted in the previous section. This is due to the fact that it is not possible to find a clean environment where signal-to-noise ratio is so high that even the lowest-energy portions of the speech signal, such as weak fricatives, significantly exceed the energy level of the background noise or silence. Although a simple energy-based algorithm that uses some types of thresholds is enough to discriminate low-energy level sounds from background noise in a high signal-to-noise case, it is usually not sufficient to do the same separation in real-life conditions.

Recall from the previous chapter that the speech data recorded for the current study were taken in an office environment and includes distortions commonly found in such environments such as other speakers, phone rings, etc… Therefore, the end point detection is not a trivial problem with the data at hand since it gives an excellent example of real-life conditions.

Examples of typical silence sections, or background noise, for the recordings via both the ear-insert microphone and the microphone placed in front of the mount are shown in Figure 3.2 and Figure 3.3, respectively, along with their estimated power spectral densities. The power spectral density estimations are obtained by using the

classical periodogram method. Both silence sections include the low-noise hum mentioned earlier, which occupies the frequency range up to 100 *Hz* in all recordings. Note that the power of the low-frequency hum present in the in-ear microphone data drops to a mean level of −90 *dB* at around 200 *Hz* and stays at that level, whereas the power of the low-frequency hum present in the outside microphone data is slow to decay to its average value of −90 *dB*.

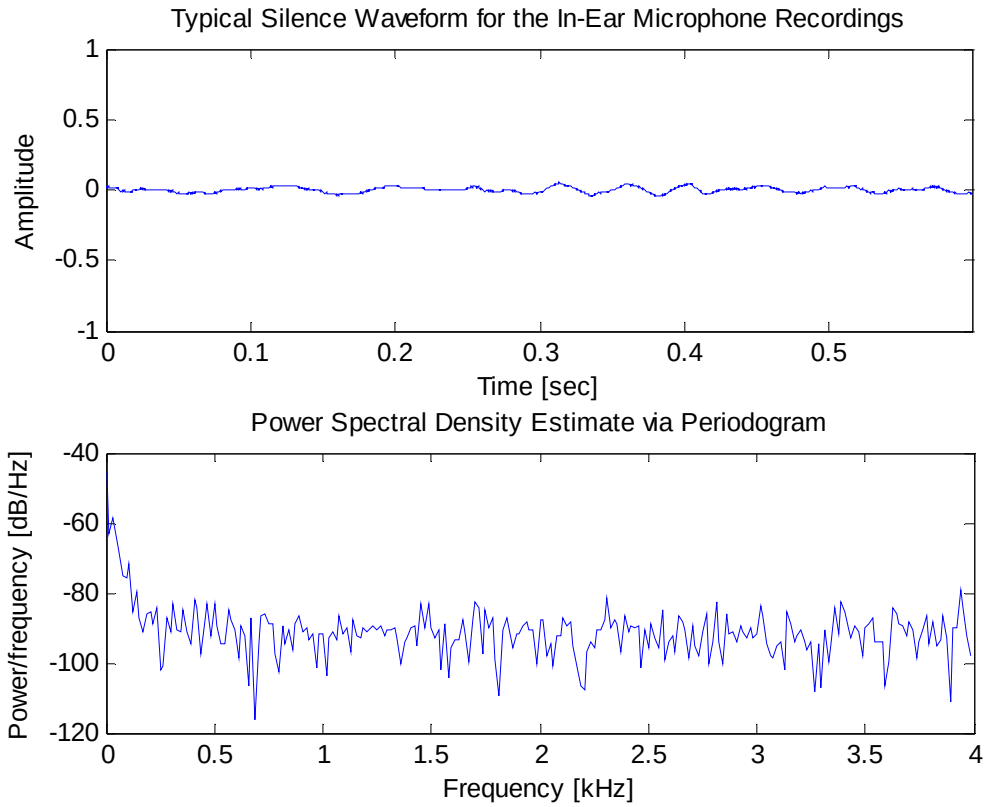

Figure 3.2. Typical silence waveform (top plot) and related power spectral density (bottom plot) for the recordings with the ear microphone (first 600 *ms* of "left_3_18_04_25_05.txt").

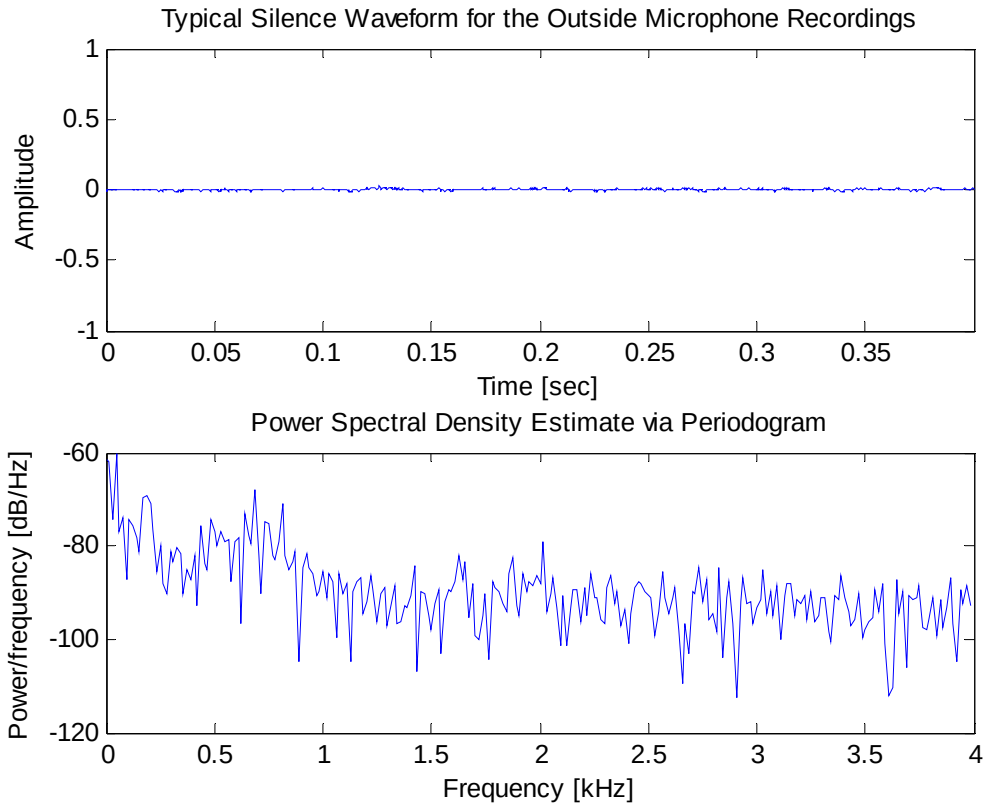

Figure 3.3. Typical silence waveform (top plot) and related power spectral density (bottom plot) for the recordings with the microphone placed in front of the mount (first 400 *ms* of "right_outside_19_17_08_22_05.txt").

The noise types encountered in real environments, which cause failures in the end point detection algorithms, can be divided into three broad categories [Taboada, Feijoo, Balsa, Hernandez, 1994]:

- Stationary noise associated with the transmission system, i.e. microphone and/or the surroundings.
- Sporadic (non-stationary) noise including people talking in the vicinity, doors opening and closing, telephone ringing, mechanical (factory) noises, and so on.
- Noise or artifacts generated by the speaker such as mouth noises, i.e., sounds made by the tongue and lips, heavy breathing noises, and breath releases.

The last two of these noise types are the most difficult ones to deal with, and increase the complexity of end point detection algorithm. Non-stationary noises are long in duration and strong in intensity when compared to the speaker-generated artifacts such

as clicks and breath noise, which are much shorter in duration but sharper in intensity. A very strong and long-lasting non-stationary noise can clutter the whole spectrum of an utterance to the point that the speech may seem to be buried in noise. A speaker-generated artifact such as a click, heavy breath release, coughing, or lip smacking attached to either the front or end of the desired speech segment can easily be detected as part of the speech segment.

Examples of speech containing non-stationary noise and user-generated artifacts are shown in Figure 3.4 and Figure 3.5, respectively. In both figures, the bandpass filtered speech waveforms are plotted on top, and their respective absolute magnitude energy contours, which will be explained later in the following section, are plotted at the bottom. The actual boundaries of the speech segments are indicated with vertical dotted lines on each plot. Noise types present in the recordings are also indicated on the energy plots. Figure 3.4 illustrates an utterance contaminated with heavy mechanical (factory) noise. Note that the mechanical noise burst preceding the speech segment is much longer in duration than the speech and half in strength. Figure 3.5 illustrates a speech recording heavily cluttered with speaker-generated artifacts. The two clicks present before the utterance are shorter in duration and less strong in energy than the speech segment as noted earlier; however, the last artifacts generated by the speaker are longer in duration and much stronger in intensity, exceeding the speech energy.

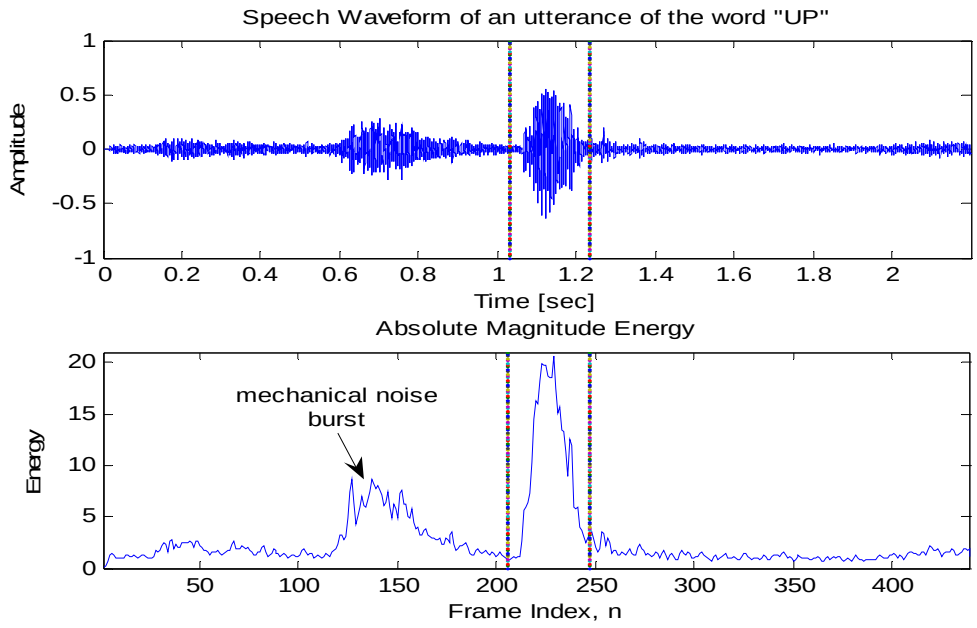

Figure 3.4. Example of utterance contaminated by mechanical noises; speech waveform (up_1_12_04_13_05.txt) (top plot), absolute magnitude energy contour (bottom plot). Vertical dotted lines indicate the utterance boundaries.

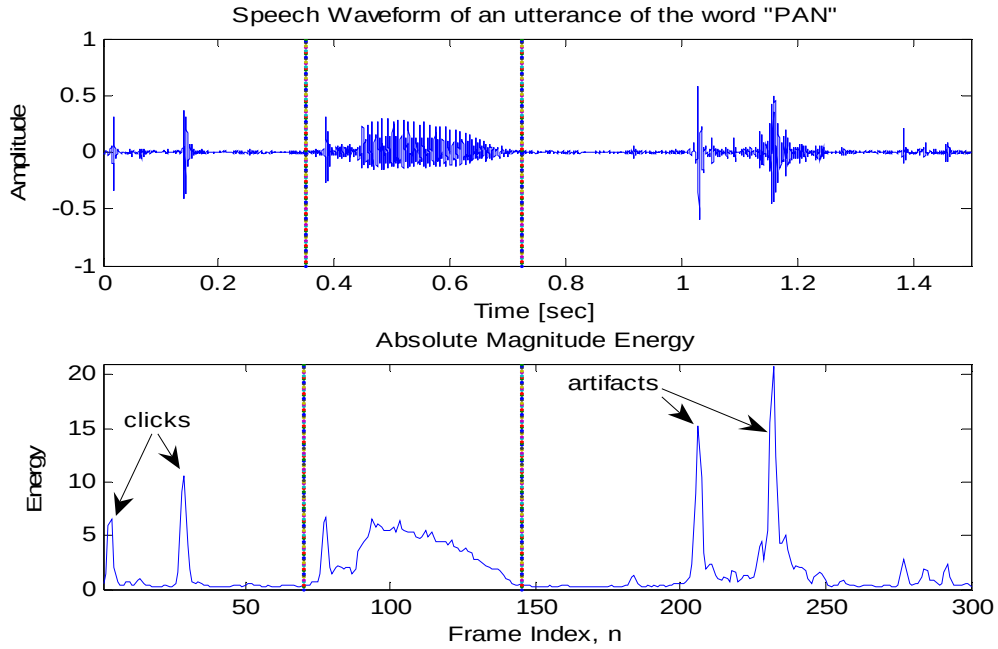

Figure 3.5. Example of speech contaminated by speaker generated artifacts; speech waveform (pan_7_4_04_29_05.txt) (top plot), absolute magnitude energy contour (bottom plot). Vertical dotted lines indicate the utterance boundaries.

Difficulties in end point detection arise not only from the different types of noise present in the recordings, but also from the vocabulary words themselves. Some phonemes or sounds have very low energy when compared to the vowel portion of the speech, and they look like background noise. It is even very hard for a naked eye to detect the beginning or the end of an utterance in the presence of these low-energy, noise-like phonemes at either the beginning or the end of the utterance. Broad categories of such problems encountered include [Rabiner, Sambur, 1975]:

- Weak fricatives (/f, th, h/) at the beginning or end of an utterance.
- Weak plosive bursts (/p, t , k/).
- Final nasals (/m, n/).
- Voiced fricatives at the end of words which become unvoiced.
- Trailing off of certain voiced sounds.

As presented in Table 2.2, the vocabulary words selected for the current study possess the first three properties. This issue also makes the end point detection to be implemented for the purpose of this study a much more challenging task, given the fact that the data has all sorts of noises as mentioned earlier.

C. END POINT DETECTION MEASURES

This section presents some of the measures that have been used for or proposed for end point detection which are considered in this study. These measures include short-time energy quantity, Teager's energy algorithm, short-time zero-crossing rate (ZCR), and energy-entropy feature (EEF). Two of these measures, short-time energy quantity and energy-entropy feature (EEF), are used for the end point detection algorithm in the current study.

1. Short-Time Energy Measure

The short-time energy measure has been used extensively in many end point detection algorithms for decades since it is computationally simple and easy to implement in both software and hardware. The short-time energy is also a natural way of representing the amplitude changes in speech signals.

As mentioned earlier in the chapter, some segments of a speech signal such as unvoiced segments have much lower amplitude than the voiced segments, resulting in these unvoiced segments with lower energy than their voiced counterparts. Therefore, the

energy measure can be used to discriminate between voiced and unvoiced segments with appropriately set thresholds, even in poor signal-to-noise levels. The short-time energy measure can also be used to discriminate between speech and silence segments with a simple threshold, especially in environments with very high signal-to-noise ratios ($30\ dB$ or higher) as in such a case the lowest energy segments of the speech signal will exceed the energy of the silence segments [Rabiner, Sambur, 1975].

Short-time energy parameters commonly used in end point detection algorithms are squared energy, logarithmic energy, root mean square (RMS) energy, and absolute magnitude energy.

The squared energy parameter is defined as:

$$E_n = \sum_{i=1}^{N} x^2(i), \tag{3.1}$$

where n is the frame index, and N is the total number of samples in a given frame.

The logarithmic energy is given by:

$$E(n) = \sum_{i=1}^{N} \log_{10} x^2(i). \tag{3.2}$$

The root mean square (RMS) energy is defined as:

$$E(n) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x^2(i)}. \tag{3.3}$$

Finally, the absolute magnitude energy parameter is given by:

$$E(n) = \sum_{i=1}^{N} \left| x(i) \right|. \tag{3.4}$$

The four types of short-time energies defined above are depicted in Figure 3.6 and Figure 3.7 for one of the recordings of the word “left” (left_3_18_04_25_05.txt). Recall that the time-domain waveform and corresponding spectrogram of the utterance was shown in Figure 2.5. The absolute magnitude energy and squared energy contours are shown in Figure 3.6. The RMS energy and logarithmic energy contours are shown in

Figure 3.7. The vertical dotted lines on each energy plot indicate the edge points of the utterance that are detected manually by listening to the utterance. Energy contours are computed using the bandpass filtered version of the utterance.

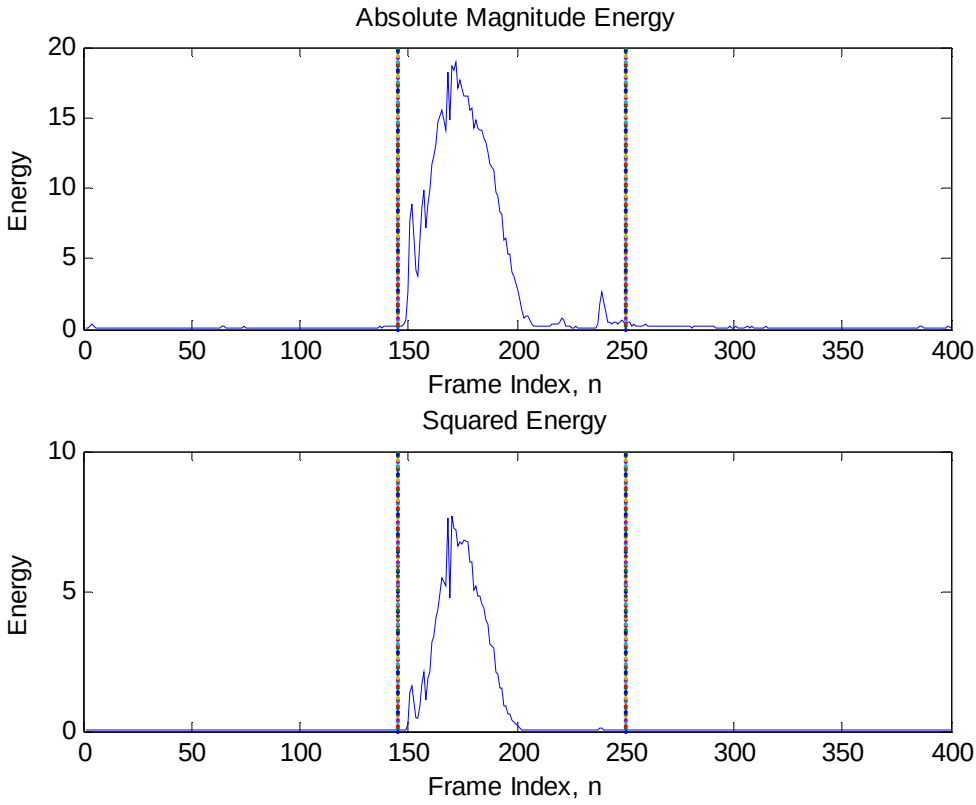

Figure 3.6. Absolute magnitude energy contour (top plot) and squared energy contour (bottom plot) of the word “left” (left_3_18_04_25_05.txt). Vertical dotted lines indicate the end points of the utterance.

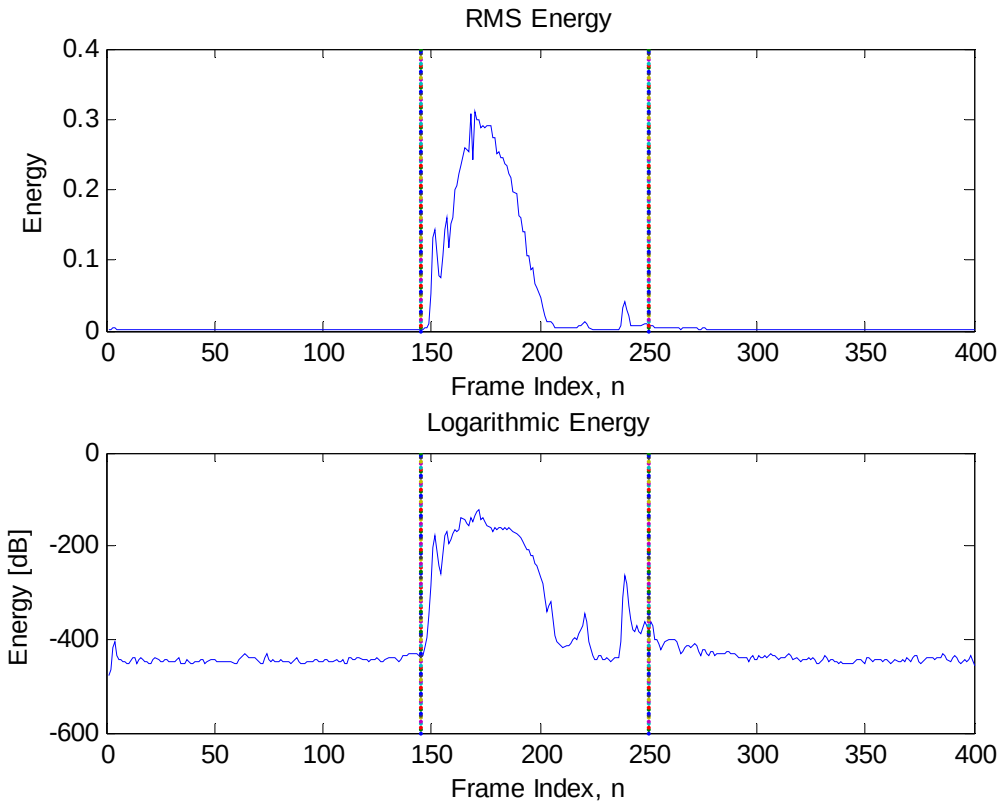

Figure 3.7. RMS energy contour (top plot) and logarithmic energy contour (bottom plot) of the word "left" (left_3_18_04_25_05.txt). Vertical dotted lines indicate the end points of the utterance.

The logarithmic energy reveals the details of the weak portions of the speech better than any other energy parameter defined above due to the nonlinear compression of the logarithm function applied to the signal amplitude. However, it also amplifies the noise or silence portions of the recording and makes it harder to set a threshold from this noisy signal [Qiang, Youwei, 1998].

The squared energy measure suppresses low-frequency noise completely, and is more stable than other measures. However, weak segments of an utterance, such as the weak fricative /f/ and stop consonant /t/ following the fricative in the word "left", are also deemphasized at the same time along with the background noise. Thus, the squared energy quantity gives very conservative edge point estimates and can be used to detect the voiced (mainly, vowel) portions of the utterances [Qiang, Youwei, 1998].

The RMS energy resembles the squared energy in the sense that it is a scaled version of the squared energy parameter. The square root operator emphasizes low energy segments while deemphasizing higher energy ones, which also makes it resemble the absolute magnitude energy. This characteristic in turn reduces the big energy difference between voiced and unvoiced segments present in an utterance, but the computational load increases at the same time.

The absolute magnitude energy reflects the sum of the magnitudes of sample amplitudes per frame; hence, the weak unvoiced segments of the utterance are not deemphasized, and this quantity reveals information about the speech segment. As a result, it is much easier to define some thresholds from the background noise for end point detection than it is for the other quantities discussed. However, the detection scheme may become unstable in strong noise cases since the background noise is not suppressed at all [Qiang, Youwei, 1998].

The above discussion highlighted the facts that each short-time energy based measure has its own advantages and disadvantages. Among these energy based quantities, the absolute magnitude energy quantity is the simplest and fastest one in implementation, making it suitable for on-line implementations while yielding the same (or sometimes better) performance as the others. Therefore, the absolute magnitude energy quantity was ultimately chosen for the end point detector implemented for the present study.

2. Teager's Energy Algorithm

The Teager's energy algorithm, which was based on a new algorithm developed by Teager [Teager, 1980] in modeling speech production, was first presented by Kaiser, [Kaiser, 1990], to compute the energy of a signal. If the samples of a signal representing the oscillatory motion of the body are given by $x_i = A\cos(\Omega i + \phi)$, where A is the sample amplitude, Ω is the digital frequency in radians/sample, and ϕ is the arbitrary initial phase in radians, then the energy of the signal is given as [Kaiser, 1990]:

$$E_i = x_i^2 - x_{i+1}x_{i-1} = A^2 \sin^2(\Omega) \approx A^2\Omega^2. \tag{3.5}$$

The above equation is known as the Teager's energy algorithm, or simply the Teager's algorithm. Some observations can be made from Equation (3.5). While calculating the instantaneous energy per sample E_i, the algorithm takes into account not only the current sample, but also two adjacent samples. Thus, the instantaneous energy computed on the time-domain samples can capture dynamic changes in a signal rapidly. The energy measure is affected by both the amplitude and the frequency of the samples. The last property also makes the algorithm track and respond rapidly to amplitude and frequency changes.

The reasons indicated above made the Teager's energy algorithm attractive for the end point detection because it can be used as a stand-alone feature for the end point detection, and replace the short-time zero-crossing rate (ZCR) quantity to be discussed next. It was first applied to the end point detection problem by Ying et al. [Ying, Mitchell, Jamieson, 1993], who implemented energy computations on a per-frame basis instead of on a per-sample basis and called the resulting algorithm the "frame-based Teager energy" approach.

The power spectrum of the samples in a frame is first estimated from the fast-Fourier transform (FFT) of the frame in the frame-based Teager energy approach [Ying, Mitchell, Jamieson, 1993]:

$$P_n(\omega) = \frac{1}{N/2} \cdot X_n(\omega) \cdot X_n^*(\omega), \quad \text{for} \quad \omega = 0, ..., \pi/2, \tag{3.6}$$

where $X_n(\omega)$ is the FFT of the n-th frame, N is the total number of the FFT points in a frame, and ω is the digital frequency in radians/sample.

Next, each sample in the power spectrum is weighted with the square of its corresponding digital frequency:

$$P_n(\omega) \cdot \omega^2, \quad \text{for} \quad \omega = 0, ..., \pi/2. \tag{3.7}$$

Finally, the frame energy is obtained by taking the square root of the sum of the weighted power spectrum defined in Equation (3.7):

$$E_n = \left(\sum_{i=1}^{N/2} P_n(\omega) \cdot \omega^2 \right)^{1/2}. \quad (3.8)$$

The Teager's algorithm is illustrated in Figure 3.8 with the same utterance used previously, i.e., one of the utterances of the word "left" (left_3_18_04_25_05.txt). The Energy contour is again computed over the bandpass filtered version of the utterance. The filtered speech waveform is also shown in the same figure on top. The vertical dotted lines on the plots indicate the actual end points of the utterance. When the Teager's energy contour is compared to the previously discussed energy contours of the same word in Figure 3.6 and Figure 3.7, it resembles the RMS energy with the exception that the latter is computed in the frequency domain.

The Teager's energy algorithm gives more conservative end point estimates than the short-time absolute magnitude energy does. In an on-line implementation, where the algorithm is forced to move from left to right and to decide as it operates on the data, the Teager's energy algorithm also fails to capture the exact end point locations when there are weak fricatives and/or stop (plosive) consonants at either the beginning or the end of an utterance, which is a common problem in algorithms operating with energy parameters.

In addition, the algorithm is not as efficient and successful in tracking the changes in signals with multiple frequency components as it is for signals with few frequency components, and it is sensitive to noise when the signal contains different frequency components [Kaiser, 1990]. Finally, Equations (3.6) through (3.8) reveal the fact that the Teager's algorithm is computationally expensive.

As a result, the Teager's energy algorithm was not applied in our end point detection scheme.

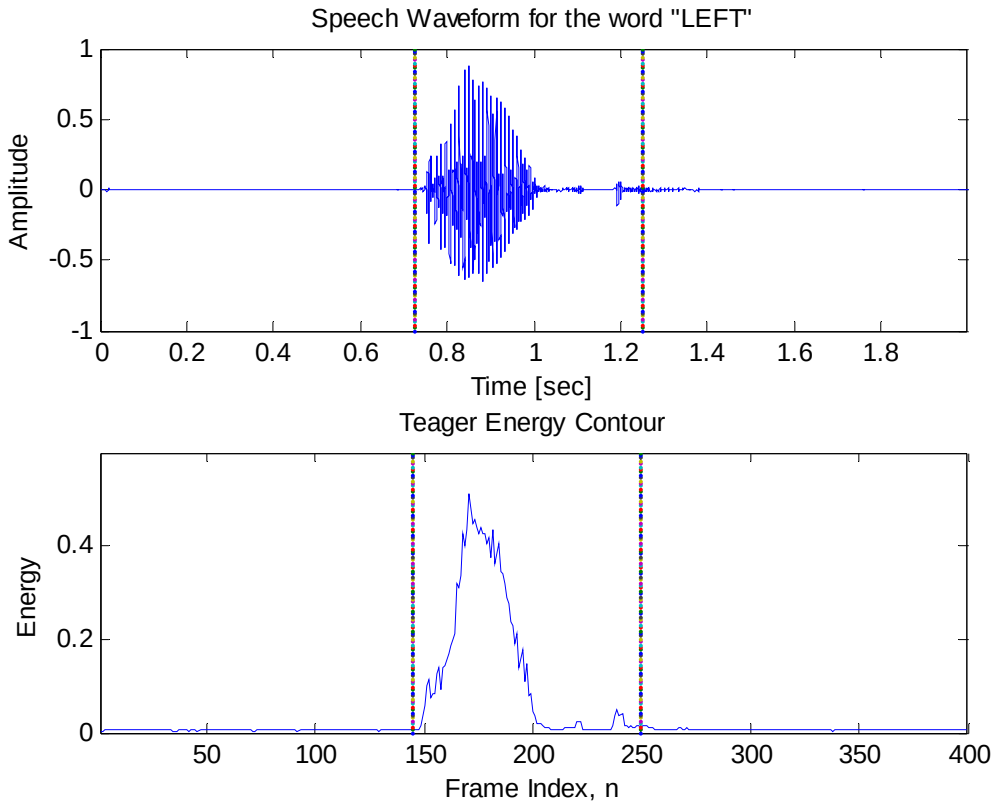

Figure 3.8. Speech waveform of one of the recordings of the word “left” (left_3_18_04_25_05.txt) (top plot), and Teager’s energy plot of the utterance (bottom plot). Vertical dotted lines indicate the end points of the utterance.

3. Short-Time Zero-Crossing Rate (ZCR)

The short-time zero-crossing rate (ZCR) quantity is another parameter that has been used frequently, together with the short-time energy, for end point detection since the 1970s. It is generally employed as a secondary parameter to refine the initial end point estimates that were obtained by using short-time energy parameter.

The short-time ZCR is defined as the number of times the successive samples of a speech sequence change sign per frame and given as:

$$Z(n) = \frac{1}{2}\sum_{m=1}^{N}\left|\operatorname{sgn}\left[x(m+1)\right] - \operatorname{sgn}\left[x(m)\right]\right|, \tag{3.9}$$

where

$$\operatorname{sgn}\left[x(m)\right] = \begin{cases} +1, & x(m) \geq 0 \\ -1, & x(m) < 0 \end{cases}. \tag{3.10}$$

The ZCR can give rough estimates of the frequency content of a speech signal, particularly in high signal-to-noise ratio environments. However, the ZCR is very sensitive to dc offset, low-frequency hum and any other type of noise that may be present in the recorded speech signal [Rabiner, Sambur, 1975], [Rabiner, Schafer, 1978]. The first two problems, i.e., dc offset and low-frequency hum, can be overcome prior to the end point detection by removing the mean of the speech and highpass (or bandpass) filtering the signal, but it is not realistic to remove all noise prior to the end point detection. Hence, the reliability and accuracy of the ZCR parameter decreases significantly as the signal-to-noise ratio decreases, and it really becomes impossible to obtain anything from the ZCR in low SNR environments.

The ZCR of a reasonably noise-free utterance, one of the recordings of the word “kill” (kill_3_20_04_25_05.txt) whose waveform and spectrogram were shown in Figure 2.7 in the previous chapter, is shown in Figure 3.9. Following Figure 3.9, the ZCR of a considerably noisy utterance, one of the recordings of the word “up” (up_1_1_04_13_05.txt), is shown in Figure 3.10. In each figure, the bandpass filtered speech waveforms are also plotted on top of the related ZCR plots. The vertical dotted lines on the plots indicate the utterance end points detected manually by listening to the utterance. The ZCR of the voiced speech segment is much lower and steady than the ZCR of the silence segments. Even in the low noise case, it is not easy to set a good threshold by looking at the ZCR of the silence sections and to determine the boundaries of the utterance, especially the end point of the utterance. Furthermore, it becomes almost impossible to say anything about the end points of the utterance from Figure 3.10 when noise is present, which illustrates how the ZCR parameter becomes unreliable in high-noise environments.

These two examples clearly show that the ZCR parameter was not a good choice for use as a secondary parameter to refine the utterance end points for the data under investigation. Thus, it was not used in our study.

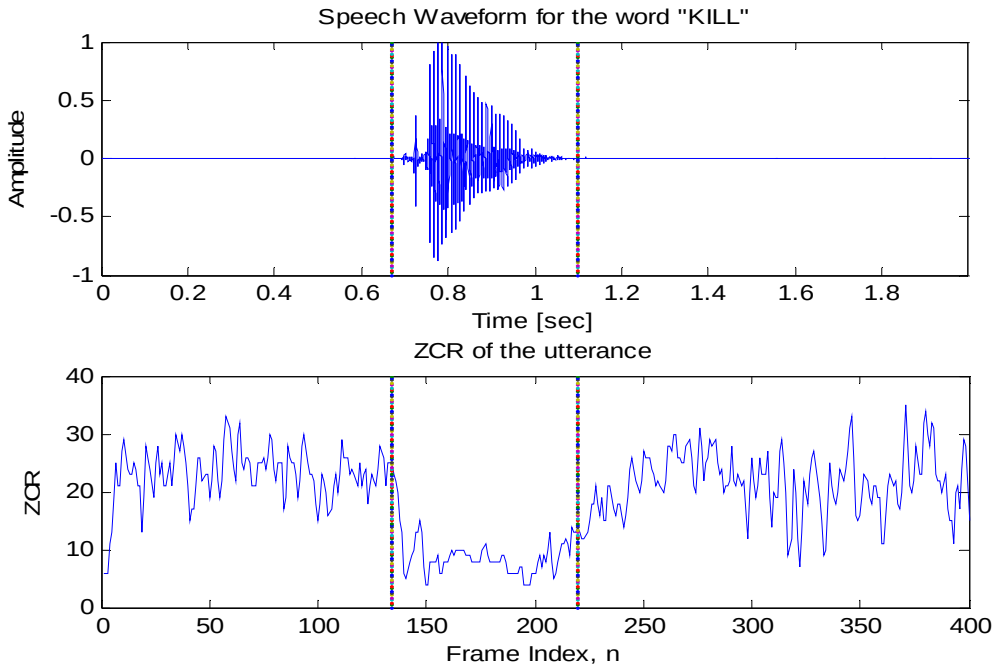

Figure 3.9. ZCR plot (bottom plot) of a noise-free utterance (kill_3_20_04_25_05.txt) (top plot). Vertical dotted lines on the plots indicate the actual end points of the utterance.

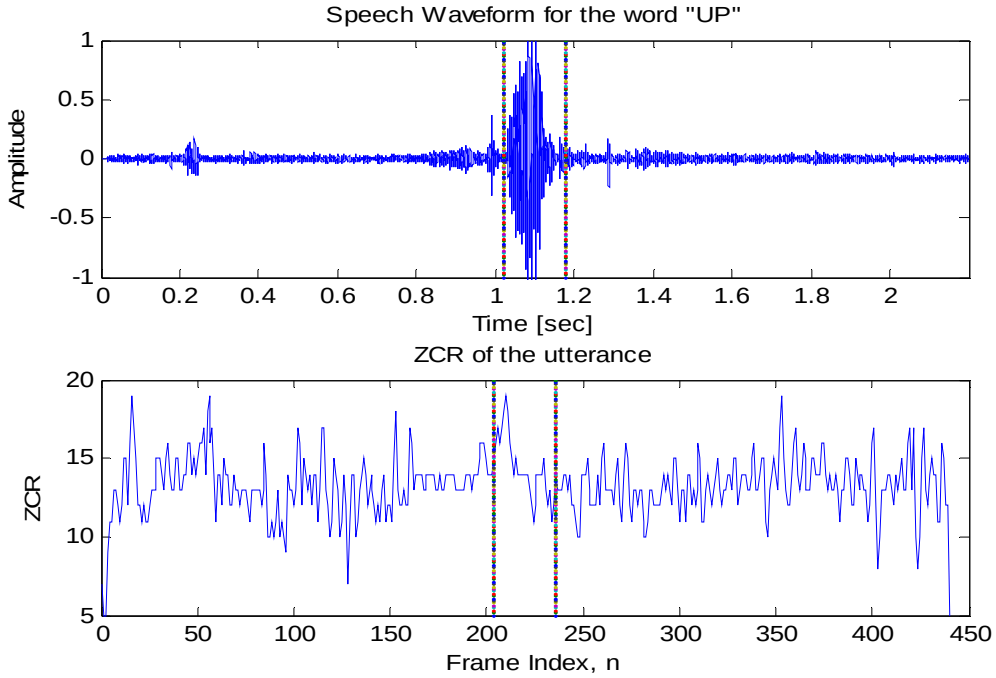

Figure 3.10. ZCR plot (bottom plot) of a noisy utterance (up_1_1_04_13_05.txt) (top plot). Vertical dotted lines on the plots indicate the actual end points of the utterance.

4. Energy-Entropy Feature

The entropy is the measure of uncertainty or the amount of unexpected information contained in a signal, and is extensively used in the fields of coding and information theory. It was first applied to the end point detection problem in speech recognition by [Shen, Hung, Lee, 1998]. The entropy is also referred to as the spectral entropy by [Shen, Hung, Lee, 1998] since all computations are carried out in the frequency domain. Although its usage in the end point detection is somewhat new, the entropy parameter has become a hot research topic in that area.

In order to obtain the entropy or the spectral entropy of a speech frame $x_n(m)$, first the fast-Fourier transform (FFT) of the frame, i.e., short-time FFT of the frame, is computed by using:

$$X_n(k) = \sum_{m=1}^{N} x_n(m) e^{-j2\pi km/N}, \quad \text{for } k = 1, 2, ..., N. \tag{3.11}$$

Then, the spectral energy of the frequency index k in each frame is estimated as:

$$S_n(k) = \left| X_n(k) \right|^2, \quad \text{for } k = 1, 2, ..., N/2. \tag{3.12}$$

The probability associated with each frequency index k, i.e., the probability density function (pdf) estimation for the spectrum, can be estimated by normalizing the spectral energies:

$$p_n(i) = \frac{S_n(i)}{\sum_{k=1}^{N/2} S_n(k)}, \quad \text{for } i = 1, 2, ..., N/2. \tag{3.13}$$

Next, additional constraints are used to enhance the discriminality of the pdf defined above [Shen, Hung, Lee, 1998], [Huang, Yang, 2000]:

$$S_n(k) = 0, \quad \text{if } f < 150 \text{ } Hz \text{ or } f > 2300 \text{ } Hz, \tag{3.14}$$

$$p_n(i) = 0, \quad \text{if } p_n(i) \geq 0.9. \tag{3.15}$$

Note that since a bandpass filter with a passband of $[150 \text{ } Hz, 2300 \text{ } Hz]$ was applied to the data to remove distortions present in all trials, as will be discussed later in

Section D.1, the frequency bands contribution outside $[150\ Hz\ ,\ 2300\ Hz]$ are eliminated.

