Feature enhancement for robust speech recognition

Zhang, Yi

2008

Zhang, Y. (2008). Feature enhancement for robust speech recognition. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/20668>

<https://doi.org/10.32657/10356/20668>

FEATURE ENHANCEMENT FOR ROBUST SPEECH RECOGNITION

ZHANG YI

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Master of Engineering

2008

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor Koh Soo Ngee, for his consistent encouragement. Professor Koh spent lots of time to discuss with me various technical problems and novel ideas pertaining to my research work. He has given me a great deal of support throughout my research. Without his help, I would not have made such a good progress.

I would also like to thank Dr. You Changhuai of Institute for Infocomm Research, Agency for Science, Technology and Research, for his support, encouragement and the discussion he had with me.

My sincere thanks also go to Mr. V. G. Reju, Ms. Zhang Xinxin and Ms. Liu Xiaopei for sharing with me their knowledge and interests.

Last but not least, I wish to thank Mr. Lim Cheng Chye, lab technician of Communication Research Laboratory, for providing me lots of support and helping me to purchase the speech databases.

Table of Contents

Summary.....	V
List of Figures.....	VII
List of Tables	IX
List of Abbreviations	XI
Chapter 1 Introduction	1
1.1 Motivation	1
1.2 Scope of the Thesis.....	2
1.3 Contributions	3
Chapter 2 Overview of Automatic Speech Recognition	5
2.1 Introduction	5
2.2 Automatic Speech Recognition System	5
2.3 Hidden Markov Model	7
2.4 Feature Extraction.....	13
2.4.1 Mel Frequency Cepstral Coefficient.....	14
2.4.1.1 Pre-emphasis.....	15
2.4.1.2 Windowing	16
2.4.1.3 Fast Fourier Transform.....	17
2.4.1.4 Mel Filter Bank Analysis.....	17
2.4.1.5 Discrete Cosine Transform.....	18
2.4.1.6 Energy Measure.....	19
2.4.1.7 Derivative Calculation.....	19

II

2.4.2	Phase Autocorrelation (PAC) Derived Speech Feature.....	20
2.4.2.1	Phase Autocorrelation.....	20
2.4.2.2	PAC Derived Feature.....	25
2.5	Database for Simulation Study	28
2.5.1	AURORA 2	28
2.5.2	AURORA 4	29
2.5.3	CENSREC-3	30
Chapter 3	System Evaluation of AURORA 4.....	33
3.1	Introduction	33
3.2	Training Procedure	34
3.2.1	<i>sil</i> and <i>sp</i> Model.....	36
3.2.2	Decision Tree Based Clustering	37
3.2.3	Mixture Increment	38
3.3	Parameter Tuning	39
3.4	System Evaluation	42
3.5	Summary.....	44
Chapter 4	Feature Normalization for Robust Speech Recognition.....	46
4.1	Introduction	46
4.2	Cepstral Mean and Variance Normalization	48
4.2.1	Segmental Mean and Variance Normalization.....	50
4.2.2	Recursive Segmental Variance Normalization.....	52
4.2.3	Variance Normalization with Reliable Frame Selection	54
4.3	Log Energy Dynamic Range Normalization	59

4.4	Histogram Equalization	65
4.5	PCA Based Feature Vector Compression.....	69
4.6	Summary.....	71
Chapter 5	Cepstral Domain Filtering for Robust Speech Recognition	72
5.1	Introduction	72
5.2	ARMA Filtering	74
5.3	Multi-Eigenvector Temporal Filtering	77
5.4	Linear Discriminant Analysis for Cepstral Filtering	82
5.5	Entropy Based Multi-Eigenvector Temporal Filtering.....	86
5.6	β -order Multi-Eigenvector Temporal Filtering.....	93
5.7	Comparative Evaluation of Hybrid Schemes	95
5.8	Summary.....	99
Chapter 6	Model Adaptation for Large Vocabulary ASR.....	100
6.1	Introduction	100
6.2	Maximum Likelihood Linear Regression (MLLR).....	101
6.3	Regression Class Trees.....	105
6.4	Model Adaptation process using HTK	107
6.5	MLLR for Speaker Adaptation.....	108
6.6	Model Adaptation for Reverberant Speech	110
6.7	Model Adaptation for Additive Noise and Channel Distortion.....	115
6.8	Summary.....	119
Chapter 7	Conclusion and Future Work.....	121
References		I

Summary

The results of investigations into some aspects of robust speech recognition are reported in this thesis. Included in the topics that have been studied are feature extraction, training and decoding procedures, speech feature enhancement and model adaptation. In an automatic speech recognition (ASR) system, feature extraction is critical to determining system performance. The most commonly used feature vectors for ASR are those based on the Mel Frequency Cepstral Coefficients (MFCC). However, it is well known that under noisy conditions, the performance of MFCC-based speech feature vectors degrades significantly. There have been many other robust features proposed in recent years and one that is derived from phase autocorrelation (PAC) was investigated.

To analyze the effectiveness of an ASR system in a large vocabulary continuous speech recognition task, an experimental framework has been built based on the AURORA 4 database. This framework uses the Hidden Markov Model based recognition engine. During the training and decoding, techniques such as decision tree based state clustering, Gaussian mixture model, and language model interpolation are applied to refine the system. Some parameters are also tuned to optimize the performance.

Feature enhancement is a broad category in robust speech recognition. Techniques in two major sub-categories of feature enhancement have been discussed: feature normalization and cepstral domain filtering. Feature normalization is generally a simple but effective method. Out of several methods discussed in this thesis, Histogram Equalization (HEQ) is found to have an advantage. In addition, Principal Component

VI

Analysis (PCA) based speech feature compression could be combined with the feature normalization. This combined approach is shown to be more efficient than the Advanced Front-end standard from ETSI for large vocabulary continuous speech recognition.

Cepstral domain filtering techniques are designed to remove the high frequency components and accentuate the speech information in the modulation frequency domain. The experimental results obtained show that both of the proposed cepstral filters, the entropy based multi-eigenvector filter and the β -order multi-eigenvector filter, outperform their linearly weighted counterparts.

Besides feature domain techniques, environmental noise could also be compensated by model adaptation. By means of an experiment in large vocabulary continuous speech recognition, it is shown that a trained ASR system could be improved significantly by adapting to a small amount of noisy data.

List of Figures

Figure 2-1: Overview of a speech recognition system	6
Figure 2-2: MFCC processing	15
Figure 2-3: Frequency response of pre-emphasis filter at $a = 0.97$	16
Figure 2-4: Mel filter bank	18
Figure 2-5: Inverse cosine function	22
Figure 2-6: Autocorrelation sequence for speech	23
Figure 2-7: Comparison of the magnitude and PAC spectrum for speech	23
Figure 2-8: The distribution of PAC spectrum against the magnitude spectrum	24
Figure 2-9: Performance of PAC MFCC and MFCC for clean training mode, the x-axis is the noise level of the utterance	26
Figure 2-10: Performance of PAC MFCC and MFCC for different frequency characteristic, the x-axis is the noise level of the utterance	27
Figure 3-1: Topologies of the HMM: (a) <i>sp</i> model, (b) <i>sil</i> model and (c) triphone or monophone model	37
Figure 3-2: Decision tree based state tying clustering	38
Figure 4-1: The sliding window in segmental variance normalization	51
Figure 4-2: Process of selecting reliable frames using Teager energy	55
Figure 4-3: Frame selection using Teager energy	56
Figure 4-4: Process of reliable frame selection process using feature entropy	57
Figure 4-5: Frame selection using feature entropy	58
Figure 4-6: Comparison of the log energy between the clean and noisy speech (10dB). ..	60
Figure 4-7: Representation of scaling effect of the log scale and the linear scale	61

VIII

Figure 4-8: Two different schemes when combining the variance normalization with log energy dynamic range normalization	63
Figure 4-9: Comparison of MVN, SVN and LEN on logE sequence	64
Figure 4-10: CHEQ evaluation on AURORA 4.....	69
Figure 5-1: MVA post processing	74
Figure 5-2: Frequency response of ARMA filter with different orders	75
Figure 5-3: Comparison of c1 sequence before and after the MVA processing, where the x-axis is the number of frames, y-axis is the magnitude of c1 sequence	75
Figure 5-4: The representation of the temporal filter analysis	77
Figure 5-5: Comparison of PCA derived temporal filter and multi-eigenvector filter.....	80
Figure 5-6: Comparison of filter response of LDA and mPCA cepstral filters.....	85
Figure 5-7: Frequency response of ePCA and PCA filter	88
Figure 5-8: Plot of the entropy contribution.....	88
Figure 5-9: Comparison between mPCA and emPCA, $M = 3$	89
Figure 5-10: Filter bank defined by 15 eigenvectors.....	90
Figure 5-11: Filters defined by first 4 eigenvectors	91
Figure 5-12: Illustration of the filter weighting.....	91
Figure 5-13: Comparison of β mPCA, emPCA and mPCA.	94
Figure 5-14: Comparison of 5 different cepstral filters	96
Figure 6-1: A simple example of regression class tree.....	106
Figure 6-2: Model Adaptation Process Using HTK	108
Figure 6-3: Relative energy decay of impulse responses from RT60 0.1s to 0.6s	113

List of Tables

Table 2-1: Recording environment for testing data.....	31
Table 2-2: Training data in CENSREC-3 database	32
Table 2-3: Testing data in CENSREC-3 database.....	32
Table 3-1: Training procedure in HTK for AURORA 4 task.....	35
Table 3-2: Performance comparison for different Gaussian mixture models	40
Table 3-3: Recognition accuracy at different language model factors and word insertion penalties (%) on evaluation set.....	42
Table 3-4: Recognition accuracy at different language model factors and word insertion penalty tuning	42
Table 3-5: Recognition accuracy (%) on baseline evaluation using AURORA 4.....	44
Table 3-6: Comparison of different mixture models in AURORA 4 (%).....	44
Table 4-1: Evaluation of MVN on AURORA 4 (%).....	49
Table 4-2: Evaluation of SVN on AURORA 4 (%)	52
Table 4-3: Different sliding window lengths in SVN (%).....	52
Table 4-4: Evaluation of recursive SVN on AURORA 4 (%)	54
Table 4-5: Evaluation of MVN with reliable frame selection on AURORA 4 (%)	58
Table 4-6: Evaluation of LEN with linear scale (%)	61
Table 4-7: Evaluation of LEN with nonlinear scale (%)	62
Table 4-8: Evaluation of LEN with variance normalization on AURORA 4 (%)	64
Table 4-9: Evaluation of HEQ on AURORA 4 (%).....	67
Table 4-10: Evaluation of PCA compression on AURORA 4 (%)	70

X

Table 4-11: Evaluation of CHEQ+PCA compression on AURORA 4 (%).....	71
Table 5-1: Evaluation of ARMA on CENSREC-3 (%).....	77
Table 5-2: Evaluation of ARMA on AURORA 4 (%).....	77
Table 5-3: Evaluation of PCA and mPCA on CENSREC-3 (%).....	81
Table 5-4: Evaluation of PCA and mPCA on AURORA 4 (%).....	81
Table 5-5: Evaluation of LDA filter on AURORA 2 (%).....	85
Table 5-6: Filter length of LDA filter in AURORA 2 Evaluation (%).....	86
Table 5-7: Evaluation of PCA and ePCA on CENSREC-3 (%).....	92
Table 5-8: Choice of M using AURORA 4 development data set	93
Table 5-9: Evaluation of mPCA and emPCA on AURORA 4 (%).....	93
Table 5-10: Selection of β order in AURORA 4 development set (%).....	94
Table 5-11: Evaluation of emPCA and β mPCA in AURORA 4 (%).....	95
Table 5-12: Evaluation of emPCA and β mPCA in CENSREC-3 (%).....	95
Table 5-13: Comparison of all cepstral filters with SVN on AURORA 4 (%).....	96
Table 5-14: Comparison of all cepstral filters with HEQ on AURORA 4 (%).....	96
Table 5-15: Evaluation of two-stage PCA processing on CENSREC-3 (%).....	97
Table 5-16: Evaluation of two-stage PCA processing on AURORA 4 (%).....	98
Table 6-1: Speaker adaptation on AURORA 4 (%).....	109
Table 6-2: Evaluation of MLLR model adaptation at different reverberant time (%)	112
Table 6-3: Evaluation of model adaptation at different amount of adaptation data (%) .	114
Table 6-4: The Euclidean distance between mean vectors of each monophone in testing and adaptation speech features	114

Table 6-5: Adaptation to the additive noise for the close-talking microphone, testing and adaptation utterances has the same type of noise	117
Table 6-6: Cross-noise Adaptation to additive noise for the close-talking microphone, testing and adaptation utterances has different types of noise.....	118
Table 6-7: Adaptation to the additive noise for the secondary microphone.....	119

List of Abbreviations

ACC	Word Recognition Accuracy
ANN	Artificial Neural Network
ARMA	Auto Regression Moving Average Filter
ASR	Automatic Speech Recognition
CDF	Cumulative Density Function
CHEQ	Class based Histogram Equalization
CT	Close Talk
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DSR	Distributed Speech Recognition
EM	Expectation and Maximization
emPCA	Entropy Based Multi-Eigenvector Temporal Filtering
ePCA	Entropy Based PCA Filter
ETSI	European Telecommunication Standard Institute
FFT	Fast Fourier Transform

XII

FIR	Finite Impulse Response
HEQ	Histogram Equalization
HF	Hands Free
HMM	Hidden Markov Model
HTK	Hidden Markov Model Toolkit
ISIP	Institute for Signal and Information Processing
LEN	Log Energy Dynamic Range Normalization
LDA	Linear Discriminant Analysis
logE	Log Energy Coefficient
MAP	Maximum a Posterior
MFCC	Mel Frequency Cepstral Coefficient
MLLR	Maximum Likelihood Linear Regression
mPCA	Multi-Eigenvector Temporal Filtering
MVA	MVN plus ARMA
MVN	Mean and Variance Normalization
PAC	Phase Autocorrelation
PCA	Principal Component Analysis
PDF	Probability Density Function
SNR	Signal to Noise Ratio
SVN	Segmental Variance Normalization
WSJ	Wall Street Journal

Chapter 1 Introduction

1.1 Motivation

Speech recognition is one of the actively researched areas of digital signal processing. In recent decades, due to raising demand from human machine interface technologies, artificial intelligence, automatic translation and automatic information retrieving, speech recognition has gained more attention. Current speech recognizers have the capability of recognizing clean speech to a very high level of accuracy. However, when environmental noise is present, the performance of the speech recognizers degrades significantly. The difficulty of maintaining high recognition accuracy under a noisy environment has proven to be a major obstacle that prevents speech recognition technology from being deployed in real world applications. In recent years, robust speech recognition has become one of the most important research topics in speech recognition. A good robust speech recognition system is expected to have a consistent recognition performance regardless of noise. Challenges facing robust speech recognition include acoustical degradations caused by additive noise, reverberation, channel distortion, and mismatch between the training and testing data.

One of the most popular techniques for robust speech recognition is speech enhancement which removes the noise in the time domain while preserving the intelligibility of the speech as much as possible. It is generally believed that in the speech feature domain, it is much more difficult to remove the noise components. In this thesis, a few feature enhancement techniques in robust speech recognition have been investigated. The scope of the investigation is not limited simply to digit string recognition as has mostly been

done in the past. Instead, it is extended to isolated word recognition and continuous large vocabulary recognition. In addition, the model adaptation approach which has previously been used for speaker adaptation is also examined for robust speech recognition in large vocabulary continuous speech recognition tasks.

1.2 Scope of the Thesis

Robust speech recognition techniques involve signal domain techniques, feature domain techniques, model domain techniques, etc. This thesis will focus on the feature domain techniques.

The thesis is organized as follows: Chapter 2 is a review of fundamental knowledge in the area of robust speech recognition. The statistical speech recognition technique based on the Hidden Markov Model is revised. The most commonly used feature vectors for speech recognition are those derived from the MFCC. The use of MFCC for deriving these feature vectors is reviewed. In addition, a recently proposed feature vector extraction algorithm that based on Phase Autocorrelation is also studied and examined. The last part of Chapter 2 deals with the databases used in our simulation studies, namely, AURORA 2, AURORA 4 and CENSREC-3. The AURORA 4 experimental framework used in this investigation is described in Chapter 3. It has to be pointed out that the exact training and testing procedure is not specified by the database and the testing framework needs to be designed by the user. AURORA 4 is used for large vocabulary speech recognition task. Many techniques have been applied to refine the system, including state tying, mixture increment and parameter tuning. Two feature enhancement techniques, feature normalization and spectral domain filtering are discussed in Chapters 4 and 5. In Chapter 4, conventional mean and variance normalization, segmental mean and variance

normalization, recursive mean and variance normalization, log energy dynamic range normalization, histogram equalization as well as the PCA (Principal Component Analysis) based feature compression are tested and compared on AURORA 4 database. In Chapter 5, ARMA filtering, multi-eigenvector filtering, linear discriminant analysis based filtering, entropy based multi-eigenvector filtering and β -order multi-eigenvector filtering are compared on AURORA 2, AURORA 4 and CENSREC-3. Chapter 6 studies model adaptation for large vocabulary robust speech recognition systems. MLLR model adaptation was examined for speaker adaptation. In addition, it has also been used to test for recognition under additive noise and channel distortion. In Chapter 7, we conclude the thesis and propose recommendations for future research.

1.3 Contributions

The major contributions of this dissertation are summarized as follows:

1. An experimental framework for large vocabulary continuous speech recognition system has been built. The system is based on the AURORA 4 database from ETSI (European Telecommunications Standards Institute).
2. Some variance normalization techniques have been tested on AURORA 2, AURORA 4 and CENSREC-3 databases. The best result obtained is compared with that of the Advanced Front-end from ETSI.
3. A novel reliable frame selection method based on feature entropy is tested on the AURORA 2 database.
4. Two novel cepstral filters, an entropy based multi-eigenvector filter and a β -order multi-eigenvector filter are proposed. The performances of these two filters have

been compared with the existing ARMA filter, a linear weighted multi-eigenvector filter, on the AURORA 4 and CENSREC-3 databases.

5. Model adaptation has been examined for robust speech recognition for large vocabulary speech recognition tasks.

Chapter 2 Overview of Automatic Speech Recognition

2.1 Introduction

Some of the fundamentals of speech recognition are outlined in this chapter. The basic elements of an automatic speech recognition system are described in Section 2.2 and some of the basic mathematics of the Hidden Markov Model (HMM) is revised in Section 2.3. The Mel Frequency Cepstral Coefficients (MFCC) used in feature extraction for speech recognition is reviewed in Section 2.4. In addition, a recently proposed feature extraction algorithm, i.e. PAC MFCC, is examined and compared with MFCC. Only established databases from well known international standards organizations and research institutes were used to perform all the simulations described in this thesis. A database contains a huge amount of speech utterances and defines a framework for the training and testing process. It makes the comparison of different algorithms easier. A summary of all the databases used in these investigations is given in the last section of this chapter.

2.2 Automatic Speech Recognition System

Figure 2-1 shows the basic structure of an automatic speech recognition (ASR) system. Firstly, the feature extraction process converts the speech waveform into feature vectors. The feature vector represents the speech information which is of importance to the recognizer. These feature vectors are then split into two groups. One group is used for training the acoustical model and the other group is used for decoding.

Speech recognition engines fall into several different categories. For simple word or digit recognition when the vocabulary size is small, and when the vocabulary contains only a few words, a dynamic time warping engine is suitable. For large vocabulary recognition

tasks, the most commonly used recognition engines are based on the Hidden Markov Model (HMM), the Artificial Neural Network (ANN) or a hybrid of the two (HMM-ANN) [51]. All of our speech recognition experiments used an HMM based software, known as HTK (Hidden Markov Model Toolkit) [1].

Figure 2-1: Overview of a speech recognition system

HMM is a mathematical model proposed in 1960s, in which the feature vectors generated by the model are described in a statistical manner. According to the Bayes optimal classification rule

$$P(M_j | x) = \frac{p(x | M_j)P(M_j)}{P(x)} \quad (2.2.1)$$

where M_j is the j -th statistical model to generate the sequence and x is a sequence of the observed symbol, e.g., speech vectors. The minimum probability of error, in classifying x into a correct category M_j , is obtained if one always assigns x to a model with the maximum posterior probability $P(M_j | x)$. This problem is equivalent to maximizing the product term $p(x | M_j)p(M_j)$, as $p(x)$ is the same for all classes. In speech recognition,

the prior term $p(M_j)$ is estimated using a language model which describes the frequency of occurrences of each word sequence; the likelihood $p(x | M_j)$ is estimated using a HMM model, which represents the statistical distribution of feature vectors.

2.3 Hidden Markov Model

Hidden Markov Models (HMM) are especially known for their applications in pattern recognition fields such as speech recognition and face recognition. It is an extension of the Markov process. In a Markov process, a model consists of a series of states. These states are visible to the outside world. The transition from one state to another state is controlled by a transition probability. In HMM, the state is “hidden”. The outside world could only observe parameters, but the undergoing state sequence which generates the parameters is not known. The challenge in HMM is to determine the hidden state sequence based on the observable parameters.

An HMM is characterized by the following parameters:

1. The number of states in a model, N . The set of all states is denoted by $S = \{S_1, S_2, \dots, S_N\}$ and at time t , the model is at the state q_t .
2. The number of observations generated by a state, M . The set of observation symbols is denoted by $V = \{v_1, v_2, \dots, v_M\}$. For simplicity, in the current discussion, an HMM is assumed to generate only discrete symbols.
3. The state transition probability, $A = \{a_{ij}\}$, where

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i], 1 \leq i, j \leq N \quad (2.3.1)$$

4. The observation probability distribution in a state, $B = \{b_j(k)\}$, where

$$b_j(k) = P[v_k \text{ at } t \mid q_t = S_j], 1 \leq j \leq N \text{ and } 1 \leq k \leq M \quad (2.3.2)$$

5. The initial state, $\pi = \{\pi_i\}$, where

$$\pi_i = P[q_1 = S_i], 1 \leq i \leq N \quad (2.3.3)$$

A complete parameter set for an HMM is usually represented as

$$\lambda = \{A, B, \pi\} \quad (2.3.4)$$

A series of observation symbols $O = O_1 O_2 \cdots O_T$ is generated by an HMM according to the following procedure

1. Initially the model is at the state q_1 according to π .
2. At time $t = 1$, it generates symbol O_1 based on the output distribution $b_j(k)$ at q_1 .
3. The model now transits to the next state q_2 based on a_{ij} , at time $t = 2$, and the observation symbol O_2 is generated by q_2 .
4. Step 3 repeats until all the observation symbols have been generated.

Although the observation symbols may be generated as described above, there are still three basic problems that need to be resolved.

Problem one: Given an observation sequence $O = O_1 O_2 \cdots O_T$ and an HMM $\lambda = \{A, B, \pi\}$, how is one to calculate $P(O \mid \lambda)$, i.e., the probability that the HMM generate the sequence.

Problem two: Given an observation sequence $O = O_1 O_2 \cdots O_T$ and an HMM $\lambda = \{A, B, \pi\}$, what is the most likely state sequence $q_1, q_2, \cdots, q_T$ that generates this observation sequence.

Problem three: Knowing an observation sequence $O = O_1 O_2 \cdots O_T$ and an HMM $\lambda = \{A, B, \pi\}$, how is one to adjust the parameters of the HMM to maximize the $P(O | \lambda)$.

These three problems relate to the training and testing process of an HMM. Problems 1 and 3 involve the evaluation and estimation problem, i.e., training process of the HMM. Problem 2 involves the decoding problem, i.e., the testing process. The solutions of these three problems are fundamental to the development of an HMM based recognition engine. Next, some basic mathematics used to solve each problem is discussed.

To solve the first problem, suppose that a state sequence

$$Q = q_1, q_2, \cdots, q_T \quad (2.3.5)$$

generates an observation sequence. The probability could be written as

$$P(O | Q, \lambda) = \prod_{t=1}^T p(O_t | q_t, \lambda) \quad (2.3.6)$$

By using the forward part of the Forward-Backward algorithm, the above probability could be solved. Define

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t, q_t = S_i | \lambda) \quad (2.3.7)$$

as the probability of generating the partial observation sequence $O_1 O_2 \cdots O_t$ at time t , and the current state is at S_i , given the model λ . The Forward algorithm has three steps:

1) Initialization:

$$\alpha_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N \quad (2.3.8)$$

2) Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), \quad \begin{matrix} 1 \leq t \leq T-1 \\ 1 \leq j \leq N \end{matrix} \quad (2.3.9)$$

3) Termination:

$$P(O|\lambda) = \sum_{i=1}^N \alpha_T(i) \quad (2.3.10)$$

The final result in 2.3.10 is the sequence probability, given the HMM. This probability may also be calculated using the backward part of the Forward-Backward algorithm in which the probability is induced from the last state to the first state. Define

$$\beta_t(i) = P(O_{t+1} O_{t+2} \cdots O_T | q_t = S_i, \lambda) \quad (2.3.11)$$

as the probability of generating the partial observation sequence $O_{t+1} O_{t+2} \cdots O_T$, given that at time t , the model is at the state S_i , and model λ . The backward part of the Forward-Backward algorithm also has 3 steps:

1) Initialization

$$\beta_T(i) = 1, \quad 1 \leq i \leq N \quad (2.3.12)$$

2) Induction

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), \quad t = T-1, T-2, \dots, 1, \quad 1 \leq i \leq N \quad (2.3.13)$$

3) Termination

$$P(O|\lambda) = \sum_{i=1}^N \pi_i b_i(O_1) \beta_1(i) \quad (2.3.14)$$

The final result in 2.3.14, which is equivalent to 2.3.10, is the sequence probability, given the HMM. Later, the Forward-Backward algorithm is applied in the third problem again to solve the estimation problem.

To solve the second problem, one has to define what is meant by “the most likely sequence” in this case. Out of the many definitions available, the most commonly used one defines it to be “the sequence with the most likely individual state”. The Viterbi algorithm is applied to solve such a problem.

Given the best state sequence, $Q = \{q_1 q_2 \cdots q_T\}$, for the observation $O = \{O_1 O_2 \cdots O_T\}$, define

$$\delta_t(i) = \max_{q_1, q_2, \dots, q_{t-1}} P[q_1 q_2 \cdots q_t = i, O_1 O_2 \cdots O_t \mid \lambda] \quad (2.3.15)$$

$\delta_t(i)$ is the most probable path at time t , which accounts for the first t observations and ends at the state S_i . By induction,

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i) a_{ij} \right] b_j(O_{t+1}) \quad (2.3.16)$$

For each time instant t and state j that maximizes equation 2.3.16, the state sequence needs to be recorded. Array $\psi_t(j)$ is used to retrieve the state sequence. The procedure for the Viterbi algorithm is as follows:

1) Initialization:

$$\delta_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N \quad \text{and} \quad \psi_1(j) = 0 \quad (2.3.17)$$

2) Recursion:

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(O_t), \quad 2 \leq t \leq T \quad 1 \leq j \leq N \quad (2.3.18)$$

$$\psi_t(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}], \quad 2 \leq t \leq T \quad 1 \leq j \leq N \quad (2.3.19)$$

3) Termination:

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)] \quad (2.3.20)$$

$$q_T^* = \arg \max_{1 \leq i \leq N} [\delta_T(i)] \quad (2.3.21)$$

4) Path (state sequence) backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \dots, 1 \quad (2.3.22)$$

The Viterbi algorithm is quite similar to the forward part of the Forward-Backward algorithm. The difference is in the recursion step, where the summation in the Forward algorithm is replaced with the maximization.

The third problem is the most difficult of the three. There are many methods of model parameters re-estimation that maximize the observation likelihood. One way to adjust $\lambda = \{A, B, \pi\}$ to maximize $P(O|\lambda)$ at each state is to use the Baum-Welch algorithm. The Baum-Welch algorithm employs the Forward-Backward algorithm, which solves the first problem, to compute the partial observation probability.

The probability of being in state S_i at time t , and in state S_j at time $t+1$, given the observation sequence and the model parameter is defined by

$$\xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda) \quad (2.3.23)$$

By using the definition of Forward-Backward variable in 2.3.7 and 2.3.11, the above formula becomes

$$\xi_t(i, j) = \frac{P(q_t = S_i, q_{t+1} = S_j, O | \lambda)}{P(O | \lambda)} = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O | \lambda)} = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^N \sum_{j=1}^N \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \quad (2.3.24)$$

The probability of being at the state S_i at time t could be defined as

$$\gamma_t(i) = \sum_{j=1}^N \xi_t(i, j) \quad (2.3.25)$$

If both $\xi_t(i, j)$ and $\gamma_t(i)$ are summed over the time index t , the results are as follows

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{expected number of transitions from } S_i \quad (2.3.26)$$

$$\sum_{t=1}^{T-1} \xi_t(i, j) = \text{expected number of transition from } S_i \text{ to } S_j \quad (2.3.27)$$

Hence a reasonable model parameter re-estimation is

$$\bar{\pi}_i = \text{expected frequency (number of times) in state } S_i \text{ at time } (t=1) = \gamma_1(i) \quad (2.3.28)$$

$$\bar{a}_{ij} = \frac{\text{expected number of transitions from state } S_i \text{ to } S_j}{\text{expected number of transitions from state } S_i} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad (2.3.29)$$

$$\bar{b}_i(k) = \frac{\text{expected number of times in state } i \text{ and observing symbol } v_k}{\text{expected number of times in state } i} = \frac{\sum_{t=1}^T \gamma_t(i)}{\sum_{t=1}^T \gamma_t(i)} \quad (2.3.30)$$

The reestimated model is defined as $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$. It may be proved that

$$P(O | \bar{\lambda}) > P(O | \lambda).$$

2.4 Feature Extraction

In an ASR system, the speech waveform is first converted into feature vectors representing the acoustic information, before being fed into the recognition engine. A feature vector should capture the inter-class discriminatory information in the speech signal and is expected to perform consistently regardless of environment noise type and SNR level. If properly designed, a good speech feature extraction algorithm could ease the burden on the recognition engine. Unfortunately, this is difficult to achieve as noise could seriously distort the speech waveform, affecting the intelligibility of speech. Generally, the recognition accuracy under noisy condition is much worse than that obtained for clean speech.

The most widely used speech feature vector is based on the Mel Frequency Cepstral Coefficients (MFCC), which is the parametrization of choice in many speech recognition applications. The other commonly used feature vector is Perceptual Linear Prediction (PLP) coefficient [52] which is based on the Linear Predictive Coefficient (LPC) [44, 45 and 46]. In recent years many new speech features have been proposed, e.g., higher lag autocorrelation coefficients [57, 58], PAC based autocorrelation coefficients [10, 13, 54] and spectral entropy based feature [56]. Method of calculating the MFCC feature vector is reviewed in this section. In addition, the new speech feature proposed recently for robust speech recognition, namely Phase Autocorrelation (PAC) derived MFCC, is examined using AURORA 2. PAC MFCC is quite similar to traditional MFCC. Instead of using the energy spectrum as in MFCC, PAC MFCC uses the phase angle spectrum to calculate the cepstral coefficients, because phase angle is believed to be less affected by noise. The experimental results show that the performance degradation of PAC MFCC is much smaller than that of MFCC in adverse acoustic environments.

2.4.1 Mel Frequency Cepstral Coefficient

MFCC is a de facto standard for feature vectors in speech recognition. Many databases use MFCC to obtain their baseline results and in most of the available literature MFCC is employed for comparison. In addition, lots of cepstral domain feature enhancement techniques, as will be discussed in chapters 4 and 5 are based on MFCC. Figure 2-2 is a diagram showing the process of calculating the MFCC_E_D_A ($c_1 \sim c_{12}$, log energy, plus their first and second derivatives). Conventionally, the resultant feature vector is 39-dimensional. Next, each block in Figure 2-2 is described in more detail.

Figure 2-2: MFCC processing

2.4.1.1 Pre-emphasis

It is well known that most of the speech energy falls below 7 kHz and because most of speech spectra have a drop off of 6-dB/octave, the high frequency components have much smaller amplitude with respect to the low frequency components. This results in a high dynamic range within the speech bandwidth. A pre-emphasis at high frequencies is therefore required to make all components similar in amplitude. The pre-emphasis filter is a high pass filter, defined in the time domain as

$$y(n) = x(n) - a \cdot x(n-1) \quad (2.4.1)$$

where a is the pre-emphasis parameter typically taken as 0.97. Equation 2.4.1 is essentially a first order FIR filter in the z-domain. Figure 2-3 shows the frequency response of the pre-emphasis filter. Generally, it suppresses the low frequency components and amplifies the high frequency components. The output signal of the pre-emphasis filter has a relatively lower dynamic range.

Figure 2-3: Frequency response of pre-emphasis filter at $a = 0.97$

2.4.1.2 Windowing

As the speech signal is non-stationary, it is processed frame by frame. Within each frame the signal's spectral components are assumed to be stationary and the frame length is typically 20 to 40 ms. The simplest window for frame processing is the rectangular window, in which magnitudes inside the window are set to be one, and those outside the window are set to be zero. However, a rectangular window has high side lobes in its frequency spectrum. This is called spectral leakage. To solve this problem, windows which taper off at both ends are generally used. In speech feature extraction, the most commonly used window shape is the Hamming window, defined by

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right), \quad n = 0, 1, \dots, N-1 \quad (2.4.2)$$

The windowed frames are overlapped, typically by 50%. For example, for speech sampled at 16,000 Hz, the window length is 25 ms and the window shift is 10 ms. There is therefore 60% overlap between adjacent windows and the number of samples within a

window is 400. All these samples are finally converted into a 39-dimensional MFCC feature vector, so the sampling rate of the feature vector is reduced to 100 Hz. Unless stated otherwise, all the feature vectors reported in this thesis have been obtained by using a sampling rate of 100 Hz.

2.4.1.3 Fast Fourier Transform

A standard transform, the Discrete Fourier Transform (DFT), converts the speech signal from the time domain to the frequency domain. DFT is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) \exp\left(-j \frac{2\pi nk}{N}\right), \quad k = 0, 1, 2, \dots, N-1 \quad (2.4.3)$$

where N is the number of samples. To enable faster processing, the Fast Fourier Transform (FFT) is applied. In FFT, N is taken to be the nearest power of 2. For a speech signal sampled at 16,000 Hz and windowed at a length of 25 ms, the number of sample in each frame is 400. To enable the fast algorithm to be applied, N is taken to be 512 (nearest power of 2) by appending zeros at the end of the sample sequence. After the FFT, only half of the coefficients are saved, as the other half is the mirror image of the first half. Some smoothing techniques could also be applied to further reduce the number of coefficients.

2.4.1.4 Mel Filter Bank Analysis

Mel filter bank analysis is used to simulate the human auditory system, since our ears resolve frequencies non-linearly across the audio spectrum. Mel filter banks provide high resolution at low frequency and low resolution at high frequency. The relation between the Hz and Mel scale is defined as

$$Mel(f) = 2595 \log_{10} \left(1 + \frac{f}{700} \right) \quad (2.4.4)$$

After FFT, all the frequency components are grouped by the Mel filter bank and components within each Mel band are weighted and summed to obtain the power of the band. The shape of a filter band is taken to be triangular as shown in Figure 2-4 and the adjacent filter bands are overlapped by 50%. After the filter bank analysis, the frequency components within a frame are converted into filter bank coefficients. This is a significant data reduction step as the number of filter bank coefficients is typically 20 – 24.

Figure 2-4: Mel filter bank

2.4.1.5 Discrete Cosine Transform

The Discrete Cosine Transform (DCT) is employed to de-correlate the filter bank coefficients. Spectral coefficients are transformed into cepstral coefficients according to:

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^N m_j \cos \left(\frac{\pi i}{N} (j - 0.5) \right) \quad \text{for } i = 1, 2, \dots, N \quad (2.4.5)$$

where m_j is the log of the coefficient of j^{th} filter band and N is the number of filter bands.

The logarithm function converts the multiplication process into addition. Usually the first

12 coefficients ($c_1 - c_{12}$) are taken as cepstral coefficients, i.e., the 22 Mel filter bank coefficients are compressed into 12 cepstral coefficients.

2.4.1.6 Energy Measure

To augment the MFCC vector generated from DCT, an energy term can be appended. There are two ways of calculating the energy coefficient: log energy (logE, _E) and zeroth coefficient (c0, _0). Log energy is calculated in the time domain as

$$E = \log \sum_{n=1}^N s_n^2 \quad (2.4.6)$$

where the s_n represent speech samples and N is the total number of samples in the current frame. The zeroth coefficient is calculated by setting i in equation 2.4.5 to 0. In fact, the zeroth coefficient is a scaled sum of the Mel filter bank coefficients. Despite the fact that c0 is often redundant when logE is used, either c0 or logE could be appended to the MFCC vector. However, both energy coefficients could be used at the same time. They can be combined linearly as in the ETSI standard [9] or they could be combined with other cepstral coefficients through compression. The method of compression will be discussed in greater detail in Section 4.5.

2.4.1.7 Derivative Calculation

The performance of a speech feature vector can be enhanced by appending the first and second time derivatives because these derivatives capture the dynamic evolution of the speech signal. After the DCT, the 13-dimensional static feature vector (MFCC_E or MFCC_0) is appended with its first and second derivatives. The resultant 39-dimensional speech features are used during the recognition process.

2.4.2 Phase Autocorrelation (PAC) Derived Speech Feature

MFCC is an energy based feature vector in which the cepstral coefficients are derived from the power spectrum. Although MFCC gives good discrimination for clean speech, it shows excessive sensitivity to environmental noise. Generally, MFCC results in poor performance under noisy conditions. A possible reason could be that the autocorrelation coefficients, which are the inverse Fourier transform of the power spectrum, are distorted by the noise. In this section a recently proposed method, PAC derived speech feature [10, 13, and 54], is discussed and examined. Instead of the power spectrum, the PAC spectrum makes use of the angle between two vectors, which is expected to be less affected by the noise. Based on the PAC spectrum, an equivalent speech feature, known as PAC MFCC, is derived for ASR applications. A series of experiments verify that PAC MFCC is more robust than conventional MFCC under certain noisy conditions.