Finally, the entropy or the spectral entropy of a speech frame is defined as:

$$H_n = -\sum_{i=1}^{N/2} p_n(i) \cdot \log_{10}\left[p_n(i) \right]. \tag{3.16}$$

The entropy curve of an utterance, one of the recordings of the word “right” (right_3_25_04_25_05.txt) whose waveform and spectrogram were shown earlier in Figure 2.9, is illustrated in Figure 3.11. The entropy of the utterance is computed on its bandpass filtered version, which is shown on the top plot of Figure 3.11. Vertical dotted lines on each plot indicate the actual beginning and end points of the utterance.

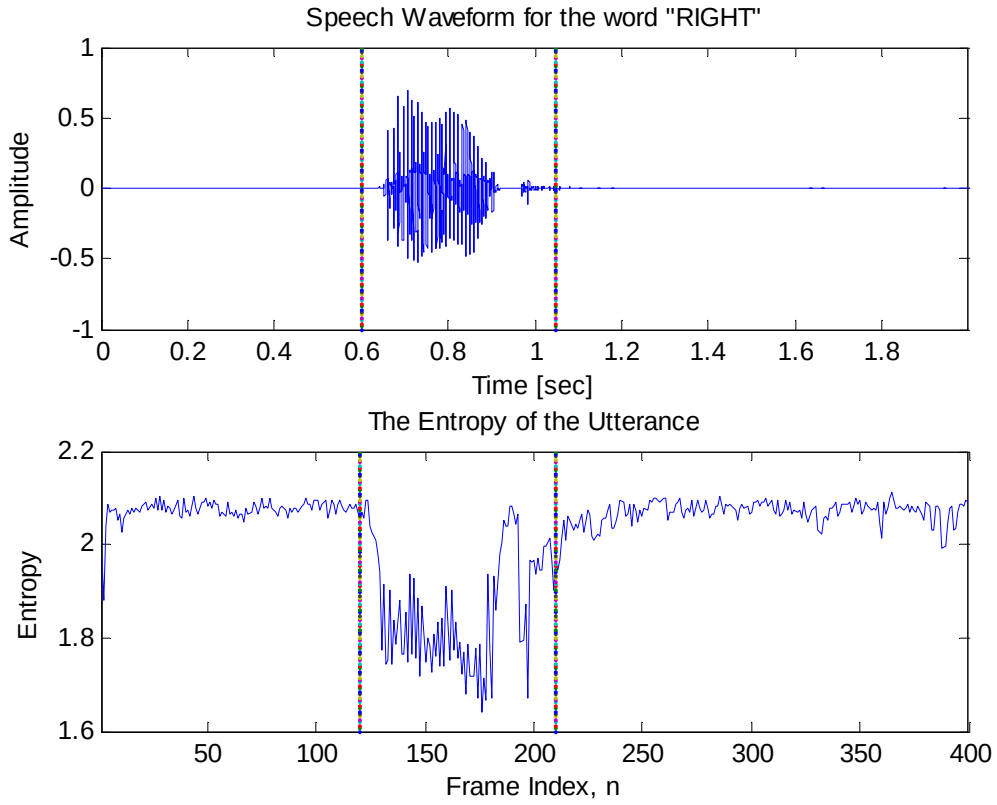

Figure 3.11. Waveform of one of the utterances of the word “right” (right_3_25_04_25_05.txt) (top plot), and corresponding entropy curve (bottom plot). Vertical dotted lines indicate the edge points of the utterance.

Results obtained by [Shen, Hung, Lee, 1998] revealed that the spectral entropy of a speech segment is quite different from that of a silence segment. Although the entropy parameter performs quite well on signals with low SNR contaminated with different types of noises such as white noise, pink noise, and gun shot [Shen, Hung, Lee, 1998], it is also known that the spectral entropy fails under multi-talker babble and background music [Huang, Yang, 2000]. However, the short-time energy quantity performs better in these noise conditions because of its additive property, which is the energy of the sum of speech plus noise always exceeds the energy of noise only [Wu, Wang, 2005]. Note that energy-based algorithms fail when the speech contains non-stationary noise such as mechanical or factory noise [Huang, Yang, 2000].

Both the short-time energy and spectral entropy have their own advantages and drawbacks depending on the noise conditions. Hence, both energy and entropy can be combined together to obtain a more reliable and robust feature, i.e., the energy-entropy feature (EEF), which takes advantage of the strengths of the two quantities while compensating for their limitations. The energy-entropy feature (EEF) of a frame, first defined by [Huang, Yang, 2000], is formed by simply multiplying the energy and the entropy computed for a specific frame.

The energy-entropy feature (EEF) is given as:

$$EEF_n = \sqrt{1 + \left| E_n \times H_n \right|}. \tag{3.17}$$

The multiplication operation defined above in Equation (3.17) emphasizes the voiced regions and attenuates the silence or noise regions. Doing so, the low-energy areas such as weak fricatives and stop consonants at the end or in front of an utterance can be emphasized over the attenuated noise regions and employed to refine the end point estimates.

Because of the reasons stated so far, the EEF was chosen as the secondary feature for the refinement of the end point estimates obtained by the short-time energy quantity. As it will be explained later in the chapter, first, the short-time magnitude energy is used to get initial estimates of the end points of an utterance, and afterwards, the EEF is used to refine the end point estimates wherever needed.

The energy-entropy (EEF) contour of the same utterance used in Figure 3.11 (right_3_25_04_25_05.txt) together with its absolute magnitude energy contour is shown in Figure 3.12. The vertical dotted lines on each plot indicate the actual end points of the utterance.

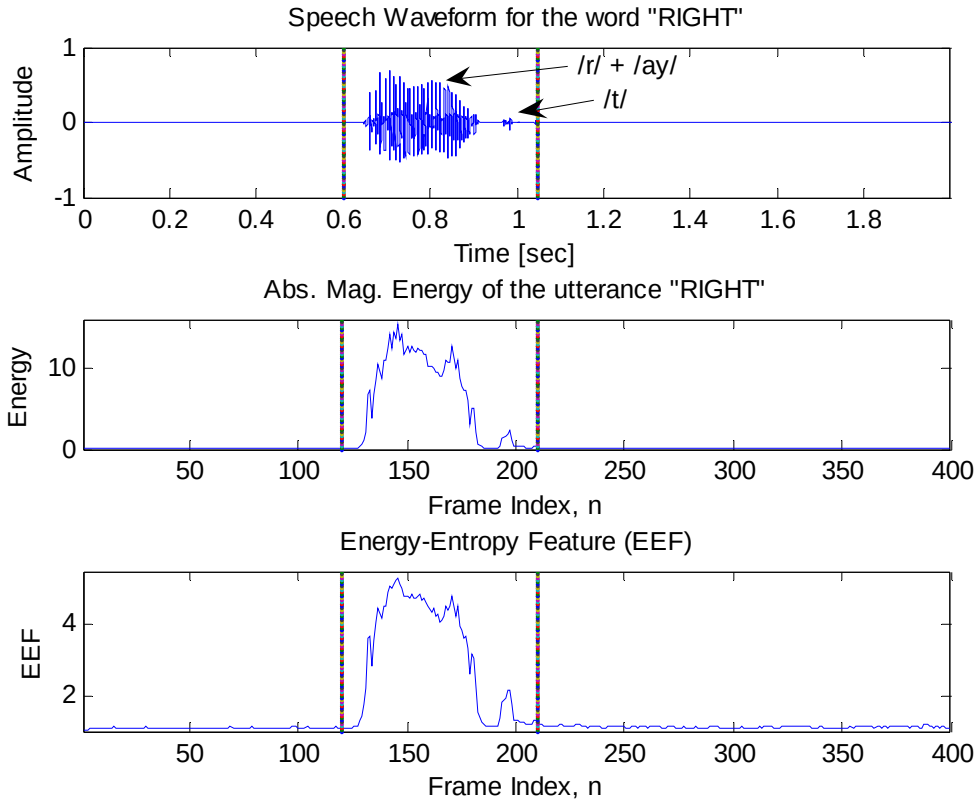

Figure 3.12. Absolute magnitude energy contour (top plot) and energy-entropy curve (bottom plot) of the waveform shown in Figure 3.11 (right_3_25_04_25_05.txt). Vertical dotted lines indicate the end points of the utterance.

Figure 3.12 shows that the low-energy stop consonant /t/ following the diphthong /ay/ is more emphasized and visually noticeable on the EEF plot than it is on the pure energy or entropy plots, since the amplitude difference between vowel portion and consonant portions is reduced. However, note that the background noise sections are also emphasized when compared to the energy contour of the utterance, which is mainly due to the multiplication operation that forms the EEF as explained earlier. The amplification of the silence sections may partially be avoided by the selection of a different short-time energy quantity for the EEF, namely the short-time squared energy quantity. Recall from

Section C.1 and Figure 3.6 that the short-time squared energy quantity provides greater attenuation of background noise than the short-time absolute magnitude energy quantity does due to the squaring operation present in its definition. Therefore, using the short-time squared energy for the EEF will yield an EEF plot with silence sections more attenuated than those obtained with the short-time absolute magnitude energy scheme. However, the short-time squared energy quantity was not used since the short-time absolute magnitude energy quantity was initially chosen for the end point detector implementation. Using a different energy parameter for the EEF would mean additional computation and reduced speed for the detection algorithm. Also, recall that threshold values for an algorithm which uses the squared energy with the entropy parameter would be set according to the silence sections. Therefore, there would be no improvement in implementation by using the short-time squared energy quantity rather than the short-time absolute magnitude energy quantity.

D. THE END POINT DETECTION ALGORITHM

This section presents the end point detection algorithm used for the present study. First, we discuss the filter used as the preprocessor before the end point detection algorithm. Next, framing issues such as frame type and size are presented. Then, the threshold mechanism employed in the algorithm is discussed. Finally, we present the overall end point detection algorithm.

1. Preprocessing Stage

As discussed in Chapter II, there is a low-frequency hum in the frequency interval $[0\ Hz\ ,\ 100\ Hz]$ present in all recordings, and the frequency content of the recorded utterances is mainly concentrated up to $2.3\ kHz$, specifically for recordings collected from the ear canal. In addition, some recordings exhibit a strong stationary noise around $2.4\ kHz$.

Thus, recordings were bandpass filtered to remove the low-frequency noise and stationary noise contributions mentioned while keeping the useful frequency contents.

The elliptical IIR bandpass filter selected has the following characteristics:

- elliptical IIR filter,
- 9^{th} order,

- its passband is from 150 *Hz* to 2.3 *kHz*,
- its stopband is $[0\ Hz\ ,\ 100\ Hz]$ and $[2.35\ kHz\ ,\ 4\ kHz]$,
- at least 50 *dB* attenuation in the stopband.

An IIR filter was preferred to an FIR filter because required specifications could be implemented with a low order IIR implementation. For comparison, a one-stage FIR filter with the same specifications as those of the IIR filter would require an order of around 300, which is not suitable for a real-world application. However, the IIR filter selection results in some end point detection accuracy issues, which will be discussed in the next section.

2. Framing

The speech signals, which are non-stationary and wideband in nature, are divided into frames in which the speech can be assumed to be stationary. A rectangular window is selected to form the speech frames since it weights all the samples in a frame equally, and is the easiest one to use in an on-line implementation. Actually, the choice of a particular window is not important because the spectral resolution is not an issue here, as the algorithm operates mainly on the time-domain information. The use of any other window rather than the rectangular one may complicate the computations and the frame shift considerations as the window weights every sample in a frame differently.

The frame length is selected to be 10 *ms*. Since the sampling frequency is 8 *kHz*, there are 80 samples per 10 *ms* long frame. The frame rate is 5 *ms*, i.e., 50% overlapping is used between frames. Overlapping allows capturing the dynamic changes from one frame to another and increases the accuracy of the end point decisions since the end point decisions are made on a frame basis.

Note that there is no actual multiplication performed to form the frames since the rectangular window is used for framing. A sliding window which takes 10 *ms* long frames or 80 samples per frame from the incoming speech signal is applied from left to right by overlapping the frames 50% each time. The whole recorded speech is not segmented into frames and not stored in a matrix prior to the detection because this would conflict with the use of an on-line implementation.

3. Threshold Mechanism

An upper and lower threshold are computed and supplied to the energy-based detection phase, as simulations showed that the algorithm often failed to detect the speech segment and the error rate was very high with one threshold only. Another advantage of using double thresholds is that the upper threshold speeds up the algorithm since the upper threshold is much higher than the maximum energy of most of the user- generated artifacts and sporadic noises. As a result, the algorithm does not spend much time on noisy sections. The lower threshold is used to refine end point estimates obtained with the upper threshold since the upper threshold is set more conservatively than the lower threshold to avoid sudden noise bursts that are strong in energy.

The background or silence energy is used to set specific upper and lower threshold values. For this purpose, the first 100 *ms* of a recording is assumed to be only silence or background noise, which corresponds to the first 20 overlapping frames. The average silence energy $E_{silence}$ is computed as the average of the short-time absolute magnitude energies of the first 20 frames. Upper and lower thresholds, denoted by ITU and ITL respectively, are then obtained by multiplying $E_{silence}$ with some scalar which is dependent on the value of $E_{silence}$. Therefore, an adaptive threshold mechanism based on the noise level is formed.

Note that it is not always true that the first 100 *ms* of a recording is purely silence because speech starts right at the beginning of the recording interval in a few occasions. In that case, $E_{silence}$ becomes excessively large, which in turn causes detection errors. In order to take these problem cases into account and prevent errors, the parameter $E_{silence}$ is reset to a predefined value of 0.5 when it is found to be higher than a predetermined value of 4.5, and thresholds are computed according to this reset value. These two predetermined values were selected after trial and error and adjusting the thresholds in this manner solved the problem of missing the entire utterance with the energy-based algorithm when speech was present in the first 20 frames.

There were also cases where the first 20 frames contained a strong user-generated noise or a sporadic noise which may be very different from the characteristics of the rest

of the silence sections. These types of noises do not represent the actual background noise or silence behavior in the rest of the recording. In addition, there were a few cases where the first 20 frames were empty due to a late start in recording. Hence, to obtain a better estimate of $E_{silence}$ that represents the actual silence sections by taking the above situations into account, the algorithm was designed to continue searching to the right until $E_{silence}$ dropped below a predefined value of 2.0, or until reaching a predetermined maximum number of frames, i.e., 70, whichever occurred first.

4. The End Point Detection Algorithm

The stages preparing the speech signals for the end point detection have been discussed so far. The actual details of the algorithm will be presented here.

The end point detection algorithm designed for this study uses two parameters presented earlier, namely the short-time absolute magnitude energy quantity, and the energy-entropy feature (EEF). The beginning and end points of an utterance are computed in two steps. First, the short-time absolute magnitude energy quantity is used to obtain the initial edge point estimates of the speech signal. Second, the refinement of the initial end point estimates is performed by the energy-entropy feature (EEF) when necessary. Then, end points are declared, and the speech segment of the recording corresponding to the section between these end points is cropped and forwarded to the feature extraction block prior to classification and recognition stages.

Some heuristic assumptions were made and incorporated into the overall detection algorithm to increase its reliability and robustness, especially for the short-time absolute magnitude energy step, which provides most of the information used by the detection algorithm. These so-called assumptions are the duration information and the maximum energy.

As mentioned earlier in the chapter, some noises present in the speech recordings, especially non-stationary noises and artifacts generated by the speakers, exceeded the upper threshold and were erroneously classified as speech. Even though most of these artifacts such as clicks, pops, and lip smacks have strong energy, they were generally short in duration, i.e., only a few frames long. In such cases, a duration count can help to not place the end points incorrectly. The duration count used in the detection algorithm is

10 frames, or 50 *ms*, and was selected empirically. Basically, the duration of the voiced section of the shortest word in the vocabulary, i.e., the word "up," is around 20 frames or 100 *ms* on average. However, using 20 frames as the duration count value directly resulted in lots of detection errors such as misses or false detections with the word "up."

The algorithm starts to count the number of frames that are above the upper threshold once the energy of a frame exceeds the upper threshold. Speech is declared to have started when the number of frames exceeding the upper threshold is 10 (or, 50 *ms* in duration).

Note that there were also a few cases where the algorithm failed to detect the presence of an utterance, or selected a noise segment as the desired speech despite the fact that the double threshold and the duration count were used. These errors were due to two reasons:

- Short but strong spikes, generally generated by the speaker, right at the beginning of the voiced region, especially for the word "up."
- Non-stationary high-energy noises such as background talks and mechanical noises which were long in duration.

We noted that the first reason mentioned above caused the energy based scheme to miss the entire utterance, especially with the word "up." To remedy this problem, as soon as the energy algorithm finished its run from left to right with a miss result, the energy contour was smoothed with a 5^{th} order median filter to push the energy level of the frames containing a spike below the upper threshold, where the filter order was selected as it yielded the best results among all filter orders investigated. At that point, the algorithm made one more pass through the energy contour to detect the speech segment.

The second reason mentioned above caused the energy algorithm to detect noise sections as speech. In order to minimize this problem, the detected section is smoothed with a 5^{th} order median filter, and the maximum energy of the detected section is compared to the maximum energy of the whole signal. Simulations showed that the maximum energy of the non-speech segment is always lower than that of the speech segment, specifically after the median filter is applied. In such a case, the algorithm

makes another pass over the smoothed previously computed energy contour to correct the initial end point detection results.

Although the energy algorithm performances significantly improved with these additional constraints, it still sometimes failed to detect the stop consonants, like /t/ that followed the weak fricative /f/ in the word “left,” or /t/ in the word “right.” This phenomenon was detected while checking and listening to the end point detected and cropped utterances after the energy-based scheme finished running on the data. Note that the final stop /t/ is separated from the vowel portion of the word “left” with the weak fricative /f/ which behaves like noise. As a result, the energy algorithm detected the vowel portion and excluded the fricatives and/or stop consonants at either end of an utterance. In such cases, the energy-entropy feature (EEF) was adapted into the detection algorithm as the secondary refinement step. The EEF step is applied after the energy algorithm returns its end point estimation.

The EEF uses two threshold values that were set according to the average energy-entropy value of the first 20 frames assumed to contain only background noise or silence. The first threshold is used only to refine the speech start point, whereas the second threshold is used only for the speech end point. Both thresholds are derived by multiplying the average energy-entropy value of the first 20 frames $EEF_{silence}$ with some fixed scalar values that were determined empirically. Since there are no fricatives at the beginning of the vocabulary words, and the energy based algorithm does an excellent job at the beginning point estimation, the EEF searches only 5 frames back from the initial beginning point for correction. Since most detection problems arose at the end points of the specific words, “left” and “right,” the EEF was set to search 70 frames to the right, starting from the initial end point estimation returned by the energy algorithm.

The overall start/end point detection process can be summarized as follows:

- From the threshold algorithm, obtain the upper and lower thresholds, ITU and ITL, and the initial frame number from which the algorithm will start the search.
- Compute the short-time absolute magnitude energy of the frame, E_n, and move to the next frame.
- Compare E_n with ITU.

- If $E_n \le ITU$, continue the search with the next frame.
- If $E_n > ITU$, then check the short-time absolute magnitude energies of the next 10 frames, i.e., the next $50\ ms$, (*duration count*). If all 10 frame energies are above ITU, declare the first frame where $E_n > ITU$ as the preliminary beginning point of the utterance $\overline{SP}$. Otherwise, continue the search.
- After the preliminary beginning point $\overline{SP}$ is found with ITU, refine it by searching back until the first point at which $E_n > ITL$ is found. Label it as the "speech start point", SP.
- Continue forward until $E_n < ITU$ to find the possible end point location.
- If $E_n < ITU$, then check the energy of the next 10 frames (i.e., the next $50\ ms$) in order to avoid a sudden energy dip. If the short-time energies of these 10 frames are not all above ITU, declare the first frame whose $E_n < ITU$ as the preliminary end point of the utterance $\overline{EP}$. Otherwise, continue the search.
- After the preliminary end point $\overline{EP}$ is determined with ITU, refine it by searching forward until the first point at which $E_n > ITL$ is found. Label it as the "speech end point", EP.
- Check the index of the speech start point. If it is found to be equal to the last frame index of the whole recording N, then the algorithm has completed a right-to-left run on the whole recording without finding any start and end point. Thus, a miss has occurred. If the speech start and end points are found to be the last frame index and zero respectively, then a miss has occurred.
- In the case of a miss, we smooth the short-time absolute magnitude energy curve with a 5^{th} order median filter, and conduct the same search procedure outlined above on the smoothed energy curve from right to left once more to find the speech end points, SP and EP.
- If there is no miss, then compute the energy of the rest of the recording, and smooth it with a 5^{th} order median filter.
- Find the maximum of the smoothed energy curve, E_{median}, of the whole recording $\max\left(E_{median}\right)$, and the maximum of the smoothed energy curve between the detected end points $\max\left[E_{median}\left(SP:EP\right)\right]$.
- If $\max\left[E_{median}\left(SP:EP\right)\right] = \max\left(E_{median}\right)$, then terminate the energy algorithm, and return the edge point estimates that will be used by the EEF algorithm to refine the estimates if necessary.

- Otherwise, perform the search procedure outlined above on E_{median} until $\max\left[E_{median}\left(SP:EP\right)\right] = \max\left(E_{median}\right)$. Terminate the energy algorithm, and return the edge point estimates for final refinement by the EEF algorithm.
- Call the EEF algorithm to refine the start/end points.
- Set the thresholds, $EETL_1$ for refining SP, and $EETL_2$ for refining EP, according to $EEF_{silence}$.
- Search five frames to the left starting from SP. If $EEF_n > EETL_1$ (n is between $SP-5$ and $SP-1$.), move SP to that point. Otherwise, do not change the initial start point SP. Declare the "final speech start point", FSP.
- Search 70 frames to the right starting from EP. If $EEF_n > EETL_2$ (n is between $EP+1$ and $EP+60$.), check the EEF of the next 10 frames (to take into account the duration of the stop consonant /t/ at the end of the words "left" and "right"), and move EP to the last frame at the far right where $EEF_n > EETL_2$. Otherwise, do not change the initial end point EP. Declare the "final speech end point", FEP.
- Complete the end point detection algorithm and crop the signal from the estimated start/end points, FSP and FEP.

The complete end point detection algorithm outlined above is shown in flowchart diagram format in Figures 3.13 through 3.16.

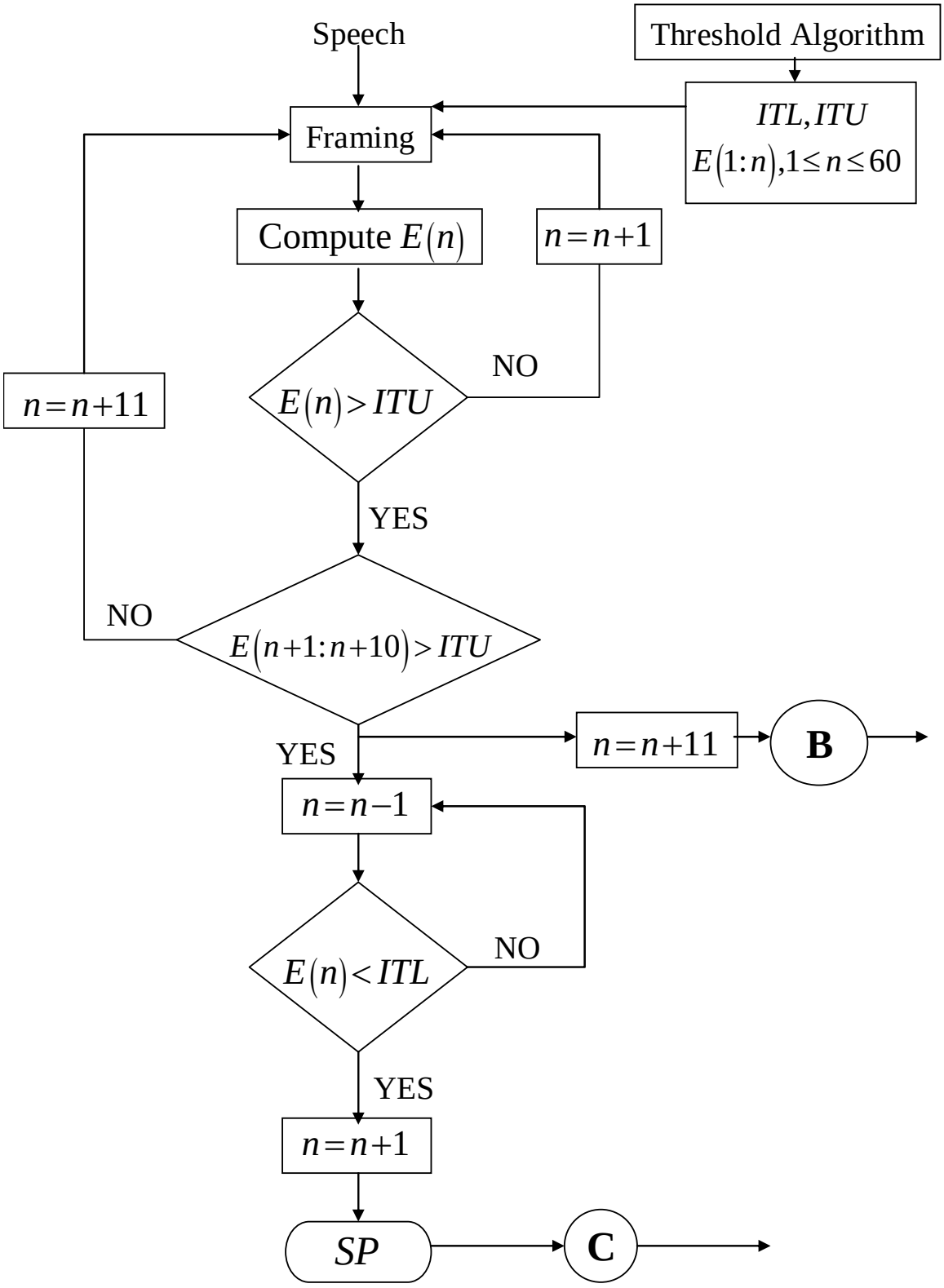

Figure 3.13. Flowchart for the initial speech start point estimation using the short-time absolute magnitude energy quantity.

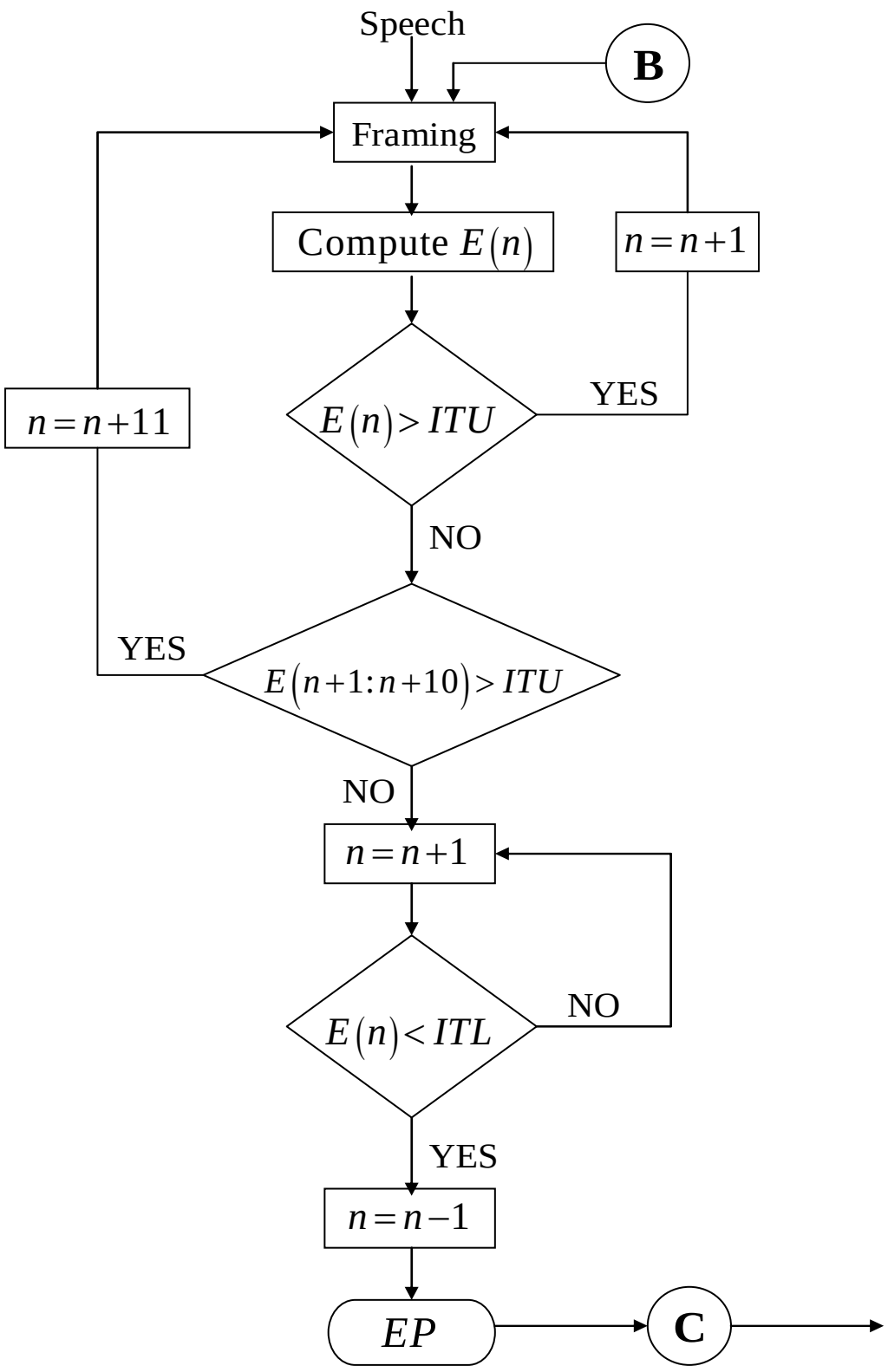

Figure 3.14. (Continued from Figure 3.13) Flowchart for the initial speech end point estimation using the short-time absolute magnitude energy quantity.

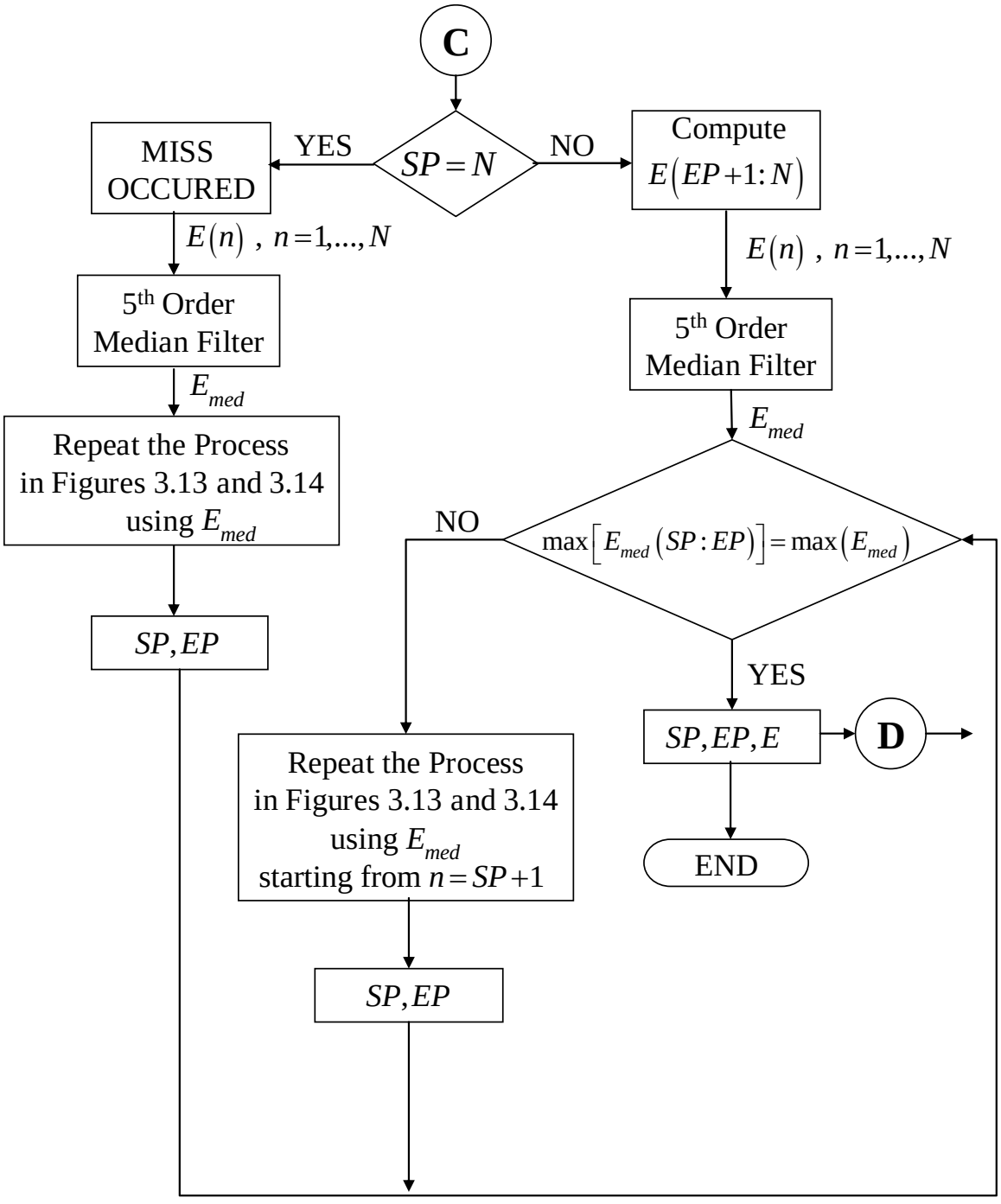

Figure 3.15. (Continued from Figures 3.13 and 3.14) Flowchart for the last part of the energy-based end point detection algorithm that addresses the miss and misdetection cases.

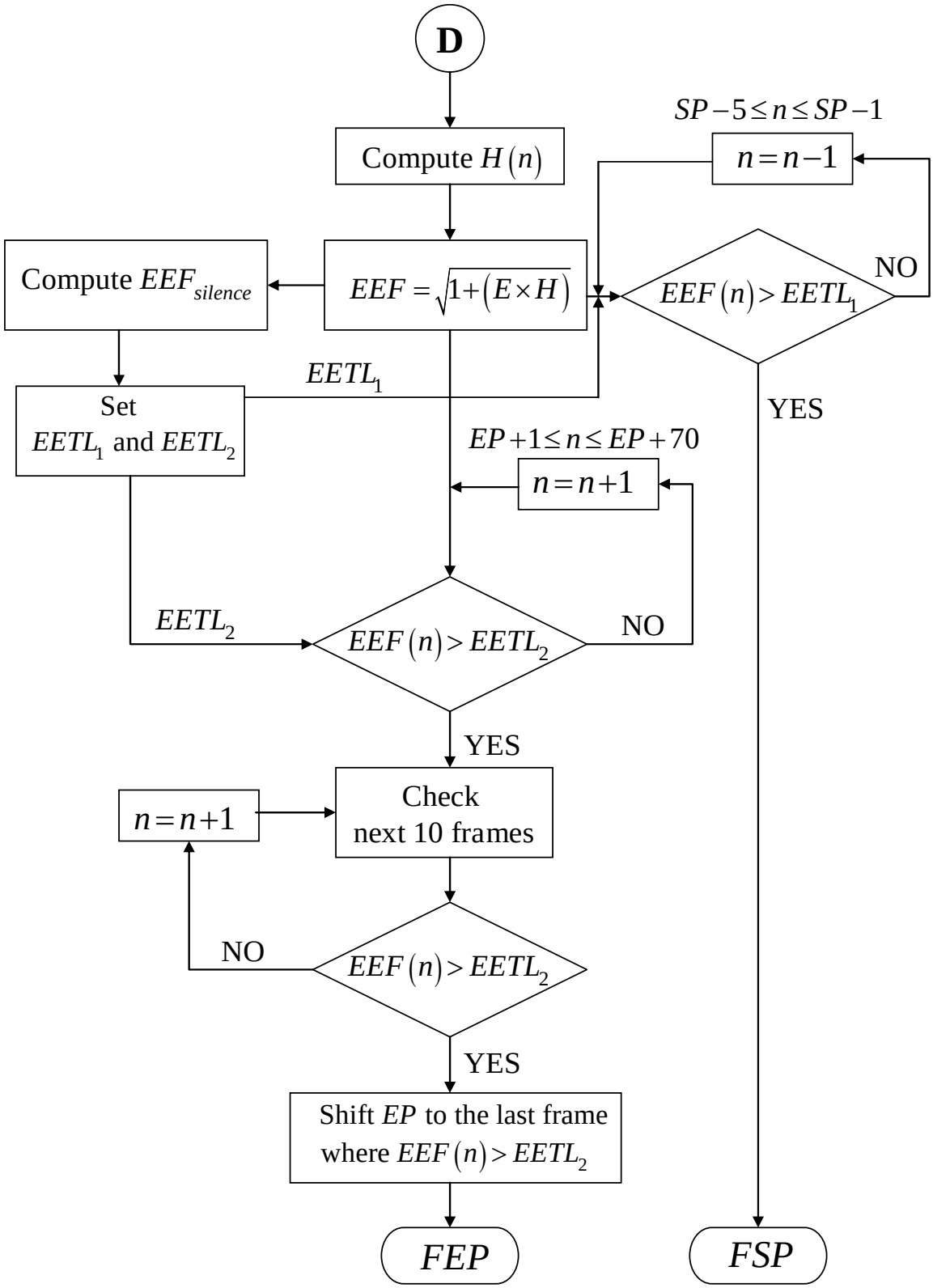

Figure 3.16. (Continued from Figures 3.13 through 3.15) Flowchart for determining the final speech edge points using the energy-entropy feature (EEF).

It may seem that the overall detection algorithm is computationally intensive and time-consuming, but the implementations conducted on a personal desktop PC with a Pentium Celeron (2.6 *GHz*) processor and 256 *MB* RAM took about 130 minutes for the algorithm to process all 8,228 utterances in the data set (which corresponds to about 1 sec per utterance).

The resulting detection algorithm is very accurate and robust in the sense that it does not return any misses, i.e., declare "no speech is present" incorrectly or select background noise as the desired utterance. Also, the audio and visual checks verified that the detected and cropped utterances closely matched manually cropped words.

E. THE EFFECT OF THE FILTER ON END POINT DETECTION

A 9^{th} order elliptical IIR filter was used for the end point detection, as explained in earlier sections. We selected this filter because it had a small number of coefficients and would be easy to implement in real-life applications. However, this particular filter also has some unwanted effects on some of the vocabulary words chosen for this study, which in turn may greatly affect the end point detection accuracies on these words.

These so-called unwanted effects appear particularly at the end of the voiced segments of an utterance when the utterance ends with a nasal or a breath release, such as for vocabulary words "down," "pan," "move," and "kill." These unwanted effects were especially noticeable when a subject releases his breath heavily after these words, or utters the final phone by lengthening it. As a result, these filter effects could become a major concern in implementation.

The IIR filter distorts the speech waveforms at the end of the voiced segments and amplifies the breath releases at the word ends. This in turn amplifies the absolute magnitude energy contour over the nasal or breath release sections right next to the voiced ones. This amplification of the energy contour causes the energy algorithm to move the end point of an utterance further right, which caused inaccurate end point estimations. In addition, it was not possible to increase threshold values to address this problem as doing so would have degraded the performance of the end point detection algorithm on other utterances, potentially missing the entire utterance or placing end points too conservatively.