2.4.2.1 Phase Autocorrelation

The new phase autocorrelation feature explores the noise robust property of the angle between two vectors. The computation of the PAC spectrum involves two operations: energy normalization and inverse cosine transform. Assume a speech waveform is divided into a series of frames given by

$$\{s_0[n], s_1[n], \dots, s_t[n], \dots, s_{T-1}[n]\} \quad (2.4.7)$$

where T is the total number of frames. Each frame is defined as

$$s_t[n] = \{s[Kt+0], s[Kt+1], \dots, s[Kt+N-1]\} \quad (2.4.8)$$

where K is the frame shift and N is the frame size. It is well known from the Wiener-Khinchine relation that the spectral density and the time average of the autocorrelation function are Fourier transform pairs. The autocorrelation function can be calculated as

$$R[k] = x_0^T x_k \quad (2.4.9)$$

where
$$x_0 = \{s_t[0], s_t[1], \dots, s_t[N-1]\} \quad (2.4.10)$$

$$x_k = \{s_t[k], \dots, s_t[N-1], s_t[0], \dots, s_t[k-1]\} \quad (2.4.11)$$

x is assumed to be a part of a periodic signal with the period of N . If noise is present, the above equations become

$$R^n[k] = x_0^{nT} x_k^n \quad (2.4.12)$$

where
$$x_0^n = \{s_t^n[0], s_t^n[1], \dots, s_t^n[N-1]\} \quad (2.4.13)$$

$$x_k^n = \{s_t^n[k], \dots, s_t^n[N-1], s_t^n[0], \dots, s_t^n[k-1]\} \quad (2.4.14)$$

Due to the noise, the autocorrelation will be different from the one derived for the clean data (2.4.9). Since the two vectors have the same magnitude, we can rewrite equation 2.4.12 as

$$R^n[k] = \|x\|^2 \cos(\theta_k) \quad (2.4.15)$$

where $\|x\|$ is the magnitude of the vector and θ_k is the angle between the two vectors.

The phase autocorrelation coefficient is defined as

$$P[k] = \theta_k = \cos^{-1} \left(\frac{R^n[k]}{\|x\|^2} \right) - \frac{\pi}{2} \quad (2.4.16)$$

where the division by $\|x\|^2$ results in normalized autocorrelation coefficients. The PAC spectrum is essentially the inverse cosine transform of the normalized autocorrelation coefficients.

Figure 2-5: Inverse cosine function

The inverse cosine transformation is plotted in Figure 2-5. The curve is linear in the middle and at both ends the slope becomes larger. Due to this property, the input values at the two extremes are magnified in the output. In an autocorrelation sequence, the first few coefficients have large magnitude; therefore they will be enhanced after the inverse cosine transformation. The first few coefficients decide the shape of the spectral envelope, as they represent the slow varying components in the spectral domain. Since the variation across these coefficients is amplified, the shape of the spectral envelope, especially at the peak region, is better enhanced in the PAC spectrum. Meanwhile, for those components close to zero, the linear transformation will not enhance any variation across them.

Figure 2-6 shows the phase and magnitude autocorrelation sequences for speech. Notice that the first few coefficients in the phase sequence are relatively enhanced with respect to those in the magnitude sequence. According to the Wiener-Khinchine theorem, the Fourier transform of the time average of the autocorrelation sequence is the power spectrum. Figure 2-7 shows the Fourier equivalent in the spectral domain. By comparing both spectrum plots, the large spectral peaks are well preserved while the small peaks

which are likely due to the noise are suppressed in the corresponding PAC spectrum. Spectral peaks are of particular interest for ASR. Hence intuitively the PAC derived feature is expected to perform better than conventional MFCC.

Figure 2-6: Autocorrelation sequence for speech

Figure 2-7: Comparison of the magnitude and PAC spectrum for speech

Figure 2-8 shows the distribution of the PAC spectrum against the corresponding normalized magnitude spectrum. This plot could further explain the superiority of the

PAC spectrum. Each point represents a sample and the x and y coordinates correspond to the logarithm of the power spectrum and the PAC spectra, respectively. At high magnitude, there is a clear linear relationship between the power and PAC spectrum. These high magnitude components are most likely from spectral peaks. They carry important speech information for ASR. The linear relationship ensures that no information is lost during the inverse cosine transform. However, the large variation of the low magnitude components in the power spectrum has been confined to a small range in the PAC equivalent. These low magnitude components represent either the noise or unimportant speech information. Hence the attenuation of these components will improve the performance of the PAC feature vector.

Figure 2-8: The distribution of PAC spectrum against the magnitude spectrum

2.4.2.2 PAC Derived Feature

The PAC derived feature is a new feature vector proposed recently by Ikbal et al [10] for robust ASR. This new feature extraction method replaces the power spectrum with the PAC spectrum described in the last section, while keeping the rest of the algorithm the same as for MFCC. The resultant feature vector is referred to as PAC MFCC.

AURORA 2 is used to verify the effectiveness of PAC MFCC. The feature vector used in the experiment is 39-dimensional PACMFCC_0_D_A (PACMFCC_0 with first and second directives appended). The acoustic model is trained with clean utterances only. The test data set involves 10 types of noise. Results show that the performance of PAC MFCC under different noisy condition is not consistent.

Figure 2-9 compares the performance of PAC MFCC and MFCC under eight different noise conditions which are included in test set A and test set B. When no channel distortion due to the use of a different microphone from that used to obtain the training data is present, PAC MFCC generally performs better than MFCC, especially at high noise level. In a particular case, the improvement of word accuracy could be as high as 30%. However, under low noise conditions PAC MFCC degrades the performance. This is because the energy component itself carries some speech information [13]. Such degradation of performance in a clean environment is common for new speech features. Out of the eight different types of noise, the worst performance of PAC MFCC happens in the presence of babble noise, airport noise and restaurant noise. As these three types of noise have similar characteristics to speech noise, they are classified as non-stationary.

Figure 2-9: Performance of PAC MFCC and MFCC for clean training mode, the x-axis is the noise level of the utterance

The plot shows that when speech-like noise is present, the performance of PCA MFCC is much worse than MFCC at low noise level. Under such a condition, PAC MFCC is not as effective as it is for the stationary noise (Car, Exhibition, Subway, Train and Street noise). The rest of the five types of noise are considered to be stationary; the performance of PCA MFCC is good except for a small drop for a relatively clean environment.

Figure 2-10: Performance of PAC MFCC and MFCC for different frequency characteristic, the x-axis is the noise level of the utterance

The set C result is illustrated in Figure 2-10. When MIRS filtering, which models a different channel characteristic, is applied to the input of a speech recognizer, PAC MFCC has no advantage at all over MFCC. The phase autocorrelation sequence i.e., the angle between the time delayed speech frames, is severely distorted by the communication channel distortion, making the PAC MFCC feature ineffective. MFCC could still maintain its good discrimination under such an operating environment.

The experimental results on AURORA 2 demonstrate some noise robust properties of the PAC derived speech features. When no channel distortion is present, PAC MFCC outperforms conventional MFCC at high noise level. However, when different channel characteristics are considered or a low level non-stationary noise is present, the efficiency of PAC MFCC is unsatisfactory. Although there are also many other novel speech

features that have been proposed recently, MFCC is still a dominant feature vector. In the rest of this report, we will use MFCC for all the speech recognition experiments. As will be seen, in Chapter 3, a large vocabulary continuous speech recognition task uses MFCC as its baseline feature. In Chapter 4 and 5, some feature enhancement techniques based on MFCC will be discussed. Through proper enhancement, the MFCC could perform much better.

2.5 Database for Simulation Study

In an ASR system, the HMM must be trained and tested under well defined conditions. A speech recognition experiment requires a large amount of speech utterances (tens of hours) during the training stage and each utterance must be labeled with phone or word level transcription. Speech recording and transcription is tedious and time consuming. Fortunately, many databases are available from research institutes around the world. A database contains a large amount of speech data. It defines a framework to perform the testing, including the topology of the HMM, the training process, the language model and parameter setting and so on. Different algorithms could then be compared fairly in the same framework. All experimental results reported in this thesis have been obtained using one of three databases: AURORA 2, AURORA 4 and CENSREC-3.

2.5.1 AURORA 2

AURORA 2 [32] is a noisy version of the TIDIGIT database [33]. The speech files are isolated English digits. Eight different types of noise, namely suburban train, babble, car, exhibition hall, restaurant, street, airport and train station were selected to represent the most probable application scenarios for telecommunication terminals. Two frequency

characteristics defined by ITU, namely G.712 and MIRS, are used to filter the speech and noise. There are two training modes: clean and multi-condition. In the clean training mode, 8440 utterances recorded from 55 male and 55 female speakers are used. These utterances are filtered with the G.712 characteristic. The same data set but contaminated by noise is used in multi-condition training mode. Noise is added artificially to the clean speech at SNR levels of 20dB, 15dB, 10dB and 5dB. The clean set is also considered as a noise condition. Speech and noise are filtered with the G.712 characteristic before adding. The test data, which are recorded from 52 male and 52 female speakers, are grouped into 3 sets. Sets A and B contain four types of noise each. Again, speech and noise are filtered with the G.712 characteristic before adding. Set C contains two types of noise and this time noise and speech are filtered with the MIRS characteristic before adding. This set is intended to test the influence on recognition performance when channel distortion is present. There are 7 SNR levels in each noise condition (20dB, 15dB, 10dB, 5dB, 0dB, -5dB and clean) and the total number of testing utterances is 70070. Unless stated otherwise in this report, the recognition accuracy obtained using AURORA 2 is the average of results for 0dB, 5dB, 10dB, 15dB and 20dB.

2.5.2 AURORA 4

The evaluation framework of AURORA 4 [3] is very much the same as that of AURORA 2, except that the target now is a large continuous speech vocabulary with a size of 5000 words. AURORA 4 was created from the WSJ database in which the speech utterances were recorded by reading articles from the Wall Street Journal. Every utterance was recorded using two microphones: a close talking microphone (Sennheiser HMD440) and a secondary microphone (one of various different microphones) mounted on a desk. The

use of the secondary microphone was to enable recognition experiments with different frequency characteristics in the transmission channel. There are two training modes. In the clean training mode, 7138 clean utterances recorded in the close talking microphone are used to train the acoustic model. In the multi-condition training mode, the same set of clean utterances with half of them recorded using the secondary microphone is applied. Seventy-five per cent of the recordings from each microphone are corrupted by noise at an average SNR level of 15dB (uniformly distributed between 10 and 20 dB).

There are 14 test sets with 330 utterances in each. For utterances recorded from each of two microphones there are one clean set and six noisy sets. The 6 different types of noise, namely car, babble, restaurant, street, airport and train station, were added artificially at SNR levels uniformly distributed between 5 and 15 dB. These noises represent realistic scenarios of application environments for telecommunication. Since the speech is made up of continuous English utterances, the training and testing procedures are more complex than that of AURORA 2 where single word acoustic model is applied. AURORA 4 adopts the triphone model which is commonly used for large vocabulary systems. Unlike the AURORA 2 where the HTK-based testing scripts are given, users must develop their own testing procedures under certain guidelines. The evaluation framework of AURORA 4 will be described in more detail in Chapter 3.

2.5.3 CENSREC-3

CENSREC-3 [12, 43] is a Japanese database for the evaluation of isolated word recognition under in-car conditions. The distinct feature of this database is that all the utterances are recorded in a real noisy car driving environment. The speech files were recorded using a close-talking (CT) microphone (Sennheiser HMD410) and hands-free

(HF) microphone (Sony EXM77B) attached to the ceiling of the driver's seat. The recording conditions are listed in Table 2-1. There are 3 different vehicle speeds (idling, low-speed and high-speed) and 6 kinds of in-car environments (normal, hazard flasher on, air-conditioner on (fan low/high), audio on and windows open). A total number of 14050 training utterances spoken by 202 males and 91 females were recorded with each microphone. And the test data set consists of 14216 utterances spoken by 8 males and 10 females recorded from each microphone.

The evaluation framework comprises six different matching conditions between training and testing data. Conditions 1, 2, and 3 are classified as “well-matched condition”, the data recorded using the same microphone under the same environment are used for both training and testing. Condition 4 is classified as “moderately-mismatched condition”, the testing and training data were collected using the same microphone but under different recording environments. Conditions 5 and 6 are classified as “highly-mismatched condition”, the training and testing data were collected using different microphones under different recording conditions. In Tables 2-2 and 2-3, the circle (O) indicates that the data are included in the condition.

Table 2-1: Recording environment for testing data

Car speed	In-car conditions
Idling (quiet)	Normal, Hazard on, Fan (low), Fan (high), Audio on, Window open
Low speed	Normal, Fan (low), Fan (high), Audio on, Window open
High speed	Normal, Fan (low), Fan (high), Audio on, Window open

Table 2-2: Training data in CENSREC-3 database

Condition	1		2		3		4		5		6	
microphone	CT	HF	CT	HF	CT	HF	CT	HF	CT	HF	CT	HF
Idling	O	O	O	—	—	O	—	O	O	—	O	—
Low speed	O	O	O	—	—	O	—	—	O	—	—	—

Table 2-3: Testing data in CENSREC-3 database

Condition	1		2		3		4		5		6	
microphone	CT	HF	CT	HF	CT	HF	CT	HF	CT	HF	CT	HF
Idling	O	O	O	—	—	O	—	O	—	O	—	—
Low speed	O	O	O	—	—	O	—	O	—	O	—	O
High speed	O	O	O	—	—	O	—	O	—	O	—	O

The complexities of these three databases are different due to the different framework structures and decoding difficulties. AURORA 2 has the lowest complexity as the test data consist of only 11 English digits. CENSREC-3 has a moderate complexity as it consists of 50 isolated Japanese words. And AURORA 4 has the highest complexity as the test data are made up of continuous English speech. Hence the experimental results obtained on AURORA 4 are considered to be most reliable and those of AURORA 2, the least reliable.

Chapter 3 System Evaluation of AURORA 4

3.1 Introduction

Using current technology, accuracies close to 100% may be achieved for isolated word or digit recognition. On the other hand, large vocabulary continuous speech recognition remains a most challenging task and the performance, even when using the best available current technology, is still generally poor. As the vocabulary size increases, the dimension of HMM becomes larger. The high complexity of the model network increases the decoding time and degrades the accuracy drastically. A good language model is also needed to select the most probable word sequence during the decoding. The other problem with a large vocabulary system is that insufficient training data is available to cover the entire model set and as a result there are always some acoustic models under-trained. Therefore, some refinement techniques are employed, specifically to improve the recognizer's performance. An experimental framework for a large vocabulary continuous speech recognition task – AURORA 4, is discussed in this chapter. AURORA 4 [3], based on the Wall Street Journal database (CSR-I (WSJ0) Complete, from Linguistic Data Consortium), has been developed by the ETSI committee. The objective of AURORA 4 is to evaluate the robustness of either a feature extraction algorithm or an entire ASR system, at a 5000 word vocabulary task. There are 7138 training utterances and 330 testing utterances. Six different types of noise were added artificially to the clean speech, at an SNR level between 5 to 20 dB. More information on AURORA 4 is given in Section 2.5.2.

The training procedures are discussed in Section 3.2. The refinement techniques, namely mixture increment and decision tree based clustering, will be explained in more detail. Various parameters that need to be tuned in order to optimize the training and testing process are explained in Section 3.3. The baseline result obtained is shown in Section 3.4 and this chapter is summarized in Section 3.5.

3.2 Training Procedure

Before a model set is trained, a phone level transcription is prepared using a standard pronunciation dictionary. In our framework, we chose the CMU pronunciation dictionary (version 0.6) [18] with some local additions made to give full coverage of the training set. The CMU dictionary contains 39 phonemes, one short pause model (*sp*) and one silence model (*sil*). There are 41 models in the entire phone set. Since detailed orthographic transcriptions are given in the WSJ0 database, we prepared the phone level transcription by replacing each word by its phone elements. Two monophone level transcriptions were prepared, one with *sp* model in between the words and one without. Multiple pronunciations for a word were taken care of by a forced alignment process in which the most likely pronunciation is picked up. During the training process, the monophone level transcription will be converted into a cross word triphone level transcription. For example, the phrase “Beat it!” is represented as “*sil b iy t ih t sil*” in monophone level transcription. The corresponding cross word triphone level transcription becomes “*sil sil-b+iy b-iy+t iy-t+ih t-ih+t ih-t+sil sil*”. The triphone model takes account of the contextual effect in the pronunciation, i.e., the pronunciation of a phone is affected by the phone before and the phone after it such that a phone could be pronounced in different ways.

The baseline evaluation procedure and test results recommended by the AURORA working group was designed using a prototype system developed at the Institute for Signal and Information Processing (ISIP) in Mississippi State University. In our evaluation, HTK (Hidden Markov Model Toolkit) [1] is used both for training and testing. It is necessary for us to adjust the procedure and some parameters to suit HTK. Our evaluation procedure is quite close to that given in [2]. The details will be discussed next. Our HTK system is based on a 3-state left-to-right continuous-density model with one entry and one exit state. The feature extraction uses ETSI WI007 Front-end [17]. The training procedure is given in Table 3-1.

Table 3-1: Training procedure in HTK for AURORA 4 task

1. **Seed model:** Create an initial model with single Gaussian mixture. Use it as the seed model for every monophone.
2. **Monophone training:** Train the initial monophone HMM set with the phone level transcription (without *sp*) for 4 iterations of Baum-Welch algorithm. At this moment, assume that the *sp* model has not been inserted between the words and *sil* model occurs only at the start and the end of an utterance.
3. ***sp* model training:** Insert the *sp* model between the words. The single state *sp* model is tied with the central state of the *sil* model. Then all HMMs are trained with the phone level transcription (with *sp*) again for 4 more iterations of Baum-Welch algorithm.
4. **Forced alignment:** The current HMM set is forced to align with the acoustic data. This is to choose the most likely pronunciation in the case that there exists multiple pronunciations for a word. The new monophone transcription generated at this step is used for the rest of the training procedure.
5. **Final monophone training:** The set of HMMs are trained again with the new phonetic transcription from last step for 5 iterations of Baum-Welch algorithm.
6. **Convert monophone to triphone:** The set of HMMs are converted to cross

word triphone models. Only those triphones appearing in the training data are created. The new set of triphone HMMs is trained with triphone level transcription for 4 iterations of Baum-Welch algorithm.

7. **State-tying clustering:** To reduce the number of states and the parameters, those states with similar parameters are tied together. Decision tree based state tying clustering is implemented to control the number of states and ensure sufficient training data for each triphone model. After state tying, the new set of HMMs is trained with 4 iterations of Baum-Welch algorithm.
8. **Mixture increment:** The number of mixtures for each state is increased from 1 to 2, then to 4, 8 and 16. After each mixture increment, 4 iterations of Baum-Welch algorithm are applied.

3.2.1 *sil* and *sp* Model

sil and *sp* are two distinct models in an entire HMM set. *Sil* is used at the start and the end of each utterance while *sp* model is used in between the words. Both of them have different topologies from normal acoustic models. Figure 3-1 shows 3 different topologies of an HMM. An empty circle represents an effective state and a solid circle denotes either an entry or an exit state. An arrow indicates a state transition. Normal triphone or monophone models are left-to-right 3-state. A state transition occurs either between two adjacent states or within the same state. The *sil* model has 3 states with bidirectional transition between states 1 and 3. During training, the *sil* model is trained before the *sp* model is inserted so that *sil* model's parameters could be accurately estimated [2]. In the *sil* model, the extra bidirectional transitions between states 1 and 3 enhance the model by allowing more impulse noises to be absorbed by individual states [1]. The *sp* model, which is a single state model with a direct transition from entry state to exit state, is to train the model of the silence/noise between the words. This transition enables the model to generate no data if there is no silence between the words. Both *sil*

and *sp* models represent non-speech regions of an utterance and they might compete with each other when modeling silence. This can be avoided by tying the center states of the *sp* model and the *sil* model [1].

Figure 3-1: Topologies of the HMM: (a) *sp* model, (b) *sil* model and (c) triphone or monophone model

3.2.2 Decision Tree Based Clustering

In a triphone based large vocabulary speech recognizer, the number of HMMs is non trivial. There are millions of parameters to be estimated but only hours of speech data are available. For a system with 40 monophones, theoretically, the number of triphone models is $40^3 = 64,000$. Not every combination exists for English speech, but the remaining part is still a huge number. The insufficient amount of data will leave some models under-trained. One approach for solving this problem is to reduce the total number of states using tree based state tying clustering. In this system refinement method, those states that are close to each other will share the same set of parameters.

The basic idea of clustering is to tie all the states that are acoustically indistinguishable [11]. A phonetic decision tree is needed for clustering. It is a binary tree with a yes/no question at each node. These questions decide which child node a state is streamed to by checking the right or left context of the triphone model. Initially, all the states are at the

root node, these states are split at each node by answering the question attached to that node. The pool of states at each node is split in a way to maximize the likelihood of the training data. Such processes continue until a state reaches a leaf node. Finally all the states in the leaf node at the bottom are tied together.

Figure 3-2: Decision tree based state tying clustering

Figure 3-2 illustrates the structure of a decision tree. The center states of phone “iy” will be clustered according to their context. The advantage of the decision tree based clustering is that the phonetic decision tree could be saved into a separate file. If there is a vocabulary change or some new triphones are introduced, the phonetic decision tree could be loaded again and new triphones which do not appear in the training data could be integrated into the current HMM set by tracing down the decision tree.

3.2.3 Mixture Increment

Mixture increment is usually a final step in HMM training. The representation of the probability distribution of speech vectors generated by a state using single Gaussian distribution is generally not accurate. As the number of observations becomes large, a single Gaussian is replaced by a mixture of Gaussian components to describe the state

emission probability distribution. The number of mixtures is increased from 1 to 2, 4, 8, and 16 and so on at each step, until the desired number of mixtures is reached. This number depends on the training vectors available associated with the model. Generally, the more training data available, the more Gaussian mixtures needed. In practice, around 10 mixture components give good performance in large vocabulary recognition systems [11]. After each mixture increment, the whole HMM set needs to be re-estimated, including the optimum mixture weightings.

3.3 Parameter Tuning

After an entire set of HMMs has been trained, some parameters need to be adjusted to optimize the system performance. These parameters are: number of Gaussian mixture components, beam pruning threshold, threshold for state clustering, word insertion penalty and language model scale factor. The development set for parameter tuning is independent of the test data. We chose a subset of 164 utterances from the test set 01 as our development set. There is no overlap between the development set and the test set.

Table 3-2 shows the recognition results of the HTK system with different numbers of mixtures. When the number of mixtures is above 4, the difference in word recognition accuracy is quite negligible. The 8- and 16-mixture model systems perform almost equivalently. However, during training, at each increment of the Gaussian mixtures, the entire model set is re-estimated for 4 iterations of the Baum-Welch algorithm, which is quite time consuming. Reducing the number of Gaussian mixtures from 16 to 4 would reduce 8 iterations of re-estimation. This will save a large amount of time. Taking into account the computation load and the processing time, the number of mixtures in the base evaluation is therefore chosen to be 4. Table 3-2 shows that the 4-mixture model has a

slight degradation in accuracy over the 16-mixture and 8-mixture models, because the result is obtained based on the development set only. In fact, in the next section it is verified that the overall performance of the 4-mixture model can be better than that of the 16-mixture model for the clean training mode.

Table 3-2: Performance comparison for different Gaussian mixture models

No. of mixtures	Word recognition accuracy (%)
2	85.63
4	88.59
8	89.73
16	89.42

The beam pruning factor is used to speed up the training and decoding process and it trades off search error for processing time [2]. The ISIP system supports 3 levels of pruning, i.e., word, phone and state, while HTK only supports state level pruning. In HTK, beam pruning limits the active path in training and decoding. Those paths whose likelihoods fall more than a beam width below the best record, will be pruned. Pruning can reduce the amount of computation required by an order of magnitude. If the beam width is too low, search error will occur or no path can survive at the end. But if the beam width is too high or no pruning is used, both the training and the decoding will take a much longer time. In our AURORA 4 training and testing procedure, the beam pruning level is 250 as suggested by the HTK book [1]. This setting would significantly reduce the time for training and decoding.

The threshold for state clustering is used to control the total number of states in the final model set. To achieve optimum state tying, the splitting of all the states in a single mode is done in such a way so as to achieve the maximum likelihood value of a single Gaussian with diagonal covariance at each child node [1]. Such a process continues until the

increase in log likelihood falls below the threshold or no question is available. In our experiment, the threshold is adjusted until the number of states in the final HMM set is as close as possible to the number (3215) given by the ISIP prototype system. The number of states before tying is 23925 and after the state tying it is reduced to approximately 3200. The compression ratio is 7.5:1.

A speech recognizer is optimized when the word insertion error and the deletion error are balanced with each other. This could be achieved by proper tuning of the word insertion penalty and the language model scaling factor in HTK. The word insertion penalty is to control the poorly articulated words like “a”, “oh”, and “the”; the product of the language model factor and the bigram probability is used to evaluate the contribution of the language model to the overall path score [2]. For example, if the word insertion penalty is set to be m and the language factor is set to be n , during the decoding, the language model log probability x would be converted to $nx+m$.

Table 3-3 shows the word recognition accuracy of the development set at different combinations of language model factor and word insertion penalty. When the language model factor is below 15, the recognition result is not so good. When the language model factor is above 15, different word insertion penalties introduce very little changes in the results. The best performance occurs at a language factor of 15 and a word penalty of -12. Table 3-4 lists some selected experimental results. Based on the results shown in Tables 3-3 and 3-4, the language model factor was set to be 15 and the word insertion penalty was set to be -12 in our baseline evaluation. At this combination, the insertion error and the deletion error are quite close to each other. In addition, the word accuracy at current setting is also the optimum.

Table 3-3: Recognition accuracy at different language model factors and word insertion penalties (%) on evaluation set

		Language model factor				
		5	10	15	20	25
Word insertion penalty	-18	-	-	-	88.10	79.04
	-16	-	84.91	88.25	88.25	79.49
	-14	-	85.03	88.36	88.40	80.89
	-12	-	85.41	88.59	88.36	-
	-10	74.34	85.22	88.36	-	-
	-8	73.62	84.80	-	-	-

Table 3-4: Recognition accuracy at different language model factors and word insertion penalty tuning

Language model factor	Word insertion penalty	Word accuracy (%)	Substitution error (%)	Insertion error (%)	Deletion error (%)
15	-16	88.25	8.6	0.9	2.3
20	-16	88.25	8.0	0.8	3.0
15	-14	88.36	8.5	0.9	2.2
20	-14	88.40	8.2	0.8	2.7
15	-12	88.59	8.4	1.0	2.0
20	-12	88.36	8.2	0.8	2.6
15	-10	88.36	8.6	1.2	1.9

3.4 System Evaluation

The AURORA 4 defines 2 training modes: clean and multi-condition. Since clean training mode is of more interest to us, only the result for the clean training mode is listed here. The test data consist of 14 test sets with 330 utterances in each set. It turns out that testing all the files for each noise condition is very time consuming. The baseline evaluation is designed to use a subset of 166 utterances from each test set. The database provides each utterance in two sampling frequencies: 8 kHz and 16 kHz. Our experiment would focus on the 16 kHz data only. The 14 test sets will give 14 test results, which is hard to manipulate. In our experiment, we regrouped these 14 test sets into 4 subsets as is done in [4]. They are

Set A: clean data from Sennheiser microphone. (Set 1)
Set B: noisy data from Sennheiser microphone. (Sets 2 to 7)
Set C: clean data from secondary microphone. (Set 8)
Set D: noisy data from secondary microphone. (Sets 9 to 14)

During the decoding, a language model is needed. In our experiment, we chose a 5000 closed non-verbalized punctuation language model with back-off mode. The language model is supplied by the MIT Lincoln Laboratory and it is included in the WSJ0 database. But it does not cover all the vocabulary in the AURORA 4 test data. Therefore, the language model used for AURORA 4 decoding is interpolated from the WSJ0 model and the model generated from the AURORA 4 test data. The interpolated language model should have a comparable perplexity as the one from WSJ0 (235.85). By using an interpolation weight of 0.999, the interpolated language model has a perplexity of 235.6. Table 3-5 shows the word recognition accuracies of our HTK baseline system. For comparison, the results of the ISIP prototype system are also listed here. Both the ISIP and our HTK system use the ETSI WI007 front-end to extract speech feature. The 39-dimension features (MFCC_E_D_A) includes 13-dimension static features ($c_1 \sim c_{12}$, plus log frame energy) with first and second derivatives appended. Our HTK system gives better results than that of the ISIP prototype system. This baseline result will be used as the benchmark for the later experimental results on AURORA 4. The performance of Advanced Front-end (ETSI ES 202 050) [9] is also shown in Table 3-5. The ETSI standard, which encompasses a noise reduction stage, has much more significant error reduction over the baseline. Note that the average recognition accuracy in the table is not the average result of 4 groups; instead, it is the average of the 14 test

sets defined by AURORA 4. All the AURORA 4 experimental results reported in this thesis will be calculated in the same way.

Table 3-5: Recognition accuracy (%) on baseline evaluation using AURORA 4

Test set	Clean training				
	A	B	C	D	Ave
ISIP	85.5	40.68	46.70	25.48	37.80
HTK	87.85	45.84	50.57	30.29	42.51
Advanced Front-end	89.87	74.16	71.45	57.99	68.16

Our evaluation runs on a computer with Pentium D CPU of 3.2 GHz and 2 GB RAM. The model training time is around 2 hours and the decoding process (based on the 166 set) takes around 13 hours. The time spent for noisy data decoding is twice as long as that for clean data. Table 3-6 is the comparison of the 4-mixture and 16-mixture systems. In clean training mode, the 4-mixture system performs better than the 16-mixture model. The reason of which is explained in section 3.2.3. The amount of training data in AURORA 4 is insufficient to support a 16-mixture system as the parameters of HMM set will be underestimated. In addition, reducing the number of mixtures could significantly reduce the training time. Therefore, all the experiments on AURORA 4 in this thesis will use 4-mixture models.

Table 3-6: Comparison of different mixture models in AURORA 4 (%)

Test condition	Clean training Average result
HTK (16-mixture)	41.46
HTK (4-mixture)	42.51

3.5 Summary

The details involved in building a large vocabulary continuous speech recognition system using AURORA 4 are described in this chapter. Because such a system has a higher complexity than that required for a simple digit recognition task, the process of building

such a system is also complicated. The decoding of speech is broken down into two levels: The feature level decoding translates the speech features into phones and the language level decoding translates the phones into words and sentences. A language model employed in the language level decoding requires a tuning process to find the optimal word insertion penalty and language model scale factor. And Gaussian mixture model and state clustering techniques in the phone level decoding require the same process to find the optimal number of mixtures and threshold for state clustering. The tuning of a speech recognition system depends on amount of the speech features and its distribution.

Although the parameters of an HMM set are at their optimal value, the baseline recognition accuracy is too low for practical use. Further improvement is therefore necessary. The usual approach of additional processing includes feature normalization, cepstral filtering or model adaptation. They will be described in the following chapters.

Chapter 4 Feature Normalization for Robust Speech Recognition

4.1 Introduction

An automatic speech recognition (ASR) system degrades severely when there is a mismatch between the training and testing features [42]. The major causes for this mismatch include additive noise, channel distortion and speaker variability. Front-end methods for improving the robustness of speech feature vectors basically fall into three categories. The first category includes noise removal methods, i.e., speech enhancement. Noise spectral subtraction is a typical example in which the clean speech spectra are estimated by subtracting the estimated noise spectra from those of the noisy speech in the spectral domain. Alternatively, adaptive filtering is applied in the time domain to remove the noise. The second category of the front end methods for improving speech vector robustness includes extraction methods of a robust speech feature that is minimally affected by the environment. MFCC has been proved to be not robust enough in an adverse noise environment and many new features have been proposed in recent years to overcome this deficiency [54, 56 and 57]. The PAC MFCC discussed in Chapter 2 has been shown to perform better than MFCC under certain noisy conditions. Unfortunately, no feature has so far been found to be completely invariant to noise. All speech features suffer performance degradation problems when noise is present. Nevertheless, a good feature vector is expected to degrade moderately in an adverse environment. Cepstral domain feature enhancement belongs to the third category of front-end techniques for achieving robust ASR. It helps to improve the robustness of extracted feature vectors. In

recent years cepstral domain feature enhancement has attracted more attention within the speech community due to its simple implementation. The objective of feature enhancement is to remove the noise components and reduce the mismatch between the clean and noisy feature trajectories in the cepstral domain. Some feature enhancement techniques are studied and compared in Chapters 4 and 5. Chapter 4 discusses some feature normalization techniques while the focus of Chapter 5 is on cepstral domain filtering methods.

While mean and variance normalization (MVN) is the most widely used feature normalization method, many improved MVN methods have been studied in past years. These include segmental variance normalization (SVN) [5, 50], recursive SVN [22] and MVN with energy-based frame selection [19]. These traditional techniques are reviewed in this chapter. In addition, an MVN with feature entropy based frame selection is proposed and test results are presented showing it to be as good as an energy-based approach for frame selection.

Of the feature coefficients currently in use, the log energy coefficient has characteristics that are different from all other coefficients. Conventional MVN is not efficient for normalizing the log energy coefficient. A log energy dynamic range normalization proposed by Zhu and O'Shaughnessy [23] has been studied. It has also been tested when combined with MVN and SVN. Histogram equalization [39, 40 and 41] is another normalization technique which is shown to have an advantage over conventional variance normalization methods, as it normalizes all the moments of a coefficient PDF.

This chapter is organized as described next. Conventional MVN and improved MVN including SVN, recursive SVN, and MVN with energy based reliable frame selection are

revised in Section 4.2. At the end of this section, MVN with feature entropy based frame selection is introduced. The effectiveness of the log energy dynamic range normalization when combined with MVN or SVN is studied in Section 4.3. HEQ is compared with conventional variance normalization in Section 4.4. Furthermore, Class-based HEQ is shown to be superior. PCA based feature vector compression is described in Section 4.5. In Section 4.6, the best results obtained when using the techniques described in previous sections are compared with those obtained when using the Advanced Front-end standard (ES 202 050) from ETSI. All experimental results have been obtained using AURORA 4.

4.2 Cepstral Mean and Variance Normalization

Cepstral Mean and Variance normalization (MVN) is the most commonly used technique in feature enhancement. It effectively reduces the mismatch between the clean and noisy feature trajectories by negating the shift in cepstral mean and reduction in dynamic range caused by the noise signals. Suppose that the test data are represented by a series of feature vectors $X = [\bar{C}(1) \ \bar{C}(2) \ \dots \ \bar{C}(t) \ \dots \ \bar{C}(T)]$, for $t = 1, 2, \dots, T$ and T is the number of frames. $\bar{C}(t) = [c(t,1) \ c(t,2) \ \dots \ c(t,k) \ \dots \ c(t,K)]^T$ is a vector representing a frame, with cepstral coefficients as its elements and K is the number of coefficients in the vector. In MVN, the mean and standard deviation are calculated for each coefficient trajectory as:

$$\mu_k = \frac{1}{T} \sum_{t=1}^T c(t,k) \quad \text{and} \quad \sigma_k = \sqrt{\frac{1}{T} \sum_{t=1}^T (c(t,k) - \mu_k)^2} \quad \text{for } k = 1, 2, \dots, K \quad (4.2.1)$$

Then every cepstral coefficient is normalized by subtracting the mean and dividing by the standard deviation

$$\hat{c}(t, k) = \frac{c(t, k) - \mu_k}{\sigma_k} \text{ for } k = 1, 2, \dots, K \quad (4.2.2)$$

The resultant cepstral coefficients have a zero mean and unit variance. The normalization of variance reduces the variability of the feature coefficients so that the parameter statistics are always the same, irrespective of noise conditions [22]. Due to their low complexity and high efficiency, many feature enhancement methods employ MVN as a preprocessing stage.

Table 4-1 shows the evaluation results of MVN on AURORA 4 in clean training mode. There are two static feature vectors to be tested: MFCC_E and MFCC_0, i.e., the cepstral coefficient $c_1 \sim c_{12}$ with either log energy (logE) or zeroth coefficient, c_0 . The 13-dimensional static feature is normalized before appending the first and second derivatives. The final feature vector used in recognition is 39-dimensional. It is observed that MVN on MFCC_0 is more effective than MVN on MFCC_E. The difference of the average recognition accuracy between the normalized MFCC_E and MFCC_0 is more than 20%. This is because the log energy coefficient has a characteristic different from cepstral coefficient $c_0 \sim c_{12}$. It is shown that MVN is not a good method for normalizing the log energy coefficient. In fact, another method exists for normalization of log energy and it is tested to be more effective. This issue will be revisited in Section 4.2. Although there are two basic features considered during the recognition, for consistency throughout the reports, all the baseline features are referred to MFCC_E, unless it is stated otherwise.

Table 4-1: Evaluation of MVN on AURORA 4 (%)

Test Set	Clean speech training				
	A	B	C	D	Ave
MFCC_E D A	87.85	45.84	50.57	30.29	42.51
+MVN	88.99	46.83	58.75	25.40	41.51
MFCC_0 D A	89.61	49.87	53.48	36.95	47.43
+MVN	89.50	67.58	74.11	49.31	61.78

The mean and variance normalization has low computation complexity yet fast processing speed in robust ASR. Furthermore, some improved variance normalization techniques have been shown to be much better than conventional MVN but with a slight increase in computation load. Some of these methods are discussed next.

4.2.1 Segmental Mean and Variance Normalization

The MVN is applied over the whole utterance, thereby introducing a long processing delay. This is not feasible in a real time application, as the recognizer cannot wait until the end of the whole utterance. An improved method is the segmental variance normalization (SVN). In SVN [5, 50], instead of the whole utterance, the variance is normalized within a sliding window. This will significantly reduce the processing delay.