An utterance of the word “pan” (pan_15_14_05_12_05.txt) is shown in Figure 3.17. Vertical dotted lines show actual beginning and end points of the utterance. Figure 3.18 plots the bandpass IIR filtered version of the utterance with its absolute magnitude energy contour underneath. Vertical dotted lines on the plots indicate the end point estimates returned by the end point detection algorithm applied to the IIR filtered speech. Note the amplification caused by the IIR filter is quite noticeable at the end sections on both the speech waveform and the energy contour. The detection error due to the IIR filter is 400 *ms*, which makes the use of an IIR filter a bit tricky.

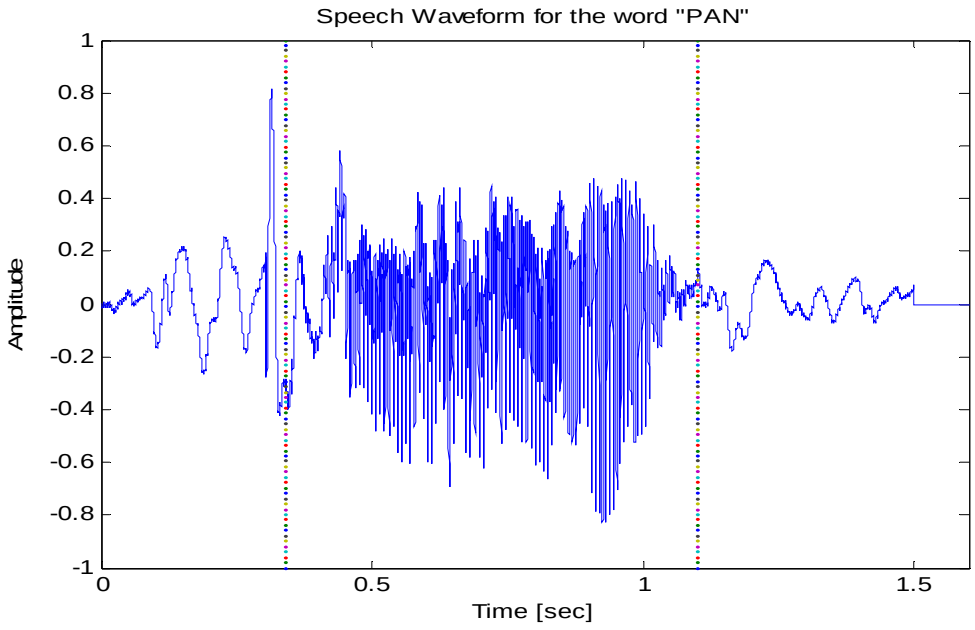

Figure 3.17. Speech waveform of one of the utterances of the word “pan” (pan_15_14_05_12_05.txt). The vertical dotted lines indicate the actual end points.

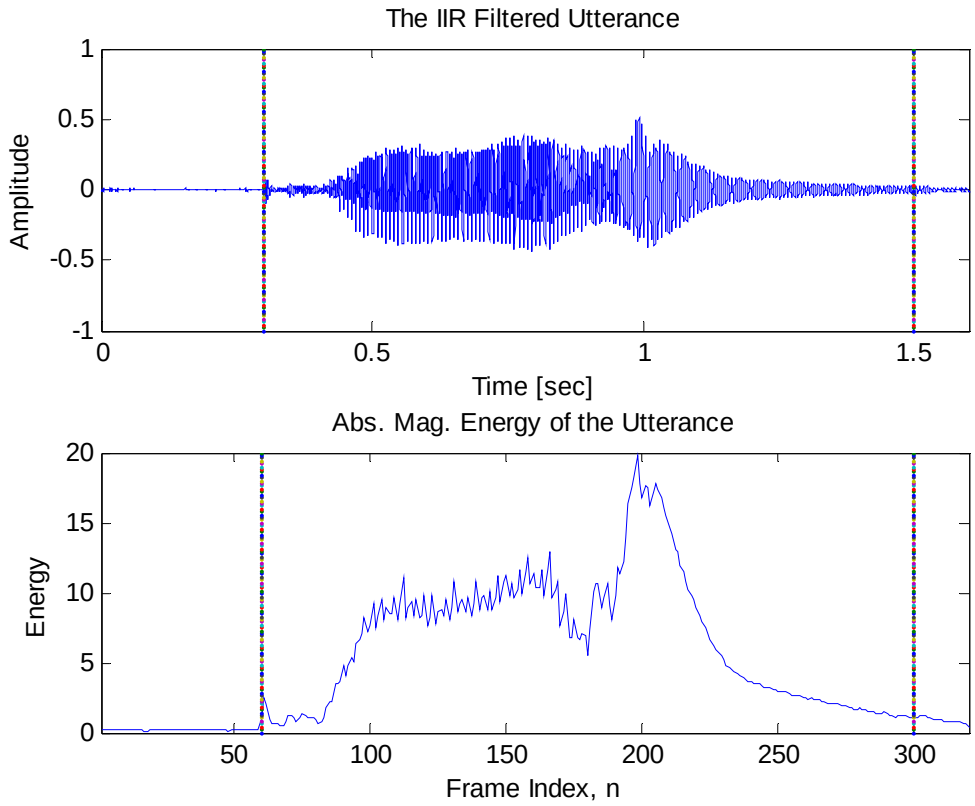

Figure 3.18. Speech waveform of the IIR filtered utterance (pan_15_14_05_12_05.txt) (top plot), and the corresponding absolute magnitude energy contour (bottom plot). Vertical dotted lines indicate the detected end points.

Note that a finite impulse response (FIR) filter does not cause the same amplification on both the speech signal and related energy curve as opposed to an IIR filter. The FIR filter selected to illustrate this phenomenon has the following characteristics:

- bandpass equiripple FIR filter,
- 300^{th} order (one stage implementation),
- passband is from $150\ Hz$ to $2.3\ kHz$,
- stopband is $[0\ Hz\ ,\ 100\ Hz]$ and $[2.35\ kHz\ ,\ 4\ kHz]$,
- at least $50\ dB$ attenuation in the stopband.

The FIR filter, therefore, has the same characteristics as the IIR filter discussed in Section D.1, except for the filter type and order.

The result of using a FIR filter on the utterance contained in Figure 3.17 (pan_15_14_05_12_05.txt) is illustrated in Figure 3.19, which plots the FIR filtered utterance and its absolute magnitude energy contour. Vertical dotted lines on the plots mark the end points detected by using the end point detection algorithm with the FIR filtered speech waveform. Results show that the end point estimates found by the end point detection algorithm using the FIR filtered speech are off by only 16 *ms* from actual speech boundaries. The unwanted effects caused by the IIR filter are not observed on either the time-domain waveform or energy contour of the FIR filtered speech. We also note that the separation between speech and silence segments is much clearer for the FIR filtered data, which makes the end point detection more reliable and accurate.

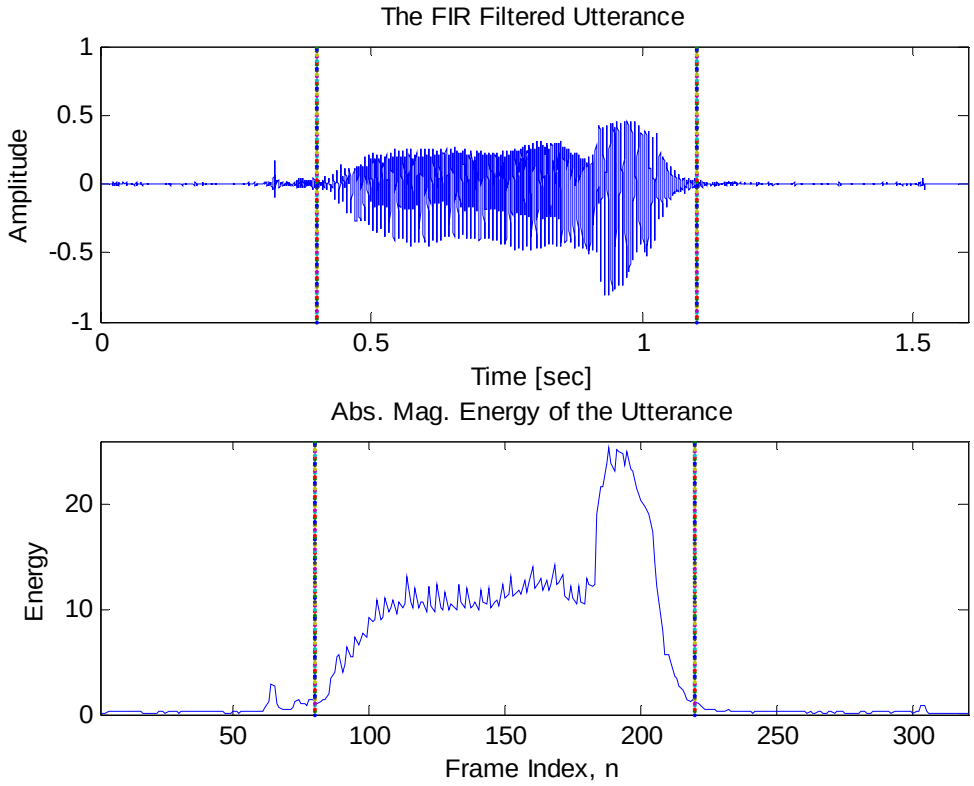

Figure 3.19. Speech waveform of the FIR filtered utterance (pan_15_14_05_12_05.txt) (top plot), and the corresponding absolute magnitude energy contour (bottom plot). Vertical dotted lines indicate the detected endpoints.

The main drawback of the FIR filter implementation is the high order required to match desired specifications, i.e., 300, which makes it unsuitable for a real-life

application. However, such a FIR filter could be implemented in stages through multi-rate signal processing techniques, which could reduce the overall complexity of the one-stage FIR filter [Mitra, 2006].

F. SUMMARY

This chapter discussed end point detection schemes, which are an essential part of any speech recognizer. Specifically, we presented the problems associated with end point detection, the parameters used for the detectors, and the specific detection algorithm ultimately used for the current recognizer.

The feature extraction stage of the recognizer, which follows the speech detector, will be presented in the next chapter.

IV. FEATURE EXTRACTION

This chapter presents the feature extraction block of the speech recognizer, which provides a compact representation of the segmented speech data to be used at the recognition stage. The chapter discusses two spectral features extensively used in speech recognition applications, real cepstrum (RC) coefficients and mel-frequency cepstral coefficients (MFCCs), which are also employed for the present study.

A. REAL CEPSTRUM

Speech production is usually modeled as the convolution of an excitation sequence $e(n)$ with the impulse response of the vocal tract, $h(n)$. Therefore, the speech $s(n)$ can be written as:

$$s(n) = e(n) * h(n). \tag{4.1}$$

In the frequency domain, Equation (4.1) is given as the multiplication of the discrete-time Fourier transforms (DTFT) of the two sequences:

$$S(\omega) = E(\omega) \cdot H(\omega). \tag{4.2}$$

The excitation sequence is considered to be a random noise sequence for unvoiced speech and a quasi-periodic impulse train with the pitch period for voiced speech, whereas the impulse response of the vocal tract is considered to be a short window. Therefore, the excitation sequence is viewed as the rapidly varying part of the speech, as opposed to the vocal tract filter which represents the slowly varying part of the speech.

In many speech applications, the separate estimation of the excitation sequence and the vocal tract model is required. Since the speech is produced by the convolution operation (or, multiplication in frequency), it is not possible to separate the speech into two sequences by well-known linear techniques. As applied first by Noll [Noll, 1967], the excitation and vocal tract model can be separated by using a nonlinear operator, namely, by taking the logarithm of the signal:

$$\log|S(\omega)| = \log|E(\omega)| + \log|H(\omega)|. \tag{4.3}$$

Cepstral analysis is motivated by the idea of separating these two components of the speech signal. It was first discovered and applied to seismic analysis by Bogert [Bogert, Healy, Tukey, 1963], but the cepstral analysis was extended and first applied to speech by Noll [Noll, 1967], [Deller, Proakis, Hansen, 1993]. The more general mathematical model which is called homomorphic (cepstral) signal processing was introduced by Oppenheim [Oppenheim, 1969]. The cepstrum derived from the homomorphic processing is usually known as the *complex cepstrum*, although its version used in practice is called the *real cepstrum* [Deller, Proakis, Hansen, 1993].

The main difference between the real cepstrum and the complex cepstrum is that the phase information about the speech signal is retained in the latter while it is discarded by the real cepstrum [Deller, Proakis, Hansen, 1993]. Even though the complex cepstrum might seem to be more attractive since it preserves more information than the real cepstrum does, it is not used much in practical applications unless it is desired to return back to the time domain [Gold, Morgan, 2000]. Real cepstrum coefficients are more commonly used in speech recognition applications as there is no return to the time domain once features are extracted.

The real cepstrum (RC) of a speech sequence $s(n)$ is defined as the inverse DTFT of the logarithm of the spectral (DTFT) magnitude:

$$c_s(n) = IDTFT\left\{\log\left|DTFT\left\{s(n)\right\}\right|\right\} = \frac{1}{2\pi}\int_{-\pi}^{\pi}\log\left|S(\omega)\right|e^{j\omega n}d\omega, \tag{4.4}$$

where the natural or base 10 logarithm is generally used in the definition. Since the speech sequence $s(n)$ is real-valued, its logarithmic spectral magnitude, $\log\left|S(\omega)\right|$, is real and even. Therefore, the real cepstrum $c_s(n)$ is real and even, i.e., the second half of the real cepstrum coefficients is redundant and repetitive of the first half. The computation process of the real cepstrum is shown as a block diagram in Figure 4.1.

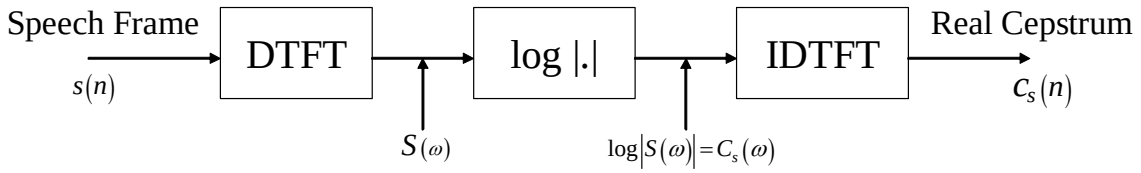

Figure 4.1. Computation of the real cepstrum using the DTFT (After: [Deller, Proakis, Hansen, 1993]).

In practical applications, the DTFT and IDTFT in the above definition are replaced by the discrete Fourier transform (DFT) and inverse discrete Fourier transform (IDFT), respectively. In this case, Equation (4.4) can be modified as:

$$c_d(n) = \frac{1}{N}\sum_{k=0}^{N-1} \log\left|S(k)\right| e^{j2\pi kn/N}, \quad \text{for} \quad n = 0,1,...,N-1. \tag{4.5}$$

Since the speech is time-varying, it is blocked into small frames where it is assumed to be stationary, thereby allowing to capture speech temporal and spectral dynamic changes. Hence, the above real cepstrum definition is also known as the *short-time real cepstrum* since it is applied to each individual frame.

The real cepstrum computation in Figure 4.1 is modified according to Equation (4.5) and shown in Figure 4.2.

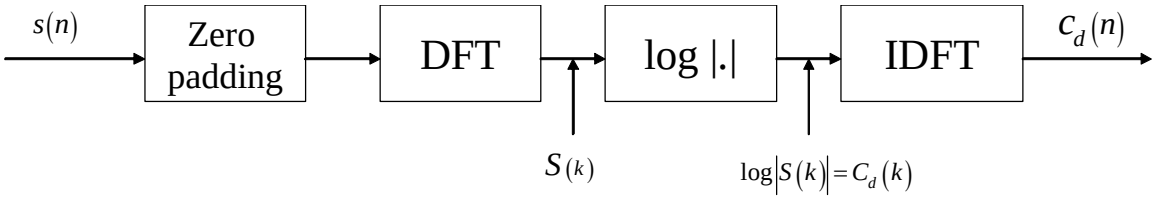

Figure 4.2. Computation of the real cepstrum using the DFT (After: [Deller, Proakis, Hansen, 1993]).

Using the DFT instead of the DTFT is similar to sampling the DTFT at N equally spaced frequencies from $-\pi$ to π. In this case, $c_d(n)$ in Equation (4.5) becomes the convolution of the original $c_s(n)$ in Equation (4.4) with a uniform impulse train of period N [Deng, O'Shaughnessy, 2003]:

$$c_d(n) = \sum_{i=-\infty}^{\infty} c_s(n + iN), \quad \text{for} \quad n = 0,1,...,N-1. \tag{4.6}$$

Therefore, the real cepstrum computed with the DFT and IDFT contains the replicas of the real cepstrum computed with the DTFT and the IDTFT at intervals of N. This in turn may potentially cause some aliasing, but can be minimized when the number of DFT points N is kept large enough, typically more than 100 points [Deng, O'Shaughnessy, 2003]. Thus avoiding aliasing can be achieved by either selecting longer frames or zero padding, as indicated in Figure 4.2.

For speech recognition purposes, the number of real cepstrum coefficients retained is generally less than 20 and typically 10 to 14. This ensures that the speech portion due to the vocal tract is kept while removing the contribution due to the excitation. Hence, for the current study, 14 real cepstrum coefficients are used for each speech segment cropped by the endpoint detection algorithm discussed in Chapter III.

The real cepstrum coefficients derived from one of the segmented utterances of the word "left" are plotted in Figure 4.3.

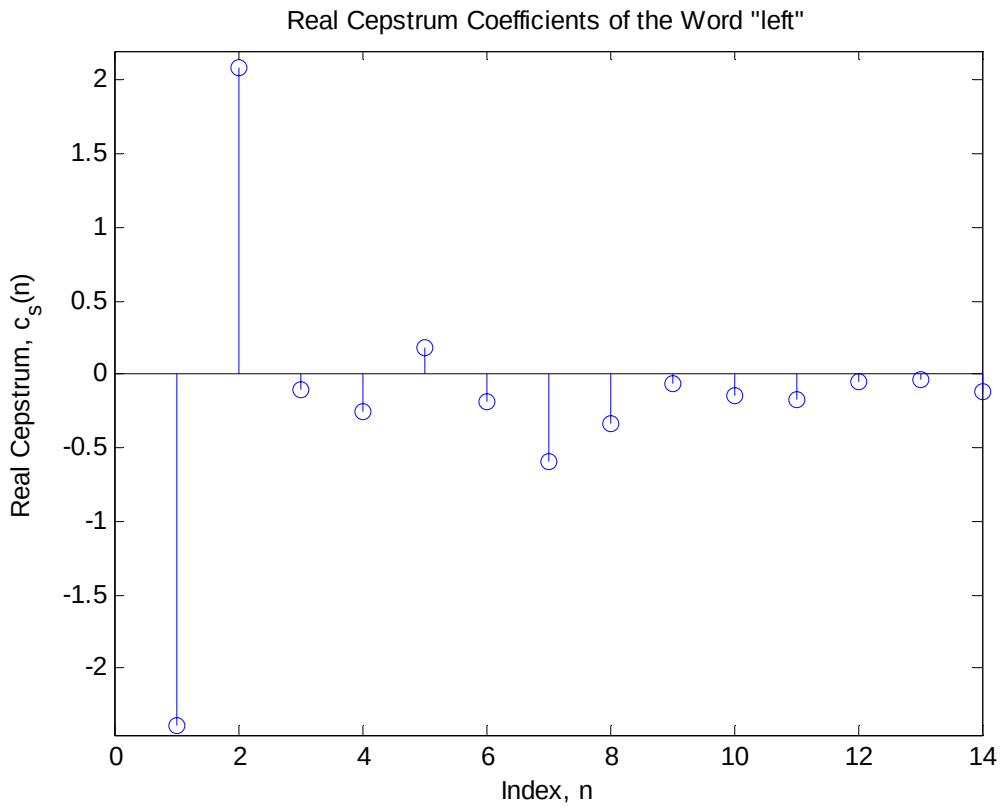

Figure 4.3. Real Cepstrum Coefficients of one of the segmented utterances of the word “left” (left_11_1_05_02_05.txt). Only the first 14 coefficients are plotted.

B. MEL-FREQUENCY CEPSTRAL COEFFICIENTS

The linear predictive coding (LPC) and real cepstrum coefficients were the major parameters used to represent utterances for speech recognizers up until the 1980s [Deller, Proakis, Hansen, 1993]. The *mel-frequency cepstral coefficients* (MFCC) were first used for a speech recognition system with a dynamic-time warping algorithm (DTW) in a study by Davis and Mermelstein in 1980 [Davis, Mermelstein, 1980]. Their study revealed the fact that MFCCs outperform any other parametric representation such as LPC and real cepstrum coefficients. MFCCs developed by Davis and Mermelstein [Davis, Mermelstein, 1980] have become the most popular features up to date.

The basic idea behind using MFCCs is to obtain a feature representation which approximates the human perception. MFCCs are the nonlinearly weighted and warped (with a nonlinear mapping scale) versions of the real cepstrum coefficients [Deng, O’Shaughnessy, 2003]. Basically, the logarithmic spectral magnitudes on the physical

frequency scale obtained by the real cepstrum are warped to corresponding magnitudes on a perceptual frequency scale since the perceived pitch or frequency of a tone is different than the physical frequency in *Hz* [Deller, Proakis, Hansen, 1993]. As a result, the *mel* scale allows for the required mapping (or warping) from the physical frequencies to the actual perceptual frequencies, as shown in Figure 4.4.

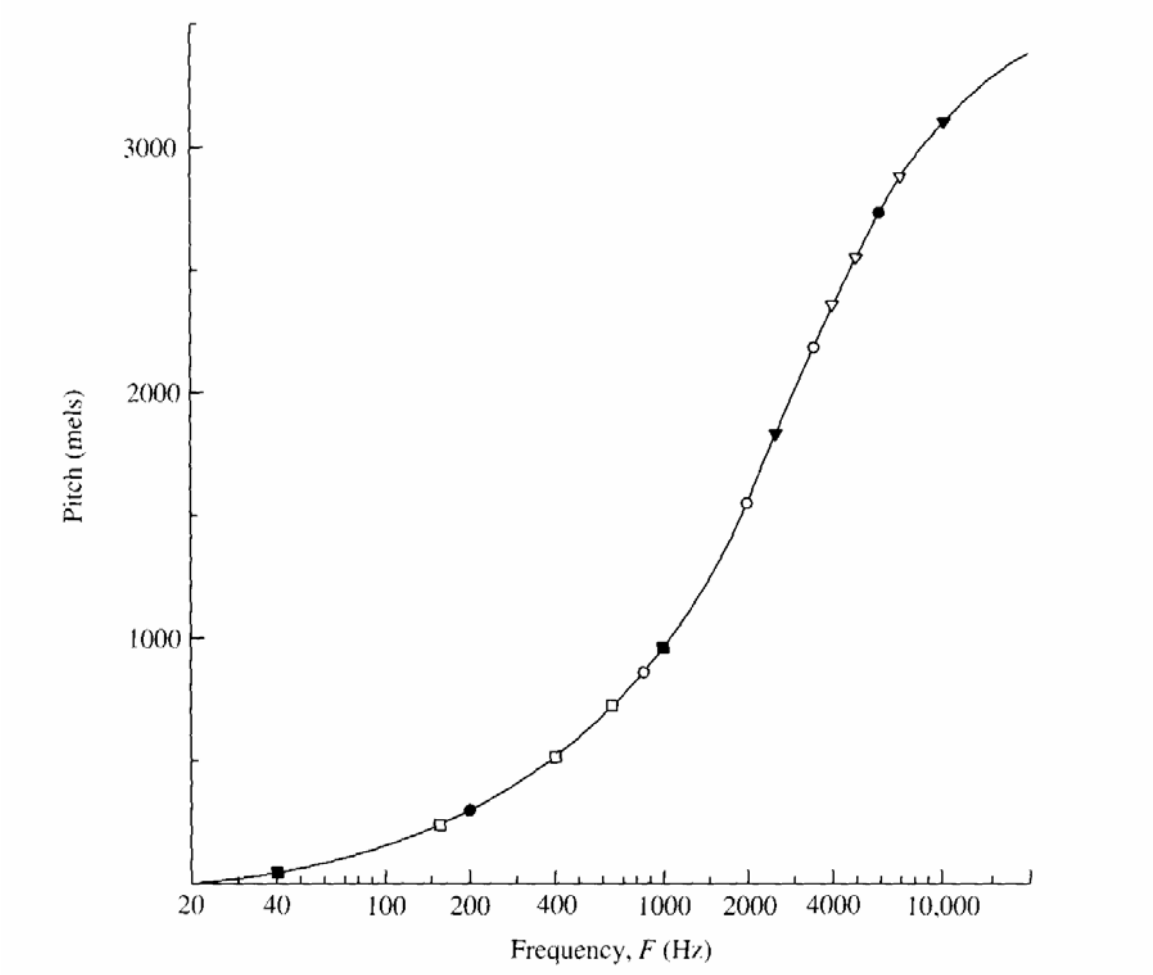

Figure 4.4. The mel scale (From: [Deller, Proakis, Hansen, 1993]).

Figure 4.4 shows that the mapping is linear below 1 *kHz* and is logarithmic above 1 *kHz*. The mapping is achieved by the formula [Picone, 1993]:

$$F_{mel} = 2595 \times \log_{10}\left(1 + f/700\right). \tag{4.7}$$

The MFCCs, however, are computed by using the human perceptual model instead of warping the real cepstrum with a mel scale as discussed above. The human

perception, or the operation of the basilar membrane in the inner ear, is generally modeled as a bank of 24 or so bandpass filters. Studies have shown that the basilar membrane is 32 *mm* long and that each bandwidth is about 1.5 *mm* in length along the basilar membrane; resulting in 24 bandpass filters (also called "critical band" filters) needed to model the basilar membrane [Deng, O'Shaughnessy, 2003]. The distribution of these critical band filters is also linear below 1 *kHz* and logarithmic above 1 *kHz*. Thus, their center frequencies and bandwidths follow the mel scale shown in Figure 4.4.

The critical band filters are conceptualized by a simple set of triangular windows (or bandpass filters), each centered on a critical band, as shown in Figure 4.5.

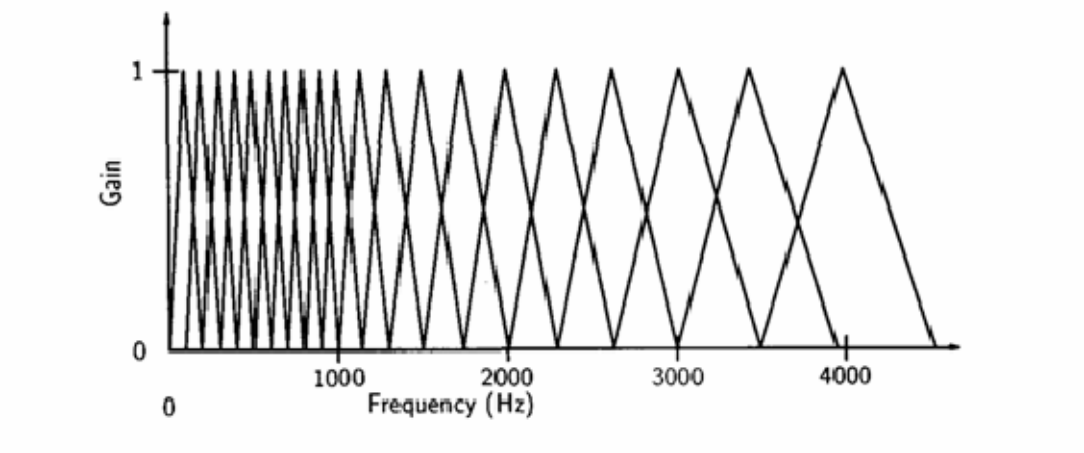

Figure 4.5. Conceptual triangular filters for extracting the MFCCs (From: [Deng, O'Shaughnessy, 2003]).

In practice, other types of filters can be applied to generate the MFCCs. However, the triangular filters are consistently used in speech recognition studies as they are especially easy to implement [Deller, Proakis, Hansen, 1993]. Thus, triangular filters were chosen to extract the MFCCs in this study.

Since the human auditory system responds to the energy in a critical band, the total logarithmic energy in a critical band is obtained through the conceptual filters, and the energy of each critical band is converted to corresponding MFCC via an IDFT. Generally, the final IDFT block is implemented with a discrete cosine transform (DCT) by replacing the complex exponential with a cosine since the logarithmic spectral energy is real and even, as explained in the previous section. The MFCC computational process

is presented next and the derivation follows closely that presented in [Vergin, O'Shaughnessy, Farhat, 1999].

First, the spectral magnitude (or energy) of a speech signal or a frame of the speech signal $s(n)$ is calculated as:

$$S_i = \left|S(k)\right|, \quad \text{for} \quad i = 0,1,...,N/2, \tag{4.8}$$

where $S(k)$ is the N-point DFT of the speech signal or a frame of the speech signal:

$$S(k) = \sum_{n=0}^{N-1} s(n) e^{j2\pi kn/N}, \quad \text{for} \quad k = 0,1,...,N-1. \tag{4.9}$$

The spectral energy, $\left|S(k)\right|^2$, can also be used in Equation (4.8) instead of the spectral magnitude.

Next, the energy in each critical band is obtained by applying the conceptual triangular windows shown in Figure 4.5 to the spectral magnitude in Equation (4.8):

$$E_j = \sum_{i=0}^{(N/2)-1} S_i \cdot h_j(i), \quad \text{for} \quad j = 1,...,J, \tag{4.10}$$

where J is the total number of triangular filters, $h_j(i)$, used.

Finally, MFCCs are calculated as:

$$c_s(n) = \sum_{j=1}^{J} \log_{10}\left(E_j\right) \cos\left[n\left(j+0.5\right)\frac{\pi}{J}\right], \tag{4.11}$$

where n is the number of MFCCs to be retained, generally 8 to 14 [Deng, O'Shaughnessy, 2003].

The computational process explained above is illustrated as a block diagram in Figure 4.6.

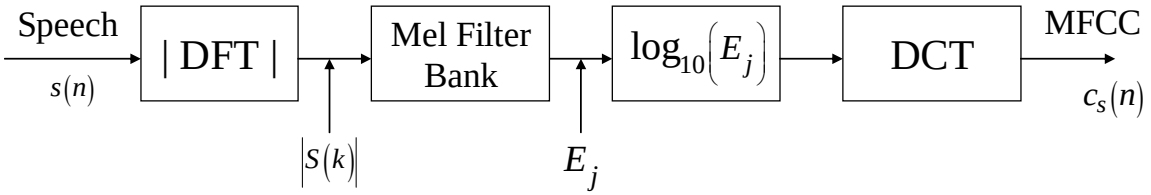

Figure 4.6. The MFCC computation as a block diagram (After: [Zhu, Alwan, 2003]).

The first coefficient $c_s(0)$ represents the average power in the speech signal. However, $c_s(0)$ is not often used in recognition applications since the average power varies considerably depending on the microphone placement and channel. The coefficients $c_s(n)$ give increasingly finer spectral details for each $n>1$ [Deng, O'Shaughnessy, 2003].

Fourteen MFCCs, excluding the first coefficient, are extracted for each segmented utterance and selected as feature vectors for the classification stage of the recognizer. The choice of fourteen MFCCs follows the above discussions, as the coefficients become negligibly small when the order, i.e., index number, increases. Even in the case of fourteen coefficients, the last few coefficients are much smaller in amplitude than the rest of the coefficients.

The MFCCs derived for one of the segmented utterances of the word "up" are illustrated in Figure 4.7.

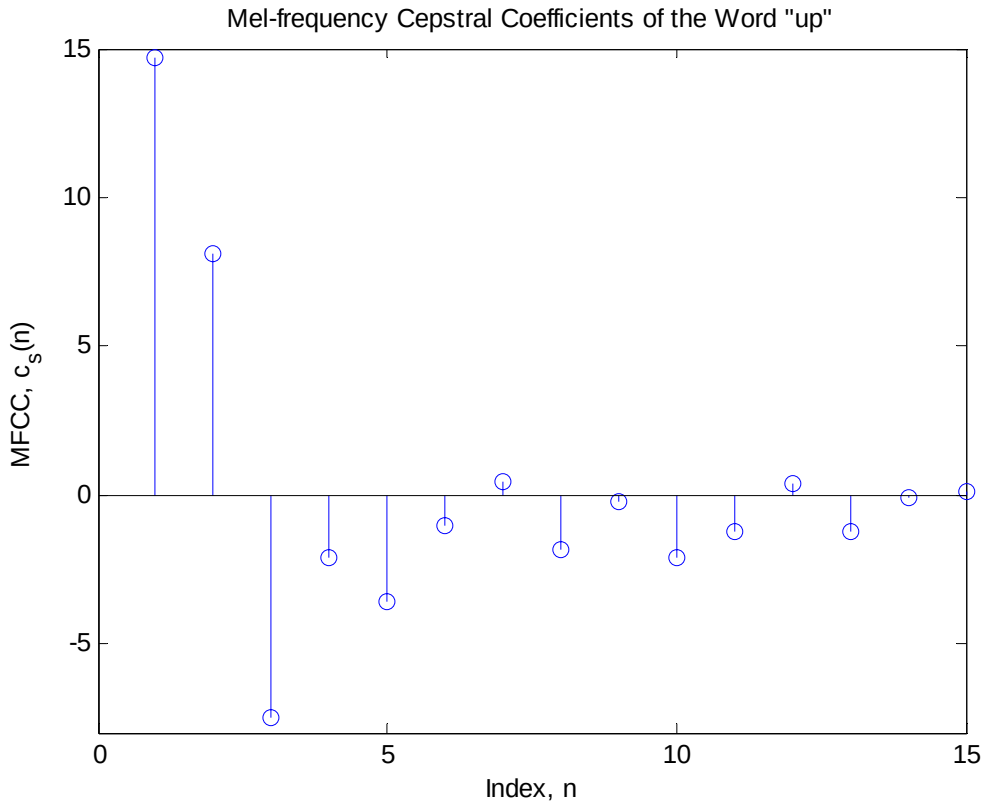

Figure 4.7. Mel-frequency Cepstral Coefficients (MFCC) of one of the segmented utterances of the word “up” (up_8_1_04_29_05.txt). 15 MFCCs, including the first coefficient, are plotted.

C. SUMMARY

This chapter presented the feature extraction section of the speech recognizer, which serves to represent the segmented utterances in both a useful and efficient way for the recognizer. Specifically, two extensively used spectral features that are also chosen for the current study were presented in detail, namely, the real cepstrum, and the mel-frequency cepstrum.

The next chapter will present the recognition stage and the recognition results obtained.

V. RECOGNITION RESULTS

This chapter describes the feedforward multi-layer neural network configuration used as speech recognizer for this study and the resulting recognition results. First, the chapter introduces the basic concepts behind artificial neural networks. Next, two learning algorithms commonly used in multi-layer neural networks (conjugate gradient and Levenberg-Marquardt algorithms) are presented. Third, we discuss implementation issues for the multi-layer neural networks used for the present study. Finally, we present recognition results.

A. INTRODUCTION

Artificial neural networks (ANNs) are inspired by the human nervous system. The human nervous system consists of approximately 10^{11} nerve cells, or neurons, each of which has 10^{4} connections with other neurons [Deller, Proakis, Hansen, 1992]. A simplified model of a biological neuron is shown in Figure 5.1.

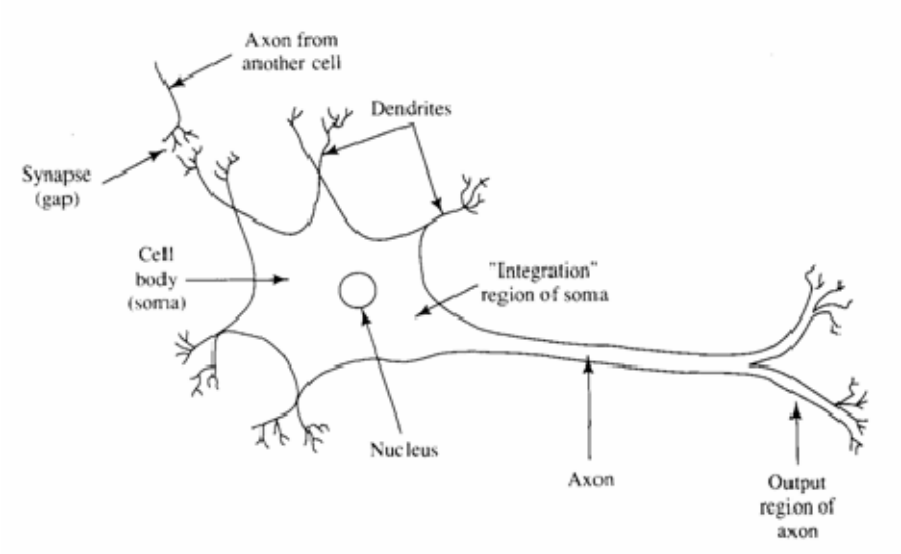

Figure 5.1. A simplified model of a biological neuron (From: [Deller, Proakis, Hansen, 1992]).

There are three main components in a nerve cell: the dendrites, the cell body (or the soma), and the axon. The dendrites are the receptive nerve fibers that carry the input signals into the cell body. The cell body sums and thresholds the received signals through

the dendrites. The axon is a long transmission line that carries the signals from one cell body to others. The synapse is the connection point between an axon of a cell and a dendrite of another. The nervous system is a highly parallel structure, which is a combination of the nerve cells described above [Hagan, Demuth, Beale, 1996].

ANNs, which are inspired by the biological neural system introduced above, are the simplified version of the complex human nervous system, although the exact mathematical behavior of the nervous system is unknown [Hagan, Demuth, Beale, 1996]. An artificial neuron accepts signals from other neurons or from its inputs, integrates or sums the incoming signals, and then the output is determined according to some sort of threshold function. A typical artificial neuron structure is illustrated in Figure 5.2.

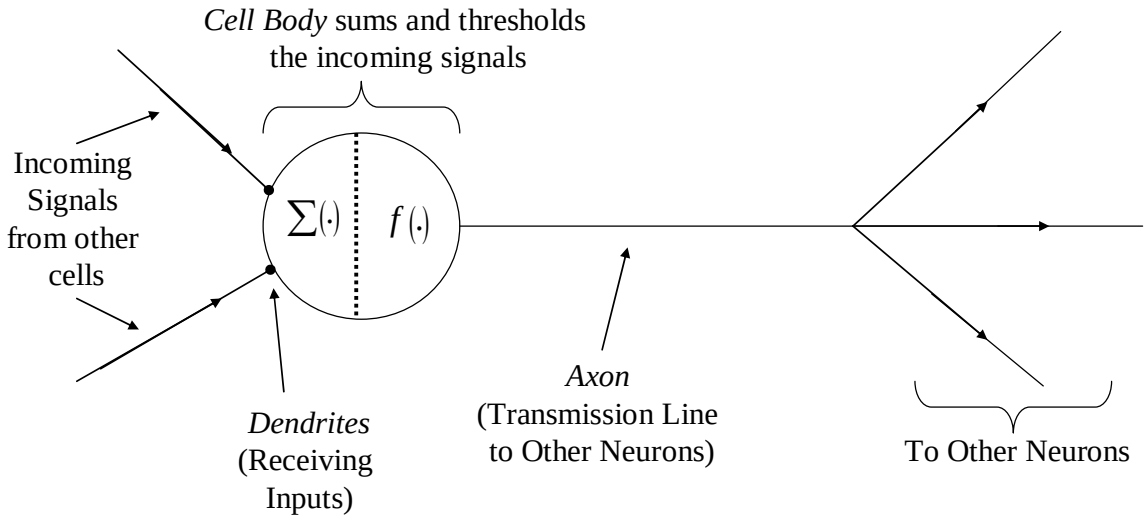

Figure 5.2. Artificial neuron model (After: [Deller, Proakis, Hansen, 1992]).