An SVN process is shown in Figure 4-1. Suppose the length of the window is N . At the beginning, the window is at the start of the utterance and feature vectors are inserted into the window one by one as the window moves to the end of the utterance. The coefficients are not normalized until there are $N/2$ vectors inside the window. Then the first vector at the center of the window is calculated. The window keeps moving. At each time, a new vector is inserted and the vector at the center is processed. When the window is not full, the mean and variance are calculated using all vectors inside the window. At the moment the window is full, the first $N/2$ vectors have already been calculated. The vector at the center is then normalized by using the mean and variance of the entire window. This process is repeated until the end of the utterance is reached when the last $N/2$ vectors are processed together.

Figure 4-1: The sliding window in segmental variance normalization

The advantage of SVN is its fast adaptation to noise especially when the noise is non-stationary across the whole utterance. The process delay is fixed at half of the window length. Conventional MVN and SVN are compared on the AURORA 4 database, in Table 4-2. In the evaluation, the feature vectors were sampled at 100 Hz and N , the window length was set to be 100, i.e., 1 second. The baseline feature was MFCC_E_D_A (c1~c12, plus the log energy, appended with first and second derivatives). The 13-dimensional static features were normalized before appending derivatives.

SVN was tested to be superior to conventional MVN. Note that in Table 4-2 the average MVN result is even worse than the baseline result. This is again because the MVN is not effective on log energy coefficient normalization. This issue will be addressed in Section 4.2, in which it shows that log energy dynamic range normalization is a better approach. Unlike MVN, SVN does not pose a problem for normalization of the log energy trajectories. This is because within a sliding window, the noise could be considered as stationary and the noise power does not vary significantly. In Table 4-2 the average recognition accuracy is successfully improved by 10%, from 41.51% to 52.29%.

Table 4-2: Evaluation of SVN on AURORA 4 (%)

Test set	Clean speech training				
	A	B	C	D	Ave
Baseline	87.85	45.84	50.57	30.29	42.51
MVN	88.99	46.83	58.75	25.40	41.51
SVN	87.79	57.11	66.27	39.24	52.29

In SVN, the window length affects the estimation accuracy of the mean and standard deviations. As the noise characteristic changes with time, too long a window will not capture the fast variation of the noise property, while too short a window will not collect sufficient statistical information of the noise. Therefore, an optimum length is expected for the sliding window. Table 4-3 compares the effectiveness of sliding windows of lengths 0.8, 1.0 and 1.2 seconds. The test used set A in AURORA 2 in clean training mode. Obviously, the 1.0 second window is best for the noise suppression. This is also consistent with the result obtained in [5].

Table 4-3: Different sliding window lengths in SVN (%)

L	AURORA 2 Set A
80 (0.8s)	74.25
100 (1s)	76.07
120 (1.2s)	74.49

4.2.2 Recursive Segmental Variance Normalization

SVN successfully reduces the delay in MVN from the whole utterance length to half of the segment size (0.5 second in our experiment), while still improving the recognition accuracy significantly. The processing delay could be shortened further. In the last section, results from experiments were presented showing that when the segment length is below 1.0 second, the estimation of mean and variance will be less accurate. Such a deficiency could be rectified using recursive segmental feature normalization (Recursive SVN) [22, 49] as it has been proved that the mean and variance of coefficients can be estimated recursively [21]. Recursive SVN, which combines both recursive algorithm and

the segmental approach, further reduces the process delay without compromising on performance.

Suppose the length of a segment is N . When the segment is full, the initial mean and variance is calculated as

$$m_t(i) = \frac{1}{N} \sum_{t=1}^N c_t(i) \quad (4.2.3)$$

$$\sigma_t(i) = \sqrt{\frac{1}{N} \sum_{t=1}^N c_t^2(i) - m_t^2(i)} = \sqrt{s_t^2(i) - m_t^2(i)} \quad (4.2.4)$$

Once the mean and variance are known, the first vector inside the segment could be calculated as

$$\hat{c}_{t-N}(i) = \frac{x_{t-N}(i) - m_t(i)}{\sigma_t(i)} \quad (4.2.5)$$

All feature vectors are thus processed with the delay of N frames. Every time a new vector is inserted into the segment, the mean and square sum are updated as

$$m_t(i) = \lambda \cdot m_{t-1}(i) + (1 - \lambda) \cdot c_t(i) \quad (4.2.6)$$

$$s_t^2(i) = \lambda \cdot s_{t-1}^2(i) + (1 - \lambda) \cdot c_t^2(i) \quad (4.2.7)$$

where λ is the forgetting factor used to reduce the effect of the past frame. Equation 4.2.4 is then used to calculate the new standard deviation before normalizing the next frame using 4.2.5. The remaining frames are calculated in the same way until the end of the utterance is reached.

Table 4-4 shows the experimental results using AURORA 4 in clean training mode. In SVN with a 1.0 second window, the process delay is 0.5 second, i.e., 50 frames at 100 Hz sampling rate. A lower delay time requires the window of recursive SVN to be shorter than 0.5 second. In our experiment, 0.3 second window ($N = 30$) was tested and different

forgetting factors were attempted. It was found that at proper settings ($\lambda = 0.93, 0.91$ or 0.88), recursive SVN could further improve the overall performance while reducing the process delay significantly.

Table 4-4: Evaluation of recursive SVN on AURORA 4 (%)

Test Set	Clean speech training				
	A	B	C	D	Ave
Baseline	87.85	45.84	50.57	30.29	42.51
MVN	88.99	46.83	58.75	25.40	41.51
SVN(N=100)	87.79	57.11	66.27	39.24	52.29
Recursive SVN, N=30, $\lambda=0.93$	83.94	58.72	65.01	43.80	54.57
Recursive SVN, N=30, $\lambda=0.91$	82.32	57.83	63.46	44.44	54.24
Recursive SVN, N=30, $\lambda=0.88$	78.31	55.74	60.59	43.86	52.61

4.2.3 Variance Normalization with Reliable Frame Selection

In MVN, an accurate estimation of mean and variance is critical. An utterance can be divided into speech and non-speech regions with most of the statistical information in the speech region. Hence it makes sense to estimate the mean and variance based on speech frames only. Such a method depends on the reliability of the speech detection algorithm. Under a clean environment, it is not difficult to identify the speech frames. However, in the case of noisy utterances, some speech frames with low energy are severely distorted by environmental noise, so they might be mistakenly classified as noise while some noise frames might be mistakenly classified as speech. There are many methods for selecting reliable frames. Two of them, energy based frame selection [19] and a proposed spectral entropy based frame selection, are briefly discussed in this section.

In energy based frame selection, Teager energy [30] which represents instantaneous energy is employed to decide the reliability of each frame. Suppose an utterance $s(n)$ is of length N . Teager energy is defined as

$$E_{Teag}(n) = |s^2(n) - s(n-1)s(n+1)|, \quad \text{for } 1 < n < N \quad (4.2.8)$$

$$E_{Teag}(1) = |s^2(1) - s(2)s(1)| \text{ and } E_{Teag}(N) = |s^2(N) - s(N)s(N-1)| \quad (4.2.9)$$

Then the Teager energy of an utterance is smoothed by a moving average filter and the smoothed energy is sorted in ascending order. The first $Q\%$ samples of the queue are assigned with indicator 0 and the rest with 1. Next, all the indicators within each frame are averaged as an indicator for the frame. Finally, any frame whose indicator is above a threshold T is taken to be a reliable frame. This process is illustrated in Figure 4-2. The Teager energy calculation for each point involves only 2 multiplications and 1 subtraction. The entire process has low complexity and is quite easy to implement. Figure 4-3 is a plot of an utterance with reliable frame selection. The dotted line represents a smoothed instantaneous energy (Teager energy); the dashed line indicates average indicators for each frame and the solid line represents the reliable frame. By comparing with the speech waveform, it can clearly be seen, that the speech frames have been correctly selected.

Figure 4-2: Process of selecting reliable frames using Teager energy

Figure 4-3: Frame selection using Teager energy

Another method of reliable frame selection is based on the feature entropy which is used in variable frame rate analysis [20] for robust speech recognition. It is found that the entropy of Mel filter bank coefficients is high in the speech region and low in the non-speech region. Such a property could be adopted to select reliable speech frames. Suppose an n -dimensional random variable follows Gaussian probability distribution

$$p(v) = \frac{1}{(2\pi)^{n/2} |\det(K)|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T K^{-1}(x - \mu)\right) \quad (4.2.10)$$

where μ is an n -dimensional mean vector, and K is an n -by- n covariance matrix. The entropy of v is defined as

$$H(v) = -\int p(v) \ln p(v) dv \approx n \ln \sqrt{2\pi} + \ln \text{tr}(K) \quad (4.2.11)$$

where $\text{tr}()$ is a trace operation, i.e., the sum of diagonal elements in a covariance matrix.

Firstly, a speech signal is converted into Mel filter bank coefficients (23 channels) at every 2.5 ms (400 Hz). Every frame is represented by a 23-dimensional feature vector. At

each frame, the covariance matrix is estimated using the adjacent 8 frames. Then, frame entropy is sorted in ascending order. The first $Q\%$ of the queue is assigned with indicator 0 and the rest with 1. Since the feature entropy is calculated at every 2.5 ms while the MFCC feature vector is sampled at every 10 ms, the average indicator r corresponding to each MFCC vector is calculated. Those frames whose average r is above threshold T will be selected. Figure 4-4 illustrates the process for reliable frame selection using feature entropy.

Figure 4-4: Process of reliable frame selection process using feature entropy

Figure 4-5 is a plot of reliable frame selection using feature entropy. The same utterance as in Figure 4-3 is used for comparison. The upper plot of Figure 4-5 is the waveform and the feature entropy. At the bottom, the dotted line is the frame indicator and the solid line represents the reliable frame. This feature entropy method could also correctly identify speech frames. However, the higher computation load makes this method inefficient when compared with the energy based method.

Figure 4-5: Frame selection using feature entropy

Table 4-5: Evaluation of MVN with reliable frame selection on AURORA 4 (%)

Test set	Clean training mode				
	A	B	C	D	Ave
baseline	87.85	45.84	50.57	30.29	42.51
MVN	88.99	46.83	58.75	25.40	41.51
MVN with frame selection (Teager)	89.65	60.92	67.05	43.24	55.83
MVN with frame selection (Entropy)	89.76	59.72	65.55	40.99	54.25

MVN with both reliable frame selection methods was tested on AURORA 4. In both approaches, $Q = 40$ and $T = 0.1$, and in the feature entropy approach, block size for the covariance calculation is 8. All these values have been set empirically. Table 4-5 shows that variance normalization with frame selection approaches is effective in improving recognition accuracy. Both methods successfully increase the overall performance by more than 10% over MVN. However, the reliable frame selection has high computational complexity. The normalization process takes a much longer time than traditional MVN because the speech waveform corresponding to every feature file must be processed to

select the reliable frames. This makes the implementation quite troublesome and inefficient.

4.3 Log Energy Dynamic Range Normalization

In Section 4.2, it was shown that MVN is not an effective normalization approach for log energy ($\log E$) coefficient (See Tables 4-1 and 4-2). This is because that log energy is calculated in a manner different from the cepstral coefficient ($c_0 \sim c_{12}$). Figure 4-6 plots the $\log E$ coefficient trajectories of an utterance from AURORA 4 at both clean (dashed line) and noisy (approximate 10 dB SNR, solid line) conditions. This plot reveals two properties of the $\log E$ coefficient trajectory: first, the peak region of the $\log E$ trajectory is not affected too much by an adverse environment; second, the valley region is elevated due to the noise power. The mismatch between the clean and noisy $\log E$ trajectories is therefore caused by the valley region. The $\log E$ trajectory of a noisy utterance has a much lower dynamic range than that of a clean one. A log energy normalization algorithm (LEN) is proposed in [23] to minimize the mismatch of the $\log E$ sequence between clean and noisy speech. In this method, the dynamic range of the noisy $\log E$ is normalized to the target level such that the peak region is kept unchanged while the valley regions are made closer to one another.

Let the dynamic range of a log energy trajectory be

$$dynamic_range(dB) = 10 \times \frac{\max(\log E)}{\min(\log E)} \quad (4.3.1)$$

where the $\max(\log E)$ and $\min(\log E)$ are the maximum and minimum values of the $\log E$ coefficients in the entire utterance. If $\min(\log E) = \alpha \times \max(\log E)$, the above formula becomes

$$\text{dynamic_range(dB)} = \frac{10}{\alpha} \quad (4.3.2)$$

Figure 4-6: Comparison of the log energy between the clean and noisy speech (10dB).

If the dynamic range of a log energy sequence is set to be a fixed level, we can determine the value of α and hence the $\min(\log E)$. The steps of log energy dynamic range normalization are as follows:

1. Given Target_range, $\alpha = \frac{10}{\text{Target_range}}$.
2. Find $Max = \max(\log E)$ and $Min = \min(\log E)$.
3. Calculate the target minimum, $T_min = \alpha \times Max$.
4. If $Min < T_min$, then for every coefficient,

$$\log E \Rightarrow \log E + \frac{T_min - Min}{Max - Min} \times (Max - \log E) \quad (4.3.3)$$

We used set A of AURORA 2 in clean training mode to test the above algorithm. Table 4-6 shows the improvement of the recognition accuracy of set A (baseline 61.34%). The speech feature during the testing is MFCC_E_D_A. Only the logE coefficient is normalized using the above algorithm and the cepstral coefficients (c1 ~ c12) remain

unchanged. The first and second derivatives are appended after the logE normalization. Table 4-6 shows that at target dynamic range from 15 to 20dB, the increase in accuracy with respect to the baseline result is at least 10%. The highest accuracy (72.71%) is obtained at the target dynamic range of 17 dB.

Table 4-6: Evaluation of LEN with linear scale (%)

Target Dynamic range	Set A
20 dB	71.83
19 dB	72.02
18 dB	72.60
17 dB	72.71
16 dB	72.30
15 dB	71.07

The linear scale used in equation 4.3.3 may not be the best solution; a nonlinear scale has been found to be more effective on testing. In nonlinear scale normalization, equation 4.3.3 of the above algorithm is changed to

$$\log E \Rightarrow \log E + \frac{T_{\min} - \text{Min}}{\log(\text{Max}) - \log(\text{Min})} \times (\log(\text{Max}) - \log(\log E)) \quad (4.3.4)$$

Figure 4-7: Representation of scaling effect of the log scale and the linear scale

Figure 4-7 shows the difference between the linear scaling and log scaling used by equation 4.3.3 and 4.3.4. The dash line represents linear scaling whereas the dotted line represents log scaling. The plot shows that the dynamic range of log scaling is slightly larger than linear scaling. Log scale is therefore expected to be better than linear scale.

Table 4-7 shows the result of nonlinear normalization using equation 4.3.4. Use of nonlinear normalization instead of linear normalization improves the accuracy by 1%. The highest accuracy (73.43%) is obtained at the target dynamic range of 12 dB. We will use nonlinear scale normalization at 12 dB target dynamic range for the later experiments.

Table 4-7: Evaluation of LEN with nonlinear scale (%)

Target Dynamic range	Set A
17 dB	69.11
16 dB	70.02
15 dB	70.71
14 dB	71.90
13 dB	73.08
12 dB	73.43
11 dB	69.90

Next, we tried to combine the log energy dynamic range normalization (LEN) with the variance normalization. Here, both conventional MVN and segmental variance normalization (SVN) were tested. The variance normalization could apply on every cepstral coefficient trajectory including the logE while LEN is specifically designed for logE. When combining both variance normalization and LEN, two schemes as shown in Figure 4-8 were attempted. In scheme 1, after LEN all 13 feature vector coefficients are normalized while in scheme 2, after LEN, logE coefficient is not processed by the variance normalization stage. In other words, in scheme 1 logE trajectory is processed by both variance normalization and LEN while in scheme 2, logE trajectory is processed by LEN only.

Figure 4-8: Two different schemes when combining the variance normalization with log energy dynamic range normalization

The results of two combination schemes on AURORA 4 in clean training mode are given in Table 4-8. When combined with MVN and SVN, LEN significantly improves the recognition accuracy and scheme 2 is better than scheme 1. Particularly, LEN+MVN improves the performance of MVN by 13%. Recall that MVN is not appropriate for normalizing the log energy trajectory as it results in performance degradation. With LEN, the logE trajectory is effectively normalized, and both schemes 1 and 2 increase the recognition accuracy significantly, with the latter being much better than the former. It is interesting to note that the difference between scheme 1 and scheme 2 of LEN+MVN is as much as 5%.

On the other hand, when combined with SVN, LEN brings an additional 3% increase in average accuracy, but the difference between schemes 1 and 2 is negligible with the latter being slightly better than the former. In addition, by comparing the results of SVN and LEN+SVN (scheme 2), it is seen that for a large vocabulary continuous speech recognition task, LEN is more effective than SVN in normalizing the logE sequence.

Table 4-8: Evaluation of LEN with variance normalization on AURORA 4 (%)

	Clean training mode				
	Set A	Set B	Set C	Set D	Average
baseline	87.85	45.84	50.57	30.29	42.51
MVN	88.99	46.83	58.75	25.40	41.51
LEN+MVN (scheme 1)	89.47	56.35	61.44	34.32	49.64
LEN+MVN (scheme 2)	89.80	60.93	63.68	41.67	54.93
SVN	87.79	57.11	66.27	39.24	52.29
LEN+SVN (scheme 1)	87.63	60.52	67.17	42.15	55.06
LEN+SVN (scheme 2)	88.51	61.41	64.20	43.12	55.71

Figure 4-9 illustrates the differences in the normalized log energy trajectory for MVN, SVN and LEN. The original logE sequences for both clean and noisy conditions (10dB SNR) are shown in Figure 4-6. Clearly, both SVN and MVN could significantly reduce the mismatch between the clean and noisy sequences. However, in peak and valley regions the two logE sequences are still apart from each other, which is the main cause of the mismatch. With LEN, the logE trajectories of both clean and noisy utterances are much more closely matched, particularly at peak regions where the gap is much smaller, so that the mismatch is reduced by a large extent.

Figure 4-9: Comparison of MVN, SVN and LEN on logE sequence

4.4 Histogram Equalization

Histogram equalization (HEQ), or histogram matching, is a common technique used in image processing to improve the brightness and contrast of a digital picture. A histogram of a digital image shows the frequency of occurrence for each gray level over the range $[0, 2^B - 1]$, where B is the number of bits in the digital representation. Loosely speaking, the histogram of an image is an estimation of the probability density function (PDF) of the gray level. The purpose of histogram equalization is to transform the PDF of an original image to that of a reference image. The formulation of the histogram equalization is as follows.

There are two continuous variables x and y . Their PDFs are defined as $p_x(x)$ and $p_y(y)$ respectively. Now we want to transform the variable x so that it has the same PDF as $p_y(y)$. First we need to find the cumulative density function (CDF) of each variable.

$$s = C_x(x) = \int_{-\infty}^x p_x(w)dw \text{ and } C_y(y) = \int_{-\infty}^y p_y(w)dw = s \quad (4.4.1)$$

where s is a random variable and w is a dummy variable for integration. If $p_x(x)$ equals $p_y(y)$, it follows that $C_x(x) = C_y(y) = s$, then

$$y = C_y^{-1}(s) = C_y^{-1}(C_x(x)) \quad (4.4.2)$$

where C_y^{-1} is the inverse function of the CDF of y . The condition for equation (4.4.2) to be valid is that C_y is monotonically increasing, so that the inverse function C_y^{-1} exists.