Therefore, an ANN is a network of these artificial neurons connected via some connections in a parallel structure. ANNs learn from examples shown to them, much like the way humans learn from the examples they see, which is called the training phase of the network. After learning from examples during training, they are capable of generalizing the learned examples to new examples that are not introduced during the training period, which is called the testing phase.

The first practical application of artificial neural networks dates back to late 1950s with the invention of the perceptron network by Rosenblatt [Rosenblatt, 1958]. The

perceptron was capable of performing pattern classification on two linearly separable classes. Artificial neural networks did not get much attention until the 1980s mainly due to the lack of powerful computers and processors needed to conduct the experiments combined with the limitations of the networks in classification, as well as the lack of powerful learning algorithms to train the networks for more complex problems. Interest in artificial neural networks increased dramatically with the advances in computing technology. New concepts introduced in the 1980s also contributed to this increase in research efforts on specific ANN types, namely, the recurrent network [Hopfield, 1982], and the backpropagation algorithm [Rumelhart, McClelland, 1986]. In particular, the discovery of the backpropagation algorithm allowed ANNs to perform complex pattern classification tasks on nonlinear data [Hagan, Demuth, Beale, 1996].

Artificial neural networks have found successful applications in a wide variety of fields for the last three decades. These fields include: aerospace, automotive, banking, defense, electronics, entertainment, financial, insurance, manufacturing, medical, oil and gas, robotics, speech, securities, telecommunications, and transportation [Hagan, Demuth, Beale, 1996]. There are many commercial implementations of artificial neural networks that are being used in the related fields mentioned above.

Since the speech recognition can basically be viewed as a pattern classification problem, and since the artificial neural networks are capable of performing complex classification tasks, ANNs have easily become a research tool for speech recognition purposes for the last two decades as an alternative to the hidden Markov model (HMM) that is the most common technique used for speech recognition. Particularly, some advantages of the ANNs which made them attractive for the speech recognition are their flexible architecture, highly parallel and regular structure, robustness to the limited training data, ability to accommodate discriminant learning, and no need to know the statistical distribution of the input features [Morgan, Bourlard, 1995].

ANN applications to speech recognition are basically divided into two broad categories; isolated word recognition and continuous speech recognition. The recognition task in the present study falls into the category of the isolated word recognition.

ANNs have not been quite successful in continuous speech recognition applications, and actually, it is not yet known how to implement a neural network based complete system for continuous speech recognition. This challenge is due to at least one fundamental problem with the training of the networks used for the continuous speech recognition: a target vector must be defined [Morgan, Bourlard, 1995].

Different types of ANNs, however, have been applied successfully to the phoneme, digit and isolated word recognition, and feedforward neural networks such as multi-layer networks with backpropagation algorithm, radial basis function (RBF) networks, probabilistic neural networks (PNN), and time-delay neural networks (TDNN) are architectures commonly used in these speech applications. A multi-layer neural network configuration with backpropagation algorithm is applied to the isolated word recognition problem of the present study. Next, we briefly introduce the basic idea behind multi-layer neural networks and the backpropagation algorithm.

B. MULTI-LAYER NEURAL NETWORKS

As indicated earlier, the artificial neuron (also called a node or a unit) is the smallest fundamental building block of an artificial neural network. It is a simple processing unit which tries to mimic the biological neuron.

In an artificial neural network model, incoming signals are multiplied by respective scalar *weights* (or *connections*) and passed to a summer which combines weighted inputs together with a scalar *bias*. The output of the summer forms the *net input*, which can also be viewed as the inner product of the inputs with the weights shifted by some bias (if any). The net input is then applied to an *activation function*, or a *transfer function*, whose output is the output of that specific neuron. The neuron output a is given by:

$$a = f\left(\underline{W} \cdot \underline{p} + b\right), \tag{5.1}$$

where $\underline{W}$ is the weight matrix, $\underline{p}$ is the input vector, b is the bias, and $f(.)$ is the activation function.

A single neuron model with multiple inputs is illustrated in Figure 5.3.

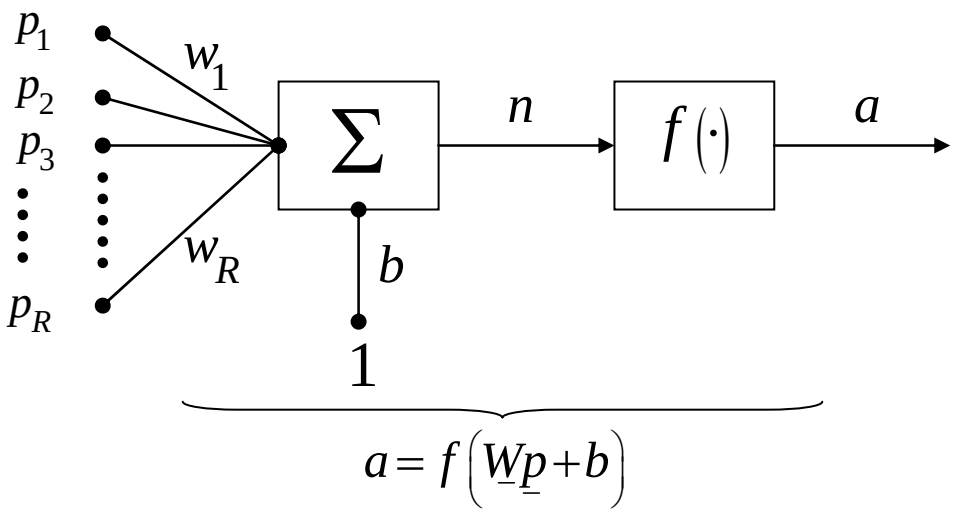

Figure 5.3. Mathematical model of a single artificial neuron with multiple inputs (After: [Hagan, Demuth, Beale, 1996]).

When the above model is related to the actual neuron model discussed in the introduction, the weights represent the strengths of the synapses, the summer and the activation function correspond to the cell body, and the net output corresponds to the signals on the axon [Hagan, Demuth, Beale, 1996]. Weights and biases are the values that have to be learned by using a learning function during training, and need to be stored for use afterwards. The bias, however, may or may not be used.

The activation function, or the transfer function, can be any type of function that fits the action desired from the respective neuron and is a design choice which depends on the specific problem. Some common types of activation functions are the hard-limit function (*hardlim*), linear function (*purelin*), log sigmoid function (*logsig*), and hyperbolic tangent sigmoid function (*tansig*). Log sigmoid and hyperbolic tangent sigmoid functions are commonly used in multi-layer neural networks with a backpropagation algorithm since they are differentiable and can form arbitrary nonlinear decision surfaces.

The collection of these artificial neurons forms the *layers* of a network. The neural networks can be classified as single-layer and multi-layer networks depending on the number of the layers. In a multi-layer structure, the last layer is called the *output layer*, while the rest are called *hidden layers* since their outputs do not have any connection with the outside environment.

Multi-layer networks are also referred to as feedforward networks since any feedback connection from the output layer to any of the hidden layers does not exist. In a feedforward multi-layer network, the output of a hidden layer effectively becomes an input to the next layer. A feedforward multi-layer neural network that has two layers is shown in Figure 5.4. The abbreviated matrix notation and network structure in Figure 5.4 will be used from now on throughout the chapter.

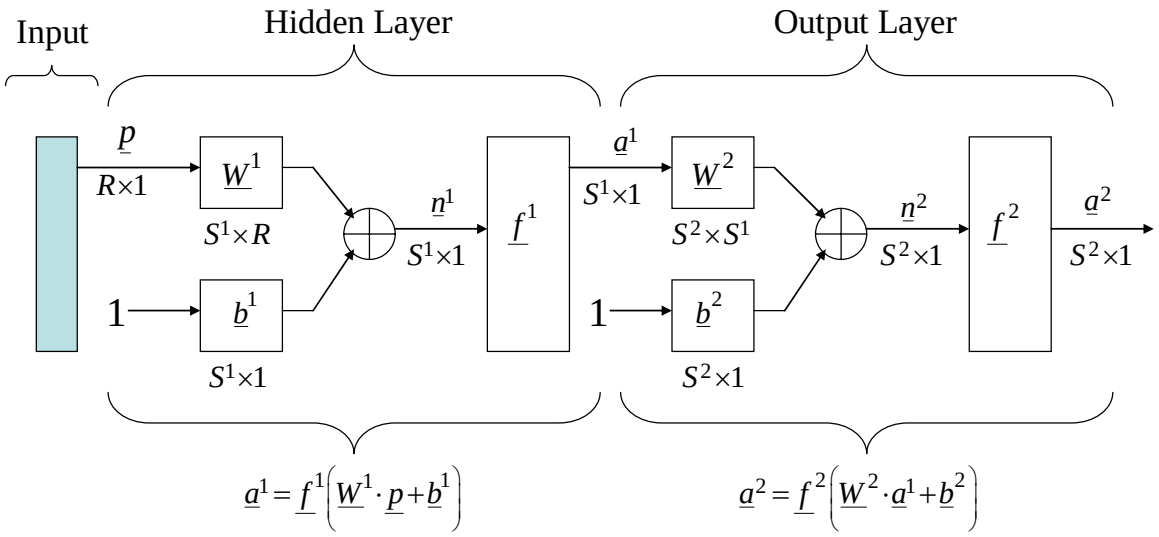

Figure 5.4. Feedforward two-layer neural network (After: [Hagan, Demuth, Beale, 1996]).

The number of layers, number of neurons in each layer, and type of activation functions per layer or per neuron in each layer are design parameters, i.e., the designer has to choose them according to the specific problem because there are no restrictions or no specific rules about the selection of these parameters. These parameters are generally set on a trial-and-error basis, which may be viewed as a drawback of the neural networks.

Although there is no specific rule on the number of layers to be used, it has been shown that two-layer networks with nonlinearities can, at least in theory, approximate any complex function given a sufficient number of neurons in the hidden layers. This property is the great computational power or expressive power of the multi-layer networks compared to the networks with no hidden layers [Duda, Hart, Stork, 2001].

C. THE BACKPROPAGATION ALGORITHM

The backpropagation algorithm is the most general and yet powerful method to train a single-layer or a multi-layer neural network. It is a generalization of the least mean square (LMS) algorithm used for linear networks, where the performance index is the mean square error (MSE) for both algorithms.

Basically, a training sequence is passed through the multi-layer network, the error between the target (desired) output and the actual output is computed, and the error is then propagated back through the hidden layers from the output to the input in order to update weights and biases in all layers.

Next, we present the algorithm. Notations and derivations below follow closely [Hagan, Demuth, Beale, 1996], and partially [Duda, Hart, Stork, 2001].

The feedforward operation of a multi-layer network can be defined as:

$$\underline{a}^{m+1} = f^{m+1}\left(\underline{W}^{m+1}\underline{a}^{m} + \underline{b}^{m+1}\right) \quad \text{for} \quad m = 0,1,...,M-1, \tag{5.2}$$

where M is the total number of layers in the network. This is simply the extension of Equation (5.1) to a multi-layer case.

As indicated earlier, the MSE is the performance index (or function) of the algorithm used to update the network parameters. The performance function for a multi-output case is given by:

$$F\left(\underline{x}\right) = E\left\{\underline{e}^{T}\cdot\underline{e}\right\} = E\left\{\left(\underline{t}-\underline{a}\right)^{T}\cdot\left(\underline{t}-\underline{a}\right)\right\}, \tag{5.3}$$

where $\underline{t}$ is the target vector, $\underline{a}$ is the actual output vector, $\underline{e}$ is the error vector (i.e., the difference between the target and actual outputs), and $\underline{x}$ is the network parameter vector containing weights and biases.

The expectation operation in Equation (5.3) can be approximated as:

$$\hat{F}\left(\underline{x}\right) = \left[\underline{t}\left(k\right)-\underline{a}\left(k\right)\right]^{T}\cdot\left[\underline{t}\left(k\right)-\underline{a}\left(k\right)\right] = \underline{e}^{T}\left(k\right)\cdot\underline{e}\left(k\right), \tag{5.4}$$

where k is the iteration number.

The weight and bias updates at iteration $(k+1)$ can be expressed in terms of the weights and biases at iteration k, and in terms of the performance function:

$$w_{i,j}^{m}(k+1) = w_{i,j}^{m}(k) - \alpha \frac{\partial \hat{F}}{\partial w_{i,j}^{m}}, \tag{5.5}$$

$$b_{i}^{m}(k+1) = b_{i}^{m}(k) - \alpha \frac{\partial \hat{F}}{\partial b_{i}^{m}}, \tag{5.6}$$

where $w_{i,j}^{m}$ is the weight associated with the j^{th} connection to the i^{th} neuron at layer m, and α is the *learning rate* that determines the amount of change to the weights and biases. The learning rate also determines how fast the algorithm will converge to the minimum point on the error surface while ensuring convergence.

Note that the last two equations are simply a re-statement of the well-known steepest descent algorithm, where the error is minimized by taking steps of α in the negative direction of the gradient of the performance function which causes a downhill movement on the performance function surface. Thus, the backpropagation algorithm is merely a gradient descent scheme.

The derivations up to this point are identical to the LMS algorithm, which is also a modified version of the steepest descent algorithm. Computing the partial derivatives in Equations (5.5) and (5.6) is the hardest part of the algorithm since the error does not explicitly depend on the weights and biases in the hidden layers. Therefore, the chain rule is used to compute these partial derivatives:

$$\frac{\partial \hat{F}}{\partial w_{i,j}^{m}} = \frac{\partial \hat{F}}{\partial n_{i}^{m}} \times \frac{\partial n_{i}^{m}}{\partial w_{i,j}^{m}}, \tag{5.7}$$

$$\frac{\partial \hat{F}}{\partial b_{i}^{m}} = \frac{\partial \hat{F}}{\partial n_{i}^{m}} \times \frac{\partial n_{i}^{m}}{\partial b_{i}^{m}}, \tag{5.8}$$

where n_{i}^{m} is the net input of the i^{th} neuron in layer m, and is given by:

$$n_{i}^{m} = \sum_{j=1}^{S^{m-1}} w_{i,j}^{m} \cdot a_{j}^{m-1} + b_{i}^{m}, \tag{5.9}$$

where S^{m-1} represents the total number of neurons in layer $m-1$.

Using the chain rule makes the error an explicit function of the weights and biases of layer m. Therefore, the partial derivatives of the performance function can be written as:

$$\frac{\partial \hat{F}}{\partial w_{i,j}^{m}} = s_i^m \cdot a_j^{m-1}, \tag{5.10}$$

$$\frac{\partial \hat{F}}{\partial b_i^{m}} = s_i^m, \tag{5.11}$$

where s_i^m is called the *sensitivity*, which describes how the error changes with the net input at layer m, and is given as [Hagan, Demuth, Beale, 1996]:

$$s_i^m = \frac{\partial \hat{F}}{\partial n_i^m}. \tag{5.12}$$

The weight and bias updates defined in Equations (5.5) and (5.6) become in matrix form:

$$\underline{W}^m(k+1) = \underline{W}^m(k) - \alpha \underline{s}^m \left(\underline{a}^{m-1}\right)^T, \tag{5.13}$$

$$\underline{b}^m(k+1) = \underline{b}^m(k) - \alpha \underline{s}^m. \tag{5.14}$$

The last two equations above are the weight and bias update equations that define the backpropagation algorithm.

The sensitivities in Equations (5.13) and (5.14) must be computed and propagated back through the network from the output nodes to the input nodes since the error is propagated back with the sensitivities, as will be apparent shortly. This is also obtained using the chain rule:

$$\underline{s}^m = \frac{\partial \hat{F}}{\partial \underline{n}^m} = \left(\frac{\partial \underline{n}^{m+1}}{\partial \underline{n}^m}\right)^T \frac{\partial \hat{F}}{\partial \underline{n}^{m+1}} = \dot{\underline{F}}^m\left(\underline{n}^m\right)\left(\underline{W}^{m+1}\right)^T \frac{\partial \hat{F}}{\partial \underline{n}^{m+1}} = \dot{\underline{F}}^m\left(\underline{n}^m\right)\left(\underline{W}^{m+1}\right)^T \underline{s}^{m+1}, \tag{5.15}$$

where the partial derivative of the net input vector at layer $m+1$ with respect to the net input vector at layer m is given by:

$$\frac{\partial \underline{n}^{m+1}}{\partial \underline{n}^{m}} = \underline{W}^{m+1}\dot{\underline{F}}^{m}\left(\underline{n}^{m}\right), \tag{5.16}$$

and the matrix that contains the partial derivatives of the activation functions at layer m is computed as:

$$\dot{\underline{F}}^{m}\left(\underline{n}^{m}\right) = \begin{bmatrix} \dot{f}^{m}\left(n_1^m\right) & 0 & \cdots & 0 \\ 0 & \dot{f}^{m}\left(n_2^m\right) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \dot{f}^{m}\left(n_{S^m}^m\right) \end{bmatrix}. \tag{5.17}$$

Equation (5.17) reveals an important property of the backpropagation algorithm: all the activation functions used for the network design must be *differentiable.*

The sensitivity of the output layer which starts the propagation of the network error back to the hidden layers can be computed by using Equation (5.15), and is defined as:

$$\underline{s}^{M} = -2\dot{\underline{F}}^{M}\left(\underline{n}^{M}\right)\underbrace{\left(\underline{t}-\underline{a}\right)}_{\underline{e}}. \tag{5.18}$$

The sensitivity at layer m is obtained from the sensitivity of layer $m+1$; hence, the name backpropagation. The last equation also reveals the fact that the sensitivities are actually a means of propagating the network error back to the hidden layers, and measure how each layer responds to the changes caused by the error. This phenomenon is illustrated in Figure 5.5, where each circle stands for a neuron, and each arrow indicates a connection between neurons of different layers. The weights associated with each connection are also shown in Figure 5.5.

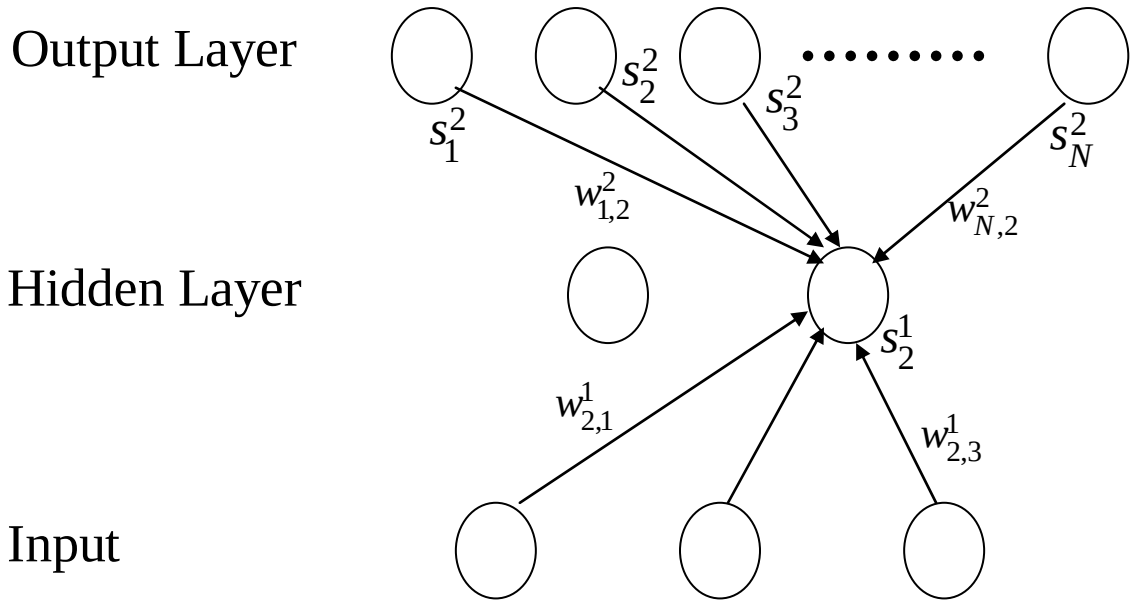

Figure 5.5. Backpropagation of sensitivities in a feedforward two-layer neural network (After: [Duda, Hart, Stork, 2001]).

There are a few issues that need to be addressed about the backpropagation algorithm. The very first issue deals with the convergence of the algorithm. There may be cases where the algorithm returns networks parameters which seem to minimize the MSE, but do not yield desired results or approximations. Since the error surface for multi-layer networks is very complex, there are multiple local minimum points in addition to the global minimum along the error surface. In practice, it is impossible to evaluate whether the algorithm converges to the global minimum or a local minimum. Furthermore, it is usually impossible to compute initial weight and bias values close to a minimum. Therefore, it is essential to iterate the algorithm multiple times over the whole training set with different initial weight and bias values.

Another issue to consider is the potential long learning time or large number of epochs needed to train the network. Thus, several variations of the basic algorithm have been proposed over the years and successfully used in most of the multi-layer networks with backpropagation to speed-up the algorithm training phase. These modifications and techniques include: adding a momentum term to the algorithm (backpropagation with momentum), variable learning rate, conjugate gradient descent, and Levenberg-Marquardt algorithm.

The conjugate gradient (CG) descent and Levenberg-Marquardt (LM) algorithms will be discussed in the following sections.

D. CONJUGATE GRADIENT ALGORITHM

The conjugate gradient (CG) algorithm is a numerical optimization technique designed to speed up the convergence of the backpropagation algorithm. It is in essence a line search technique along any set of conjugate directions, instead of along the negative gradient direction as is done in the steepest descent approach. The power of the CG algorithm comes from the fact that it avoids the calculation of the Hessian matrix or second order derivatives, which are required in the LM derivation, yet it still converges to the exact minimum of a quadratic function with n parameters in at most n steps [Hagan, Demuth, Beale, 1996].

The algorithm starts by selecting the negative gradient direction as the initial descent direction, i.e., the initial search direction. Next, the algorithm moves along the initial search direction until the local minimum in error is reached in that direction. At that point, the next search direction (i.e., the *conjugate direction*) is computed by selecting a direction orthogonal to the previous one and the following iteration selected as leading to the minimum value along that direction [Duda, Hart, Stork, 2001].

A detailed discussion of the steps taken to accomplish the algorithm will be presented next, where the notations and derivations follow closely those presented in [Hagan, Demuth, Beale, 1996].

The conjugate gradient algorithm starts by selecting the initial search direction as the negative of the gradient:

$$\underline{p}_0 = -\underline{g}_0, \tag{5.19}$$

and

$$\underline{g}_i = \nabla \underline{F}\left(\underline{x}\right)\Big|_{\underline{x}=\underline{x}_k}, \tag{5.20}$$

where $\underline{x}$ is the vector containing the weights and biases and $\underline{F}\left(\underline{x}\right)$ is the performance function, i.e., the mean square error (MSE).

The search directions $\underline{p}_i$ are called *conjugate* with respect to a positive definite Hessian matrix if and only if

$$\underline{p}_i^T \underline{A}\underline{p}_j = 0 \quad \text{for} \quad i \neq j, \tag{5.21}$$

where $\underline{A}$ represents the Hessian matrix $\nabla^2 \underline{F}(\underline{x})$.

The above condition can be modified to avoid the calculation of the Hessian matrix for practical purposes, and is given as:

$$\Delta\underline{g}_i^T \underline{p}_j = 0 \quad \text{for} \quad i \neq j. \tag{5.22}$$

The new weights and biases are computed by taking a step with respect to the learning rate α_i along the search direction that minimizes the error:

$$\underline{x}_{i+1} = \underline{x}_i + \alpha_i \underline{p}_i, \tag{5.23}$$

where the learning rate α_i for the current step is given by:

$$\alpha_i = -\frac{\underline{g}_i^T \underline{p}_i}{\underline{p}_i^T \underline{A}_i \underline{p}_i}. \tag{5.24}$$

Next, the new conjugate search direction is selected to continue the algorithm:

$$\underline{p}_{i+1} = -\underline{g}_{i+1} + \beta_{i+1}\underline{p}_i, \tag{5.25}$$

where the scalar β_i, which can be viewed as a momentum added to the algorithm [Duda, Hart, Beale, 2001], is given by one of three common choices (only *Fletcher and Reeves formula* is shown here since it is used for the current implementation):

$$\beta_i = \frac{\underline{g}_{i+1}^T \underline{g}_{i+1}}{\underline{g}_i^T \underline{g}_i}. \tag{5.26}$$

The algorithm iterates along successive conjugate directions until it converges to the minimum, or a predefined error criterion is achieved.

As is obvious from the above steps, the conjugate gradient algorithm requires a batch mode training, where weight and bias updates are applied after the whole training

set is passed through the network, since the gradient is computed as an average over the whole training set [Duda, Hart, Beale, 2001].

The conjugate gradient algorithm outlined above is guaranteed to converge to the minimum in n iterations if the performance function is quadratic with n parameters, as indicated earlier. The algorithm, however, may not converge to the minimum in n iterations if the network is a multi-layer network with many hidden neurons. This is due to the fact that the performance function is not quadratic for multi-layer networks, but may exhibit many local minima. Therefore, the conjugate gradient method was modified to be applied to multi-layer networks. The algorithm does not specify what to do if convergence is not reached after n iterations. As a result, one of the possible approaches to force the algorithm to continue in the case of multi-layer networks is to simply reset the search direction to the negative of the gradient after n iterations [Hagan, Demuth, Beale, 1996].

Although the conjugate gradient algorithm requires many computations to reach convergence, it is one of the fastest batch training algorithms, and has very useful properties, such as avoiding the computation and storage of second order derivatives, while preserving a quadratic convergence property [Hagan, Demuth, Beale, 1996].

E. LEVENBERG-MARQUARDT ALGORITHM

The Levenberg-Marquardt (LM) algorithm is a modified version of Newton's method which finds the minimum of a quadratic function in one iteration only by using the second order derivatives information. Newton's method approximates the performance function as a sum of squares, i.e., as a quadratic, which makes the LM algorithm suitable to the training of multi-layer networks as they have complex nonlinear performance surfaces.

The basic Newton's iteration can be written as:

$$\underline{x}_{k+1} = \underline{x}_k - \left[\nabla^2 \underline{F}(\underline{x})\Big|_{\underline{x}=\underline{x}_i}\right]^{-1} \times \left[\nabla \underline{F}(\underline{x})\right], \tag{5.27}$$

where the performance function $\underline{F}(\underline{x})$ is defined as [Hagan, Demuth, Beale, 1996]:

$$\underline{F}\left(\underline{x}\right) = \underline{v}^T\left(\underline{x}\right) \cdot \underline{v}\left(\underline{x}\right). \tag{5.28}$$

Newton's method requires the computation and storage of the second order derivatives, i.e., the Hessian matrix, as well as the inverse of the Hessian matrix. Newton's scheme is, therefore, expensive and not desirable in large real-life applications. The LM scheme modifies the original scheme given in Equation (5.27) by approximating the Hessian matrix with the *Jacobian matrix* that contains only first order derivatives.

The weight and bias update formula for the LM scheme can be defined as [Hagan, Demuth, Beale, 1996]:

$$\underline{x}_{i+1} = \underline{x}_i - \left[\underline{J}^T\left(\underline{x}_i\right) J\left(\underline{x}_i\right) + \mu_i \underline{\mathrm{I}}\right]^{-1} \underline{J}^T\left(\underline{x}_i\right) \underline{v}\left(\underline{x}_i\right), \tag{5.29}$$

where μ_i is a small scalar added to the Hessian matrix estimation in order to insure it is invertible, and $\underline{J}\left(\underline{x}\right)$ represents the Jacobian matrix containing the first order derivatives [Hagan, Demuth, Beale, 1996]:

$$\underline{J}\left(\underline{x}\right) = \begin{bmatrix} \dfrac{\partial \underline{v}_1\left(\underline{x}\right)}{\partial x_1} & \cdots & \dfrac{\partial \underline{v}_1\left(\underline{x}\right)}{\partial x_N} \\ \vdots & \vdots & \vdots \\ \dfrac{\partial \underline{v}_N\left(\underline{x}\right)}{\partial x_1} & \cdots & \dfrac{\partial \underline{v}_N\left(\underline{x}\right)}{\partial x_N} \end{bmatrix}. \tag{5.30}$$

The algorithm approaches the steepest descent algorithm with a small learning rate when μ_i increases, whereas it approaches Newton's scheme when μ_i decreases. The LM algorithm, therefore, combines the speed of Newton's scheme and the guaranteed convergence of the steepest descent [Hagan, Demuth, Beale, 1996].

The LM scheme is the fastest algorithm among the possible selection of algorithms for networks with moderate to small size parameters. However, the algorithm has two main drawbacks. First, the algorithm is computationally intensive as it requires more computations per iteration, including the matrix inversion, than other schemes such as the conjugate gradient. Second, it requires the storage of the Hessian matrix estimation, which is a $n \times n$ matrix, where n represents the number of network parameters (i.e., the weights and biases), whereas other schemes such as the conjugate

gradient requires only the storage of the gradient that is a n dimensional vector. As a result, it becomes impractical to use the LM scheme for large network configurations [Hagan, Demuth, Beale, 1996].

F. IMPLEMENTATION

Multi-layer neural networks with the backpropagation algorithm are used in this speech recognition study. Two different neural network structures were implemented: a two-layer feedforward neural network with one hidden and one output layer, and a three-layer feedforward neural network with two hidden layers and one output layer. Note that the two neural network structures considered in this study use same inputs, target assignments, activation functions, output layer structure, network parameters, and differ only in the number of layers and hidden neurons in the hidden layers.

For both network types, the output layer has seven neurons, each of which corresponds to one word in the vocabulary. Recall from Chapter II that the vocabulary chosen for this study has only seven words. For the classification purpose, each word is assumed to correspond to a class, and each word belonging to its respective class is labeled with an integer number from one to seven. More on the class and target assignments will be explained later.

The numbers of hidden neurons in the hidden layers were varied in each trial of two multi-layer networks in order to evaluate their performance. For the two-layer network, 50, 100, and 150 hidden neurons in the hidden layer were selected in the implementation. Therefore, the two-layer networks employed for this study are denoted as $(50-7)$, $(100-7)$, and $(150-7)$. Four different structures with different number of hidden neurons were implemented for the three-layer network structure: $(30-20-7)$, $(40-20-7)$, $(50-30-7)$, and $(60-40-7)$.

The hyperbolic tangent sigmoid function (*tansig*) was selected as the activation function for the hidden neurons, as it provides the necessary nonlinearities in the network to solve the classification problem. The log sigmoid function (*logsig*) was used for the neurons at the output layer in order to restrict the network outputs to the interval $[\,0\,,\,1\,]$.

Recall from Chapter IV that both real cepstrum (RC) coefficients and mel-frequency cepstral coefficients (MFCCs) were extracted from each segmented utterance to be used as feature vectors. The MFCCs are the primary features used as the input vectors that represent the segmented utterances for all the multi-layer networks implemented. In addition, the RC coefficients are used with one of each of the two-layer and three-layer configurations, namely the $(150-7)$ and $(60-40-7)$ networks for comparison purposes. As explained in Chapter IV, each segmented speech is represented with 14 spectral coefficients, either MFCC or RC coefficients. Therefore, the network input vectors which represent the segmented utterances are 14 dimensional, i.e., 14×1 column vectors. Finally, inputs were preprocessed to have zero mean and unit variance before being fed into the network to enhance the network performance.

The architectures of the multi-layer neural networks implemented for the word recognition are shown in Figure 5.6 and Figure 5.7, which illustrate the (150-7) and (60-40-7) configurations.

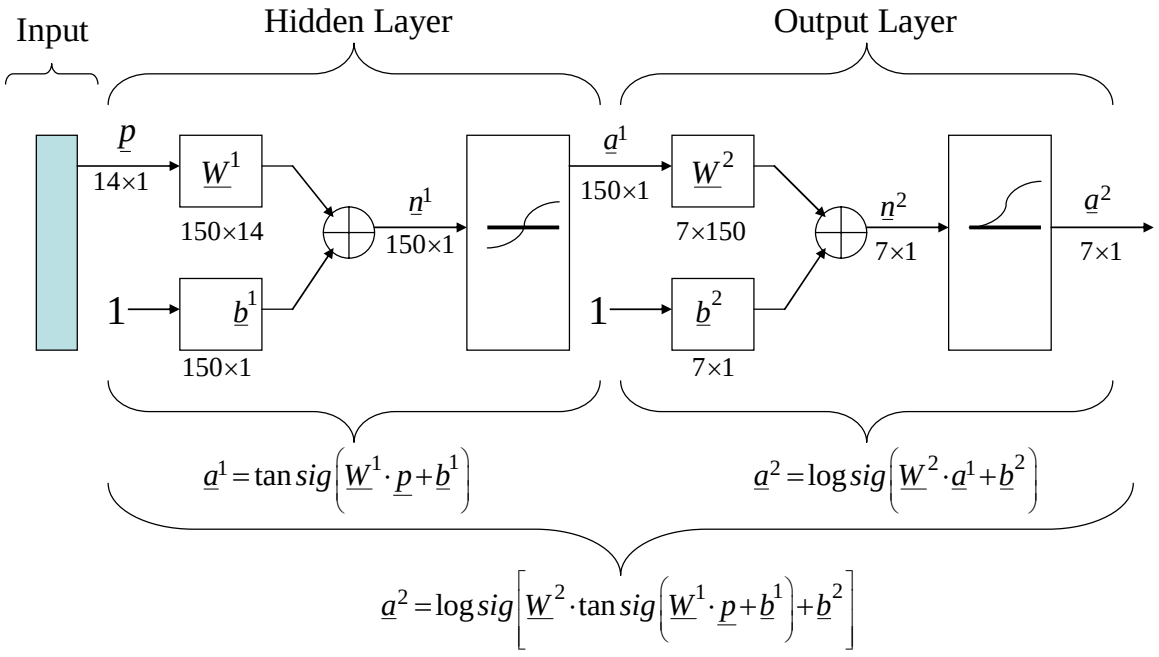

Figure 5.6. Two-layer feedforward neural network architecture implemented; $(150-7)$ configuration.

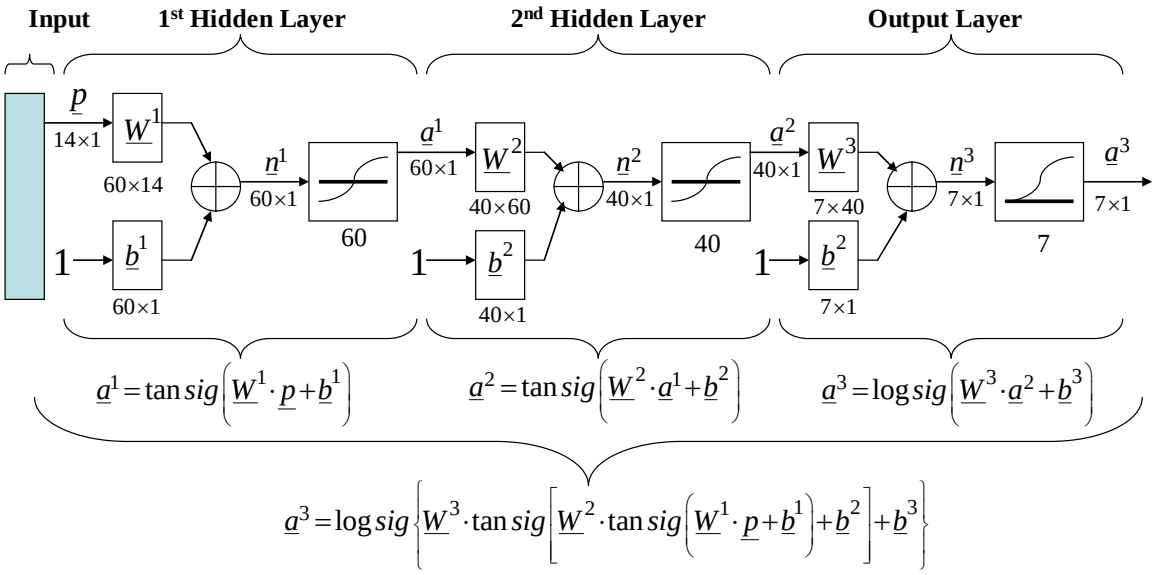

Figure 5.7. Three-layer feedforward neural network architecture implemented; $(60-40-7)$ configuration.

1-of-n coding was selected for the target representation, where $n=7$ is the total number of classes. In the 1-of-7 representation, the output of one of the neurons at the output layer which corresponds to one of the seven classes is set to one, with the output of the rest set to zero. For instance, the word "*up*" is labeled as *Class 1*, and its associated target vector is defined as $[\,0\;0\;0\;0\;0\;0\;1\,]^T$. The class number and target vector assignment for each word in the vocabulary are shown in Table 5.1.

Vocabulary Words	Class Number	Associated Target Vector
up	1	$[\,0\;0\;0\;0\;0\;0\;1\,]^T$
down	2	$[\,0\;0\;0\;0\;0\;1\;0\,]^T$
left	3	$[\,0\;0\;0\;0\;1\;0\;0\,]^T$
right	4	$[\,0\;0\;0\;1\;0\;0\;0\,]^T$
kill	5	$[\,0\;0\;1\;0\;0\;0\;0\,]^T$
pan	6	$[\,0\;1\;0\;0\;0\;0\;0\,]^T$
move	7	$[\,1\;0\;0\;0\;0\;0\;0\,]^T$

Table 5.1. Class numbers and target vectors associated with the vocabulary words.

Network outputs are continuous between zero and one, as the logsig function is used in the output layer. In order to achieve 1-of-n coding, network outputs were converted to zeros and ones by passing them through a simple maximum detector which assigns one to the maximum output value and zero to the rest.

The conjugate gradient (CG) algorithm was selected as the main backpropagation learning function instead of the Levenberg-Marquardt (LM) algorithm because the CG approach was computationally much faster and led to better classification results. Recall that the CG algorithm possesses two useful properties: the memory requirement is minimal when compared to the LM algorithm, and the CG method is much faster than the LM scheme although it usually requires a larger number of epochs for convergence. However, we experimentally observed that the longer the network was trained, the better the results were. One of the multi-layer network configurations, namely the $(40-20-7)$ network, was also tested with the LM scheme to compare the results of the LM scheme with those obtained with the CG scheme. All these issues will be addressed in the next section.

All network configurations considered in the study were applied to 80 experiments (i.e., iterations) to obtain statistically meaningful results. As will be explained later in this section, the training phase used a randomly selected fixed percentage of the data, while the rest was used for the testing phase. The maximum number of epochs for network training was set to 1,000 after observing that convergence was reached within that range.

One of the problems associated with the neural network training is called *overfitting*, where the error on the training set approaches to zero, but the error becomes very large on the testing set as the network memorizes patterns shown in the training set but is unable to generalize to slightly different patterns contained in the testing set and to classify new samples in the testing set. As a result, the best way to tackle the generalization issue is to select a network configuration that is just large enough to provide the desired result. However, determining the right size of a multi-layer network beforehand is very difficult unless the problem is easy to solve, and the network size is usually determined experimentally. Two commonly used methods to improve

generalization are *regularization* and *early stopping*. In this study, we selected the regularization technique to prevent overfitting. Regularization improves generalization by modifying the performance function (i.e., the MSE) and adding an additional term that contains the mean of the sums of the network parameters. The modified performance function *msereg* is given as [Demuth, Beale, 2005]:

$$msereg = \gamma mse + (1-\gamma) msw, \tag{5.31}$$

where γ is the performance ratio, which is also determined experimentally, and set to 0.85 for this study. Using *msereg* causes the weights and biases to be smaller, which in turn yields a smoother network response that is better designed to avoid overfitting [Demuth, Beale, 2005].