HEQ could also be applied in robust speech recognition in which the histogram of each cepstral coefficient in a speech feature vector is matched to the reference [39, 40]. In equation 4.4.2, C_y is referred to as a reference CDF and C_x is the CDF of a testing

utterance. The transformation reduces the mismatch between the training and testing features by equalizing the PDF of every cepstral component.

HEQ is an extension of the mean and variance normalization. The additive noise and channel distortion shift the cepstral mean and change the cepstral dynamic range. Mathematically, the mean normalization normalizes the first moment of the cepstral PDF by subtracting the mean from each feature component. The variance normalization normalizes the dynamic range, which is the second moment of the PDF. The HEQ method normalizes all the moments of the PDF [40]. As a result, HEQ is more effective in reducing the mismatch of speech features between clean and noisy conditions.

In the case of HEQ in image processing, the PDF can be accurately estimated because of the existence of a sufficient number of pixels in a picture (usually several thousand). On the other hand, in the case of robust speech recognition, HEQ is applied on an utterance-by-utterance basis so that only hundreds of samples are available (recall that speech feature vectors are calculated every 10 ms). This makes the accurate estimation of the test PDF very difficult although the reference CDF could be calculated using all the training features. To overcome the problem of there being insufficient data in the testing utterances, an order-statistic-based CDF estimation [35, 42] is employed.

For an utterance with N frames, all the components along a trajectory are denoted as

$$S = \{x_1, x_2, \dots, x_n, \dots, x_N\} \quad (4.4.3)$$

The order statistics of these components is

$$x_{(1)} \leq x_{(2)} \leq \dots \leq x_{(r)} \leq \dots \leq x_{(N)} \quad (4.4.4)$$

where r denotes the rank of x . The direct estimation of the CDF is

$$C_x(x) = \frac{r - 0.5}{N} \quad (4.4.5)$$

Substituting equation (4.4.5) into (4.4.2), the transformed value y becomes

$$y = C_y^{-1}(C_x(x)) = C_y^{-1}\left(\frac{r-0.5}{N}\right) \quad (4.4.6)$$

So equation 4.4.6 is used to transform the cepstral coefficients in test utterances. AURARA 4 was used to test the effectiveness of HEQ as compared with MVN and SVN. The baseline feature is 39-dimensional MFCC_E_D_A (c1 ~ c12, logE with first and second derivatives appended). MVN, SVN and HEQ were applied before the derivatives were appended. Table 4-9 shows that the performance of HEQ is outstanding in comparison with the other two. The average accuracy of the HEQ method is 3% more than the SVN method. Recall that the degradation in performance of MVN is due to the improper log energy normalization.

Table 4-9: Evaluation of HEQ on AURORA 4 (%)

Test set	Clean speech training				
	A	B	C	D	Ave
Baseline	87.85	45.84	50.57	30.29	42.51
MVN	88.99	46.83	58.75	25.40	41.51
SVN	87.79	57.11	66.27	39.24	52.29
HEQ	88.07	60.87	67.55	42.06	55.23

HEQ assumes that: firstly, the phonetic or acoustic class distributions between the training and testing data are identical or similar. Secondly, the acoustic mismatch causes nonlinear transforms of the test feature distributions and the matching between the training and testing data provides the inverse transformation to reduce the mismatch. If the distributions of phonetic or acoustic classes between the training and testing data are not identical, the inverse transformation could map the feature component into a wrong class. In order to solve this problem, a class-based HEQ (CHEQ) [29, 36] method has been proposed.

In class-based HEQ, a reliable assigned class index is critical for compensating the noisy feature. Training data are clustered into a few acoustic classes using K-means algorithm in feature vector basis. An acoustic class index $\hat{i}$ is assigned to a feature vector $\hat{V}_n$ according to

$$\hat{i} = \arg \min_{1 \leq i \leq I_H} d(\hat{V}_n, z_i) \quad (4.4.7)$$

where $d(\cdot, \cdot)$ represents the Euclidean distance. z_i is the centroid of the i_{th} class from the K-means algorithm. I_H is the number of acoustic classes. $\hat{V}_n$ is the HEQ processed feature vector to remove the effect of noise during the clustering.

An estimate of the reference feature component using class-based HEQ given the test feature component y_n is

$$\hat{x}_{H,n} = C_{H,X(\hat{i})}^{-1} [\hat{C}_{H,Y(\hat{i})}(y_n)] = C_{H,X(\hat{i})}^{-1} \left(\frac{R_i(y_n) - 0.5}{N_i} \right) \quad (4.4.8)$$

where $\hat{C}_{H,Y(\hat{i})}(y_n)$, $R_i(y_n)$ and N_i denote the test CDF of the class $\hat{i}$, the rank of y_n in class $\hat{i}$ and the number of test feature components in class $\hat{i}$. $C_{H,X(\hat{i})}^{-1}$ is the inverse of the reference CDF in class $\hat{i}$. The reference CDF is obtained using training data only.

CHEQ was evaluated on AURORA 4. During the acoustic classification using K-means algorithm, 14 dimensional feature vector MFCC_0_E (c0 ~ c12, logE) was used. The first and second derivatives were appended after the equalization. Figure 4-10 demonstrate effectiveness of CHEQ. The acoustic model was trained using clean data only. The plotted data is the average recognition accuracy from all 14 test sets in AURORA 4. When the number of classes is one, CHEQ degenerates to conventional HEQ method. As the number of classes increases from 2 onwards, a significant improvement can be

observed. Generally, CHEQ increases the recognition accuracy by more than 5%. It is interesting to note that as the number of classes changes from 2 to 36, the overall performance is not affected drastically.

Figure 4-10: CHEQ evaluation on AURORA 4

4.5 PCA Based Feature Vector Compression

A feature vector for an automatic speech recognition task usually consists of cepstral coefficient $c_1 \sim c_{12}$ with either log energy (logE) or zeroth coefficient c_0 . Both c_0 and logE represent the energy of a speech frame but they are defined in different domains: c_0 is defined in the frequency domain where it is calculated by summing up all the Mel filter bank coefficients and logE is directly calculated by summing up all the samples in the time domain. The question here is: could both coefficients be used together in a speech feature vector? The answer is yes. In Advanced Front-End (ETSI ES 202 050), c_0 and logE are linearly combined [9] and in [19], a simple but effective PCA (Principal Component Analysis) based feature compression is employed. In PCA based feature compression, the 14-dimensional input speech feature (MFCC_0_E) is compressed into a 13-dimensional one by dropping the last eigen-dimension.

PCA based feature compression was tested on AURORA 4 (Table 4-10). The baseline static features are the normalized 13-dimensional MFCC_E, MFCC_0 or 14-dimensional MFCC_0_E. When combining with PCA, the SVN normalized feature vector (MFCC_0_E) is compressed into different dimensions ranging from 10 to 13. The first and second derivatives were appended after the compression. Among three different static feature vectors, MFCC_0 performs the best and MFCC_E the worst. However, when both logE and c0 are used (MFCC_0_E), the 14-dimensional static feature vector has redundant information for speech recognition. This result is much worse than MFCC_0, although it is slightly better than MFCC_E. When the dimensionality of a feature vector is reduced using PCA, there is a significant improvement in the overall recognition accuracy, which is 10% more than that obtained using the original MFCC_0_E vector. The compressed feature is also more effective than MFCC_E and MFCC_0. The most significant improvement comes from the noisy testing sets (B & D). When the dimensionality of features is further reduced to 11 or 12, the overall performance starts to drop. At a dimensionality less than 11, the performance drops much more quickly. Hence, the PCA compressed MFCC feature vector reaches its optimal performance at a dimensionality of 13.

Table 4-10: Evaluation of PCA compression on AURORA 4 (%)

Test set	A	B	C	D	Ave
SVN(MFCC_E)	87.79	57.11	66.27	39.24	52.29
SVN(MFCC_0)	88.75	65.70	71.98	50.68	61.36
SVN(MFCC_0_E)	88.03	59.78	68.78	39.86	53.90
SVN+PCA(13/14)	86.25	67.50	72.90	55.03	63.88
SVN+PCA(12/14)	85.94	66.21	74.28	54.64	63.23
SVN+PCA(11/14)	85.53	67.22	73.46	54.10	63.35
SVN+PCA(10/14)	81.48	60.44	69.00	52.21	59.02

4.6 Summary

A few feature normalization techniques for robust ASR have been discussed in this chapter. They are MVN, SVN, LEN and HEQ. It is found that class based HEQ is so far the best feature normalization technique. In addition, PCA based feature compression is also tested to be effective in enhancing speech features for recognition. To end this chapter, the best recognition performance on AURORA 4 is attempted by combining CHEQ with PCA based feature compression. The results in Table 4-11 show that the combined approach could outperform the Advanced Front-end standard (ETSI ES 202 050) on AURORA 4. This shows that proper feature normalization could outperform the feature enhancement offered by the ETSI standard.

Table 4-11: Evaluation of CHEQ+PCA compression on AURORA 4 (%)

	Clean speech training				
	A	B	C	D	Ave
baseline	87.85	45.84	50.57	30.29	42.51
Baseline+CHEQ	87.85	64.64	69.21	50.38	60.51
Baseline+CHEQ+PCA(13/14)	86.41	72.20	75.36	60.76	68.54
Advanced Front-end	89.87	74.16	71.45	57.99	68.16

However, the drawback with the HEQ method and PCA based feature compression is that they are not suitable for real time processing, as they require a large amount of clean data to calculate some parameters before processing each utterance. Unlike the HEQ and PCA related methods, MVN, SVN and LEN are all sentence based techniques, and no prior information is required. Hence the processing is fast and efficient but at the expense of recognition accuracy.

Chapter 5 Cepstral Domain Filtering for Robust Speech Recognition

5.1 Introduction

Some common feature normalization techniques for robust speech recognition were discussed in Chapter 4. The focus in this chapter will be on the cepstral domain filtering techniques for robust speech recognition. A cepstral domain filter is designed in the modulation frequency domain [37, 38], and is achieved by means of spectral analysis of the temporal trajectories of the spectral envelopes of speech. A cepstral filter removes the high frequency noise components and emphasizes the low frequency components which contain the useful speech information. A widely used temporal filter in the modulation frequency domain is the representation of relative spectra (RASTA) [8]. RASTA is a bandpass filter with a passband of 0.26 to 14.3 Hz, and operating on the log filter bank coefficient domain. It takes advantage of the fact that the rate of change of some of the non-linguistic parts of the message lie outside the range of the rate of change of the speech information. The spectral components that vary more slowly or faster are suppressed, thereby improving the recognition accuracy.

Later, many feature domain filtering techniques are proved to be more effective than RASTA processing. In [6, 15 and 16], a very simple IIR (Infinite Impulse Response) filter, namely ARMA (Autoregressive Moving Average) filter, is proved to be highly effective in removing the high frequency components, although no prior information is required to design the filter. In [64], a reference PSD (Power Spectral Density) function is calculated from clean training features. A square-root Wiener filter is designed for each

utterance under test by normalizing the feature PSD to the reference. This utterance dependent technique is also proved to be effective.

In [7, 14, 38 and 59], a few different feature domain filters are designed based on linear discriminant analysis (LDA), principal component analysis (PCA) and minimum classification error (MCE) to improve the robustness of speech features. They are all proved to be superior when combined with cepstral mean and variance normalization. The basic objective of these filters is to project the multi-dimensional feature into the highly discriminative feature space.

Some cepstral domain filters described in the literature are considered in this chapter and their performances compared. In addition, two proposed temporal filters are analyzed and test results are presented. The performances of all of the filters are compared on AURORA 2, AURORA 4 and CENSREC-3 databases.

This chapter is organized as described next. Section 5.2 deals with the revision of the ARMA filter which is a simple IIR filter. Section 5.3 is a description of multi-eigenvector temporal filter design based on Principal Component Analysis (PCA). Section 5.4 forms a study of the Linear Discriminant Analysis (LDA) based filter. Sections 5.5 and 5.6 introduce two proposed filters: the entropy weighted multi-eigenvector temporal filter and the β -order multi-eigenvector temporal filter. Section 5.6 is a summary of the entire chapter, comparing all of the previously discussed temporal filters.

5.2 ARMA Filtering

Figure 5-1: MVA post processing

ARMA is a filter designed for cepstral domain processing. This technique is simple yet effective. The computation load for ARMA filtering is quite negligible. Figure 5-1 shows the MVA (Mean subtraction, Variance normalization and ARMA filtering) post processing procedure [6, 15, and 16]. The MVN process firstly brings feature sequence at different noise levels to the same level and scale, followed by the ARMA filtering, defined as

$$\tilde{c}(t, k) = \begin{cases} \frac{\sum_{i=1}^M \tilde{c}(t-i, k) + \sum_{j=0}^M \hat{c}(t+j, k)}{2M+1} & \text{if } M < t \leq T-M, \\ \hat{c}(t, k) & \text{otherwise} \end{cases} \quad (5.2.1)$$

where M is the order of the filter, $\hat{c}$ is the cepstral coefficient after MVN and $\tilde{c}$ is the cepstral coefficient after the filtering. The ARMA filter is essentially a low pass filter, smoothing out spikes in the cepstral coefficient trajectory. Since these spikes are likely to be due to noise, removing these peaks would be expected intuitively to improve the recognition accuracy. However if a spike is from the original clean speech, it carries important information which is of interest to ASR and removing these spikes would then unfortunately distort the speech information. Hence, there is a trade-off between the filter order and the speech recognition accuracy. Figure 5-2 shows the frequency response of ARMA filters at orders 2 and 3. If the order is low, the filter will have a larger bandwidth and some noise components will still exist in the filtered cepstral trajectory. On the other

hand, if the order is high, the filter will have a narrower bandwidth and some speech information will be removed.

Figure 5-2: Frequency response of ARMA filter with different orders

Figure 5-3: Comparison of c1 sequence before and after the MVA processing, where the x-axis is the number of frames, y-axis is the magnitude of c1 sequence

Figure 5-3 is a comparison of the trajectories of the cepstral coefficient c_1 at different noise levels, with and without MVA processing. The 2nd order ARMA filter is used for this plot. Clearly, the cepstral trajectories after processing have very clear and smooth contours. This results in a very good match between the noisy and clean feature sequences.

The implementation of ARMA processing is simple because the filter is not data dependant and no prior information is required. In addition, the order of the ARMA filter is low; only a few coefficients are required to define the filter so that the processing speed is fast. Tables 5-1 and 5-2 show the evaluation results of ARMA filtering on CENSREC-3 and AURORA 4 databases. The baseline feature is MFCC_E ($c_1 \sim c_{12}$, plus log energy) with the first and second derivatives appended. The variance normalization and the ARMA processing were applied before the derivatives were appended. Both tables show that the improvement due to the ARMA filtering is significant. In CENSREC-3, this is better illustrated under highly mismatched conditions (conditions 5 and 6) in which ARMA increases the accuracy by 15%. In well matched conditions (conditions 1, 2 and 3), the improvement due to ARMA filtering is trivial. This proves that ARMA is more effective under mismatched conditions. With CENSREC-3, the 2nd order ARMA filter produces slightly better results than that 4th order. However, with AURORA 4 the difference between 2nd and 4th order ARMA filtering is seen to be quite significant. The 4th order filter does not have enough bandwidth to preserve all the useful speech information, resulting in a severe degradation of system performance. In conclusion, a 2nd order ARMA filter is shown to be quite effective in removing the noise components.

Table 5-1: Evaluation of ARMA on CENSREC-3 (%)

Condition	1	2	3	4	5	6	Ave
baseline	86.06	99.10	79.28	56.53	40.70	30.77	65.41
MVN	93.48	99.04	91.29	81.32	57.67	57.05	79.98
MVN+ARMA ($M=4$)	94.05	98.23	91.26	85.23	72.60	71.40	85.46
MVN+ARMA ($M=2$)	94.41	99.40	91.35	84.60	69.82	75.22	85.80

Table 5-2: Evaluation of ARMA on AURORA 4 (%)

Test set	Clean speech training				
	A	B	C	D	Ave
SVN	87.79	57.11	66.27	39.24	52.29
SVN+ARMA($M=4$)	83.76	38.24	36.34	23.81	35.17
SVN+ARMA($M=2$)	86.45	60.06	67.97	43.21	55.29

5.3 Multi-Eigenvector Temporal Filtering

Figure 5-4: The representation of the temporal filter analysis

The multi-eigenvector temporal filter [7] is based on Principal Component Analysis (PCA). This method uses eigenvectors to design the filter coefficients. Firstly, a series of vectors (length L) are extracted along the k_{th} time trajectory in the cepstral domain as shown in Figure 5-4. For $t = 1, 2, \dots, T-L+1$, the vector is defined as $z_k(t) = [c(t, k) \ c(t+1, k) \ c(t+2, k) \ \dots \ c(t+L-1, k)]^T$. The mean and covariance matrix are calculated for all the vectors z along each trajectory:

$$\mu_k = \frac{1}{T-L+1} \sum_{t=1}^{T-L+1} z_k(t) \quad (5.3.1)$$

$$\Sigma_k = \frac{1}{T-L+1} \sum_{t=1}^{T-L+1} (z_k(t) - \mu_k)(z_k(t) - \mu_k)^T \quad (5.3.2)$$

Next, apply PCA to the covariance matrix of each trajectory. Suppose vector $\phi_{i,k}$, ($i = 1, 2, \dots, L$) are L normalized eigenvectors of the covariance matrix Σ_k , with corresponding eigenvalues $\lambda_{i,k}$ in descending order, i.e., $\lambda_{1,k}$ is the maximum and $\lambda_{L,k}$ is the minimum. The multi-eigenvector temporal filter (mPCA) is given by combining the first M eigenvectors weighted by the corresponding eigenvalues

$$w_k = \frac{\bar{w}_k}{|\bar{w}_k|} = \frac{\bar{w}_k}{\sqrt{\sum_{i=1}^M \lambda_{i,k}^2}} \quad (5.3.3)$$

where $\bar{w}_k = \sum_{i=1}^M \lambda_{i,k} \phi_{i,k}$ and w_k is a normalized filter for trajectory k . If M is taken to be 1, the mPCA filter degenerates into the first eigenvector of the covariance matrix, i.e., the PCA filter. Note that the first principal component has the maximum variance. Hence the PCA filter will project the multivariate variables, in this case z , into the direction where it is maximally distributed. If M is not equal to 1, the vector z is projected into the direction corresponding to the linear combination of eigenvectors with the M largest variances. In other words, the projection will be a linear combination of the first M principal components.

$$v_k = w_k^T z_k = \frac{\bar{w}_k^T z_k}{\sqrt{\sum_{i=1}^M \lambda_{i,k}^2}} = \frac{1}{\sqrt{\sum_{i=1}^M \lambda_{i,k}^2}} \sum_{i=1}^M \lambda_{i,k} \phi_{i,k} z_k = \frac{1}{\sqrt{\sum_{i=1}^M \lambda_{i,k}^2}} \sum_{i=1}^M \lambda_{i,k} y_{i,k} \quad (5.3.4)$$

where $y_{i,k}$ is the projection of z_k on the i_{th} eigen-dimension. It is known that most of the speech information is contained between 0 and 16 Hz in the modulation frequency

domain with the part of 2 – 4 Hz spectrum having the most important information [8]. In Figure 5-5, temporal filters for the 13 trajectories with M equal to 1 and 3 are plotted in the modulation frequency domain. Each plot contains 13 filters for each speech vector component. It is observed that the 13 temporal filters are quite similar. The narrow bandwidth of the PCA filter suppresses some speech information which is of importance to ASR. When the second and third principal components which carry useful speech information are used, the mPCA filter demonstrates its superiority. Obviously, the multi-eigenvector temporal filter in the bottom plot has a wider bandwidth; hence it preserves more speech information. Both filters are low pass filters; therefore they will not attenuate the slowly varying components which are undesirable to ASR. As the human auditory system is insensitive to slowly varying stimuli, these low frequency components should be suppressed. To achieve this, cepstral mean normalization is performed. After mean normalization, cepstral variance normalization is also performed before temporal filtering.