The training set selected for each iteration was formed by randomly picking 15 repetitions of a word for each subject. As a result, the total size of the training set was 2100, or $(20 \times 7 \times 15)$, since there are 20 subjects and 7 words in the vocabulary. The remaining repetitions of a word for each subject were assigned to the testing set for each experiment (i.e., iteration). As a result, the relative percentages of the training and testing sets were 25.52% and 74.48%, respectively. The total data size was 8,228 as indicated in Chapter II, resulting in the total size of 6,128 for the testing set.

The various network configurations considered in this study are listed in Table 5.2.

Structure	**Neuron #s**	**Features**	**Activation Function**	**Training Function**	**Perform. Function**	**Iteration #s**
Two-layer	50-7	MFCC	tansig - logsig	CG	msereg	80
	100-7	MFCC		CG		
	150-7	MFCC & RC		CG		
Three-layer	30-20-7	MFCC	tansig - tansig - logsig	CG		
	40-20-7	MFCC		CG & LM		
	50-30-7	MFCC		CG		
	60-40-7	MFCC & RC		CG		

Table 5.2. Multi-layer network structures considered in the study.

G. RECOGNITION RESULTS

The word recognition results obtained with the different multi-layer neural network configurations considered are presented in this section. Performance measures include:

- The confusion matrix for the training set,
- The confusion matrix for the testing set,
- The confusion matrix when the network is tested on "*kill_noise*" and "*right_outside*," on which it is not trained,
- Average classification rate and 95% confidence interval plot for the testing set, and
- Average classification rate and 95% confidence interval table for the testing sets of all the configurations used in the study.

1. Network Configurations Considered

Table 5.3 shows the overall average classification rates for both the training and testing sets, and the 95% confidence intervals for the average classification rates for the testing sets obtained after 80 iterations with each network configuration considered in the study. Results show that overall average classification rates increase for two-layer and three-layer network configurations as the number of neurons in the hidden layers increases. The best two-layer network average classification rates are obtained with the $(150-7)$ configuration, and obtained with the $(60-40-7)$ configuration among the three-layer networks. Results also show their performances to be very similar with 94.731% and 94.61%, for the two- and three-layer network structures, respectively. Results in Table 5.3 also show that MFCCs lead to better recognition rates than RCs do. We note there is about an 8% difference between the average classification rates obtained with the best network structures, i.e., the $(150-7)$ and $(60-40-7)$ networks, using the MFCCs and RCs as features. Another important point to note is the performance difference between the network trained using the CG or the LM scheme. The network trained using the CG algorithm yields around a 3.5% higher recognition result than that obtained with the LM algorithm on the same network configuration, i.e., the $(40-20-7)$ network that operates on the MFCC. Results also show that the 95%

confidence interval (CI) for the LM scheme is the largest of all CIs obtained with MFCCs as input features in this study, which makes the network configuration obtained with the LM scheme much less desirable.

Network Configuration	Training Function	Features	Average Classification Rate for Training Set (in %)	Average Classification Rate and 95% Confidence Interval for Testing Set (in %)
50 - 7	CG	MFCC	93.17	89.281 [88.226 , 90.32]
100 - 7		MFCC	97.941	93.578 [92.727 , 94.559]
150 - 7		MFCC	99.166	94.731 [93.647 , 95.47]
150 - 7		RC	95.04	86.252 [84.926 , 87.546]
30 - 20 -7		MFCC	97.498	92.297 [91.22 , 93.319]
40 - 20 -7		MFCC	98.847	93.405 [91.954 , 94.217]
50 - 30 - 7		MFCC	99.711	94.306 [93.335 , 95.048]
60 - 40 -7		MFCC	99.944	94.61 [93.795 , 95.381]
60 - 40 - 7		RC	98.782	86.7 [85.708 , 87.783]
40 - 20 - 7	LM	MFCC	97.048	89.792 [78.436 , 92.74]

Table 5.3. Average recognition results obtained for the different multi-layer neural network configurations considered in this study; 80 experiments.

2. Computational Time Issues

One last note that has to be addressed about the use of LM algorithm is the choice of the $(40-20-7)$ network with the LM scheme and the amount of time needed to complete all 80 experiments with the LM scheme. The choice of the $(40-20-7)$ network is due to the large memory requirements of the implementations with the LM scheme, as discussed in earlier sections. As a result, that network configuration was the largest complexity network we could run with LM scheme with a Pentium 4 ($3\ GHz$) processor with $512\ MB$ DDR2 RAM for 80 iterations without early termination due to

"out-of-memory" problems. We noted that using the reduced memory LM option in the LM algorithm did not help with the out-of-memory problem for the configurations considered. We also noted that the CG algorithm applied to the same complexity network considered in this study, $(40-20-7)$, converged with about 1000 epochs in about eight minutes, while the LM algorithm took on average 15 minutes to compute 20-30 epochs. Therefore, the time required for a multi-layer network with LM algorithm to complete an implementation with multiple iterations was not very practical as the network complexity increased.

3. Results

Recognition results obtained with each multi-layer network configuration shown in Table 5.3 will be presented in this section. Confusion matrices for both training and testing sets, confusion matrices for "*kill_noise*" and "*right_outside*" are given in Tables 5.4 through 5.33. Average classification rate and 95% confidence intervals for the testing sets of all configurations used in the study are presented in Table 5.34. Finally, Figures 5.8 through 5.17 present the 95% confidence intervals for the average classification rates for the testing set of each network configuration.

Overall Classification Rate = 93.17%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	**95.88333**	0.279167	3.875	0.258333	0.275	0.004167	0.579167
	DOWN	1.345833	**95.47917**	2.4625	2.295833	2.033333	1.658333	0.05
	LEFT	1.9125	0.845833	**89.475**	3.670833	0.258333	0.075	0.15
	RIGHT	0.016667	0.7	1.683333	**89.10417**	3.458333	3.325	0.133333
	KILL	0	0.908333	0.575	2.3	**89.10833**	0.591667	0.058333
	PAN	0.145833	1.7625	1.170833	1.333333	3.658333	**94.20833**	0.1
	MOVE	0.695833	0.025	0.758333	1.0375	1.208333	0.1375	**98.92917**

Table 5.4. Average recognition rates for Training data; $(50-7)$ network configuration; MFCCs as input features; 80 experiments.

Overall Classification Rate = 89.281%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	**94.29979**	0.406204	6.20283	0.650362	0.368732	0.005564	1.049475
	DOWN	2.19914	**91.62851**	3.931422	3.811594	2.367257	1.854599	0.140555
	LEFT	2.65043	1.578656	**82.34761**	5.021739	0.363201	0.072329	0.723388
	RIGHT	0.112822	1.488183	3.350871	**81.68478**	4.972345	4.037463	0.47976
	KILL	0.010745	1.838996	1.444122	4.461957	**86.30531**	1.075668	1.193778
	PAN	0.127149	2.861891	1.357039	3.039855	4.109513	**92.64095**	0.356072
	MOVE	0.599928	0.197563	1.36611	1.32971	1.513643	0.313427	**96.05697**

Table 5.5. Average recognition rates for Testing data; $(50-7)$ network configuration; MFCCs as input features; 80 experiments.

		DATA BELONGING TO:	
		KILL_NOISE	**RIGHT_OUTSIDE**
DATA IDENTIFIED AS :	**UP**	12.59076763	3.801150895
	DOWN	12.51426349	24.86892583
	LEFT	9.220695021	2.845268542
	RIGHT	12.99144191	**53.73721228**
	KILL	**22.84621369**	2.33056266
	PAN	7.575207469	12.4168798
	MOVE	22.26141079	0

Table 5.6. Average recognition rates for the words "*kill_noise*" and "*right_outside*;" $(50-7)$ network configuration; MFCCs as input features; 80 experiments.

Overall Classification Rate = 97.941%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	**99.05417**	0.045833	1.033333	0.033333	0.029167	0	0.066667
	DOWN	0.3375	**98.80833**	1.279167	0.733333	0.6875	0.55	0.008333
	LEFT	0.479167	0.283333	**96.67083**	1.0875	0.033333	0	0.033333
	RIGHT	0.004167	0.204167	0.625	**96.61667**	1.65	0.875	0.0625
	KILL	0	0.354167	0.066667	0.75	**96.17917**	0.108333	0.0125
	PAN	0.05	0.304167	0.158333	0.5125	1.175	**98.44167**	0
	MOVE	0.075	0	0.166667	0.266667	0.245833	0.025	**99.81667**

Table 5.7. Average recognition rates for Training data; $(100-7)$ network configuration; MFCCs as input features; 80 experiments.

Overall Classification Rate = 93.578%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	**96.44699**	0.251108	3.454282	0.322464	0.138274	0.012982	0.41042
	DOWN	1.085244	**94.92061**	2.882801	2.338768	1.076696	1.400223	0.074963
	LEFT	1.855301	0.865953	**89.37772**	2.684783	0.178835	0.016691	0.386057
	RIGHT	0.159384	0.976736	2.66328	**89.1087**	4.035767	1.817507	0.509745
	KILL	0.008954	1.366322	0.522496	3.251812	**91.87869**	0.563798	1.017616
	PAN	0.159384	1.495569	0.488026	1.699275	2.15708	**96.03858**	0.326087
	MOVE	0.284742	0.123708	0.611393	0.594203	0.534661	0.150223	**97.27511**

Table 5.8. Average recognition rates for Testing data; $(100-7)$ network configuration; MFCCs as input features; 80 experiments.

		DATA BELONGING TO:	
		KILL_NOISE	**RIGHT_OUTSIDE**
DATA IDENTIFIED AS :	**UP**	10.27748963	3.983375959
	DOWN	11.8218361	27.10038363
	LEFT	8.539937759	5.006393862
	RIGHT	17.07728216	**48.20652174**
	KILL	**25.08428423**	2.586317136
	PAN	7.139522822	12.99872123
	MOVE	20.0596473	0.118286445

Table 5.9. Average recognition rates for the words "*kill_noise*" and "*right_outside*;" $(100-7)$ network configuration; MFCCs as input features; 80 experiments.

Overall Classification Rate = 99.166%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	**99.72083**	0.008333	0.358333	0.004167	0.004167	0	0.016667
	DOWN	0.075	**99.65417**	0.8875	0.304167	0.245833	0.183333	0
	LEFT	0.1625	0.1375	**98.43333**	0.591667	0.029167	0.004167	0.0125
	RIGHT	0	0.045833	0.208333	**98.41667**	0.725	0.308333	0.045833
	KILL	0	0.058333	0.008333	0.454167	**98.525**	0.0125	0
	PAN	0.0125	0.095833	0.0375	0.129167	0.416667	**99.4875**	0
	MOVE	0.029167	0	0.066667	0.1	0.054167	0.004167	**99.925**

Table 5.10. Average recognition rates for Training data; $(150-7)$ network configuration; MFCCs as input features; 80 experiments.

Overall Classification Rate = 94.731%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	**96.90007**	0.199409	2.686865	0.228261	0.127212	0.02411	0.30922
	DOWN	0.863181	**96.05244**	2.378447	1.728261	0.711652	1.192507	0.058096
	LEFT	1.68159	0.760709	**91.28991**	2.277174	0.154867	0.011128	0.296102
	RIGHT	0.166547	0.891802	2.403846	**90.8279**	3.408923	1.164688	0.528486
	KILL	0.016117	0.915805	0.410015	3.074275	**93.50479**	0.548961	0.927661
	PAN	0.170129	1.091211	0.362845	1.425725	1.749631	**96.94547**	0.284858
	MOVE	0.202364	0.088626	0.46807	0.438406	0.34292	0.113131	**97.59558**

Table 5.11. Average recognition rates for Testing data; $(150-7)$ network configuration; MFCCs as input features; 80 experiments.

		DATA BELONGING TO:	
		KILL_NOISE	**RIGHT_OUTSIDE**
DATA IDENTIFIED AS :	**UP**	9.408713693	4.216751918
	DOWN	10.94268672	26.93734015
	LEFT	8.550311203	7.036445013
	RIGHT	19.51763485	**46.17327366**
	KILL	**25.77282158**	3.391943734
	PAN	7.196576763	12.06202046
	MOVE	18.61125519	0.182225064

Table 5.12. Average recognition rates for the words "*kill_noise*" and "*right_outside*;" $(150-7)$ network configuration; MFCCs as input features; 80 experiments.

Overall Classification Rate = 95.04%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	**97.78333**	0.070833	1.766667	0.041667	0.179167	0.004167	0.320833
	DOWN	0.45	**96.10417**	1.925	1.191667	3.0875	1.954167	0.1875
	LEFT	1.141667	0.879167	**93.24167**	1.4	0.545833	0.091667	0.695833
	RIGHT	0.025	0.654167	1.045833	**92.44583**	3.508333	1.1375	0.020833
	KILL	0.0125	0.35	0.666667	3.295833	**90.75**	0.35	0.004167
	PAN	0.179167	1.55	0.733333	0.529167	1.679167	**96.25833**	0.070833
	MOVE	0.408333	0.391667	0.620833	1.095833	0.25	0.204167	**98.7**

Table 5.13. Average recognition rates for Training data; $(150-7)$ network configuration; RCs as input features; 80 experiments.

Overall Classification Rate = 86.252%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	**92.69162**	0.409897	5.462627	0.289855	0.291298	0.046365	0.699025
	DOWN	1.484599	**86.43464**	4.963716	3.286232	4.664454	4.324926	0.635307
	LEFT	4.222779	3.6226	**80.72932**	3.798913	1.430678	0.218843	1.84033
	RIGHT	0.232808	2.762186	3.216618	**78.29529**	9.105826	2.349777	1.043853
	KILL	0.112822	1.240768	2.463716	10.11775	**80.19358**	1.656157	0.845202
	PAN	0.433381	4.67873	1.050435	2.367754	3.20059	**90.90319**	0.421664
	MOVE	0.821991	0.851182	2.11357	1.844203	1.113569	0.500742	**94.51462**

Table 5.14. Average recognition rates for Testing data; $(150-7)$ network configuration; RCs as input features; 80 experiments.

		DATA BELONGING TO:	
		KILL_NOISE	**RIGHT_OUTSIDE**
DATA IDENTIFIED AS :	**UP**	11.15793568	8.903452685
	DOWN	13.12240664	19.58439898
	LEFT	17.43775934	6.988491049
	RIGHT	17.06431535	**35.96867008**
	KILL	**21.81924274**	7.826086957
	PAN	4.914419087	18.24488491
	MOVE	14.48392116	2.484015345

Table 5.15. Average recognition rates for the words "*kill_noise*" and "*right_outside*;" $(150-7)$ network configuration; RCs as input features; 80 experiments.

Overall Classification Rate = 97.498%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	**98.93333**	0.091667	1.004167	0.029167	0.058333	0.004167	0.075
	DOWN	0.3125	**98.04583**	1.533333	0.983333	0.695833	0.954167	0.020833
	LEFT	0.566667	0.4125	**95.7625**	1.558333	0.066667	0.008333	0.029167
	RIGHT	0.0125	0.4	1.070833	**95.575**	1.1625	0.966667	0.0625
	KILL	0	0.516667	0.1875	1.1125	**96.5875**	0.216667	0.029167
	PAN	0.104167	0.525	0.254167	0.479167	1.283333	**97.81667**	0.020833
	MOVE	0.070833	0.008333	0.1875	0.2625	0.145833	0.033333	**99.7625**

Table 5.16. Average recognition rates for Training data; $(30-20-7)$ network configuration; MFCCs as input features; 80 experiments.

Overall Classification Rate = 92.297%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	**95.57486**	0.328656	3.835269	0.358696	0.114307	0.025964	0.573463
	DOWN	1.332378	**93.67245**	3.354499	2.405797	1.15413	1.904674	0.140555
	LEFT	2.587751	1.172452	**87.75218**	3.681159	0.230457	0.040801	0.496627
	RIGHT	0.15043	1.504801	3.011611	**86.21196**	4.044985	1.845326	0.633433
	KILL	0.023281	1.412482	0.769231	4.369565	**91.26291**	0.827151	1.36057
	PAN	0.162966	1.861152	0.636792	2.429348	2.505531	**95.21513**	0.406672
	MOVE	0.168338	0.048006	0.640421	0.543478	0.687684	0.14095	**96.38868**

Table 5.17. Average recognition rates for Testing data; $(30-20-7)$ network configuration; MFCCs as input features; 80 experiments.

		DATA BELONGING TO:	
		KILL_NOISE	**RIGHT_OUTSIDE**
DATA IDENTIFIED AS :	**UP**	9.1156639	4.862531969
	DOWN	9.463174274	26.95012788
	LEFT	11.07624481	6.297953964
	RIGHT	20.3591805	**46.08375959**
	KILL	**22.41830913**	2.80370844
	PAN	7.441649378	12.82928389
	MOVE	20.12577801	0.172634271

Table 5.18. Average recognition rates for the words "*kill_noise*" and "*right_outside*;" $(30-20-7)$ network configuration; MFCCs as input features; 80 experiments.

Overall Classification Rate = 98.847%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	**99.6125**	0.058333	0.408333	0	0.020833	0	0.025
	DOWN	0.116667	**99.31667**	1.054167	0.516667	0.304167	0.4875	0.004167
	LEFT	0.183333	0.166667	**97.79167**	0.95	0.020833	0.004167	0.008333
	RIGHT	0.004167	0.116667	0.479167	**97.7375**	0.545833	0.416667	0.025
	KILL	0.004167	0.1625	0.058333	0.408333	**98.58333**	0.1	0.0125
	PAN	0.066667	0.179167	0.125	0.25	0.491667	**98.9625**	0
	MOVE	0.0125	0	0.083333	0.1375	0.033333	0.029167	**99.925**

Table 5.19. Average recognition rates for Training data; $(40-20-7)$ network configuration; MFCCs as input features; 80 experiments.

Overall Classification Rate = 93.405%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	**96.16404**	0.306499	3.136792	0.253623	0.105088	0.027819	0.371064
	DOWN	1.054799	**94.60672**	2.835631	1.994565	0.866519	1.680267	0.108696
	LEFT	2.310172	1.028434	**89.44122**	3.001812	0.160398	0.046365	0.395427
	RIGHT	0.17192	1.286928	2.942671	**88.00543**	3.545354	1.511499	0.655922
	KILL	0.016117	1.152142	0.649492	4.181159	**92.78577**	0.771513	1.161919
	PAN	0.145057	1.569424	0.46807	2.103261	2.04646	**95.86795**	0.344828
	MOVE	0.137894	0.049852	0.526125	0.460145	0.490413	0.094585	**96.96214**

Table 5.20. Average recognition rates for Testing data; $(40-20-7)$ network configuration; MFCCs as input features; 80 experiments.

		DATA BELONGING TO:	
		KILL_NOISE	**RIGHT_OUTSIDE**
DATA IDENTIFIED AS :	**UP**	8.311721992	5.047953964
	DOWN	9.544865145	27.11317136
	LEFT	9.71473029	7.893222506
	RIGHT	21.84128631	**43.54539642**
	KILL	**24.09880705**	3.318414322
	PAN	7.021524896	12.81969309
	MOVE	19.46706432	0.262148338

Table 5.21. Average recognition rates for the words "*kill_noise*" and "*right_outside*;" $(40-20-7)$ network configuration; MFCCs as input features; 80 experiments.

Overall Classification Rate = 99.711%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	**99.95**	0.0125	0.0875	0.008333	0.008333	0	0
	DOWN	0.008333	**99.88333**	0.495833	0.1125	0.05	0.1125	0
	LEFT	0.025	0.029167	**99.26667**	0.3625	0	0	0
	RIGHT	0.004167	0.033333	0.116667	**99.225**	0.0875	0.041667	0.008333
	KILL	0	0.0375	0.0125	0.2	**99.8125**	0	0
	PAN	0.008333	0.004167	0.016667	0.079167	0.041667	**99.84583**	0
	MOVE	0.004167	0	0.004167	0.0125	0	0	**99.99167**

Table 5.22. Average recognition rates for Training data; $(50-30-7)$ network configuration; MFCCs as input features; 80 experiments.

Overall Classification Rate = 94.306%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	**96.61891**	0.238183	2.570755	0.268116	0.092183	0.016691	0.324213
	DOWN	0.757521	**95.51329**	2.58164	1.788043	0.663717	1.687685	0.118066
	LEFT	2.179441	0.94904	**90.90348**	2.494565	0.164086	0.04822	0.279235
	RIGHT	0.148639	1.172452	2.734035	**89.79891**	3.438422	1.003338	0.631559
	KILL	0.025072	0.696086	0.471698	3.615942	**93.56563**	0.60089	1.154423
	PAN	0.155802	1.388479	0.312046	1.643116	1.698009	**96.55972**	0.30922
	MOVE	0.114613	0.042467	0.426343	0.391304	0.37795	0.083457	**97.18328**

Table 5.23. Average recognition rates for Testing data; $(50-30-7)$ network configuration; MFCCs as input features; 80 experiments.

		DATA BELONGING TO:	
		KILL_NOISE	**RIGHT_OUTSIDE**
DATA IDENTIFIED AS :	**UP**	7.625778008	4.894501279
	DOWN	9.207728216	25.57225064
	LEFT	9.9468361	8.81713555
	RIGHT	23.00440871	**44.12723785**
	KILL	**24.69528008**	3.670076726
	PAN	6.758298755	12.58951407
	MOVE	18.76167012	0.329283887

Table 5.24. Average recognition rates for the words "*kill_noise*" and "*right_outside*;" $(50-30-7)$ network configuration; MFCCs as input features; 80 experiments.

Overall Classification Rate = 99.944%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	100	0	0.004167	0.004167	0.004167	0	0
	DOWN	0	99.99583	0.154167	0.025	0.008333	0.0125	0
	LEFT	0	0	99.83333	0.0875	0	0	0
	RIGHT	0	0.004167	0.004167	99.825	0.008333	0	0
	KILL	0	0	0	0.045833	99.96667	0	0
	PAN	0	0	0.004167	0.0125	0.0125	99.9875	0
	MOVE	0	0	0	0	0	0	100

Table 5.25. Average recognition rates for Training data; $(60-40-7)$ network configuration; MFCCs as input features; 80 experiments.

Overall Classification Rate = 94.61%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	**96.6798**	0.258493	2.583454	0.21558	0.094027	0.025964	0.23988
	DOWN	0.728868	**95.64993**	2.155298	1.539855	0.678466	1.350148	0.118066
	LEFT	2.15437	1.054284	**91.44049**	2.335145	0.178835	0.04822	0.260495
	RIGHT	0.146848	1.043205	2.685051	**90.33333**	3.442109	0.847552	0.562219
	KILL	0.025072	0.655465	0.471698	3.641304	**93.62647**	0.623145	0.983883
	PAN	0.155802	1.281388	0.313861	1.572464	1.699853	**97.00482**	0.297976
	MOVE	0.109241	0.057238	0.350145	0.362319	0.280236	0.100148	**97.53748**

Table 5.26. Average recognition rates for Testing data; $(60-40-7)$ network configuration; MFCCs as input features; 80 experiments.

		DATA BELONGING TO:	
		KILL_NOISE	**RIGHT_OUTSIDE**
DATA IDENTIFIED AS :	**UP**	7.378112033	4.721867008
	DOWN	9.918309129	24.31905371
	LEFT	9.802904564	10.39002558
	RIGHT	22.68931535	**44.07289003**
	KILL	**25.48495851**	4.124040921
	PAN	6.790715768	12.17071611
	MOVE	17.93568465	0.20140665

Table 5.27. Average recognition rates for the words "*kill_noise*" and "*right_outside*;" $(60-40-7)$ network configuration; MFCCs as input feature; 80 experiments.

Overall Classification Rate = 98.782%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	**99.521**	0.004167	0.2125	0	0.10833	0	0.058333
	DOWN	0.0875	**99.042**	0.44583	0.19583	0.76667	0.52083	0.075
	LEFT	0.27083	0.27083	**98.171**	0.3875	0.37917	0.041667	0.0875
	RIGHT	0.004167	0.17917	0.34167	**98.387**	0.77917	0.17917	0.029167
	KILL	0.0125	0.066667	0.35	0.58333	**97.487**	0.079167	0.004167
	PAN	0.079167	0.21667	0.38333	0.1875	0.45833	**99.121**	0.004167
	MOVE	0.025	0.22083	0.095833	0.25833	0.020833	0.058333	**99.742**

Table 5.28. Average recognition rates for Training data; $(60-40-7)$ network configuration; RCs as input features; 80 experiments.

Overall Classification Rate = 86.7%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	**92.676**	0.56499	5.361	0.26449	0.28208	0.057493	0.56784
	DOWN	1.1479	**86.257**	3.8806	2.837	3.3647	4.0078	0.62219
	LEFT	4.7529	3.6097	**81.402**	3.4058	1.8252	0.41728	1.578
	RIGHT	0.22564	2.9985	3.6847	**79.618**	9.8138	1.9381	1.4561
	KILL	0.18625	1.5306	2.7141	10.118	**81.217**	1.8509	1.0926
	PAN	0.38682	4.2559	1.0849	2.3659	2.6088	**91.419**	0.37106
	MOVE	0.625	0.78287	1.8723	1.3913	0.88864	0.30972	**94.312**

Table 5.29. Average recognition rates for Testing data; $(60-40-7)$ network configuration; RCs as input features; 80 experiments.

		DATA BELONGING TO:	
		KILL_NOISE	**RIGHT_OUTSIDE**
DATA IDENTIFIED AS :	**UP**	9.8561	8.9898
	DOWN	11.677	18.632
	LEFT	18.474	7.8485
	RIGHT	20.502	**34.386**
	KILL	**21.272**	10.444
	PAN	5.0726	17.42
	MOVE	13.147	2.2794

Table 5.30. Average recognition rates for the words "*kill_noise*" and "*right_outside*;" $(60-40-7)$ network configuration; RCs as input feature; 80 experiments.

Overall Classification Rate = 97.048%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	**97.91667**	0.441667	0.716667	0.066667	0.0625	0.1625	0.2375
	DOWN	0.358333	**95.8875**	1.25	0.954167	0.354167	0.945833	0.179167
	LEFT	1	1.045833	**96.85833**	1.141667	0.095833	0.195833	0.391667
	RIGHT	0.191667	0.9125	0.704167	**95.87917**	1.0625	1.0625	0.295833
	KILL	0.170833	0.5	0.154167	1.183333	**97.71667**	0.558333	0.541667
	PAN	0.220833	1.0875	0.191667	0.529167	0.6125	**96.8875**	0.1625
	MOVE	0.141667	0.125	0.125	0.245833	0.095833	0.1875	**98.19167**

Table 5.31. Average recognition rates for Training data; $(40-20-7)$ network configuration; LM algorithm; MFCCs as input features; 80 experiments.

Overall Classification Rate = 89.792%		DATA BELONGING TO :						
		UP	**DOWN**	**LEFT**	**RIGHT**	**KILL**	**PAN**	**MOVE**
DATA IDENTIFIED AS :	**UP**	**93.44377**	0.925037	4.003991	0.492754	0.250737	0.320846	0.927661
	DOWN	1.260745	**88.93833**	3.352685	3.471014	0.973451	2.674332	0.404798
	LEFT	3.794771	2.9339	**86.11756**	3.612319	0.628687	0.50816	0.938906
	RIGHT	0.444126	2.629247	3.889695	**83.62138**	4.935472	2.173591	0.878936
	KILL	0.222063	1.33678	1.012337	5.480072	**89.82485**	1.730341	1.997751
	PAN	0.363539	2.944978	0.71299	2.61413	2.54056	**92.14763**	0.401049
	MOVE	0.470989	0.291728	0.91074	0.708333	0.846239	0.445104	**94.4509**

Table 5.32. Average recognition rates for Testing data; $(40-20-7)$ network configuration; LM algorithm; MFCCs as input features; 80 experiments.

		DATA BELONGING TO:	
		KILL_NOISE	**RIGHT_OUTSIDE**
DATA IDENTIFIED AS :	**UP**	8.606068465	7.007672634
	DOWN	9.014522822	23.38554987
	LEFT	12.40923237	7.909207161
	RIGHT	19.27645228	**42.55754476**
	KILL	**23.74610996**	4.846547315
	PAN	8.292271784	13.48465473
	MOVE	18.65534232	0.808823529

Table 5.33. Average recognition rates for the words "*kill_noise*" and "*right_outside*;" $(40-20-7)$ network configuration; LM algorithm; RCs as input features; 80 experiments.

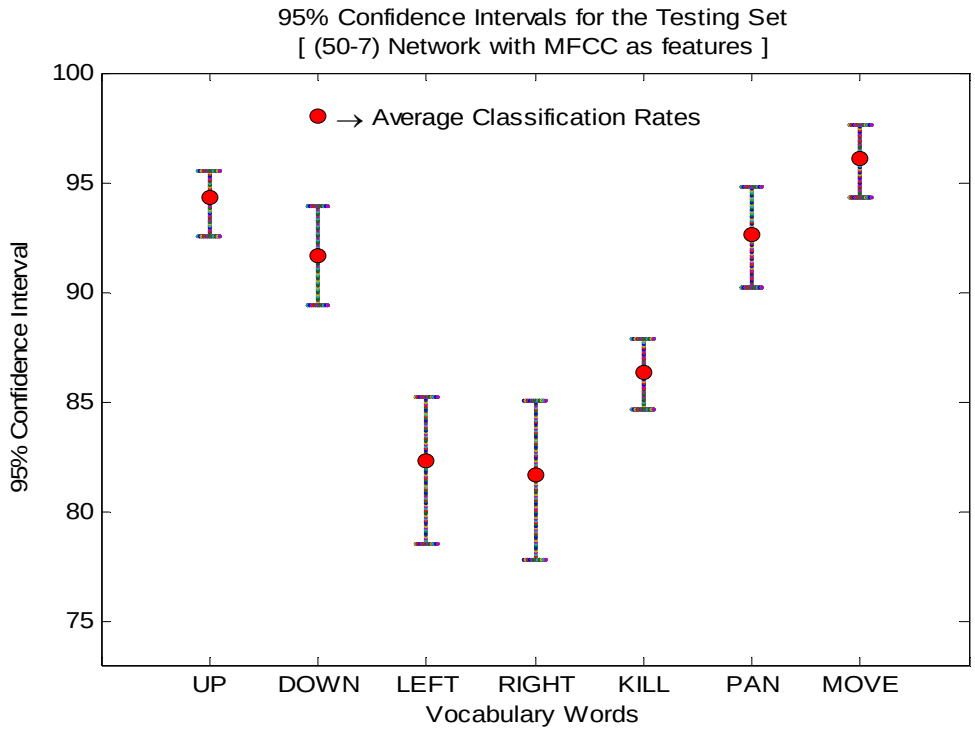

Figure 5.8. Average Classification Rates and 95% Confidence Intervals; Testing data; $(50-7)$ network; 14 MFCCs as input features; 80 experiments.

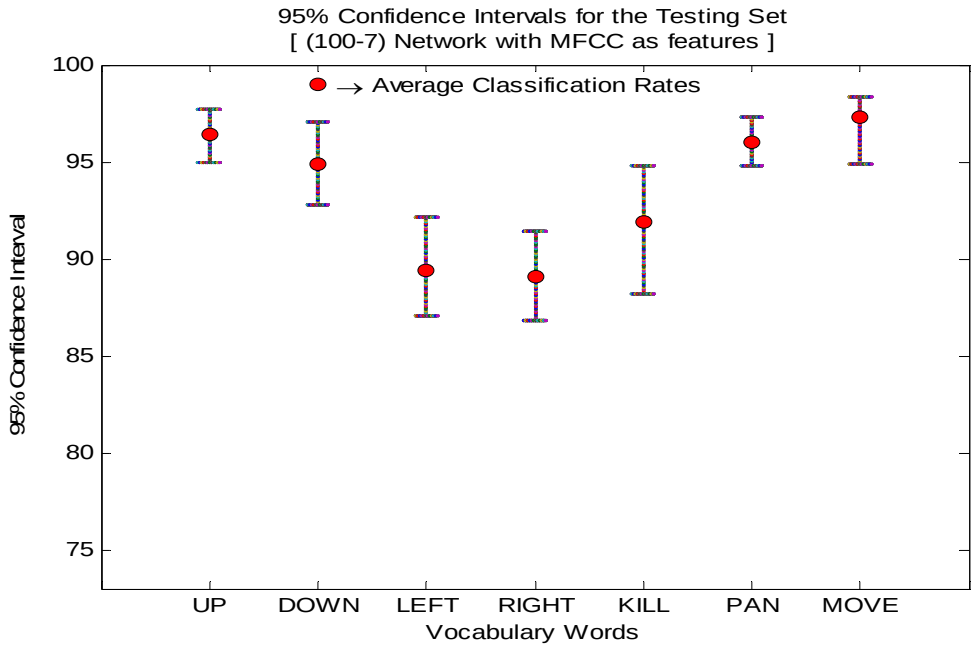

Figure 5.9. Average Classification Rates and 95% Confidence Intervals; Testing Set; $(100-7)$ network; 14 MFCCs as input features; 80 experiments.

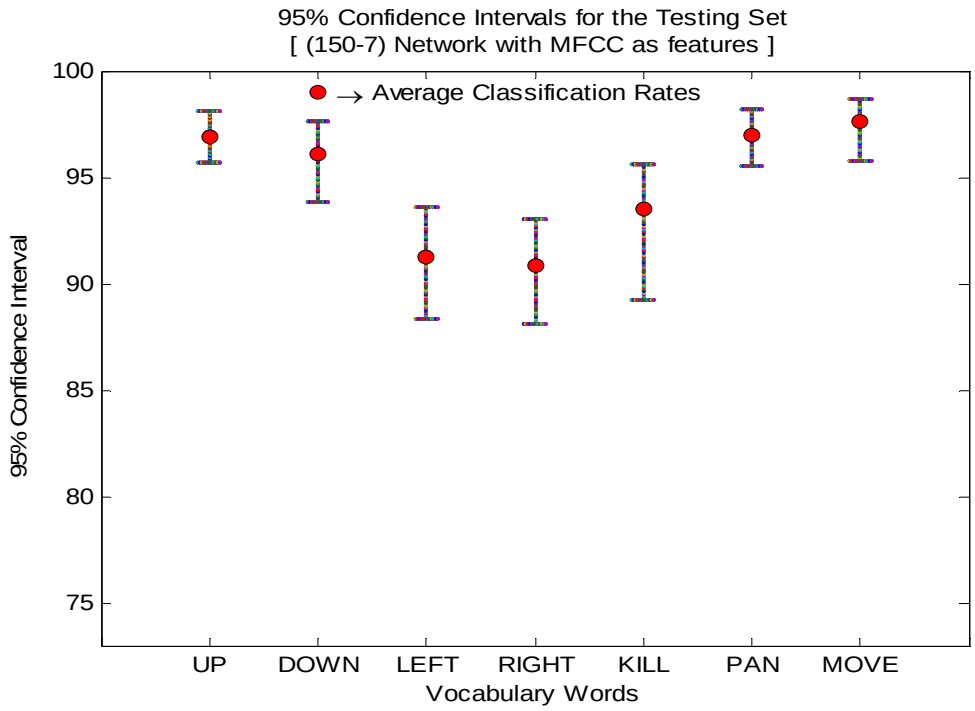

Figure 5.10. Average Classification Rates and 95% Confidence Intervals; Testing Set; $(150-7)$ network; 14 MFCCs as input features; 80 experiments.

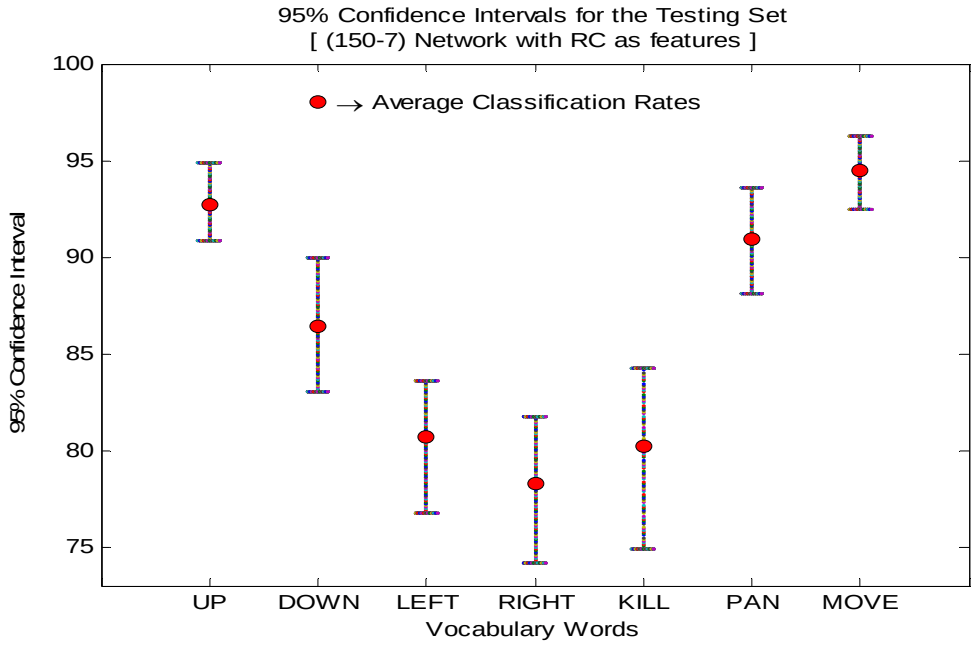

Figure 5.11. Average Classification Rates and 95% Confidence Intervals; Testing Set; $(150-7)$ network; 14 RCs as input features; 80 experiments.

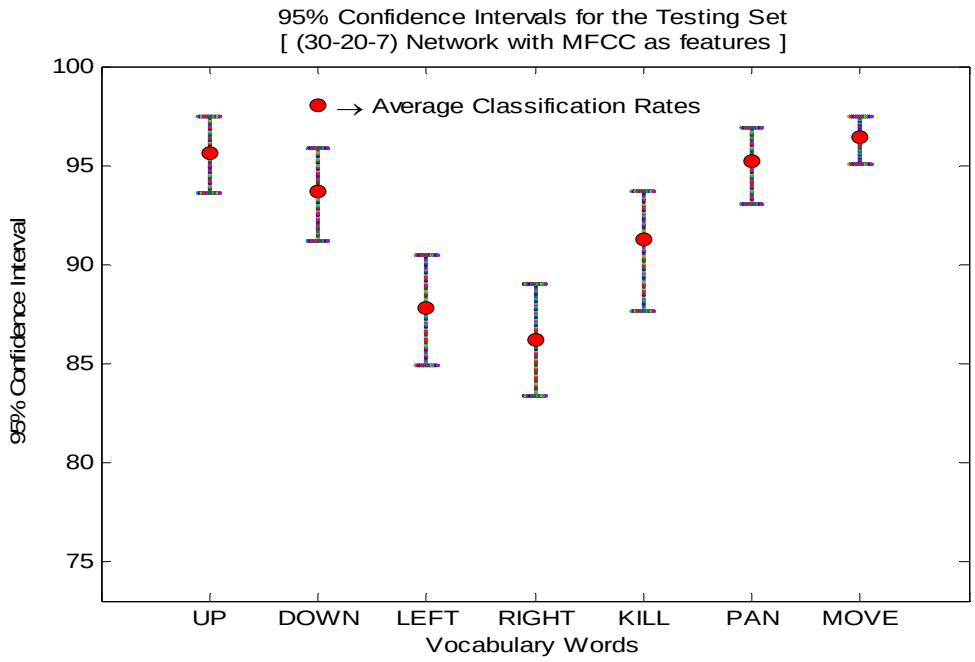

Figure 5.12. Average Classification Rates and 95% Confidence Intervals; Testing Set; $(30-20-7)$ network; 14 MFCCs as input features; 80 experiments.