Both PCA and mPCA were tested on CENSREC-3 and AURORA 4 databases. The baseline feature was MFCC_E_D_A (c1~c12, log energy, plus the first and second derivatives). Variance normalization and cepstral temporal filtering were applied before appending the derivatives. Through experiments using AURORA 2, the filter length is optimized to be 15 and M is set to be 3 [7].

Figure 5-5: Comparison of PCA derived temporal filter and multi-eigenvector filter

In CENSREC-3 and AURORA 4 evaluation, we used SVN to normalize the variance. Since in CENSREC-3, there were no clean data collected under the in-car environment, we used the data set with the lowest noise level, i.e., the 3608 utterance recorded by close talk (CT) microphone at idling (quiet) condition, to calculate the filter coefficients. In Table 5-3, it is seen that both the PCA and mPCA filters could increase recognition accuracy under highly-mismatched conditions (condition 5 and 6) by more than 10%. In moderately-mismatched testing conditions, the PCA filter degrades the performance and mPCA only increases the accuracy marginally. In well-matched conditions (condition 1, 2 and 3), both PCA and mPCA filters degrade the performance with the latter being slightly better than the former. This is because under such a condition, the good match between the training and testing data are disturbed by the filtering process. In terms of the overall average performance, PCA does not increase the recognition accuracy at all while mPCA successfully improves the accuracy by 4%. This again proves that the first principal component alone is insufficient to represent the speech information for ASR.

Table 5-3: Evaluation of PCA and mPCA on CENSREC-3 (%)

Condition	1	2	3	4	5	6	Ave
baseline	86.06	99.10	79.28	56.53	40.70	30.77	65.41
SVN	94.85	99.60	91.73	85.55	63.98	60.21	82.60
SVN+PCA	85.19	92.03	86.51	79.98	71.48	70.37	80.93
SVN+mPCA($M=3$)	91.66	97.56	90.55	85.91	75.27	78.66	86.60

The result obtained with AURORA 4 (Table 5-4) is quite similar to that with CENSREC-3. The performance of the PCA filter is very poor. The overall accuracy is even lower than the baseline result. Much useful speech information has been removed in this case, whereas it is observed that mPCA increases the overall accuracy by 2%. Particularly under noisy conditions (sets B & D), mPCA is observed to successfully improve the recognition performance. The result on set A again degrades with respect to the baseline because for the clean features, the filtering process will inherently remove some useful speech information.

Table 5-4: Evaluation of PCA and mPCA on AURORA 4 (%)

Test set	Clean speech training				
	A	B	C	D	Ave
Baseline	87.85	45.84	50.57	30.29	42.51
SVN	87.79	57.11	66.27	39.24	52.29
SVN+PCA	68.95	42.58	52.19	30.92	40.15
SVN+mPCA($M=3$)	83.02	58.43	66.56	42.78	54.06

A series of experiments shows the superiority of mPCA over the PCA filter. The performance of the mPCA filter is consistent across AURORA 4 and CENSREC-3 databases. However, the problem with mPCA is that under clean conditions, the recognition performance degrades. This is a common observation for many feature enhancement methods for robust ASR. A good cepstral domain filtering process is expected to improve the performance under mismatched conditions while maintaining a good performance under well matched and clean conditions. Hence, the current mPCA

filter needs to be modified to improve its performance under well matched or clean conditions. Multi-eigenvector filter design will be revisited in Section 5.4.

5.4 Linear Discriminant Analysis for Cepstral Filtering

Linear Discriminant Analysis (LDA) [14, 59] could also be applied to design a cepstral domain filter to improve the robustness of the feature vector. In LDA, the first step is to perform the feature vector alignment with the pre-trained HMM model set. This will classify the input vectors into different acoustic classes, each of which is represented by an HMM. In each class, a column vector $z_k(t)$ is formed in the same way as shown in Figure 5-4.

$$z_k(t) = [c(t, k) \quad c(t+1, k) \quad c(t+2, k) \quad \dots \quad c(t+L-1, k)]^T, t = 1, 2, \dots, T-L+1. \quad (5.4.1)$$

where L is the vector length. For all the vectors within each class, the mean and covariance matrices are calculated as

$$\mu_k^j = \frac{1}{N_j} \sum_{n=1}^{N_j} z_k^j(n) \text{ and } \Sigma_k^j = \sum_{n=1}^{N_j} (z_k^j(n) - \mu_k^j)(z_k^j(n) - \mu_k^j)^T \quad (5.4.2)$$

where N_j is the number of input vectors belonging to class j . z_k^j is the vector in class j .

μ_k^j and Σ_k^j are class mean and covariance. Then the between-class matrix and within-class matrix are defined as

$$S_{B,k} = \sum_{j=1}^J N_j (\mu_k^j - \mu_k)(\mu_k^j - \mu_k)^T \quad (5.4.3)$$

$$S_{W,k} = \sum_{j=1}^J N_j \Sigma_k^j \quad (5.4.4)$$

where $\mu_k = \frac{1}{\sum_{j=1}^J N_j} \sum_{j=1}^J N_j \mu_k^j$. The total number of acoustic classes is J . The LDA filter

is defined using Fisher's Linear Discriminant Analysis

$$w_{k,LDA} = \arg \max_w \frac{w^T S_{B,k} w}{w^T S_{W,k} w} \quad (5.4.5)$$

Equation 5.4.5 is a maximization problem. It could be transformed into a constrained optimization problem, where $-\frac{1}{2} w^T S_{B,k} w$ is minimized with respect to w , subject to the condition $w^T S_{W,k} w = 1$. The corresponding Lagrangian [68] is

$$L_p = -\frac{1}{2} w^T S_{B,k} w + \frac{1}{2} \lambda (w^T S_{W,k} w - 1) \quad (5.4.6)$$

By applying the KKT (Karush Kuhn Tucker) condition, at the optimum point, $\nabla L_p = 0$, i.e.,

$$-S_{B,k} w + \lambda S_{W,k} w = 0 \Rightarrow S_{W,k}^{-1} S_{B,k} w = \lambda w \quad (5.4.7)$$

This is a standard eigenvector problem where w and λ are the corresponding eigenvector and eigenvalue of the square matrix $S_{W,k}^{-1} S_{B,k}$. If we substitute 5.4.7 into equation 5.4.5, we could get the solution of this optimization problem, which is the eigenvector of $S_{W,k}^{-1} S_{B,k}$ corresponding to the largest eigenvalue.

$$w_{k,LDA} = \arg \max_w \frac{w^T S_{B,k} w}{w^T S_{W,k} w} = \arg \max_w \lambda \frac{w^T S_{W,k} w}{w^T S_{W,k} w} = \arg \max_w \lambda \quad (5.4.8)$$

In LDA cepstral filter design, the training features are first labeled with different classes by forced alignment with the pre-trained model. In AURORA 2, the number of classes is 12: digit 0-9, plus "oh" and "sil" models. Then the data in each class are converted into

the column vectors as defined in equation 5.4.1. L is the vector length of z . The mean and covariance for each class are derived using equation 5.4.2. And the within-class matrix and between-class matrix are calculated using equations 5.4.3 and 5.4.4. The LDA filter coefficient is then the eigenvector of the matrix $S_{W,k}^{-1}S_{B,k}$ corresponding to the largest eigenvalue. The same process is repeated for each trajectory of the MFCC feature vector. Figure 5-6 is a plot of the LDA and mPCA filter responses in the modulation frequency domain for each static feature coefficient (MFCC_E). The frequency axis is in log scale. The two filters have quite different characteristics. The mPCA filters of the 13 trajectories are quite consistent and have small side lobes, while the LDA filters of the 13 trajectories have different passband gains and relatively larger side lobes. The bandwidth of the LDA filter is also narrower than that of the mPCA filter. So the performance of the LDA filter is expected to be worse than that of the mPCA filter.

Table 5-5 shows the evaluation results of LDA cepstral filtering on AURORA 2. The MFCC feature vectors were normalized before the LDA filtering in each trajectory. The result shown here is at the optimal filter length. It can be observed that the LDA filter improves the recognition accuracy significantly. However, under clean training mode, the LDA filter is not as effective as the mPCA filter. This is consistent with the analysis in Figure 5-6. In addition, the LDA filter design has one additional step of pre-alignment which has a high computation load. As a result, mPCA has an advantage over LDA in terms of computation complexity.

Figure 5-6: Comparison of filter response of LDA and mPCA cepstral filters

Table 5-5: Evaluation of LDA filter on AURORA 2 (%)

	Clean speech training			
	Set A	Set B	Set C	Ave
MVN	70.18	70.77	66.37	69.65
MVN+LDA	77.46	78.03	76.54	77.50
MVN+mPCA($M=3$)	82.10	83.03	80.93	82.24

The filter length could affect the performance of feature vectors. Table 5-6 shows the recognition accuracies at different LDA filter lengths. The testing data are from set A of AURORA 2. The highest performance gain is achieved at the filter length of 17.

The drawback with LDA cepstral filtering is the high computation load. In the LDA filter design, the complexity increases drastically with the number of classes. The experimental results shown in this section are based on AURORA 2, which has only 11 classes. If the same experiment is carried out on AURORA 4 or CENREC-3 databases, in which the number of classes is several thousands, the processing time, computation load as well as the required memory space would be prohibitively high. This would make the LDA

cepstral filtering difficult to be implemented in large vocabulary speech recognition tasks.

For this reason, the results of CENREC-3 and AURORA 4 are not shown here.

Table 5-6: Filter length of LDA filter in AURORA 2 Evaluation (%)

L	Set A
11	76.05
13	76.84
15	77.38
17	77.46
19	76.53

5.5 Entropy Based Multi-Eigenvector Temporal Filtering

In PCA based filter design, it is believed that the first few principal components carry the most useful information that is critical to ASR, whereas the distributions in other eigen-dimensions are more likely to be associated with noise. The PCA filter in Section 5.3 projects the multivariate variable into the direction where it has the maximum variance. Here a new PCA based temporal filter is proposed based on an entropy concept which has previously been used in speech processing. In [56, 65], the entropy contribution from each sub-band is calculated as a feature for robust ASR, because the formant positions of the spectrum could be captured by spectral entropy. In [66], two different entropy expressions, namely $-\text{SlogS}$ and logS where S is the power spectrum, have been discussed for speech spectral estimation. In our proposed PCA based filter design, we also employ an entropy concept, and instead of choosing eigenvectors based on the variance, we choose eigenvectors based on the entropy contribution. The entropy contribution is given by

$$H_{i,k} = -\hat{\lambda}_{i,k} \times \log(\hat{\lambda}_{i,k}) \quad (5.5.1)$$

where $\hat{\lambda}_{i,k} = \frac{\lambda_{i,k}}{\sum_{i=1}^M \lambda_{i,k}}$ for $i = 1, 2, \dots, M$. M is taken to be 15, i.e., dimension of the

covariance matrix. Then the PCA filter is modified as an ePCA filter. It is observed that in some time trajectories, the ePCA filter will select the eigenvector that corresponds to the second largest eigenvalue. Figure 5-7 compares both ePCA and PCA filters. The filter on the top is based on the largest entropy contribution. Because for certain time trajectories, instead of a low pass filter, the band pass filter has been applied, the region between 3 and 8 Hz with useful speech information is emphasized, and the ePCA filter is expected to be more effective than the PCA filter.

In mPCA filter design described in Section 5.3, if M is not equal to 1, the data are projected into the direction corresponding to the linear combination of eigenvectors with the M largest variances. We also extended the entropy concept into mPCA filter design. In the new entropy based multi-eigenvector temporal filter (emPCA), we use the entropy based weighting to design the filter. Supposing, $\lambda_{i,k}$ ($i = 1, \dots, M$) are the first M largest eigenvalues and assuming that the distributions in the higher eigen-dimensions are mainly due to noise, the contribution to the overall entropy from an individual eigen-dimension is then calculated as

$$H_{i,k} = -\hat{\lambda}_{i,k} \times \log(\hat{\lambda}_{i,k}) \quad (5.5.2)$$

where $\hat{\lambda}_{i,k} = \frac{\lambda_{i,k}}{\sum_{i=1}^M \lambda_{i,k}}$. Figure 5-8 is a plot of the entropy contribution. The x-axis

represents the amount of variance (percentage) and the y-axis represents the entropy contribution. Note that the entropy contribution is high when the amount of variance percentage is at a moderate level. A too high or too low a variance makes a low contribution to the entropy.

Figure 5-7: Frequency response of ePCA and PCA filter

Figure 5-8: Plot of the entropy contribution

The entropy concept is used to define how much information is contained in a signal. In the principal component analysis, the eigenvalue represents the distribution of the signal

in the dimension defined by the corresponding eigenvector. The direction having the most speech information is the one with the highest entropy contribution. The y-axis of Figure 5-8 could be interpreted as the amount of information given the variance. If the entropy contribution is high, the information from the corresponding direction is more. So the modified multi-eigenvector filter (emPCA) becomes

$$w_k = \frac{\bar{w}_k}{|\bar{w}|} = \frac{\bar{w}_k}{\sqrt{\sum_{i=1}^M H_{i,k}^2}} \quad (5.5.3)$$

where $\bar{w}_k = \sum_{i=1}^M H_{i,k} \varphi_{i,k}$.

Figure 5-9: Comparison between mPCA and emPCA, $M=3$

Figure 5-9 shows both mPCA and emPCA at $M=3$ for all the 13 trajectories. It may be observed that emPCA has a noticeably wider bandwidth; it enhances the signal in the frequency range of 0 – 10 Hz and therefore the useful speech information is well retained. Intuitively, emPCA is expected to perform better than mPCA. This may be explained as follows: looking at Figure 5-10 and Figure 5-11, which plot the filter banks defined by 15 eigenvectors, it is obvious that only the first 3 eigenvectors are useful since only they

cover the speech region in the modulation spectrum. In Figure 5-10, the first low pass filter has a much higher passband gain than the rest of the band pass filters. Figure 5-12 (on the left) is a typical example of the first 3 variances (eigenvalue) in PCA. Note that the first variance is larger than the second and the third; in addition, the first 3 variances are much larger than the rest. In mPCA design, the individual eigenvector is weighted with the corresponding eigenvalue. It will inherently emphasize the low pass filter, i.e., the first eigenvector. In emPCA, after the transformation, the difference between the entropy contributions of each eigenvector is much smaller. A typical example is given in the Figure 5-12 (on the right). It is apparent that the second and third bandpass filters are relatively enhanced. Hence more speech information is retained in the region of 5 to 10 Hz and the recognition accuracy is expected to increase.

Figure 5-10: Filter bank defined by 15 eigenvectors

Figure 5-11: Filters defined by first 4 eigenvectors

Figure 5-12: Illustration of the filter weighting

First, PCA and ePCA were compared using the CENSREC-3 database. The baseline feature is MFCC_E (c0 ~ c12, plus log energy) with the first and second derivatives. L is set to be 15. The training and testing features were filtered before appending derivatives. In Table 5-7, both of PCA and ePCA filters degrade the performance with respect to the baseline due to the narrow bandwidth. It may be observed that the overall performance of the ePCA filter is better than that of the PCA filter. This verifies that the previous

analysis is correct. The direction with the highest entropy contribution has more speech information than that with the largest variance.

Table 5-7: Evaluation of PCA and ePCA on CENSREC-3 (%)

Condition	1	2	3	4	5	6	Ave
SVN	94.85	99.60	91.73	85.55	63.98	60.21	82.60
SVN+PCA	85.19	92.03	86.51	79.98	71.48	70.37	80.93
SVN+ePCA	86.36	96.24	86.21	80.43	72.92	71.67	82.31
SVN+mPCA	91.66	97.56	90.55	85.91	75.27	78.66	86.60
SVN+emPCA	94.48	99.40	93.11	87.39	84.12	78.72	89.54

Next, we tested the emPCA filter on CENSREC-3. In Table 5-7, it may be observed that under all the six testing conditions, emPCA performs better than mPCA. Under well-matched conditions (conditions 1 & 2), degradation of mPCA with respect to the SVN process is quite considerable while the emPCA results in quite negligible degradation. Under mismatched conditions (conditions 3, 4, 5 and 6), emPCA successfully surpasses mPCA with the highest improvement of 11%. The overall result of emPCA is 3% more than mPCA.

The same experiment was carried out on AURORA 4 to test the effectiveness of emPCA for a large vocabulary continuous speech recognition task. In Table 5-8, different numbers of eigenvectors are tested using the development set. When M is 1, the filter degenerates to the eigenvector with the largest entropy contribution, i.e., ePCA filter. The sharp improvement is observed when M increases from 1 to 3. When M continues to increase, the performance saturates. So for all the results of emPCA filtering obtained in this thesis, M is set to be 3. In AURORA 4 evaluation (Table 5-9), emPCA outperforms mPCA in all 4 testing sets. Again, like with the well-matched conditions in CENSREC-3, the filtered feature vectors in set A lose some useful speech information, resulting in a performance degradation, but the emPCA filtered feature still maintains a higher

accuracy than the mPCA filtered feature. In noisy testing sets (B, C & D), the emPCA filter successfully outperforms the mPCA filter. The average overall improvement of emPCA with respect to the mPCA is 2%.

Table 5-8: Choice of M using AURORA 4 development data set

M	1	2	3	4	5
Ave	70.55	79.42	85.75	84.91	83.47

Table 5-9: Evaluation of mPCA and emPCA on AURORA 4 (%)

Test set	Clean speech training				
	A	B	C	D	Ave
SVN	87.79	57.11	66.27	39.24	52.29
SVN+mPCA	83.02	58.43	66.56	42.78	54.06
SVN+emPCA	86.52	60.78	67.85	44.64	56.21

5.6 β -order Multi-Eigenvector Temporal Filtering

From Figures 5-10 and 5-11, it is observed that each eigenvector covers a different region in the modulation frequency domain. The importance of the spectrum in each region is quite different from that of any other. To generalize the entropy weighted multi-eigenvector filter, the eigenvalues are raised to the power of β which can be made to be equal to or less than 1. A proper β order could increase the importance of a small eigenvalue and decrease the importance of a high eigenvalue. The experimental results show that the optimum value of β is not equal to 1 for the case of mPCA. The coefficients of this type of filter (β mPCA) are given by

$$w_k = \frac{\bar{w}_k}{|\bar{w}_k|} = \frac{\bar{w}_k}{\sqrt{\sum_{i=1}^M \lambda_{i,k}^{\beta^2}}} \quad (5.6.1)$$

where $\bar{w}_k = \sum_{i=1}^M \lambda_{i,k}^{\beta} \varphi_{i,k}$.