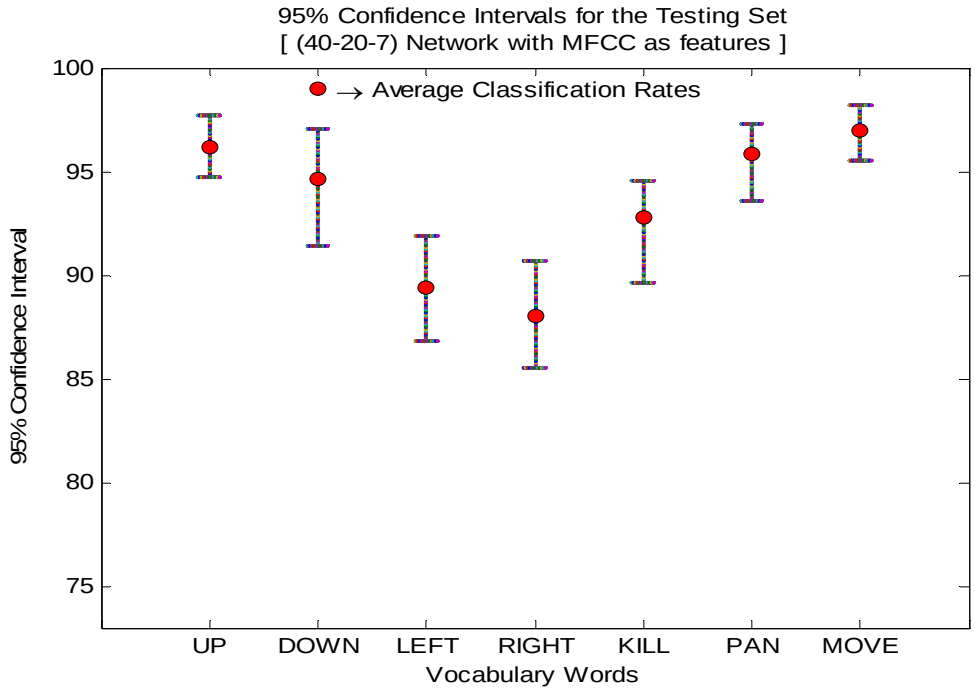

Figure 5.13. Average Classification Rates and 95% Confidence Intervals; Testing Set; $(40-20-7)$ network; 14 MFCCs as input features; 80 experiments.

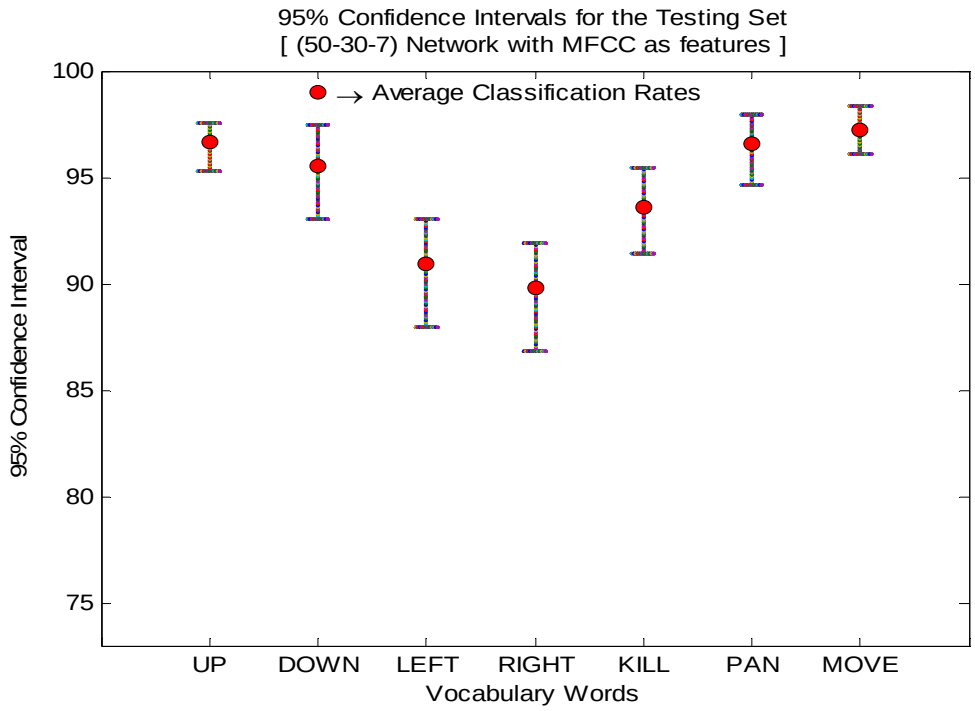

Figure 5.14. Average Classification Rates and 95% Confidence Intervals; Testing Set; $(50-30-7)$ network; 14 MFCCs as input features; 80 experiments.

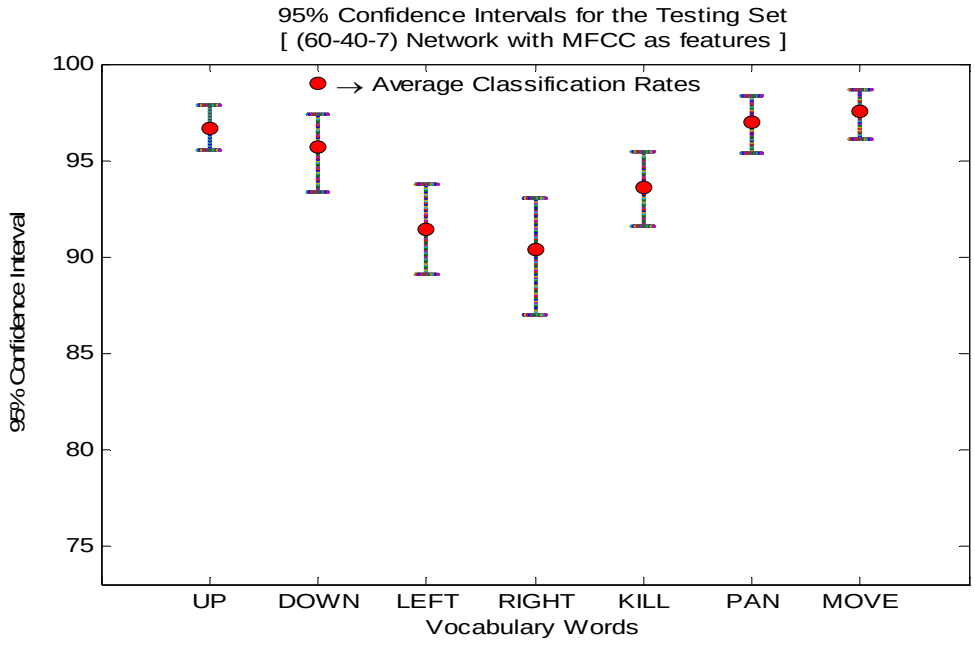

Figure 5.15. Average Classification Rates and 95% Confidence Intervals; Testing Set; $(60-40-7)$ network; 14 MFCCs as input features; 80 experiments.

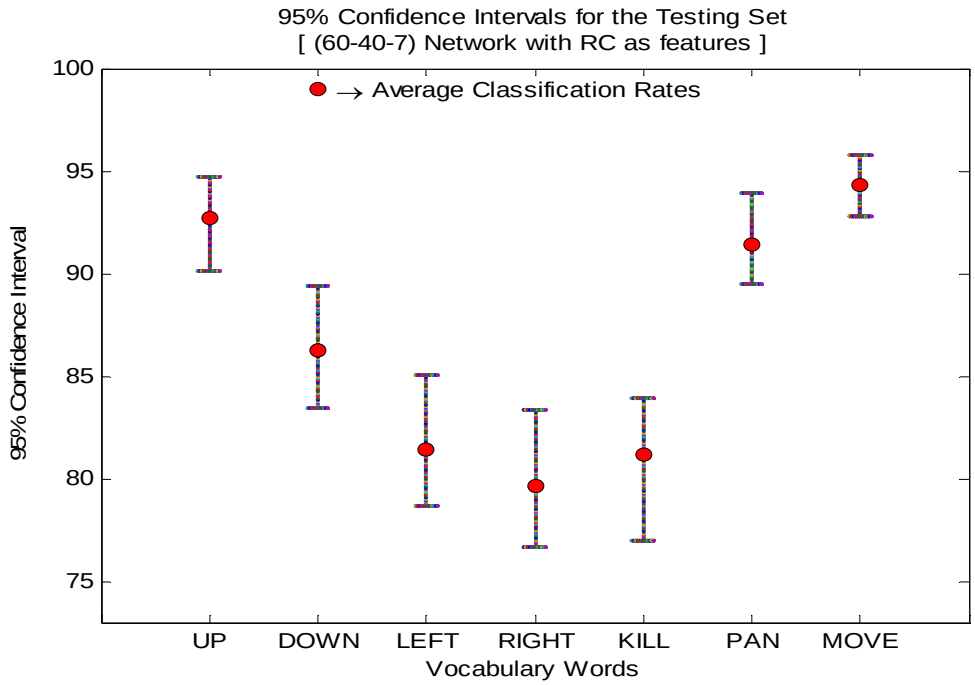

Figure 5.16. Average Classification Rates and 95% Confidence Intervals; Testing Set; $(60-40-7)$ network; 14 RCs as input features; 80 experiments.

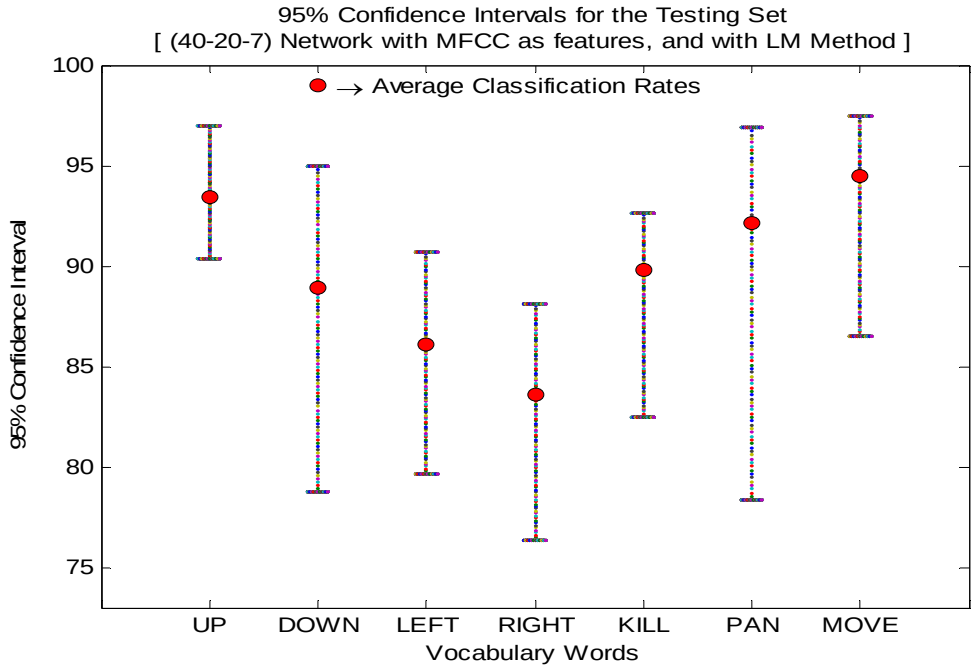

Figure 5.17. Average Classification Rates and 95% Confidence Intervals; Testing Set; $(40-20-7)$ network; LM method; 14 MFCCs as input features; 80 experiments.

Network Configuration	VOCABULARY WORDS							Overall Average Classification Rate and 95% Confidence Interval
	UP	DOWN	LEFT	RIGHT	KILL	PAN	MOVE	
50 - 7	94.3 [92.55 , 95.559]	91.629 [89.365 , 93.944]	82.348 [78.52 , 85.196]	81.685 [77.826 , 85.072]	86.305 [84.661 , 87.906]	92.641 [90.208 , 94.807]	96.057 [94.303 , 97.601]	89.281 [88.226 , 90.32]
100 - 7	96.447 [94.986 , 97.708]	94.921 [92.762 , 97.046]	89.378 [87.083 , 92.163]	89.109 [86.812 , 91.449]	91.879 [88.201 , 94.838]	96.039 [94.807 , 97.329]	97.275 [94.303 , 98.351]	93.578 [92.727 , 94.559]
150 - 7	96.9 [95.702 , 98.138]	96.052 [93.796 , 97.637]	91.29 [88.389 , 93.614]	90.828 [88.116 , 93.043]	93.505 [89.233 , 95.575]	96.945 [95.549 , 98.22]	97.596 [95.802 , 98.651]	94.731 [93.647 , 95.47]
150 - 7 (RC)	92.692 [90.831 , 94.842]	86.435 [83.013 , 89.956]	80.729 [76.778 , 83.559]	78.295 [74.203 , 81.739]	80.194 [74.926 , 84.218]	90.903 [88.131 , 93.62]	94.515 [92.504 , 96.252]	86.252 [84.926 , 87.546]
30 - 20 -7	95.575 [93.553 , 97.421]	93.672 [91.137 , 95.864]	87.752 [84.906 , 90.421]	86.212 [83.333 , 88.986]	91.263 [87.611 , 93.658]	95.215 [93.027 , 96.884]	96.389 [95.052 , 97.451]	92.297 [91.22 , 93.319]
40 - 20 -7	96.164 [94.699 , 97.708]	94.607 [91.433 , 97.046]	89.441 [86.792 , 91.872]	88.005 [85.507 , 90.725]	92.786 [89.676 , 94.543]	95.868 [93.62 , 97.329]	96.962 [95.502 , 98.201]	93.405 [91.954 , 94.217]
50 - 30 - 7	96.619 [95.272 , 97.564]	95.513 [93.058 , 97.489]	90.903 [87.954 , 93.033]	89.799 [86.812 , 91.884]	93.566 [91.445 , 95.428]	96.56 [94.659 , 97.923]	97.183 [96.102 , 98.351]	94.306 [93.335 , 95.048]
60 - 40 -7	96.68 [95.559 , 97.851]	95.65 [93.353 , 97.341]	91.44 [89.115 , 93.759]	90.333 [86.957 , 93.043]	93.626 [91.593 , 95.428]	97.005 [95.401 , 98.368]	97.537 [96.102 , 98.651]	94.61 [93.795 , 95.381]
60 - 40 - 7 (RC)	92.676 [90.115 , 94.699]	86.257 [83.456 , 89.365]	81.402 [78.665 , 85.051]	79.618 [76.667 , 83.333]	81.217 [76.991 , 83.923]	91.419 [89.466 , 93.917]	94.312 [92.804 , 95.802]	86.7 [85.708 , 87.783]
40 - 20 - 7 (LM)	93.444 [90.401 , 96.991]	88.938 [78.73 , 94.978]	86.118 [79.681 , 90.771]	83.621 [76.377 , 88.116]	89.825 [82.448 , 92.625]	92.148 [78.338 , 96.884]	94.451 [86.507 , 97.451]	89.792 [78.436 , 92.74]

Table 5.34. Average Classification Rates and 95% Confidence Intervals for the Testing sets of all configurations considered.

The following overall comments can be made from the results:

- The recognition performance of the multi-layer networks gradually increases as the number of hidden neurons in the hidden layer(s) increases for both two-layer and three-layer network configurations.
- The best recognition performances are obtained with the $(150-7)$ network for the two-layer network configurations, and with the $(60-40-7)$ network for the three-layer networks when MFCCs are used as input features. The overall average recognition rates obtained with the $(150-7)$ and $(60-40-7)$ networks are 94.761% and 94.61%, respectively. It is not, however, possible to tell which one is superior to the other since their recognition performances are very close to each other.
- The best network configurations, i.e., the $(150-7)$ and $(60-40-7)$ networks that yield the highest average recognition results with the MFCCs, produce 86.252% and 86.7% respectively when RC coefficients are used for input features. There is about an 8% difference between the overall average recognition rates obtained with the MFCCs and RCs. The performance degradation is more severe and noticeable on the words "*left,*" "*right,*" and "*kill.*" The 95% average classification rates confidence interval lower bounds for the networks with the RC coefficients goes down to 74%, and we note that the confidence interval spreads are much

larger than those obtained with the MFCCs, sometimes 8% to 10% for the words "*left*," "*right*," and "*kill*." Therefore, results show that the MFCCs are superior to the RCs in recognition performance when used with the multi-layer neural networks for the data under investigation.

- The average recognition rate obtained for the CG algorithm is about 3.5% better than that obtained with the LM algorithm for the $(40-20-7)$ network configuration. Although the difference in overall classification performances may seem small, the difference between the two algorithms becomes more significant when the individual confusion matrices and 95% confidence interval plots are examined. The degradation in recognition is very noticeable on all the vocabulary words except "move" and "up." The spreads in confidence intervals of the words "down" and "pan" obtained with the LM method are 16.25% and 18.55% respectively, whereas the spreads for the same words obtained with the CG method are 5.6% and 3.7%, respectively. Therefore, the CG algorithm leads to a more accurate and reliable network configuration for this word recognition study than the LM algorithm does.

- All network configurations considered here produce the same recognition trend for the seven vocabulary words. The highest recognition results are consistently obtained with the words "move," "up," "pan," "down," and "kill," respectively. Results show that the recognition rate is always lower for the words "right" and "left," respectively. Although there might be multiple reasons for this performance degradation on these two words, one possible cause may be the gap (due to weak fricative or diphthong) between the voiced portion of these words and the stop consonant / t / at the end of the same words, resulting in incorrect end-point detection.

- Recognition performances become very poor, when the networks are tested on types of data that they were not trained on, i.e., when tested on the "kill_noise" and "right_outside" data. As explained in Chapter II, the frequency contents the words "right," collected within the ear canal and "right_outside," collected from a microphone placed in front of the mouth, are quite different due to the low-pass nature of the auditory canal. The spectrograms for the two versions of the same word, shown in Figures 2.9 and 2.10, show that higher frequencies are dampened for the signal collected within the ear canal, resulting in potential large differences in the spectral-based RC and MFCC coefficients. Such differences are very likely responsible for performance degradations observed when the networks are tested on the "right_outside" data. These results also show that additional training with signals generated in front of the mouth would be required if a dual input mode microphone set-up (i.e., either data collected at mouth or within the ear canal) was to be considered. The highly degraded recognition performance observed for the "kill_noise" data is due to the noise embedded in these utterances. Although all other utterances on which the networks are trained and tested contain some amount of noise, the noise is not as severe as that present in the

"kill_noise" data, as illustrated in Figure 5.18 to be compared to Figure 2.7. The amount of noise result in RC and MFCC coefficients obtained for the "kill_noise" data to be different from those obtained with the "kill" data which in turn results in very poor recognition rates. Therefore, some sort of speech enhancement or noise cancellation techniques should be incorporated prior to the word recognition step.

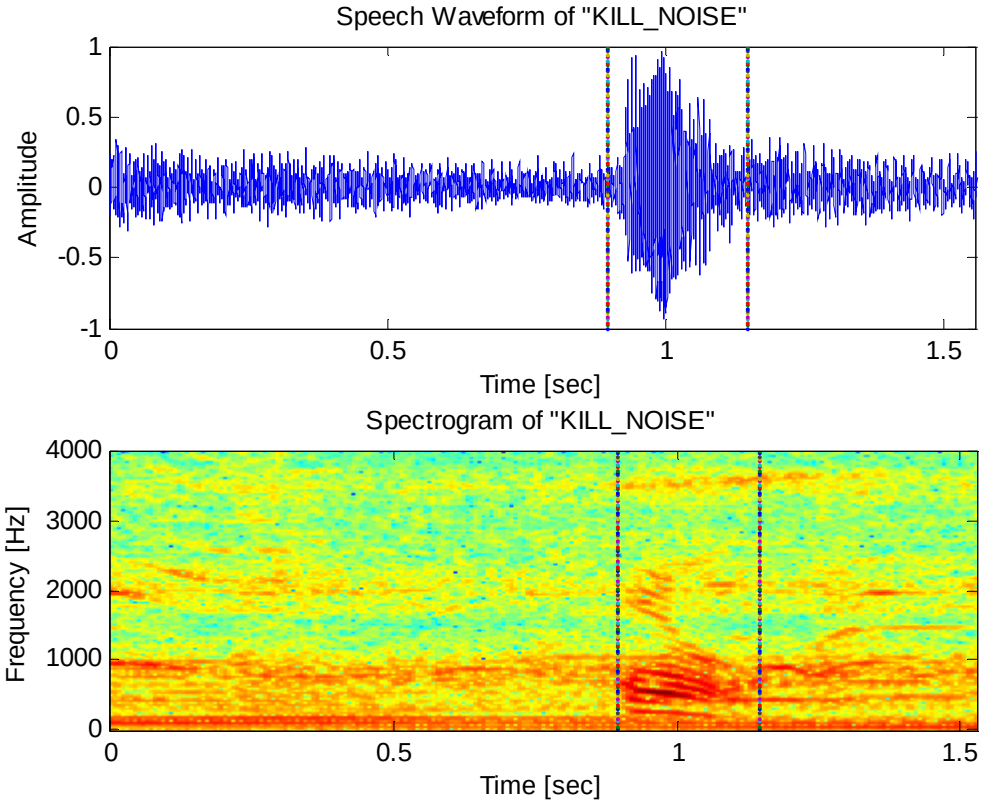

Figure 5.18. Speech waveform (top plot) and associated spectrogram (bottom plot) of "kill_noise" (kill_noise_16_35_08_25_05.txt).

H. SUMMARY

This chapter presented the multi-layer neural networks and the related learning algorithms considered in this study. Next, we presented the recognition results obtained with the different multi-layer neural network configurations considered and two different spectral-based representations, i.e., the MFCCs and RC coefficients.

Results show that the $(150-7)$ network among all two-layer network configurations and the $(60-40-7)$ network among three-layer network configurations

considered yielded highest and similar average recognition results when MFCCs were used as input features. As a result, either of these two configurations could be selected for word recognition. Results showed that MFCCs lead to better classification rates than RCs do for all network configurations investigated, as commonly observed in typical speech signal studies. Results also showed that the CG algorithm led to more accurate and reliable network configurations than the LM algorithm did for the network complexities and data size considered in this study. We also noted that the performance of the multi-layer networks degraded significantly under severe noise conditions, which points out the need for some type of noise cancellation to be applied prior to the speech recognition stage.

VI. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

This study implemented a basic isolated word recognition system operating on utterances collected from the external auditory canals of speakers via an ear-insert microphone. The speech recognition system considered here includes preprocessing, end point detection, feature extraction, and recognition stages.

Initial preprocessing was achieved by a simple IIR bandpass filter to remove a low-frequency hum present in the recordings and high frequencies that did not carry much speech information. Next, the speech was passed through an end point detector that isolated speech from non-speech segments.

Next, real cepstrum (RC) coefficients and mel-frequency cepstral coefficients (MFCCs) were extracted from each segmented utterance and used as input features for different neural network classifiers. Ten different two- and three-layer neural network configurations with different numbers of hidden neurons were implemented for the word recognition task and their resulting classification performances compared.

Recognition results showed that increasing the numbers of hidden neurons (while avoiding overfitting) increased the performance of both two-layer and three-layer networks. Best overall recognition rates were obtained for the two-layer network with the $(150-7)$ configuration using MFCCs as input features (94.731%), and for the three-layer network with the $(60-40-7)$ configuration (94.61%), respectively. Best overall recognition rates obtained with the same $(150-7)$ and $(60-40-7)$ networks using RCs as input features were 86.252% and 86.7% respectively. Therefore, results showed that MFCCs are a better choice for the data under investigation, as observed in other speech studies where data collection is done at the mouth. Implementation results also showed that the conjugate gradient algorithm led to more accurate and reliable network configuration than the Levenberg-Marquardt algorithm did for the network complexities and data size considered in this study. Finally, we noted that the performance of the multi-layer networks degraded significantly under severe noise conditions, i.e., when

tested on the "*kill_noise*," data, thereby showing the need for some noise cancellation prior to data processing.

In conclusion, the study showed that a simple feedforward back propagation configuration can be used for this isolated word recognition problem, given a sufficient number of neurons in the hidden layers and MFCCs are selected as input features.

B. RECOMMENDATIONS

There are still a wide variety of issues that can be addressed in order to build a robust and reliable speech recognizer. Suggestions for future studies include:

- Expanding the vocabulary used for speech recognition with new words and increasing the database.
- Incorporating speech enhancement techniques to the preprocessing stage to enhance the recognizer accuracy and reliability in the presence of noise.
- Using a frame-based feature extraction method instead of computing the input features on one frame only which includes the entire word as done in this study.
- Investigating the performance of other neural network types for word recognition, especially those that take the dynamic nature of speech into account, such as time-delay neural networks (TDNN) for example.

APPENDIX A. IN-EAR MICROPHONE SPECIFICATIONS

This appendix lists the specifications of the in-ear microphone used for recordings.

Electro-Acoustic

Supply Voltage Range	0.9 to 1.6VDC
Current Drain @ 1.3VDC	24µA typical, 50µA maximum
Frequency Response/Sensitivity	Displayed on chart
Sensitivity Tolerance	± 3dB @ 1kHz
Dynamic Output Impedance	2.8 to 6.8kΩ, 4.4 typical
DC Output Voltage Range	0.2 to 0.7VDC, 0.5VDC typical
Power Supply Feedthrough Attenuation	23dB typical (ouput-referred)
Input-Referred Self-Noise Level (A-weighted, 1kHz reference)	27dB typical, 30dB maximum
Output Self-Noise	-100dBV maximum, A-weighted
Acoustic Polarity	Increased pressure at sound inlet causes a positive-going voltage to appear at the output,terminal, relative to the negative terminal.

Environmental

Operating Temperature	-17 to 63˚C
Storage Temperature	-40 to 63˚C
Humidity Coefficient of Sensitivity	0.06dB/%RH typical, non-condensing
Vibration Sensitivity	30dBSPL maximum @ 1g acceleration, non-acoustic, SPL equivalent, input-referred at 1kHz sensitivity
ESD Tolerance	MIL-STD-750 Class 1 rating EOS/ESD-S5.1-1993 Class 2 rating
RF Immunity	Integrated RFI suppression filters

Mechanical

Weight	0.08 grams
Case Material	Type 305 stainless steel
Solder Content	Sn63Pb37

Optional Soldering Fixture

Base	ET-3042
Nest Plate	ET-814

Figure A.1. Specifications of the in-ear microphone used for recordings (From: www.knowlesacoustics.com, last accessed on February 15, 2006).

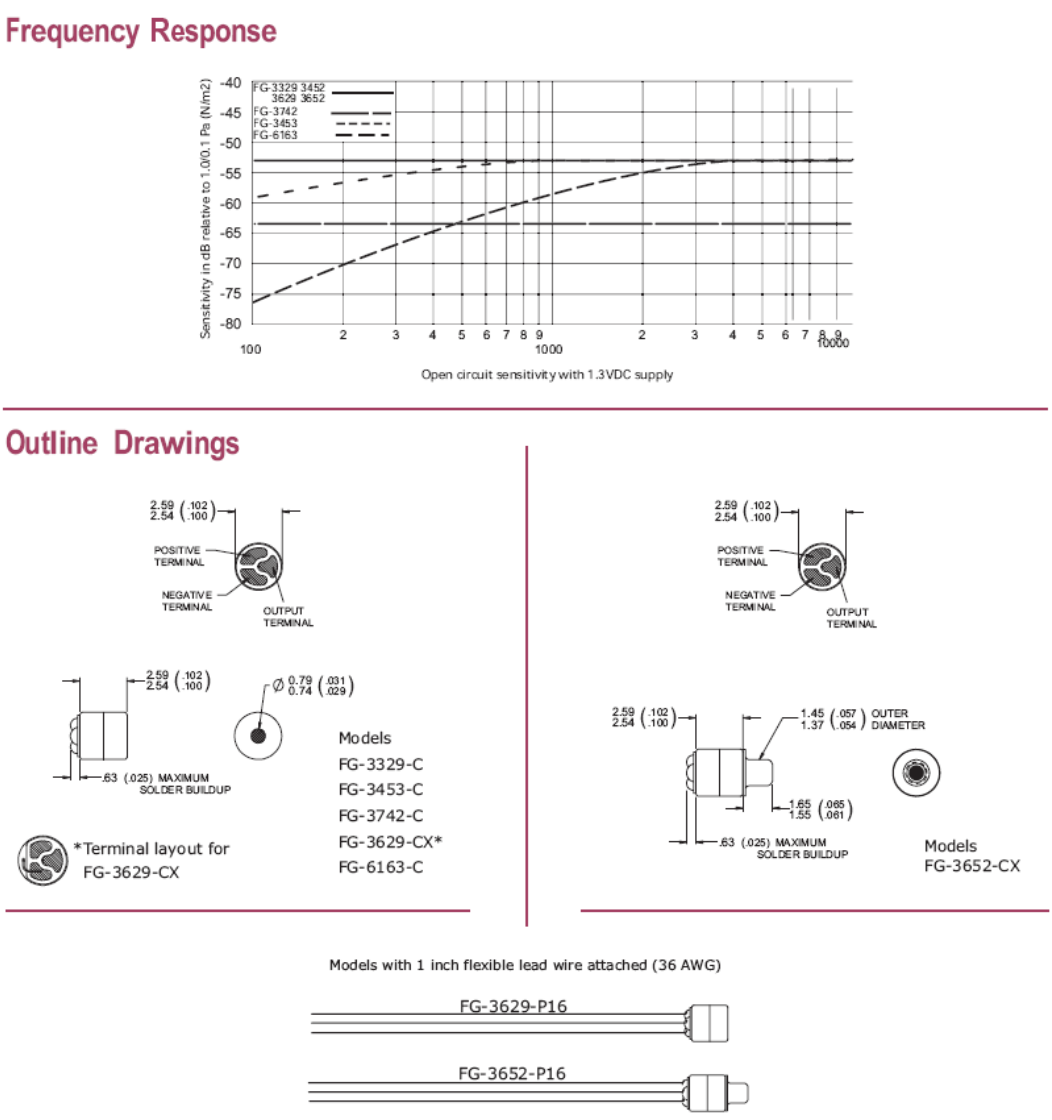

Figure A.2. (Continued from Figure A.1.) Specifications of the in-ear microphone used for recordings (From: www.knowlesacoustics.com, last accessed on February 15, 2006).

APPENDIX B. MATLAB SOURCE CODE

This appendix lists all Matlab programs and functions used for this study. These programs have been developed according to the problem definition of the current study. Two of the functions used for feature extraction task are from the "Voicebox Toolbox" by Brookes [Brookes, 1997] and modified according to the requirements of this study.

Listed below are the three stages of the word recognizer implemented for this study. The Matlab programs and functions are presented along with their descriptions under the stage they belong to. For the first two stages, the first program is the main program to be executed, and the following are the functions that are being called by the main program. For the last stage, all the programs listed are the main programs to be executed, and only two programs (one for two-layer network and one for three-layer network) for two networks out of ten configurations implemented for this study are shown. The other configurations for both network types can easily be obtained by changing a few parameters in the programs.

1. DESCRIPTION

a. End Point Detection

- *speech_segmentation.m*: Performs the speech segmentation task on the whole data set.
- *energy_threshold.m*: Computes the required thresholds, and passes them to the following energy-based end point detection algorithm.
- *endpoint_detection.m*: Conducts the complete energy-based end point detection algorithm and finds the initial end point estimates of an utterance. This function calls two sub-functions in itself, *endpoint_detector.m*, and *endpoint_correct.m*.
- *endpoint_detector.m*: Conducts the actual energy-based end point detection, computes the utterance energy, and returns the initial end point estimates if a miss does not occur.
- *endpoint_correct.m*: Complements the energy-based end point detection algorithm, and resumes operation if a miss or a false alarm occurs at the completion of *endpoint_detector.m*. This sub-function uses the median filtered short-time energy quantity to find the utterance end points.
- *endpoint_refine*: Refines the initial end point estimates returned by the energy-based end point detection algorithm (*endpoint_detection.m*), as the secondary measure. This function uses the energy-entropy feature (EEF).

b. Feature Extraction

- *Feature_extract.m*: Extracts both the MFCCs and RC coefficients for each segmented utterance, and saves all extracted MFCC and RC features in two separate *mat* files for use in the classification stage.
- *melcepstrum.m*: Computes the MFCCs of an input speech signal (From [Brookes, 1997]).
- *melbank.m*: Creates the conceptual triangular filters that are used as the filterbank to compute the MFCCs (From [Brookes, 1997]).

c. Recognition

- *BPNN_MFCC_2_Layer.m*: Performs the word recognition task. This program contains the $(150-7)$ network that uses the MFCCs as its input features.
- *BPNN_MFCC_3_Layer.m*: Performs the word recognition task. This program contains the $(60-40-7)$ network that uses the MFCCs as its input features.

2. MATLAB CODE LISTING

- ***speech_segmentation.m:***

```
% ********************************************************************
% Filename            : speech_segmentation.m
%                       (Speech Segmentation Main Program)
% Thesis Advisor      : Prof. Monique P. Fargues
% Author              : LTJG Gokhan Bulbuller
%                       Turkish Navy
% Date                : November, 2005
% Description         : The main program for the speech segmentation task. It takes a
%                       recorded speech file from the folder that belongs to a word
%                       uttered by a subject in the "Split" folder, passes it through an
%                       elliptical bandpass IIR filter, finds the utterance end points, crops
%                       the utterance from the detected end points, and saves the
%                       segmented utterance into a folder belonging to the same word
%                       uttered by the same subject in the "Crop" folder.
% Inputs              : Recorded utterances in the database.
% Outputs             : Segmented utterances.
% Functions Used      : energy_threshold.m, endpoint_detection.m, endpoint_refine.m.
% ********************************************************************

clear all, close all, clc,

Main_loc = ['C:\Documents and Settings\Owner\My Documents\thesis\ear_mic_data\
data\' ] ;
```

```
% Specify the full path names for subjects, recording dates associated with subjects, and
% the vocabulary words.

path_name(1,1)  = { Main_loc '1'  } ;
path_name(2,1)  = { Main_loc '2'  } ;
path_name(3,1)  = { Main_loc '3'  } ;
path_name(4,1)  = { Main_loc '4'  } ;
path_name(5,1)  = { Main_loc '5'  } ;
path_name(6,1)  = { Main_loc '6'  } ;
path_name(7,1)  = { Main_loc '7'  } ;
path_name(8,1)  = { Main_loc '8'  } ;
path_name(9,1)  = { Main_loc '9'  } ;
path_name(10,1) = { Main_loc '10' } ;
path_name(11,1) = { Main_loc '11' } ;
path_name(12,1) = { Main_loc '12' } ;
path_name(13,1) = { Main_loc '13' } ;
path_name(14,1) = { Main_loc '14' } ;
path_name(15,1) = { Main_loc '15' } ;
path_name(16,1) = { Main_loc '16' } ;
path_name(17,1) = { Main_loc '17' } ;
path_name(18,1) = { Main_loc '18' } ;
path_name(19,1) = { Main_loc '19' } ;
path_name(20,1) = { Main_loc '20' } ;

date(1,1)  = { '04_13_05' } ;
date(2,1)  = { '04_25_05' } ;
date(3,1)  = { '04_25_05' } ;
date(4,1)  = { '04_28_05' } ;
date(5,1)  = { '04_28_05' } ;
date(6,1)  = { '04_29_05' } ;
date(7,1)  = { '04_29_05' } ;
date(8,1)  = { '04_29_05' } ;
date(9,1)  = { '04_29_05' } ;
date(10,1) = { '05_02_05' } ;
date(11,1) = { '05_02_05' } ;
date(12,1) = { '05_02_05' } ;
date(13,1) = { '05_04_05' } ;
date(14,1) = { '05_05_05' } ;
date(15,1) = { '05_12_05' } ;
date(16,1) = { '08_25_05' } ;
date(17,1) = { '08_22_05' } ;
date(18,1) = { '08_22_05' } ;
date(19,1) = { '08_22_05' } ;
date(20,1) = { '08_25_05' } ;
```

```
word(1,1) = { 'up'    } ;
word(2,1) = { 'down'  } ;
word(3,1) = { 'left'  } ;
word(4,1) = { 'right' } ;
word(5,1) = { 'kill'  } ;
word(6,1) = { 'pan'   } ;
word(7,1) = { 'move'  } ;
word(8,1) = { 'kill_noise'    } ;
word(9,1) = { 'right_outside' } ;

fs = 8e3 ;      % Sampling Frequency used for recordings.

% -------- Preprocessing Stage ----------
% Elliptical bandpass IIR filter
% Passband = 150 - 2300 Hz , and Transition Band = 50 Hz (on both sides).

[m,Wn] = ellipord([0.0375 0.575],[0.025 0.5875],1,50) ;
[b,a] = ellip(m,1,50,Wn) ;

% ------------------------------------------------

% The loop below implements the speech segmentation task for the whole data set. The
% outer loop is for the 20 subjects, and the inner loop is for nine words uttered by each
% subject.

for i = 1:20

for j = 1:9

% Define the Split folder location to be opened, the folder location to be created to save
% the segmented utterances.

open_loc  = [ path_name{i,1} '\Split\' word{j,1} ] ;
save_loc = [ path_name{i,1} '\Cropped\' word{j,1} ] ;

mkdir(save_loc) ;

Open_dir = dir(open_loc) ;              % List the contents of the folder.
[m,n] = size(Open_dir) ;          % Size of the folder.

% Implement the end point detection and segmentation for each recorded utterance.

for trial=1:(m-2)

% Specify the file to be opened of the form:
%                 "word_subject #_trial #_month_day_year.txt."
```

```
open_split_file_loc = [open_loc,'\',word{j,1},'_',num2str(i),'_',...
                                        num2str(trial),'_',date{i,1},'.txt'];

fid = fopen(open_split_file_loc,'r');                % Open text file for read.

%If the maximum number of recorded utterances in the folder is exceeded, then
%terminate the loop.

if fid == -1
   break ;
end

sig_str = fscanf(fid,'%c');            % Read the file as string characters, and
sig = str2num(sig_str);                % Convert it to a numeric vector.

if size(sig,1) ~= 1, sig = sig' ; end        % Work with row vectors.

sig = sig - mean(sig) ;      % Remove the DC Offset from the signal.

sig_filtered = filter(b,a,sig) ; % Pass the signal through the bandpass filter.

sig1 = sig ; sig = sig_filtered ;

N = fs/100 ;           % Total number of samples in each overlapping frame.
n = floor(2*length(sig)/N) ;          % Total number of overlapping frames.

%Calculate the silence energy, computed portion of the energy curve, upper and lower
%thresholds, starting frame number for end point detection algorithm, and related index
%numbers.

[E,E_silence,ITU,ITL,start,p,k] = energy_threshold(sig,N,n) ;

% Implement the end point detection algorithm using the short-time absolute magnitude
% energy parameter.

[startpt,endpt,E] = endpoint_detection(sig,ITL,ITU,E,start,p,k,N,n) ;

% Refine the end points using energy-entropy feature (EEF).

[startpt_rev,endpt_rev] = endpoint_refine (sig,startpt,endpt,E,N,n) ;

% Crop the recording from the end points detected by the algorithm employed above to
% isolate the utterance from silence sections.

sig_chopped = sig(startpt_rev:endpt_rev) ;
```

```
sig_chop_dur (trial) = ( (endpt_rev - startpt_rev) / N ) * 10 ; % Segmented utterance durations.

% Specify the location to save the segmented utterance.

cropped_loc = [save_loc,'\',word{j,1},'_',num2str(i),'_',...
                                    num2str(trial),'_',date{i,1},'.txt'];

save(cropped_loc,'sig_chopped','-ASCII') ;              % Save the segmented utterance.

end

end

end

% ********************** END OF MAIN PROGRAM **********************
```

- ***energy_threshold.m:***

```
function [E,E_silence_20,E_silence,ITU,ITL,start,p,k] = energy_threshold(sig,N,n)
% ************************************************************************
% Filename            : energy_threshold.m
%                       (Function)
% Thesis Advisor      : Prof. Monique P. Fargues
% Author              : LTJG Gokhan Bulbuller
%                       Turkish Navy
% Date                : October, 2005
% Description         : Threshold function for the energy-based end point detection
%                       algorithm.
% Inputs              : 1. sig : Filtered speech signal,
%                       2. N   : Total number of samples in a frame.
%                       3. n   : Total number of overlapping frames.
% Outputs             : 1. ITL              : Lower Threshold,
%                       2. ITU              : Upper Threshold,
%                       3. E                : Short-time absolute magnitude energy,
%                       4. E_silence_20     : Average silence energy derived from
%                                             the first 20 frames of a recording,
%                       5. E_silence        : Modified average silence energy,
%                       6. start            : Frame index to start the end point search,
%                       7. p and k          : Sample points to start to compute the
%                                             energy in the end point detection
%                                             algorithm.
% Functions Used      : N/A
% ************************************************************************
```

```
p = 1 ; k = N ;      % Edges of the first frame.

% Compute the short-time absolute magnitude energies of the first 20 frames.

for i=1:20
   E(i) = sum(abs(sig(p:k))) ;
   k = k + 40 ; p = p + 40 ;        % Shift frame to the right.
   if k > length(sig)
     sig(length(sig):k) = 0 ;
   end
end

start = i ;

E_silence = (1/20) * sum(E) ;      % Average silence energy.
E_silence_20 = E_silence ;

% Modify E_silence if necessary.

if (3.0 < E_silence & E_silence < 4.5) | (E_silence == 0)
  for i= 21:60
      E(i) = sum(abs(sig(p:k))) ;
      k = k + 40 ; p = p + 40 ;
      if k > length(sig)
        sig(length(sig):k) = 0 ;
      end
      E_silence = (1/20) * sum(E((i - 19) : i)) ;
      start = i ;
      if E_silence < 2.0
        break
      end
  end
end

% Set the upper and lower thresholds, ITL and ITU, with respect to E_silence.

if E_silence >= 3.0 & E_silence < 4.5
   ITL = 1.2 * E_silence ;
   ITU = 3 * E_silence ;
elseif 2.0 <= E_silence & E_silence < 3.0
   ITL = E_silence ;
   ITU = 2.5 * E_silence ;
elseif 1.0 < E_silence & E_silence < 2.0
   ITL = 1.5 * E_silence ;
   ITU = 3 * E_silence ;
```

```
elseif E_silence <= 1.0 & E_silence >= 0.5
   ITL = 2 * E_silence ;
   ITU = 4 * E_silence ;
elseif E_silence >= 0.25 & E_silence < 0.5
   ITL = 2 * E_silence ;
   ITU = 3 * ITL ;
elseif E_silence < 0.25
   ITL = 3 * E_silence ;
   ITU = 4 * ITL ;
end

if E_silence >= 4.5
   E_silence = 0.5 ;
   ITL = 3 * E_silence
   ITU = 6 * E_silence ;
   start = 1 ;
end

% ************************ END OF FUNCTION ************************
```

- ***endpoint_detection.m:***

```
function [startpt,endpt,E] = endpoint_detection(sig,ITL,ITU,E,start,p,k,N,n)
% ***********************************************************************
% Filename          : endpoint_detection.m
%                     (Function)
% Thesis Advisor    : Prof. Monique P. Fargues
% Author            : LTJG Gokhan Bulbuller
%                     Turkish Navy
% Date              : November, 2005
% Description       : Energy-based end point detection algorithm that finds the initial
%                     end point estimates of an utterance and computes the utterance
%                     energy.
% Inputs            : 1. sig        : Filtered speech signal,
%                     2. E          : Short-time absolute magnitude energy of the
%                                     recording computed up to sample index k,
%                     3. ITL        : Lower Threshold,
%                     4. ITU        : Upper Threshold,
%                     5. N          : Total number of samples in a frame,
%                     6. n          : Total number of overlapping frames.
%                     7. start      : Frame index to start the end point search,
%                     8. p and k    : Sample indices to start to compute energy.
% Outputs           : 1. startpt    : Initial estimate of the speech start point,
%                     2. endpt      : Initial estimate of the speech end point,
%                     3. E          : Short-time absolute magnitude energy.
% Functions Used    : endpoint_detector.m, endpoint_correct.m (Sub-functions)
```

```
% ************************************************************************

start2 = start ; endfr = 0 ;

% Conduct the end point detection algorithm, and find the initial edge point estimates.

[start,endfr,E,p,k,i] = endpoint_detector (sig,ITL,ITU,E,start,endfr,p,k,n,N) ;

% If a "miss" has occured (the first condition below), then pass the energy function
% through a fifth order median filter, and repeat the end point detection scheme with
% "endpoint_correct.m". Otherwise, compute the rest of the energy curve.

if ( start == n-1 ) & ( i == n )        % ==> Then, a "miss" has occured!
   sig( length(sig):k ) = 0 ;
   E (i) = sum( abs(sig(p:k)) ) ;
   E_med = medfilt1 (E,5) ;
   [start,endfr] = endpoint_correct (E_med,ITL,ITU,n,N,start2) ;
else
    while ( start ~= n-1 ) & ( i <= n )        % ==> Detection case,
      E(i) = sum(abs(sig(p:k))) ;                    % Continue the normal operation.
      k = k + 40 ; p = p + 40 ;
      if k > length(sig)
        sig(length(sig):k) = 0 ;
      end
      i = i + 1 ;
    end
    E_med = medfilt1 (E,5) ;             % Median filter the energy function.
end

% Find the maximum of median filtered energy curve between the initial end point
% estimates.

max_chop = max( E_med(start:endfr) ) ;

% If a false alarm occurs (i.e., a nonspeech segment was declared as speech), then repeat
% the detection scheme starting from the previously declared speech ending point.

while max_chop < max(E_med)        % ==> False Detection
  start_old = start ; endfr_old = endfr ;             % Keep the old estimates.
  start2 = endfr ;
  [start,endfr] = endpoint_correct (E_med,ITL,ITU,n,N,start2) ;
  max_chop = max( E(start:endfr) ) ;
  if start > endfr                                % In case of error, use the old estimates.
     start = start_old ; endfr = endfr_old ;
  end
end
```

```
startpt = start * (N/2) ;  % Initial estimate of the utterance start point detected by the
                           % energy-based algorithm.

endpt = endfr * (N/2) ;  % Initial estimate of the utterance end point detected by the
                         % detected by the energy-based algorithm.

% ********************** END OF MAIN FUNCTION ***********************
```

- ***endpoint_detector.m:***

```
function [start,endfr,E,p,k,i] = endpoint_detector(sig,ITL,ITU,E,start,endfr,p,k,n,N)
% ********************************************************************
% Filename            : endpoint_detector.m
%                       (Sub-function)
% Thesis Advisor      : Prof. Monique P. Fargues
% Author              : LTJG Gokhan Bulbuller
%                       Turkish Navy
% Date                : November, 2005
% Description         : The first sub-function used by the main function.
%                       This sub-function implements the actual energy-based end point
%                       detection algorithm, and returns the initial end point estimates
%                       together with the utterance energy.
% Inputs              : 1. sig        : Filtered speech signal,
%                       2. E          : Short-time absolute magnitude energy of the
%                                       recording computed up to sample index k,
%                       3. ITL        : Lower Threshold,
%                       4. ITU        : Upper Threshold,
%                       5. N          : Total number of samples in a frame,
%                       6. n          : Total number of overlapping frames,
%                       7. start      : Frame index to start the end point search,
%                       8. endfr      : Frame index for the end point,
%                       9. p and k    : Sample indices to start to compute energy.
% Outputs             : 1. start      : Estimate of the speech start point,
%                       2. endfr      : Estimate of the speech end point,
%                       3. E          : Short-time absolute magnitude energy,
%                       4. p and k    : Sample indices for the frame start and end,
%                       5. i          : Frame index.
% Functions Used      : N/A
% ********************************************************************

t = 0 ;

if start == 1    % Determine the initial starting frame number.
  i = 21 ;
elseif start ~= 1
```

```
  i = start + 1 ;
end

% Search to the right for the preliminary start point.

while i < n
   E(i) = sum(abs(sig(p:k))) ;        % Compute the frame energy.
   k = k + 40 ; p = p + 40 ;       % Shift frame to the right.
   if k > length(sig)
     sig(length(sig):k) = 0 ;
   end
   if E(i) < ITU                % ==> Continue to search for a possible start point.
     start = start + 1 ; i = i + 1 ;
   else
   if E(i) > ITU              % ==> Possible start point.
                               %     Check the next 10 frames if E(i+1:i+10)>ITU.
         for j = (i+1):(i+10)
            E(j) = sum(abs(sig(p:k))) ;
            k = k + 40 ; p = p + 40 ;
            if k > length(sig)
              sig(length(sig):k) = 0 ;
            end
            if E(j) > ITU
              t = t + 1 ;
            end
         end
         if t ~= 10           % ==> Continue to search for a possible start point.
           start = start + 11 ; i = i + 11 ;
           t = 0 ;
         end
   end
   end
         if t == 10                % ==> Preliminary start point. Terminate the loop.
           start = start + 1 ; break
         end
end

% Move backwards to find where the energy first goes under ITL. Mark it as the initial
% utterance start point.

while E(start) > ITL
  start = start - 1;
  if start == 1
     break
  end
end
```

```
if ( start == n-1 ) & ( i == n )          % ==> A "miss" has occured!
   detect_case = 0 ;
else detect_case = 1 ;                % ==> Detection case.
end

switch detect_case
   case (1)                  % ==> Continue to search for the utterance end point.

% Search to the right for the preliminary end point starting from the point where the
% algorithm has stopped the energy computation.

t = 0 ; endfr = j ; i = j + 1 ;            % ==> Set the starting frame index.

while i < n
   E(i) = sum(abs(sig(p:k))) ;         % Compute the frame energy.
   k = k + 40 ; p = p + 40 ;          % Shift frame to the right.
   if k > length(sig)
     sig(length(sig):k) = 0 ;
   end
   if E(i) > ITU            % ==> Continue to search for a possible end point.
      endfr = endfr + 1 ; i = i + 1 ;
   else
   if E(i) < ITU            % ==> Possible end point.
                             %     Check the next 10 frames if E(i+1:i+10)>ITU.
        for j = (i+1):(i+10)
           E(j) = sum(abs(sig(p:k))) ;
           k = k + 40 ; p = p + 40 ;
           if k > length(sig)
             sig(length(sig):k) = 0 ;
           end
           if E(j) > ITU
             t = t + 1 ;
           end
        end
        if t ~= 10              % ==> Preliminary end point. Terminate the loop.
          endfr = endfr + 1 ; break
        end
        if t == 10              % ==> Continue to search for a possible end point.
          endfr = endfr + 11 ; i = i + 11 ;
          t = 0 ;
        end
   end
   end
end
```

```
% Move forwards to find where the energy first goes under ITL. Mark it as the initial
% utterance end point.

while E(endfr) > ITL
  endfr = endfr + 1;
  if endfr > j            % ==> If frame index exceeds the computed energy index.
     E(endfr) = sum(abs(sig(p:k))) ;
     k = k + 40 ; p = p + 40 ;
     if k > length(sig)
       sig(length(sig):k) = 0 ;
     end
  end
end

% Adjust frame index numbers for subsequent computations.

if endfr <= j
   i = j + 1 ;
else i = endfr + 1 ;
end

case (0)              % ==> "Miss case". Do not conduct the end point search,
                      %     and return to the main function.
    endfr = 0 ;
end

% ************************* END OF FUNCTION *************************
```

- ***endpoint_correct.m:***

```
function [start,endfr] = endpoint_correct (E_med,ITL,ITU,n,N,start2)
% ********************************************************************
% Filename            : endpoint_correct.m
%                       (Sub-function)
% Thesis Advisor      : Prof. Monique P. Fargues
% Author              : LTJG Gokhan Bulbuller
%                       Turkish Navy
% Date                : November, 2005
% Description         : The second sub-function used by the main function, which
%                       complements the energy-based end point detection scheme.
%                       This sub-function resumes operation if a "miss" or "false alarm"
%                       occurs at the completion of energy-based end point detection
%                       function. It searches for the end points using the median filtered
%                       energy function. Corrected or detected edge points are returned to
%                       the main program.
% Inputs              : 1. E_med      : Median filtered energy function,
```

```
%                       2. ITL        : Lower Threshold,
%                       3. ITU        : Upper Threshold,
%                       4. N          : Total number of samples in a frame,
%                       5. n          : Total number of overlapping frames,
%                       6. start2     : Frame index to start the end point search.
% Outputs               : 1. start     : Estimate of the speech start point,
%                       2. endfr      : Estimate of the speech end point.
% Functions Used      : N/A
% ************************************************************************

% Adjust the upper threshold if it exceeds the maximum of the energy function after
% median filtering.

if ITU >= max (E_med)
   ITU = 0.8 * ( max(E_med) - ITL ) ;
end

start = start2 ; endfr = 0 ; i = start2 + 1 ;

t = 0 ;

% Search forwards for the preliminary start point.

while i < n
  if E_med(i) < ITU             % ==> Continue to search for a possible start point.
    start = start + 1 ; i = i + 1 ;
   else
   if E_med(i) > ITU              % ==> Possible start point.
                                  %   Check the next 10 frames if E_med(i+1:i+10)>ITU.
        for j = (i+1):(i+10)
           if E_med(j) > ITU
              t = t + 1 ;
           end
        end
        if t ~= 10            % ==> Continue to search for a possible start point.
          start = start + 11 ; i = i + 11 ;
          t = 0 ;
        end
   end
   end
   if t == 10                   % ==> Preliminary start point. Terminate the loop.
     start = start + 1 ; break
   end
end

% Move backwards to find where the smoothed energy first goes under ITL. Mark it as
```

```
% the initial utterance start point.

while E_med(start) > ITL
  start = start - 1;
  if start == 1
     break
  end
end

% Search forwards for the preliminary end point.

endfr = j ; i = j + 1 ; t = 0 ;

while i < n
   if E_med(i) > ITU             % ==> Continue to search for a possible end point.
      endfr = endfr + 1 ; i = i + 1 ;
   else
   if E_med(i) < ITU               % ==> Possible end point.
                                    %  Check the next 10 frames if E_med(i+1:i+10)>ITU.
         for j = (i+1):(i+10)
            if E_med(j) > ITU
               t = t + 1 ;
            end
         end
         if t ~= 10                 % ==> Preliminary end point. Terminate the loop.
           endfr = endfr + 1 ; break
         end
         if t == 10                 % ==> Continue to search for a possible end point.
           endfr = endfr + 11 ; i = i + 11 ;
           t = 0 ;
         end
   end
   end
end

% Move forwards to find where the smoothed energy first goes under ITL. Mark it as the
% initial utterance end point.

while E_med(endfr) > ITL
  endfr = endfr + 1;
  if endfr == n
     break
  end
end

% ************************** END OF FUNCTION **************************
```

- ***endpoint_refine.m:***

```
function [startpt_rev,endpt_rev] = endpoint_refine (sig,startpt,endpt,E,N,n)
% ********************************************************************
% Filename            : endpoint_refine.m
%                       (Function)
% Thesis Advisor      : Prof. Monique P. Fargues
% Author              : LTJG Gokhan Bulbuller
%                       Turkish Navy
% Date                : November, 2005
% Description         : Secondary measure to refine the initial end point estimates
%                       obtained from the energy-based end point detection algorithm
%                       (endpoint_detection.m). It relies on the energy-entropy feature
%                       (EEF).
% Inputs              : 1. sig          : Filtered speech signal,
%                       2. E            : Short-time absolute magnitude energy,
%                       3. startpt      : Initial estimate of the speech start point,
%                       4. endpt        : Initial estimate of the speech end point,
%                       5. N            : Total number of samples in a frame,
%                       6. n            : Total number of overlapping frames.
% Outputs             : 1. startpt_rev : Final (refined) speech start point,
%                       2. endpt_rev  : Final (refined) speech end point.
% Functions Used      : N/A
% ********************************************************************

start = startpt / (0.5*N) ;          % Frame indices of utterance boundaries.
endfr = endpt / (0.5*N) ;

start_rev = start ; end_rev = endfr ;        % Initial search points.

% Below loop computes the entropy of the whole recording.

p = 1 ; k = N ;

for i=1:n
   sig_fft = fft(sig(p:k),512) ;              % 512-point FFT of a speech frame.
   sig_fft_1 = sig_fft(18:127) ;          % Retain only 150 Hz <= f <= 2300 Hz.
   norm_fft = sum ( abs(sig_fft_1(:)).^2 ) ;          % Spectral energy.
   pp = abs( sig_fft_1(:) ).^2 / norm_fft ;          % PDF estimation.
   for j = 1:110
      if pp (j) >= 0.9 | pp (j) == 0.0
         pp (j) = 1e-12 ;
      end
   end
   pk = log10 ( pp ) ;
```

```
   H (i) = - ( sum ( pp .* pk) ) ;          % Entropy of a frame
   k = k + 40 ; p = p + 40 ;            % Shift frame to the right.
   if k > length(sig)
     sig(length(sig):k) = 0 ;
   end
end

EEF = sqrt ( 1 + ( E .* H ) ) ;          % Compute the EEF.

EEF_silence = (1/20) * sum( EEF (1:20) ) ;            % Average silence EEF.

% Set the thresholds with respect to the average silence EEF.

EETL1 = 1.2 * EEF_silence ;

EETL2 = 2 * EEF_silence ;

i = start - 1 ;              % Initial search point for refining the start point.

% Search back only 5 frames to refine the speech start point.

while ( i > 20 ) & ( i >= (start - 5) )
   if EEF (i) < EETL1          % ==> Continue to search.
      i = i - 1 ;
   elseif EEF (i) > EETL1        % ==> Shift the initial start point to left.
      start_rev = i ;
      i = i - 1 ;
   end
end

startpt_rev = start_rev * (0.5*N) ;            % Refined speech start point.

i = endfr + 1 ;          % Initial search point for refining the end point.

% Search forwards 70 frames to refine the speech end point.

while ( i < (n-20) ) & ( i <= ( endfr + 70 ) )
   if EEF (i) < EETL2             % ==> Continue to search.
      i = i + 1 ;
   end
   if EEF (i) > EETL2     % Possible end point. Check the next 10 frames.
      for j=(i+1):(i+10)
         if EEF (j) > EETL2
           end_rev = j + 1 ;          % ==> Shift the initial end point to right.
         end
      end
```

```
   break
   end
end

endpt_rev = end_rev * (0.5*N) ;            % Refined speech end point.

% ************************* END OF FUNCTION **************************
```

- ***Feature_extract.m:***

```
% ********************************************************************
% Filename            : Feature_extract.m
%                       (Feature Extraction Main Program)
% Thesis Advisor      : Prof. Monique P. Fargues
% Author              : LTJG Gokhan Bulbuller
%                       Turkish Navy
% Date                : November 27, 2005 17:30 PST
% Description         : The main program for feature extraction.
%                       It takes a segmented (end point detected and cropped) speech
%                       from the folder that belongs to a word uttered by a subject in the
%                       "Cropped" folder. 15 MFCCs (including the first coefficient) and
%                       15 RC coefficients are extracted for each segmented utterance.
%                       These MFCCs and RC coefficients are stored in separate
%                       20*9 cell arrays, and saved as individual "mat" files for use in
%                       classification stage.
% Inputs              : Segmented utterances in the "Cropped" folder.
% Outputs             : 1. RC_Features.mat,
%                       2. MFCC_Features.mat.
% Functions Used      : melcepstrum.m (modified from Voicebox)
% ********************************************************************

clear all, close all, clc,

Main_loc = [ 'C:\Documents and Settings\Owner\My Documents\thesis' ] ;
Main_loc_2 = [ 'C:\Documents and Settings\Owner\My Documents\thesis\ear_mic_data\
data\' ] ;

% Specify the full path names for subjects, recording dates associated with subjects, and
% the vocabulary words.

path_name(1,1)  = { Main_loc_2 '1'  } ;
path_name(2,1)  = { Main_loc_2 '2'  } ;
path_name(3,1)  = { Main_loc_2 '3'  } ;
path_name(4,1)  = { Main_loc_2 '4'  } ;
path_name(5,1)  = { Main_loc_2 '5'  } ;
path_name(6,1)  = { Main_loc_2 '6'  } ;
```

```
path_name(7,1)  = { Main_loc_2 '7'  } ;
path_name(8,1)  = { Main_loc_2 '8'  } ;
path_name(9,1)  = { Main_loc_2 '9'  } ;
path_name(10,1) = { Main_loc_2 '10' } ;
path_name(11,1) = { Main_loc_2 '11' } ;
path_name(12,1) = { Main_loc_2 '12' } ;
path_name(13,1) = { Main_loc_2 '13' } ;
path_name(14,1) = { Main_loc_2 '14' } ;
path_name(15,1) = { Main_loc_2 '15' } ;
path_name(16,1) = { Main_loc_2 '16' } ;
path_name(17,1) = { Main_loc_2 '17' } ;
path_name(18,1) = { Main_loc_2 '18' } ;
path_name(19,1) = { Main_loc_2 '19' } ;
path_name(20,1) = { Main_loc_2 '20' } ;

date(1,1)  = { '04_13_05' } ;
date(2,1)  = { '04_25_05' } ;
date(3,1)  = { '04_25_05' } ;
date(4,1)  = { '04_28_05' } ;
date(5,1)  = { '04_28_05' } ;
date(6,1)  = { '04_29_05' } ;
date(7,1)  = { '04_29_05' } ;
date(8,1)  = { '04_29_05' } ;
date(9,1)  = { '04_29_05' } ;
date(10,1) = { '05_02_05' } ;
date(11,1) = { '05_02_05' } ;
date(12,1) = { '05_02_05' } ;
date(13,1) = { '05_04_05' } ;
date(14,1) = { '05_05_05' } ;
date(15,1) = { '05_12_05' } ;
date(16,1) = { '08_25_05' } ;
date(17,1) = { '08_22_05' } ;
date(18,1) = { '08_22_05' } ;
date(19,1) = { '08_22_05' } ;
date(20,1) = { '08_25_05' } ;

word(1,1) = { 'up'    } ;
word(2,1) = { 'down'  } ;
word(3,1) = { 'left'  } ;
word(4,1) = { 'right' } ;
word(5,1) = { 'kill'  } ;
word(6,1) = { 'pan'   } ;
word(7,1) = { 'move'  } ;
word(8,1) = { 'kill_noise'    } ;
word(9,1) = { 'right_outside' } ;
```

```
fs = 8e3 ;                % Sampling Frequency used for recordings

% 20*9 cell arrays for storing the MFCCs and RCs.

MFCC_Feature_cell(20,9) = {[]} ;
RC_Feature_cell(20,9)       = {[]} ;

% Below loop implements the feature extraction for the whole segmented data set.
% The outer loop is for the 20 subjects, and the inner loop is for 9 words uttered by each
% subject.

for i = 1:20

for j = 1:9

% Define the cropped folder location to be opened.

open_loc = [ path_name{i,1} '\Cropped\' word{j,1} '\'] ;

Open_dir = dir(open_loc) ;          % List the content of the folder.
[m,n] = size(Open_dir) ;        % Size of the folder.

% Extract both features (MFCCs and RCs) for each segmented utterance in a segmented
% word folder belonging to a subject.

for k = 1:(m-2)

% Specify the cropped file to be opened of the form:
%                          "word_subject #_trial #_month_day_year.txt."

open_cropped_file_loc = [open_loc,word{j,1},'_',num2str(i),'_',...
                                                num2str(k),'_',date{i,1},'.txt'] ;

fid = fopen(open_crop_file_loc,'r') ;          % Open text file for read.

% If the maximum number of cropped utterances in the folder is exceeded, then terminate
% the loop.

if fid == -1
   break ;
end

sig_str = fscanf(fid,'%c');              % Read the file as string characters, and
sig = str2num(sig_str);                  % convert it to a numeric vector.

status = fclose(fid) ;          % Close the text file.
```

```
if size(sig,1) ~= 1, sig = sig' ; end          % Work with row vectors.

c_mel(:,k) = ( melcepstrum (sig,fs,'Ratz0',14,20,length(sig),0,0,0.5) )' ;

cc = rceps(sig) ; c_rc(:,k) = cc(1:15)' ;

end

MFCC_Feature_cell(i,j) = { c_mel } ;
RC_Feature_cell(i,j)       = { c_rc } ;

clear c_mel c_rc ;

end

end

save_loc_1 = [ Main_loc '\' 'MFCC_Features.mat' ] ;
save(save_loc_1,'MFCC_Feature_cell') ;              % Save the MFCCs for classification.

save_loc_2 = [ Main_loc '\' 'RC_Features.mat' ] ;
save(save_loc_2,'RC_Feature_cell') ;                % Save the RCs for classification.

% This section computes the overall data size by counting the feature vectors in one of
% the feature cells computed above.

data_size = 0 ;

for i=1:20
   for j=1:9
      data_size = data_size + size(MFCC_Feature_cell{i,j},2) ;
   end
end

disp('Overall Data Size = ') ; disp(data_size) ;

% ********************** END OF MAIN PROGRAM **********************
```

- ***melcepstrum.m:***

```
function c_mel = melcepstrum (sig,fs,w,nc,p,n,inc,fl,fh)
% ************************************************************************
% Filename            : melcepstrum.m
%                       (Function)
% Thesis Advisor      : Prof. Monique P. Fargues
```

```
% Author               : Mike Brookes, (melcepst.m, v 1.3 2005/02/21 15:22:13)
%                        http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/ voicebox.html
%                        (Last accessed on November 05, 2005)
% Modified by          : LTJG Gokhan Bulbuller
%                        Turkish Navy
% Date Modified        : November 2005
% Description          : Function for extracting the MFCC features.
%                        It computes the MFCCs for a given segmented utterance by
%                        assuming the whole segmented utterance as one rectangular frame
%                        of speech. The triangular filters are used as the filterbank for
%                        feature extraction.
% Inputs               : 1. sig        : Segmented speech signal,
%                        2. fs         : Sampling frequency in Hz (default fs=8e3 Hz),
%                        3. nc         : Number of MFCCs excluding the first coefficient
%                                        (default 14),
%                        4. p          : Number of filters in filterbank (default 14),
%                        5. n          : Frame length (default length(sig)),
%                        6. inc        : Frame increment (default 0),
%                        7. fl         : Low end of the lowest filter as a fraction of fs
%                                        (default 0),
%                        8. fh         : High end of highest filter as a fraction of fs
%                                        (default 0.5),
%                        9. w          : Any sensible combination of the following:
%
%                               'R'     Rectangular window in time domain (default).
%                               't'     Triangular shaped filters in mel domain (default).
%                               'p'     Filters act in the power domain.
%                               'a'     Filters act in the absolute magnitude domain
%                                       (default).
%                               '0'     Include 0'th order cepstral coefficient.
%                               'e'     Include log energy.
%                               'z'     Highest and lowest filters taper down to zero
%                                       (default).
% Outputs              : c_mel : MFCCs of the segmented utterance.
% Functions Used       : melbank.m (modified from Voicebox)
% ********************************************************************

% Set the default values.

if nargin<2 fs = 8e3    ;   end
if nargin<3 w = 'Ratz0' ;   end
if nargin<4 nc = 14     ;   end
if nargin<5 p = 14      ;   end
if nargin<6 n = length(sig) ; end
if nargin<9
  fh = 0.5 ;
```

```
  if nargin<8
    fl = 0 ;
    if nargin<7
      inc = 0 ;
    end
  end
end

f = fft(sig') ;

[m,a,b] = melbank(p,n,fs,fl,fh,w) ;          % Create the conceptual filterbank.

pw = f(a:b) .* conj(f(a:b)) ;          % Signal power.

pth = max(pw) * 1E-6 ;             % Small control value to avoid "log (zero)".

% Compute the logarithm of the spectral energies of each conceptual filter output.

if any(w=='p')               % ==> Use the power domain for feature extraction.
  y = log( max(m*pw,pth) ) ;
else                     % ==> Use the absolute magnitude domain for feature extraction.
  ath = sqrt( pth ) ;                  % Small control value to avoid "log (zero)".
  y = log( max( m*abs(f(a:b)) , ath ) ) ;
end

c_mel = dct(y)' ;              % The MFCCs of the utterance.

nc = nc + 1 ;
nf = 1 ;

% Adjust the coefficients.

if p > nc                % ==> Discard the extra coefficients.
  c_mel(nc+1:end) = [] ;
elseif p < nc                % ==> Add zeros to the coefficient vector.
  c_mel = [ c_mel 0 ] ;
end

if ~any(w=='0')              % ==> Discard the 0'th order coefficient if not requested.
  c_mel(1) = [] ;
  nc = nc - 1 ;
end

if any(w=='e')                   % ==> Include the signal log energy if requested.
  c_mel = [ log(sum(pw)).' c_mel ];
  nc = nc + 1 ;
```

```
end

% ************************* END OF FUNCTION *************************
```

- ***melbank.m:***

```
function [x,mn,mx]=melbank(p,n,fs,fl,fh,w)
% ********************************************************************
% Filename            : melbank.m
%                       (Function)
% Thesis Advisor      : Prof. Monique P. Fargues
% Author              : Mike Brookes, (melbankm.m,v 1.3 2005/02/21 15:22:13)
%                       http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/ voicebox.html
%                       (Last accessed on November 05, 2005)
% Modified by         : LTJG Gokhan Bulbuller
%                       Turkish Navy
% Date Modified       : November 2005
% Description         : Creates the conceptual triangular shaped filters to be used as
%                       filterbank for MFCC extraction by "melcepstrum.m".
% Inputs              : 1. p    : Number of filters in filterbank,
%                       2. n    : Frame length (default length(sig)),
%                       3. fs   : Sampling frequency in Hz (default fs=8e3),
%                       4. fl   : Low end of the lowest filter as a fraction of fs (default 0),
%                       5. fh   : High end of highest filter as a fraction of fs (default 0.5),
%                       6. w    : any sensible combination of the following:

%                                 't'     Triangular shaped filters in mel domain (default).
%                                 'z'     Highest and lowest filters taper down to zero
%                                         (default).
% Outputs             : 1. x    : Sparse matrix containing the filterbank amplitudes,
%                       2. mn   : The lowest fft bin with a non-zero coefficient,
%                       3. mx   : The highest fft bin with a non-zero coefficient.
% Functions Used      : N/A
% ********************************************************************

% Set the default values.

if nargin<2 n = length(sig) ; end
if nargin<3 fs = 8e3       ; end
if nargin<4 fl = 0         ; end
if nargin<5 fh = 0.5       ; end
if nargin<6 w='tz'         ; end

f0=700/fs;
fn2=floor(n/2);
lr=log((f0+fh)/(f0+fl))/(p+1);
```

```
% Convert to FFT bin numbers with 0 for DC term.

bl=n*((f0+fl)*exp([0 1 p p+1]*lr)-f0);
b2=ceil(bl(2));
b3=floor(bl(3));

b1=floor(bl(1))+1;
b4=min(fn2,ceil(bl(4)))-1;
pf=log((f0+(b1:b4)/n)/(f0+fl))/lr;
fp=floor(pf);
pm=pf-fp;
k2=b2-b1+1;
k3=b3-b1+1;
k4=b4-b1+1;
r=[fp(k2:k4) 1+fp(1:k3)];
c=[k2:k4 1:k3];
v=2*[1-pm(k2:k4) pm(1:k3)];
mn=b1+1;
mx=b4+1;


if nargout > 1
  x=sparse(r,c,v);
else
  x=sparse(r,c+mn-1,v,p,1+fn2);
end

% ************************ END OF FUNCTION **************************
```

- ***BPNN_MFCC_2_Layer.m:***

```
% ********************************************************************
% Filename            : BPNN_MFCC_2_Layer.m
%                       (Recognition Main Program)
% Thesis Advisor      : Prof. Monique P. Fargues
% Author              : LTJG Gokhan Bulbuller
%                       Turkish Navy
% Date                : January 14, 2006 21:30 PST
% Description         : The main program for word recognition.
%                       The (150-7) network implemented below is one of the ten
%                       configurations considered for recognition purposes. MFCCs
%                       stored in "MFCC_Features.mat" are used as input features. The
%                       features. The network is iterated 80 times on data by randomly
%                       selecting the training set each time, and results are saved in
%                       related ".mat" files.
```

```
% Inputs              : MFCC_Features.mat (14 Dimensional MFCCs)
% Outputs             : 1. Results_MEL_2_150.mat : Contains all the confusion matrices
%                                                  for every individual iteration,
%                       2. Conf_MEL_2_150.mat    : Contains the overall average
%                                                  confusion matrices of all 80
%                                                  iterations,
%                       3. Overall_Class_train   : The overall average classification
%                                                  rate for the training data set,
%                       4. Overall_Class_test    : The overall average classification
%                                                  rate for the testing data set.
% Functions Used      : N/A
% ************************************************************************

clear all, close all, clc,

Main_loc = [ 'C:\Documents and Settings\Owner\My Documents\thesis' ] ;
Feature_loc = [ Main_loc '\' 'MFCC_Features.mat' ] ;

load (Feature_loc)  ;          % Load the MFCCs of all data set.

results(80,3) = { [] } ;           % Cell array that will store the results.

for trial=1:80

p = randperm(39) ;

%                                  Create the Training Data Set
% To construct the training set, select randomly 15 out of all recorded repetitions of a
% subject for a word. The training matrix size is 14*2100.

train_cell(20,7) = {[]} ;

k = 1:15 ;

for i=1:20
   for j=1:7
      x = MFCC_Feature_cell{i,j} ;
      train_cell(i,j) = { x(2:15,p(k)) } ;
   end
end

cl1 = 1 ; cl2 = 15 ;

for i=1:7
   for j=1:20
      train_mtx(:,cl1:cl2) = train_cell{j,i} ;
```

```
      cl1 = cl1 + 15 ; cl2 = cl2 + 15 ;
   end
end

% Make the training set zero mean and unit variance.

for i=1:14
   train_mtx(i,:) = train_mtx(i,:) - mean(train_mtx(i,:)) ;
   train_mtx(i,:) = train_mtx(i,:)./std(train_mtx(i,:)) ;
end

%                              Create the Testing Data Set
% To construct the testing set, assign all the remaining utterances to the testing set
% matrix.

test_cell(20,7) = {[]} ;

for i=1:20
   for j=1:7
      x = MFCC_Feature_cell{i,j} ;
      X = x(2:15,p(16:end)) ;
      if size(x,2) > 39
         X = [ X x(2:15,40:end) ] ;
      end
      test_cell(i,j) = { X } ;
   end
end

m = 1 ; n = size(test_cell{1,1},2) ;

for i=1:7
   if i ~= 1
      m = n + 1 ; n = n + size(test_cell{1,i},2) ;
   end
   for j=1:20
      test_mtx(:,m:n) = test_cell{j,i} ;
      if j ~= 20
         m = n + 1 ; n = n + size(test_cell{j+1,i},2) ;
      end
   end
   N(i) = n ;
end

% Make the testing set zero mean and unit variance.

for i=1:14
```

```
   test_mtx(i,:) = test_mtx(i,:) - mean(test_mtx(i,:)) ;
   test_mtx(i,:) = test_mtx(i,:)./std(test_mtx(i,:)) ;
end

% Create the target matrix, which is (7*2100). 1-of-7 coding is used to represent each of
% seven words.

tgt = zeros(7,size(train_mtx,2));
tgt(7,1:300) = 1 ; tgt(6,301:600) = 1 ;
tgt(5,601:900) = 1 ; tgt(4,901:1200) = 1 ;
tgt(3,1201:1500) = 1 ; tgt(2,1501:1800) = 1 ;
tgt(1,1801:2100) = 1 ;

% Create, train and test the (150-7) network for word recognition.

z = [ min(train_mtx')' max(train_mtx')' ] ;

net = newff(z,[150,7],{'tansig','logsig'},'traincgf') ;
net.performFcn = 'msereg' ;
net.performParam.ratio = 0.85 ;
net.trainParam.show = 10 ;
net.trainParam.epochs = 1000 ;

[net,tr,Y,E] = train(net,train_mtx,tgt) ;        % Train the network.

A = sim(net,test_mtx) ;            % Test the network.

% Convert the network outputs obtained from training to binary outputs.

Y_bin_out=zeros(size(Y));

for i=1:7
   for j=1:size(train_mtx,2)
      if Y(i,j)==max(Y(:,j))
        Y_bin_out(i,j)=1;
      else Y_bin_out(i,j)=0;
      end
   end
end

% Convert the network outputs obtained from testing to binary outputs.

A_bin_out = zeros(size(A)) ;

for i=1:7
   for j=1:size(test_mtx,2)
```

```
      if A(i,j)==max(A(:,j))
        A_bin_out(i,j)=1;
      else A_bin_out(i,j)=0;
      end
   end
end

% Construct the confusion matrix for the training set, and compute the related average
% classification rate.

M = zeros(7,7) ;
[I,J] = find(Y_bin_out) ;

for i=1:size(Y_bin_out,2)
   if ( i <= 300 )
     if I(i)==1, M(7,1)=M(7,1)+1; end
     if I(i)==2, M(6,1)=M(6,1)+1; end
     if I(i)==3, M(5,1)=M(5,1)+1; end
     if I(i)==4, M(4,1)=M(4,1)+1; end
     if I(i)==5, M(3,1)=M(3,1)+1; end
     if I(i)==6, M(2,1)=M(2,1)+1; end
     if I(i)==7, M(1,1)=M(1,1)+1; end
   end
   if ( i > 300 ) & ( i <= 600 )
     if I(i)==1, M(7,2)=M(7,2)+1; end
     if I(i)==2, M(6,2)=M(6,2)+1; end
     if I(i)==3, M(5,2)=M(5,2)+1; end
     if I(i)==4, M(4,2)=M(4,2)+1; end
     if I(i)==5, M(3,2)=M(3,2)+1; end
     if I(i)==6, M(2,2)=M(2,2)+1; end
     if I(i)==7, M(1,2)=M(1,2)+1; end
   end
   if ( i > 600 ) & ( i <= 900 )
     if I(i)==1, M(7,3)=M(7,3)+1; end
     if I(i)==2, M(6,3)=M(6,3)+1; end
     if I(i)==3, M(5,3)=M(5,3)+1; end
     if I(i)==4, M(4,3)=M(4,3)+1; end
     if I(i)==5, M(3,3)=M(3,3)+1; end
     if I(i)==6, M(2,3)=M(2,3)+1; end
     if I(i)==7, M(1,3)=M(1,3)+1; end
   end
   if ( i > 900 ) & ( i <= 1200 )
     if I(i)==1, M(7,4)=M(7,4)+1; end
     if I(i)==2, M(6,4)=M(6,4)+1; end
     if I(i)==3, M(5,4)=M(5,4)+1; end
     if I(i)==4, M(4,4)=M(4,4)+1; end
```

```
      if I(i)==5, M(3,4)=M(3,4)+1; end
      if I(i)==6, M(2,4)=M(2,4)+1; end
      if I(i)==7, M(1,4)=M(1,4)+1; end
   end
   if ( i > 1200 ) & ( i <= 1500 )
      if I(i)==1, M(7,5)=M(7,5)+1; end
      if I(i)==2, M(6,5)=M(6,5)+1; end
      if I(i)==3, M(5,5)=M(5,5)+1; end
      if I(i)==4, M(4,5)=M(4,5)+1; end
      if I(i)==5, M(3,5)=M(3,5)+1; end
      if I(i)==6, M(2,5)=M(2,5)+1; end
      if I(i)==7, M(1,5)=M(1,5)+1; end
   end
   if ( i > 1500 ) & ( i <= 1800 )
      if I(i)==1, M(7,6)=M(7,6)+1; end
      if I(i)==2, M(6,6)=M(6,6)+1; end
      if I(i)==3, M(5,6)=M(5,6)+1; end
      if I(i)==4, M(4,6)=M(4,6)+1; end
      if I(i)==5, M(3,6)=M(3,6)+1; end
      if I(i)==6, M(2,6)=M(2,6)+1; end
      if I(i)==7, M(1,6)=M(1,6)+1; end
   end
   if ( i > 1800 ) & ( i <= 2100 )
      if I(i)==1, M(7,7)=M(7,7)+1; end
      if I(i)==2, M(6,7)=M(6,7)+1; end
      if I(i)==3, M(5,7)=M(5,7)+1; end
      if I(i)==4, M(4,7)=M(4,7)+1; end
      if I(i)==5, M(3,7)=M(3,7)+1; end
      if I(i)==6, M(2,7)=M(2,7)+1; end
      if I(i)==7, M(1,7)=M(1,7)+1; end
   end
end

Train_Conf_matrix = ( M ./300 ) .* 100 ;
Class_rate_train = sum(diag(Train_Conf_matrix)) / 7 ;

% Construct the confusion matrix for the testing set, and compute the related average
% classification rate.

M_tst = zeros(7,7) ;
[II,JJ] = find(A_bin_out) ;

for i=1:size(A_bin_out,2)
   if ( i <= N(1) )
      if II(i)==1, M_tst(7,1)=M_tst(7,1)+1; end
      if II(i)==2, M_tst(6,1)=M_tst(6,1)+1; end
```

```
   if II(i)==3, M_tst(5,1)=M_tst(5,1)+1; end
   if II(i)==4, M_tst(4,1)=M_tst(4,1)+1; end
   if II(i)==5, M_tst(3,1)=M_tst(3,1)+1; end
   if II(i)==6, M_tst(2,1)=M_tst(2,1)+1; end
   if II(i)==7, M_tst(1,1)=M_tst(1,1)+1; end
end
if ( i > N(1) ) & ( i <= N(2) )
   if II(i)==1, M_tst(7,2)=M_tst(7,2)+1; end
   if II(i)==2, M_tst(6,2)=M_tst(6,2)+1; end
   if II(i)==3, M_tst(5,2)=M_tst(5,2)+1; end
   if II(i)==4, M_tst(4,2)=M_tst(4,2)+1; end
   if II(i)==5, M_tst(3,2)=M_tst(3,2)+1; end
   if II(i)==6, M_tst(2,2)=M_tst(2,2)+1; end
   if II(i)==7, M_tst(1,2)=M_tst(1,2)+1; end
end
if ( i > N(2) ) & ( i <= N(3) )
   if II(i)==1, M_tst(7,3)=M_tst(7,3)+1; end
   if II(i)==2, M_tst(6,3)=M_tst(6,3)+1; end
   if II(i)==3, M_tst(5,3)=M_tst(5,3)+1; end
   if II(i)==4, M_tst(4,3)=M_tst(4,3)+1; end
   if II(i)==5, M_tst(3,3)=M_tst(3,3)+1; end
   if II(i)==6, M_tst(2,3)=M_tst(2,3)+1; end
   if II(i)==7, M_tst(1,3)=M_tst(1,3)+1; end
end
if ( i > N(3) ) & ( i <= N(4) )
   if II(i)==1, M_tst(7,4)=M_tst(7,4)+1; end
   if II(i)==2, M_tst(6,4)=M_tst(6,4)+1; end
   if II(i)==3, M_tst(5,4)=M_tst(5,4)+1; end
   if II(i)==4, M_tst(4,4)=M_tst(4,4)+1; end
   if II(i)==5, M_tst(3,4)=M_tst(3,4)+1; end
   if II(i)==6, M_tst(2,4)=M_tst(2,4)+1; end
   if II(i)==7, M_tst(1,4)=M_tst(1,4)+1; end
end
if ( i > N(4) ) & ( i <= N(5) )
   if II(i)==1, M_tst(7,5)=M_tst(7,5)+1; end
   if II(i)==2, M_tst(6,5)=M_tst(6,5)+1; end
   if II(i)==3, M_tst(5,5)=M_tst(5,5)+1; end
   if II(i)==4, M_tst(4,5)=M_tst(4,5)+1; end
   if II(i)==5, M_tst(3,5)=M_tst(3,5)+1; end
   if II(i)==6, M_tst(2,5)=M_tst(2,5)+1; end
   if II(i)==7, M_tst(1,5)=M_tst(1,5)+1; end
end
if ( i > N(5) ) & ( i <= N(6) )
   if II(i)==1, M_tst(7,6)=M_tst(7,6)+1; end
   if II(i)==2, M_tst(6,6)=M_tst(6,6)+1; end
   if II(i)==3, M_tst(5,6)=M_tst(5,6)+1; end
```

```
      if II(i)==4, M_tst(4,6)=M_tst(4,6)+1; end
      if II(i)==5, M_tst(3,6)=M_tst(3,6)+1; end
      if II(i)==6, M_tst(2,6)=M_tst(2,6)+1; end
      if II(i)==7, M_tst(1,6)=M_tst(1,6)+1; end
   end
   if ( i > N(6) ) & ( i <= N(7) )
      if II(i)==1, M_tst(7,7)=M_tst(7,7)+1; end
      if II(i)==2, M_tst(6,7)=M_tst(6,7)+1; end
      if II(i)==3, M_tst(5,7)=M_tst(5,7)+1; end
      if II(i)==4, M_tst(4,7)=M_tst(4,7)+1; end
      if II(i)==5, M_tst(3,7)=M_tst(3,7)+1; end
      if II(i)==6, M_tst(2,7)=M_tst(2,7)+1; end
      if II(i)==7, M_tst(1,7)=M_tst(1,7)+1; end
   end
end

Test_Conf_matrix(:,1) = ( M_tst(:,1) ./ N(1) ) .* 100 ;
Test_Conf_matrix(:,2) = ( M_tst(:,2) ./ ( N(2)-N(1) ) ) .* 100 ;
Test_Conf_matrix(:,3) = ( M_tst(:,3) ./ ( N(3)-N(2) ) ) .* 100 ;
Test_Conf_matrix(:,4) = ( M_tst(:,4) ./ ( N(4)-N(3) ) ) .* 100 ;
Test_Conf_matrix(:,5) = ( M_tst(:,5) ./ ( N(5)-N(4) ) ) .* 100 ;
Test_Conf_matrix(:,6) = ( M_tst(:,6) ./ ( N(6)-N(5) ) ) .* 100 ;
Test_Conf_matrix(:,7) = ( M_tst(:,7) ./ ( N(7)-N(6) ) ) .* 100 ;

Class_rate_test = sum(diag(Test_Conf_matrix)) / 7 ;

% Test the (150-7) network on "kill_noise" and "right_outside" on which it was not
% trained.

% Construct the new testing set from "kill_noise" and "right_outside".

test_cell_2(20,2) = {[]} ;

for i=1:20
   XX = MFCC_Feature_cell{i,8} ; XX(1,:) = [] ;
   test_cell_2(i,1) = { XX } ;
   XX = MFCC_Feature_cell{i,9} ; XX(1,:) = [] ;
   test_cell_2(i,2) = { XX } ;
end

m = 1 ; n = size(test_cell_2{1,1},2) ;

for i=1:2
   if i ~= 1
      m = n + 1 ; n = n + size(test_cell_2{1,i},2) ;
   end
```

```
   for j=1:20
      test_mtx_2(:,m:n) = test_cell_2{j,i} ;
      if j ~= 20
         m = n + 1 ; n = n + size(test_cell_2{j+1,i},2) ;
      end
   end
   NN(i) = n ;
end

% Make the new testing set zero mean and unit variance.

for i=1:14
   test_mtx_2(i,:) = test_mtx_2(i,:) - mean(test_mtx_2(i,:)) ;
   test_mtx_2(i,:) = test_mtx_2(i,:)./std(test_mtx_2(i,:)) ;
end

AA = sim(net,test_mtx_2) ;   % Test the network.

% Convert the network outputs obtained from testing to binary outputs.

A_bin_out2 = zeros(size(AA)) ;

for i=1:7
   for j=1:size(test_mtx_2,2)
      if AA(i,j)==max(AA(:,j))
         A_bin_out2(i,j)=1;
      else A_bin_out2(i,j)=0;
      end
   end
end

% Construct the confusion matrix for the testing set obtained from "kill_noise" and
% "right_outside".

M_tst2 = zeros(7,2) ;
[II,JJ] = find(A_bin_out2) ;

for i=1:size(A_bin_out2,2)
   if ( i <= NN(1) )
      if II(i)==1, M_tst2(7,1)=M_tst2(7,1)+1; end
      if II(i)==2, M_tst2(6,1)=M_tst2(6,1)+1; end
      if II(i)==3, M_tst2(5,1)=M_tst2(5,1)+1; end
      if II(i)==4, M_tst2(4,1)=M_tst2(4,1)+1; end
      if II(i)==5, M_tst2(3,1)=M_tst2(3,1)+1; end
      if II(i)==6, M_tst2(2,1)=M_tst2(2,1)+1; end
      if II(i)==7, M_tst2(1,1)=M_tst2(1,1)+1; end
```

```
   end
   if ( i > NN(1) ) & ( i <= NN(2) )
      if II(i)==1, M_tst2(7,2)=M_tst2(7,2)+1; end
      if II(i)==2, M_tst2(6,2)=M_tst2(6,2)+1; end
      if II(i)==3, M_tst2(5,2)=M_tst2(5,2)+1; end
      if II(i)==4, M_tst2(4,2)=M_tst2(4,2)+1; end
      if II(i)==5, M_tst2(3,2)=M_tst2(3,2)+1; end
      if II(i)==6, M_tst2(2,2)=M_tst2(2,2)+1; end
      if II(i)==7, M_tst2(1,2)=M_tst2(1,2)+1; end
   end
end

Test_Conf_matrix_2(:,1) = ( M_tst2(:,1) ./ NN(1) ) .* 100 ;
Test_Conf_matrix_2(:,2) = ( M_tst2(:,2) ./ ( NN(2)-NN(1) ) ) .* 100 ;

% Store the confusion matrices obtained for each iteration in a cell array.

results(trial,1) = { Train_Conf_matrix  } ;
results(trial,2) = { Test_Conf_matrix   } ;
results(trial,3) = { Test_Conf_matrix_2 } ;

end

% Save all the confusion matrices obtained for 80 iterations.

save_loc = [ Main_loc '\' 'Results_MEL_2_150.mat' ] ;
save(save_loc,'results') ;

% This section computes the overall average confusion matrices, and related average
% classification performances.

conf_train = results{1,1}  ;
conf_test = results{1,2}  ;
conf_test2 = results{1,3}  ;

for i=2:80
   conf_train = conf_train + results{i,1} ;
   conf_test = conf_test + results{i,2} ;
   conf_test2 = conf_test2 + results{i,3} ;
end

conf_train = conf_train ./ 80 ;
conf_test = conf_test ./ 80 ;
conf_test2 = conf_test2 ./ 80 ;

Conf_all(1,3) = { [] } ;
```

```
Conf_all(1,1) = { conf_train } ;
Conf_all(1,2) = { conf_test } ;
Conf_all(1,3) = { conf_test2 } ;

% Save the overall average confusion matrices.

save_loc_2 = [ Main_loc '\' 'Conf_MEL_2_150.mat' ] ;
save(save_loc_2,'Conf_all') ;

% Overall classification rates for the training and testing sets.

Overall_Class_train = sum(diag(conf_train)) / 7 ;
Overall_Class_test = sum(diag(conf_test)) / 7 ;

clc,

disp(' Overall Average Classification Rate for Training = ') ;
disp(Overall_Class_train) ;
disp(' Overall Average Classification Rate for Testing = ') ;
disp(Overall_Class_train) ;

% ********************** END OF MAIN PROGRAM **********************
```

- ***BPNN_MFCC_3_Layer.m:***

```
% ********************************************************************
% Filename          : BPNN_MFCC_3_Layer.m
%                     (Recognition Main Program)
% Thesis Advisor    : Prof. Monique P. Fargues
% Author            : LTJG Gokhan Bulbuller
%                     Turkish Navy
% Date              : January 12, 2006 21:30 PST
% Description       : The main program for word recognition.
%                     The (60-40-7) network implemented below is one of the ten
%                     configurations considered for recognition purposes. MFCCs
%                     stored in "MFCC_Features.mat" are used as input features.
%                     The network is iterated 80 times on data by randomly selecting
%                     the training set each time, and results are saved in related ".mat"
%                     related ".mat" files.
% Inputs            : MFCC_Features.mat (14 Dimensional MFCCs)
% Outputs           : 1. Results_MEL_64.mat    : Contains all the confusion matrices
%                                                for every individual iteration,
%                     2. Conf_MEL_64.mat       : Contains the overall average
%                                                confusion matrices of all 80
%                                                iterations,
```

```
%                       3. Overall_Class_train      : The overall average classification
%                                                     rate for the training data set,
%                       4. Overall_Class_test       : The overall average classification
%                                                     rate for the testing data set.
% Functions Used      : N/A
% ************************************************************************

clear all, close all, clc,

Main_loc = [ 'C:\Documents and Settings\Owner\My Documents\thesis' ] ;
Feature_loc = [ Main_loc '\' 'MFCC_Features.mat' ] ;

load (Feature_loc)  ;                   % Load the MFCCs of all data set.

results(80,3) = { [] } ;          % Cell array that will store the results.

for trial=1:80

p = randperm(39) ;

%                                   Create the Training Data Set
% To construct the training set, select randomly 15 out of all recorded repetitions of a
% subject for a word. The training matrix size is 14*2100.

train_cell(20,7) = {[]} ;

k = 1:15 ;

for i=1:20
   for j=1:7
      x = MFCC_Feature_cell{i,j} ;
      train_cell(i,j) = { x(2:15,p(k)) } ;
   end
end

cl1 = 1 ; cl2 = 15 ;

for i=1:7
   for j=1:20
      train_mtx(:,cl1:cl2) = train_cell{j,i} ;
      cl1 = cl1 + 15 ; cl2 = cl2 + 15 ;
   end
end

% Make the training set zero mean and unit variance.
```

```
for i=1:14
   train_mtx(i,:) = train_mtx(i,:) - mean(train_mtx(i,:)) ;
   train_mtx(i,:) = train_mtx(i,:)./std(train_mtx(i,:)) ;
end

%                                   Create the Testing Data Set
% To construct the testing set, assign all the remaining utterances to the testing set
% matrix.

test_cell(20,7) = {[]} ;

for i=1:20
   for j=1:7
      x = MFCC_Feature_cell{i,j} ;
      X = x(2:15,p(16:end)) ;
      if size(x,2) > 39
         X = [ X x(2:15,40:end) ] ;
      end
      test_cell(i,j) = { X } ;
   end
end

m = 1 ; n = size(test_cell{1,1},2) ;

for i=1:7
   if i ~= 1
      m = n + 1 ; n = n + size(test_cell{1,i},2) ;
   end
   for j=1:20
      test_mtx(:,m:n) = test_cell{j,i} ;
      if j ~= 20
        m = n + 1 ; n = n + size(test_cell{j+1,i},2) ;
      end
   end
   N(i) = n ;
end

% Make the testing set zero mean and unit variance.

for i=1:14
   test_mtx(i,:) = test_mtx(i,:) - mean(test_mtx(i,:)) ;
   test_mtx(i,:) = test_mtx(i,:)./std(test_mtx(i,:)) ;
end

% Create the target matrix, which is (7*2100). 1-of-7 coding is used to represent each of
% seven words.
```

```
tgt = zeros(7,size(train_mtx,2));
tgt(7,1:300) = 1 ; tgt(6,301:600) = 1 ;
tgt(5,601:900) = 1 ; tgt(4,901:1200) = 1 ;
tgt(3,1201:1500) = 1 ; tgt(2,1501:1800) = 1 ;
tgt(1,1801:2100) = 1 ;

% Create, train and test the (60-40-7) network for word recognition.

z = [ min(train_mtx')' max(train_mtx')' ] ;

net = newff(z,[60,40,7],{'tansig','tansig','logsig'},'traincgf') ;
net.performFcn = 'msereg' ;
net.performParam.ratio = 0.85 ;
net.trainParam.show = 10 ;
net.trainParam.epochs = 1000 ;

[net,tr,Y,E] = train(net,train_mtx,tgt) ;          % Train the network.

A = sim(net,test_mtx) ;          % Test the network.

% Convert the network outputs obtained from training to binary outputs.

Y_bin_out=zeros(size(Y));

for i=1:7
   for j=1:size(train_mtx,2)
      if Y(i,j)==max(Y(:,j))
        Y_bin_out(i,j)=1;
      else Y_bin_out(i,j)=0;
      end
   end
end

% Convert the network outputs obtained from testing to binary outputs.

A_bin_out = zeros(size(A)) ;

for i=1:7
   for j=1:size(test_mtx,2)
      if A(i,j)==max(A(:,j))
        A_bin_out(i,j)=1;
      else A_bin_out(i,j)=0;
      end
   end
end
```

```
% Construct the confusion matrix for the training set, and compute the related average
% classification rate.

M = zeros(7,7) ;
[I,J] = find(Y_bin_out) ;

for i=1:size(Y_bin_out,2)
   if ( i <= 300 )
      if I(i)==1, M(7,1)=M(7,1)+1; end
      if I(i)==2, M(6,1)=M(6,1)+1; end
      if I(i)==3, M(5,1)=M(5,1)+1; end
      if I(i)==4, M(4,1)=M(4,1)+1; end
      if I(i)==5, M(3,1)=M(3,1)+1; end
      if I(i)==6, M(2,1)=M(2,1)+1; end
      if I(i)==7, M(1,1)=M(1,1)+1; end
   end
   if ( i > 300 ) & ( i <= 600 )
      if I(i)==1, M(7,2)=M(7,2)+1; end
      if I(i)==2, M(6,2)=M(6,2)+1; end
      if I(i)==3, M(5,2)=M(5,2)+1; end
      if I(i)==4, M(4,2)=M(4,2)+1; end
      if I(i)==5, M(3,2)=M(3,2)+1; end
      if I(i)==6, M(2,2)=M(2,2)+1; end
      if I(i)==7, M(1,2)=M(1,2)+1; end
   end
   if ( i > 600 ) & ( i <= 900 )
      if I(i)==1, M(7,3)=M(7,3)+1; end
      if I(i)==2, M(6,3)=M(6,3)+1; end
      if I(i)==3, M(5,3)=M(5,3)+1; end
      if I(i)==4, M(4,3)=M(4,3)+1; end
      if I(i)==5, M(3,3)=M(3,3)+1; end
      if I(i)==6, M(2,3)=M(2,3)+1; end
      if I(i)==7, M(1,3)=M(1,3)+1; end
   end
   if ( i > 900 ) & ( i <= 1200 )
      if I(i)==1, M(7,4)=M(7,4)+1; end
      if I(i)==2, M(6,4)=M(6,4)+1; end
      if I(i)==3, M(5,4)=M(5,4)+1; end
      if I(i)==4, M(4,4)=M(4,4)+1; end
      if I(i)==5, M(3,4)=M(3,4)+1; end
      if I(i)==6, M(2,4)=M(2,4)+1; end
      if I(i)==7, M(1,4)=M(1,4)+1; end
   end
   if ( i > 1200 ) & ( i <= 1500 )
      if I(i)==1, M(7,5)=M(7,5)+1; end
```

```
      if I(i)==2, M(6,5)=M(6,5)+1; end
      if I(i)==3, M(5,5)=M(5,5)+1; end
      if I(i)==4, M(4,5)=M(4,5)+1; end
      if I(i)==5, M(3,5)=M(3,5)+1; end
      if I(i)==6, M(2,5)=M(2,5)+1; end
      if I(i)==7, M(1,5)=M(1,5)+1; end
   end
   if ( i > 1500 ) & ( i <= 1800 )
      if I(i)==1, M(7,6)=M(7,6)+1; end
      if I(i)==2, M(6,6)=M(6,6)+1; end
      if I(i)==3, M(5,6)=M(5,6)+1; end
      if I(i)==4, M(4,6)=M(4,6)+1; end
      if I(i)==5, M(3,6)=M(3,6)+1; end
      if I(i)==6, M(2,6)=M(2,6)+1; end
      if I(i)==7, M(1,6)=M(1,6)+1; end
   end
   if ( i > 1800 ) & ( i <= 2100 )
      if I(i)==1, M(7,7)=M(7,7)+1; end
      if I(i)==2, M(6,7)=M(6,7)+1; end
      if I(i)==3, M(5,7)=M(5,7)+1; end
      if I(i)==4, M(4,7)=M(4,7)+1; end
      if I(i)==5, M(3,7)=M(3,7)+1; end
      if I(i)==6, M(2,7)=M(2,7)+1; end
      if I(i)==7, M(1,7)=M(1,7)+1; end
   end
end

Train_Conf_matrix = ( M ./300 ) .* 100 ;
Class_rate_train = sum(diag(Train_Conf_matrix)) / 7 ;

% Construct the confusion matrix for the testing set, and compute the related average
% classification rate.

M_tst = zeros(7,7) ;
[II,JJ] = find(A_bin_out) ;

for i=1:size(A_bin_out,2)
   if ( i <= N(1) )
      if II(i)==1, M_tst(7,1)=M_tst(7,1)+1; end
      if II(i)==2, M_tst(6,1)=M_tst(6,1)+1; end
      if II(i)==3, M_tst(5,1)=M_tst(5,1)+1; end
      if II(i)==4, M_tst(4,1)=M_tst(4,1)+1; end
      if II(i)==5, M_tst(3,1)=M_tst(3,1)+1; end
      if II(i)==6, M_tst(2,1)=M_tst(2,1)+1; end
      if II(i)==7, M_tst(1,1)=M_tst(1,1)+1; end
   end
```

```
if ( i > N(1) ) & ( i <= N(2) )
   if II(i)==1, M_tst(7,2)=M_tst(7,2)+1; end
   if II(i)==2, M_tst(6,2)=M_tst(6,2)+1; end
   if II(i)==3, M_tst(5,2)=M_tst(5,2)+1; end
   if II(i)==4, M_tst(4,2)=M_tst(4,2)+1; end
   if II(i)==5, M_tst(3,2)=M_tst(3,2)+1; end
   if II(i)==6, M_tst(2,2)=M_tst(2,2)+1; end
   if II(i)==7, M_tst(1,2)=M_tst(1,2)+1; end
end
if ( i > N(2) ) & ( i <= N(3) )
   if II(i)==1, M_tst(7,3)=M_tst(7,3)+1; end
   if II(i)==2, M_tst(6,3)=M_tst(6,3)+1; end
   if II(i)==3, M_tst(5,3)=M_tst(5,3)+1; end
   if II(i)==4, M_tst(4,3)=M_tst(4,3)+1; end
   if II(i)==5, M_tst(3,3)=M_tst(3,3)+1; end
   if II(i)==6, M_tst(2,3)=M_tst(2,3)+1; end
   if II(i)==7, M_tst(1,3)=M_tst(1,3)+1; end
end
if ( i > N(3) ) & ( i <= N(4) )
   if II(i)==1, M_tst(7,4)=M_tst(7,4)+1; end
   if II(i)==2, M_tst(6,4)=M_tst(6,4)+1; end
   if II(i)==3, M_tst(5,4)=M_tst(5,4)+1; end
   if II(i)==4, M_tst(4,4)=M_tst(4,4)+1; end
   if II(i)==5, M_tst(3,4)=M_tst(3,4)+1; end
   if II(i)==6, M_tst(2,4)=M_tst(2,4)+1; end
   if II(i)==7, M_tst(1,4)=M_tst(1,4)+1; end
end
if ( i > N(4) ) & ( i <= N(5) )
   if II(i)==1, M_tst(7,5)=M_tst(7,5)+1; end
   if II(i)==2, M_tst(6,5)=M_tst(6,5)+1; end
   if II(i)==3, M_tst(5,5)=M_tst(5,5)+1; end
   if II(i)==4, M_tst(4,5)=M_tst(4,5)+1; end
   if II(i)==5, M_tst(3,5)=M_tst(3,5)+1; end
   if II(i)==6, M_tst(2,5)=M_tst(2,5)+1; end
   if II(i)==7, M_tst(1,5)=M_tst(1,5)+1; end
end
if ( i > N(5) ) & ( i <= N(6) )
   if II(i)==1, M_tst(7,6)=M_tst(7,6)+1; end
   if II(i)==2, M_tst(6,6)=M_tst(6,6)+1; end
   if II(i)==3, M_tst(5,6)=M_tst(5,6)+1; end
   if II(i)==4, M_tst(4,6)=M_tst(4,6)+1; end
   if II(i)==5, M_tst(3,6)=M_tst(3,6)+1; end
   if II(i)==6, M_tst(2,6)=M_tst(2,6)+1; end
   if II(i)==7, M_tst(1,6)=M_tst(1,6)+1; end
end
if ( i > N(6) ) & ( i <= N(7) )
```

```
      if II(i)==1, M_tst(7,7)=M_tst(7,7)+1; end
      if II(i)==2, M_tst(6,7)=M_tst(6,7)+1; end
      if II(i)==3, M_tst(5,7)=M_tst(5,7)+1; end
      if II(i)==4, M_tst(4,7)=M_tst(4,7)+1; end
      if II(i)==5, M_tst(3,7)=M_tst(3,7)+1; end
      if II(i)==6, M_tst(2,7)=M_tst(2,7)+1; end
      if II(i)==7, M_tst(1,7)=M_tst(1,7)+1; end
   end
end

Test_Conf_matrix(:,1) = ( M_tst(:,1) ./ N(1) ) .* 100 ;
Test_Conf_matrix(:,2) = ( M_tst(:,2) ./ ( N(2)-N(1) ) ) .* 100 ;
Test_Conf_matrix(:,3) = ( M_tst(:,3) ./ ( N(3)-N(2) ) ) .* 100 ;
Test_Conf_matrix(:,4) = ( M_tst(:,4) ./ ( N(4)-N(3) ) ) .* 100 ;
Test_Conf_matrix(:,5) = ( M_tst(:,5) ./ ( N(5)-N(4) ) ) .* 100 ;
Test_Conf_matrix(:,6) = ( M_tst(:,6) ./ ( N(6)-N(5) ) ) .* 100 ;
Test_Conf_matrix(:,7) = ( M_tst(:,7) ./ ( N(7)-N(6) ) ) .* 100 ;

Class_rate_test = sum(diag(Test_Conf_matrix)) / 7 ;

% Test the (60-40-7) network on "kill_noise" and "right_outside" on which it was not
% trained.

% Construct the new testing set from "kill_noise" and "right_outside".

test_cell_2(20,2) = {[]} ;

for i=1:20
   XX = MFCC_Feature_cell{i,8} ; XX(1,:) = [] ;
   test_cell_2(i,1) = { XX } ;
   XX = MFCC_Feature_cell{i,9} ; XX(1,:) = [] ;
   test_cell_2(i,2) = { XX } ;
end

m = 1 ; n = size(test_cell_2{1,1},2) ;

for i=1:2
   if i ~= 1
      m = n + 1 ; n = n + size(test_cell_2{1,i},2) ;
   end
   for j=1:20
      test_mtx_2(:,m:n) = test_cell_2{j,i} ;
      if j ~= 20
         m = n + 1 ; n = n + size(test_cell_2{j+1,i},2) ;
      end
   end
```

```
   NN(i) = n ;
end

% Make the new testing set zero mean and unit variance.

for i=1:14
   test_mtx_2(i,:) = test_mtx_2(i,:) - mean(test_mtx_2(i,:)) ;
   test_mtx_2(i,:) = test_mtx_2(i,:)./std(test_mtx_2(i,:)) ;
end

AA = sim(net,test_mtx_2) ; % Test the network.

% Convert the network outputs obtained from testing to binary outputs.

A_bin_out2 = zeros(size(AA)) ;

for i=1:7
   for j=1:size(test_mtx_2,2)
      if AA(i,j)==max(AA(:,j))
        A_bin_out2(i,j)=1;
      else A_bin_out2(i,j)=0;
      end
   end
end

% Construct the confusion matrix for the testing set obtained from "kill_noise" and
% "right_outside".

M_tst2 = zeros(7,2) ;
[II,JJ] = find(A_bin_out2) ;

for i=1:size(A_bin_out2,2)
   if ( i <= NN(1) )
     if II(i)==1, M_tst2(7,1)=M_tst2(7,1)+1; end
     if II(i)==2, M_tst2(6,1)=M_tst2(6,1)+1; end
     if II(i)==3, M_tst2(5,1)=M_tst2(5,1)+1; end
     if II(i)==4, M_tst2(4,1)=M_tst2(4,1)+1; end
     if II(i)==5, M_tst2(3,1)=M_tst2(3,1)+1; end
     if II(i)==6, M_tst2(2,1)=M_tst2(2,1)+1; end
     if II(i)==7, M_tst2(1,1)=M_tst2(1,1)+1; end
   end
   if ( i > NN(1) ) & ( i <= NN(2) )
     if II(i)==1, M_tst2(7,2)=M_tst2(7,2)+1; end
     if II(i)==2, M_tst2(6,2)=M_tst2(6,2)+1; end
     if II(i)==3, M_tst2(5,2)=M_tst2(5,2)+1; end
     if II(i)==4, M_tst2(4,2)=M_tst2(4,2)+1; end
```

```
      if II(i)==5, M_tst2(3,2)=M_tst2(3,2)+1; end
      if II(i)==6, M_tst2(2,2)=M_tst2(2,2)+1; end
      if II(i)==7, M_tst2(1,2)=M_tst2(1,2)+1; end
   end
end

Test_Conf_matrix_2(:,1) = ( M_tst2(:,1) ./ NN(1) ) .* 100 ;
Test_Conf_matrix_2(:,2) = ( M_tst2(:,2) ./ ( NN(2)-NN(1) ) ) .* 100 ;

% Store the confusion matrices obtained for each iteration in a cell array.

results(trial,1) = { Train_Conf_matrix  } ;
results(trial,2) = { Test_Conf_matrix   } ;
results(trial,3) = { Test_Conf_matrix_2 } ;

end

% Save all the confusion matrices obtained for 80 iterations.

save_loc = [ Main_loc '\' 'Results_MEL_64.mat' ] ;
save(save_loc,'results') ;

% This section computes the overall average confusion matrices, and related average
% classification performances.

conf_train = results{1,1}  ;
conf_test = results{1,2}  ;
conf_test2 = results{1,3}  ;

for i=2:80
   conf_train = conf_train + results{i,1} ;
   conf_test = conf_test + results{i,2} ;
   conf_test2 = conf_test2 + results{i,3} ;
end

conf_train = conf_train ./ 80 ;
conf_test = conf_test ./ 80 ;
conf_test2 = conf_test2 ./ 80 ;

Conf_all(1,3) = { [] } ;

Conf_all(1,1) = { conf_train } ;
Conf_all(1,2) = { conf_test  } ;
Conf_all(1,3) = { conf_test2 } ;

% Save the overall average confusion matrices.
```

```
save_loc_2 = [ Main_loc '\' 'Conf_MEL_64.mat' ] ;
save(save_loc_2,'Conf_all') ;

% Overall classification rates for the training and testing sets.

Overall_Class_train = sum(diag(conf_train)) / 7 ;
Overall_Class_test = sum(diag(conf_test)) / 7 ;

clc,

disp(' Overall Average Classification Rate for Training = ') ;
disp(Overall_Class_train) ;
disp(' Overall Average Classification Rate for Testing = ') ;
disp(Overall_Class_train) ;

% ********************** END OF MAIN PROGRAM **********************
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Black, R. D., "Ear-insert microphone," *Journal of the Acoustical Society of America*, Vol. 29, No. 2, pp. 260-264, 1957.

Bogert, B., Healy, M., and Tukey, J., "The quefrency analysis of time series for echos," In Rosenblatt, M., editor, *Proceedings of Symposium on Time Series Analysis*, Chapter 15, pp. 209-243, Wiley, New York, 1963.

Brookes, M., *Voicebox: A Matlab Toolbox for Speech Processing*, 1997, [http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html], last accessed on November 05, 2005.

Davis, S. B., and Mermelstein, P., "Comparison of parametric representations of monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-28, No. 4, pp. 357-366, August 1980.

Deller, J. R., Proakis, J. G., and Hansen, J. H. L, *Discrete-Time Processing of Speech Signals*, Macmillan, New York, 1993.

Demuth, H. B., and Beale, M. H., *Neural Network Toolbox User's Guide*, Version 4, The MathWorks, 2005.

Deng, L., and O'Shaughnessy, D., *Speech Processing*, Marcel Dekker, New York, 2003.

Duda, R. O., Hart, P. E., and Stork, D. G., *Pattern Classification*, 2nd Edition, Wiley-Interscience, New York, 2001.

Gold, B., and Morgan, N., *Speech and Audio Signal Processing*, Wiley, New York, 2001.

Graciarena, M., Franco, H., Sonmez, K., and Bratt, H., "Combining standard and throat microphones for robust speech recognition," *IEEE Signal Processing Letters*, Vol. 10, No. 3, pp. 72-74, March 2003.

Hagan, M. T., Demuth, H. B., and Beale, M. H., *Neural Network Design*, Campus Publishing Service, University of Colorado, Boulder, Colorado, 1996.

Hopfield, J. J., "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of The National Academy of Sciences*, Vol. 79, pp. 2554-2558, 1982.

Huang, L-S., and Yang, C-H., "A novel approach to robust speech endpoint detection in car environments," *Proceedings of The IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 3, pp. 1751-1754, June 2000.

Junqua, J-C., "Robustness and cooperative multimodel man-machine communication applications," *Proceedings of Second Venaco Workshop and ESCA ETRW*, pp. 101-112, September 16-20, 1991.

Kaiser, J. F., "On a simple algorithm to calculate the 'energy' of a signal," *Proceedings of The IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 1, pp. 381-384, April 1990.

Knowles Acoustics, "FG Series Microphone Specifications," [www.knowlesacoustics.com], last accessed on February 15, 2006.

Lamel, L. F., Rabiner L. R., Rosenberg, A. E., and Wilpon J. G.,"An improved endpoint detector for isolated word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-29, No. 4, pp. 777-785, August 1981.

Mitra, S. K., *Digital Signal Processing: A Computer-Based Approach*, 3rd Edition, McGraw-Hill, New York, 2006.

Morgan, N., and Bourlard, H. A., "Neural networks for statistical recognition of continuous speech," *Proceedings of The IEEE*, Vol. 83, No. 5, pp. 742-770, May 1995.

Newton, M., M.S. Electrical Engineering Thesis, Naval Postgraduate School, Monterey California, 2006 (to be completed in 2006).

O'Neill, J., "A comparison of mouth, ear, and contact microphones," *The Journal of The Acoustical Society of America*, Vol. 30, No. 7, p. 682, July 1958.

Oppenheim, A. V., "Generalized linear filtering," In Gold, B., and Rader, C. M., editors, *Digital Processing of Signals*, Chapter 8, pp. 233-264, McGraw-Hill, New York, 1969.

Picone, J. W., "Signal modeling techniques in speech recognition," *Proceedings of The IEEE*, Vol. 81, No. 9, pp. 1215-1247, September 1993.

Qiang, H., and Youwei, Z., "On prefiltering and endpoint detection of speech signal," *Proceedings of The Fourth International Conference on Signal Processing*, Vol. 1, pp. 749-752, October 1998.

Rabiner, L. R., and Sambur, M.R., "An algorithm for determining the endpoints of isolated utterances," *The Bell System Technical Journal*, Vol. 54, pp. 297-315, February 1975.

Rabiner, L. R., and Schafer, R. W., *Digital Processing of Speech Signals*, Prentice-Hall, New Jersey, 1978.

Rafaely, B., and Furst, M., "Audiometric ear canal probe with active ambient noise control," *IEEE Transactions on Speech and Audio Processing*, Vol. 4, No. 3, pp. 224-230, May 1996.

Rosenblatt, F., "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, Vol. 65, pp. 386-408, 1958.

Rumelhart, D. E., and McClelland, J. L., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol.1, Cambridge, MA: MIT Press, 1986.

Shahina, A., and Yegnanarayana, B., "Language identification in noisy environments using throat microphone signals," *Proceedings of 2005 International Conference on Intelligent Sensing and Information Processing*, pp. 400-403, January 4-7, 2005.

Shen, J. L., Hung, J. W., and Lee, L. S., "Robust entropy-based endpoint detection for speech recognition in noisy environments," *Proceedings of The International Conference on Spoken Language Processing*, November-December 1998.

Taboada, J., Feijoo, S., Balsa, R., and Hernandez C., "Explicit estimation of speech boundaries," *IEE Proceedings – Science, Measurement, and Technology*, Vol. 141, No. 3, pp. 153-159, May 1994.

Teager, H. M., "Some observations on oral air flow during phonation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 28, No. 5, pp. 599-601, October 1980.

Vaidyanathan, R., Gupta, L., Chung, B., Allen, T. J., Quinn, R. D., Tabib-Azar, M., Zarycki, J., and Levin, J., "Human-machine interface for tele-robotic operation: mapping of tongue movements based on aural flow monitoring," *Proceedings of The IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 1, pp. 859-865, September-October 2004.

Vaidyanathan, R., Kook, H., Gupta, L., and West, J., "Parametric and non-parametric signal analysis for mapping air flow in the ear-canal to tongue movement: a new strategy for hands-free human-machine interface," *Proceedings of The IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 5, pp. 613-616, May 2004.

Vergin, R., O'Shaughnessy, D., and Farhat, A., "Generalized mel frequency cepstral coefficients for large-vocabulary speaker-independent continuous-speech recognition," *IEEE Transactions on Speech and Audio Processing*, Vol. 7, No. 5, pp. 525-532, September 1999.

Westerlund, N., Dahl, M., and Claesson, I., *In-Ear Microphone Techniques for Severe Noise Conditions*, Research Report, November 2005.

Westerlund, N., Dahl, M., and Claesson, I., "Speech recognition in severely disturbed environments combining ear-mic and active noise control," *Proceedings of The 2002 International Congress and Exposition on Noise Control Engineering*, Dearborn, MI, August 2002.

Westerlund, N., Dahl, M., and Claesson, I., "In-ear microphone hybrid speech enhancement," *Proceedings of SIP*, Kauai, Hawai, USA, August 2002.

Wu, B. F., and Wang, K. C., "Robust endpoint detection algorithm based on the adaptive band-partitioning spectral entropy in adverse environments," *IEEE Transactions on Speech and Audio Processing*, Vol. 13, No. 5, pp. 762-775, September 2005.

Ying, G. S., Mitchell, C. D., and Jamieson, L. H., "Endpoint detection of isolated utterances based on a modified Teager energy measurement," *Proceedings of The IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 2, pp.732-735, April 1993.

Zhang, Y., Zhu., X., Hao, Y., and Luo, Y., "A robust and fast endpoint detection algorithm for isolated word recognition," *Proceedings of The IEEE International Conference on Intelligent Processing Systems*, Vol. 2, pp. 1819-1822, October 1997.

Zhu, Q., and Alwan, A., "Non-linear feature extraction for robust speech recognition in stationary and non-stationary noise," *Computer Speech and Language*, Vol. 17, No. 4, pp. 381-402, October 2003.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor Jeffrey B. Knorr
   Chairman, Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

4. Professor Monique P. Fargues
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

5. Professor Ravi Vaidyanathan
   Department of Systems Engineering
   Naval Postgraduate School
   Monterey, California

6. Gokhan Bulbuller
   Yildiz Mah. 223 Sokak Yalcin Apt. No:16/2
   Antalya 07050
   TURKEY