We used the development set of AURORA 4 (Section 3.3) to find the optimum β order. Different combinations of β order and number of eigenvectors were tested (Table 5-10).

The optimum performance is given by 3 eigenvectors ordered by 0.1. It is observed that when the β value is above 0.3, the performance starts to drop. When 2 eigenvectors are used, the performance is not as good as when 3 or 4 are used.

Table 5-10: Selection of β order in AURORA 4 development set (%)

	β	0.05	0.1	0.3	0.5	0.8	1.0	1.5
M	2	79.98	79.80	79.34	78.73	78.58	77.71	75.89
	3	84.76	84.84	84.69	84.12	83.02	81.20	77.82
	4	83.17	83.40	83.81	83.78	83.13	81.69	78.39

In Figure 5-13, all mPCA, emPCA and β mPCA for all 13 trajectories are compared. It is observed that the emPCA and β mPCA have quite similar characteristics in the frequency domain. Both of them have larger bandwidths than the mPCA filter. The performance of β mPCA is therefore expected to be similar to emPCA.

Figure 5-13: Comparison of β mPCA, emPCA and mPCA.

β mPCA was tested on the CENSREC-3 and AURORA 4 databases. On AURORA 4 (Table 5-11), β mPCA with optimum order is better than emPCA. The recognition results in CENSREC-3 are given in Table 5-12. Unlike the case in AURORA 4, where β mPCA

surpasses emPCA at the optimum setting, emPCA and β mPCA perform equally well in this case.

Table 5-11: Evaluation of emPCA and β mPCA in AURORA 4 (%)

Test set	A	B	C	D	Ave
Baseline	87.85	45.84	50.57	30.29	42.51
SVN+emPCA	86.52	60.78	67.85	44.64	56.21
SVN+ β mPCA ($\beta = 0.1$)	86.81	61.36	69.32	46.22	57.26

Table 5-12: Evaluation of emPCA and β mPCA in CENSREC-3 (%)

Condition	1	2	3	4	5	6	Ave
baseline	86.06	99.10	79.28	56.53	40.70	30.77	65.41
SVN+emPCA	94.48	99.40	93.11	87.39	84.12	78.72	89.54
SVN+ β mPCA ($\beta = 0.1$)	94.91	99.59	92.78	87.11	82.91	78.61	89.32

5.7 Comparative Evaluation of Hybrid Schemes

In this chapter, a few types of cepstral domain filters for robust speech recognition have been discussed and tested. They are ARMA filter, LDA filter, mPCA filter, emPCA filter and β mPCA filter. Their performances are now assessed for the most difficult task – AURORA 4. Note that the LDA filter is not shown here due to the high computation load. It is also believed that on AURORA 4, the LDA filter will not perform as well as the mPCA filter (section 5.3). All the cepstral filtering processes were preceded by the SVN process. For completeness, RASTA filtering result is also presented here. In Table 5-13, the RASTA filter is not as good as the other 4 filters. The ARMA filter is clearly better than the mPCA filter and both of them are not as good as the proposed emPCA and β mPCA filters. Figure 5-14 is a comparison of these 5 cepstral filters in the modulation frequency domain. As explained in previous sections, emPCA and β mPCA have much wider bandwidths than mPCA filter, and hence they preserve more speech information. On the other hand, the RASTA filter has a much wider bandwidth than the other 4 filters.

As a result, the mild smoothing will retain some noise components, thus reducing the recognition performance.

Table 5-13: Comparison of all cepstral filters with SVN on AURORA 4 (%)

Test set	A	B	C	D	Ave
SVN	87.79	57.11	66.27	39.24	52.29
SVN+RASTA	85.60	57.82	66.37	41.48	53.41
SVN+ARMA($M=2$)	86.45	60.06	67.97	43.21	55.29
SVN+mPCA	83.02	58.43	66.56	42.78	54.06
SVN+emPCA	86.52	60.78	67.85	44.64	56.21
SVN+ β mPCA	86.81	61.36	69.32	46.22	57.26

Figure 5-14: Comparison of 5 different cepstral filters

Table 5-14: Comparison of all cepstral filters with HEQ on AURORA 4 (%)

Test set	A	B	C	D	Ave
HEQ	88.07	60.87	67.55	42.06	55.23
HEQ+RASTA	85.12	62.11	66.81	44.97	56.74
HEQ+ARMA($M=2$)	87.33	57.63	66.54	40.08	52.86
HEQ+mPCA	80.92	51.81	66.60	36.83	48.52
HEQ+emPCA	87.37	58.09	66.80	41.26	53.59
HEQ+ β mPCA	86.59	56.47	66.78	40.75	52.62

It was shown experimentally that HEQ is better than SVN in Chapter 4. Intuitively, if SVN is replaced with HEQ in Table 5-13, the results are expected to be better. However, this is not so as shown in Table 5-14. Note that the HEQ process has already improved

the recognition performance significantly over that from SVN. Under such a circumstance, further cepstral filtering is too aggressive if the filters have low bandwidths. The results shown in Table 5-14 are therefore opposite to those given in Table 5-13. After the HEQ process, the mild filtering from RASTA can further improve the performance whereas the aggressive filtering from the other 4 filters degrades the performance. There is therefore a tradeoff between feature normalization and cepstral filtering. Either strong feature normalization may be combined with mild cepstral filtering, or weak feature normalization may be combined with aggressive cepstral filtering.

The PCA based feature vector compression was introduced in Chapter 4. Here, PCA based feature compression (denoted as PCA(13/14)) is combined with the mPCA, emPCA and β mPCA to form a two-stage PCA processing. Table 5-15 shows the evaluation results on CENSREC-3. Surprisingly all mPCA, emPCA and β mPCA filters outperform the Advanced Front-end (ETSI ES 202 050). This shows that for isolated word recognition, feature enhancement could be more effective than speech enhancement offered by ETSI. In terms of the overall performance, emPCA is superior to mPCA and β mPCA.

Table 5-15: Evaluation of two-stage PCA processing on CENSREC-3 (%)

Condition	1	2	3	4	5	6	Ave
Baseline	86.06	99.10	79.28	56.53	40.70	30.77	65.41
SVN+PCA(13/14)	95.86	99.69	93.08	88.29	85.34	76.22	89.75
SVN+PCA(13/14)+mPCA	94.56	99.16	92.53	89.06	86.74	83.51	90.93
SVN+PCA(13/14)+emPCA	96.25	99.81	93.79	87.74	89.25	82.73	91.60
SVN+PCA(13/14)+ β mPCA	96.06	99.73	93.54	86.76	88.30	80.06	90.74
Advanced Front-end	95.8	99.78	91.93	85.92	86.89	79.3	89.94

The two-stage PCA process was also tested on large vocabulary continuous speech recognition task. Table 5-16 shows that in AURORA 4, both β mPCA and emPCA perform equally well and both of them are much better than mPCA. mPCA actually

degrades the performance with respect to the PCA compressed feature vector, whereas emPCA and β mPCA successfully improves the overall recognition accuracy. They both outperform mPCA by 4%. This proves again that the proposed filters could preserve more speech information than mPCA. The ETSI Advanced Front-end is still the best choice for feature enhancement when compared with the emPCA and β mPCA approaches. It outperforms the PCA(13/14)+emPCA(β mPCA) by 4%. Even when using the log energy trajectory specially processed by the algorithm discussed in Section 4.2, the average overall accuracy increases by only 0.5%. It is believed that for large vocabulary continuous speech recognition tasks under noisy environments, time domain filtering (i.e., two-stage Wiener filtering in Advanced Front-end) is still more effective than cepstral domain filtering.

Table 5-16: Evaluation of two-stage PCA processing on AURORA 4 (%)

Test set	Clean speech training				
	A	B	C	D	Ave
Baseline	87.85	45.84	50.57	30.29	42.51
SVN+PCA(13/14)	86.25	67.50	72.90	55.03	63.88
SVN+PCA(13/14)+mPCA	82.47	64.44	68.47	51.20	60.34
SVN+PCA(13/14)+emPCA	86.19	68.34	71.31	55.02	64.12
SVN+PCA(13/14)+ β mPCA	85.41	68.71	72.01	54.87	64.20
LEN+SVN+PCA(13/14)+emPCA	86.48	68.93	71.42	55.54	64.62
Advanced Front-end	89.87	74.16	71.45	57.99	68.16

Note that under well matched conditions (e.g., Group A in AURORA 4, Conditions 1 & 2 in CENSREC-3), it is not a surprise to find that the enhanced feature always degrade the recognition performance with respect to the baseline. This is observed in many experimental results reported in Chapters 4 & 5. The objective of the speech feature enhancement is to reduce the intra-class variability and increase the inter-class variability. Unfortunately, these two targets may not always be reached at the same time, particularly under well matched conditions. As the baseline training and testing features are relatively

well matched, so although the variability within an acoustic class could be reduced through aggressive feature normalization and trajectory filtering, the inter-class variability may be weakened instead of strengthened if too many speech components are removed. Hence in spite of the fact that the overall average recognition accuracy has been significantly improved, under well matched conditions the processed feature may not perform as well as the baseline feature. Many proposed new techniques in fact suffer from such a problem [67].

5.8 Summary

In this chapter, a few cepstral domain filtering techniques for speech feature enhancement are discussed and their performances compared. Experimental results show that under certain circumstances, e.g., for isolated word recognition with car noise (CENSREC-3), the cepstral domain filtering scheme could perform better than the Advanced Front-end scheme. Nevertheless, in the most difficult large vocabulary continuous speech recognition task, the best combination of feature normalization and cepstral filtering is still not as effective as the Advanced Front-end scheme.

It is interesting to note that there is a compromise between feature normalization and cepstral domain filtering. The effectiveness of cepstral domain filtering not only depends on its bandwidth but also depends on the “extent” of feature normalization, which is a pre-processing stage before the filtering.

Chapter 6 Model Adaptation for Large Vocabulary ASR

6.1 Introduction

Very often in ASR systems, the training and testing data are not recorded in the same environment. Typically, a system trained using clean speech utterances may be tested under a noisy environment. The mismatch between the training and testing data causes the recognition accuracy to drop drastically. The feature enhancement techniques that may be used to reduce this mismatch were described in Chapters 4 and 5. The noise components in the cepstral domain may be partially removed by using feature normalization and cepstral domain filtering. Experimentally, the feature enhancement methods have been shown to be very effective. However, the speech waveform has some special characteristics that need to be taken into account. Some speech sounds are pronounced with very low power, which is even lower than that of the environmental noise. As a result, in the cepstral domain, the noise and speech components are not easily separable. This makes any attempt to separate speech and noise rather imperfect. The feature enhancement techniques could mistakenly remove some useful but weak speech signals. Hence the improvement due to feature enhancement has its limitations.

A good way to reduce the mismatch between the training and testing features is to use noisy data to train an ASR system. But the potential for mismatch may still exist as the noise present in the testing environment is quite unpredictable. It is impossible to train the model for all the noise conditions. A better solution to the mismatch problem, known as model adaptation, will be discussed in this chapter. Model adaptation is the term used to refer to the adapting or tailoring of an ASR system to suit a specific speaker or testing

condition. An early example of model adaptation was speaker adaptation [24, 60], where a speaker-independent system was adapted to suit a particular speaker. Accent adaptation [60, 61], where the performance of non-native speech recognition is improved, is another example of model adaptation. Later, the model adaptation approach also proved to be effective in robust speech recognition [24, 25]. Instead of adapting to a speaker, the system was now adjusted to fit a particular testing condition.

Common examples of model adaptation include Maximum Likelihood Linear Regression (MLLR) adaptation [60] and Maximum A Posteriori (MAP) [63] adaptation. In this chapter, MLLR will be studied for large vocabulary continuous speech recognition tasks. It is investigated under both reverberant and additive noise conditions. The experimental results show that by using a small amount of adaptation data, the improvement that could be achieved by using MLLR is significant.

The chapter is organized as described next. A description of the mathematical procedure for achieving MLLR is given in Section 6.2. One of most important concepts – regression class tree is outlined in Section 6.3. The detailed speech model adaptation process using HTK software is described in Section 6.4. The use of MLLR for speaker adaptation on AURORA 4 is investigated in Section 6.5. The performance of MLLR adaptation for large vocabulary continuous speech recognition under reverberant noise and additive noise conditions are examined in Section 6.6 and Section 6.7 respectively. Section 6.8 concludes this chapter.

6.2 Maximum Likelihood Linear Regression (MLLR)

The basic objective of MLLR is to transform the mean and the variance of each model in the original HMM set so that the transformed model set is more likely to generate the

adaptation data. The key to MLLR is the estimation of the transformation matrix. The transformation matrices are calculated using an expectation-maximization (EM) algorithm. The matrices to transform the mean and variance are obtained in separate steps. First, the mean transformation matrix is found given the current variance. Second, the variance is transformed after the mean has been updated. The whole process could be repeated such that the following relationship is satisfied [69]:

$$L(O_T | \tilde{M}) \geq L(O_T | \hat{M}) \geq L(O_T | M) \quad (6.2.1)$$

where L is the likelihood of generating sequence O_T given a model. O_T is the adaptation data given by $O_T = \{o(1), \dots, o(T)\}$, in which $o(t)$ is speech feature vector at time t . Usually O_T is a set of data independent from both training and testing data. It has the similar noise property as the testing data. M is the original model set, $\hat{M}$ is the model set with mean updated, and $\tilde{M}$ is the model set with both mean and variance updated.

The detailed estimation of transformation matrix in adaptation process is described in [1]. The main procedure and equations are summarized below for readers' convenience. The estimation for each of mean and variance involves two passes. The first pass is to generate a global transform for all the Gaussian components in the HMM set. The second pass is to generate a more specific transformation matrix for each group of Gaussian components. The end result will be a more accurate model adaptation. Within a group, Gaussian components are close to each other so that they can share the same transformation matrix. Such a group is referred to as a regression class. Regression classes are formed by splitting all the Gaussian components into the required number of clusters using the Centroid Splitting Algorithm [1], next section gives more details.

The standard auxiliary function [1] used to estimate the transformation matrix is:

$$Q(M, \hat{M}) = -\frac{1}{2} \sum_{r=1}^R \sum_{m_r=1}^{M_r} \sum_{t=1}^T L_{m_r}(t) \left[K^{(m)} + \log(\hat{\Sigma}_{m_r}) + (o(t) - \hat{\mu}_{m_r})^T \hat{\Sigma}_{m_r}^{-1} (o(t) - \hat{\mu}_{m_r}) \right] \quad (6.2.2)$$

where M and $\hat{M}$ are the original and adapted model sets, M_r is the number of tied Gaussian components $\{m_1, m_2, \dots, m_{M_r}\}$ in each regression class, R is the number of regression classes and T is the number of observations. $K^{(m)}$ subsumes all the constants. Σ_{m_r} and μ_{m_r} are the covariance matrix and mean vector respectively of a Gaussian component. The small hat on the top of the symbol indicates the adapted value. $L_{M_r}(t)$ is the occupation likelihood, defined as:

$$L_{M_r}(t) = p(q_{m_r}(t) | M, O_T) \quad (6.2.3)$$

where $q_{m_r}(t)$ indicates Gaussian component m_r at time t . Suppose that the mean vector is n -dimensional and that the new estimated mean value is:

$$\hat{\mu}_{m_r} = W_r \xi_{m_r} \quad (6.2.4)$$

where W_m is an n by $n+1$ transformation matrix to be estimated and ξ_m is an extended vector given by

$$\xi_{m_r} = [1 \quad \mu_1 \quad \dots \quad \mu_n]^T \quad (6.2.5)$$

Substitute equation 6.2.4 into the auxiliary function 6.2.2 to adapt the mean while the covariance matrix is still the original one, i.e.,

$$Q(M, \hat{M}) = -\frac{1}{2} \sum_{r=1}^R \sum_{m_r=1}^{M_r} \sum_{t=1}^T L_{m_r}(t) \left[K^{(m)} + \log(\Sigma_{m_r}) + (o(t) - W_r \xi_{m_r})^T \Sigma_{m_r}^{-1} (o(t) - W_r \xi_{m_r}) \right] \quad (6.2.6)$$

By using the property that the covariance matrix is diagonal, equation 6.2.6 could be simplified as

$$Q(M, \hat{M}) = -\frac{1}{2} \sum_{r=1}^R \sum_{m_r=1}^{M_r} \sum_{t=1}^T L_{m_r}(t) \left[K^{(m)} + \sum_{j=1}^d \log \sigma_{m_r,j}^2 + \sum_{j=1}^d \frac{(o_j(t) - w_{rj} \xi_{m_r})^2}{\sigma_{m_r,j}^2} \right] \quad (6.2.7)$$

where w_{rj} is the j^{th} row of the transformation matrix W_r . Since the unknown variable is W_r , all the other terms could be ignored and represented as K . Therefore, only the last term inside the square bracket is left. Equation 6.2.7 is therefore simplified as

$$\begin{aligned} Q(M, \hat{M}) &= K - \frac{1}{2} \sum_{r=1}^R \sum_{m_r=1}^{M_r} \sum_{t=1}^T L_{m_r}(t) \left[\sum_{j=1}^d \frac{o_j^2(t) - 2w_{rj} \xi_{m_r} o_j(t) + w_{rj} \xi_{m_r} \xi_{m_r}^T w_{rj}^T}{\sigma_{m_r,j}^2} \right] \\ &= K_1 - \frac{1}{2} \sum_{r=1}^R \sum_{j=1}^d \left(w_{rj} \left[\sum_{m_r=1}^{M_r} \frac{1}{\sigma_{m_r,j}^2} \xi_{m_r} \xi_{m_r}^T \sum_{t=1}^T L_{m_r}(t) \right] w_{rj}^T - 2w_{rj} \left[\sum_{m_r=1}^{M_r} \sum_{t=1}^T L_{m_r}(t) \frac{1}{\sigma_{m_r,j}^2} o_j(t) \xi_{m_r} \right] \right) \\ &= K_1 - \frac{1}{2} \sum_{r=1}^R \sum_{j=1}^d (w_{rj} G_r^{(j)} w_{rj}^T - 2w_{rj} k_r^{(j)}) \end{aligned} \quad (6.2.8)$$

$$\text{where } G_r^{(j)} = \sum_{m_r=1}^{M_r} \frac{1}{\sigma_{m_r,j}^2} \xi_{m_r} \xi_{m_r}^T \sum_{t=1}^T L_{m_r}(t) \quad \text{and} \quad k_r^{(j)} = \sum_{m_r=1}^{M_r} \sum_{t=1}^T L_{m_r}(t) \frac{1}{\sigma_{m_r,j}^2} o_j(t) \xi_{m_r} \quad (6.2.9)$$

Note that K_1 in equation 6.2.8 includes both K and $o_j^2(t)$ terms. Differentiating the auxiliary function with respect to W_r and maximizing it with respect to the transformed mean yields

$$w_{rj} = k_r^{(j)} G_r^{(j)-1} \quad (6.2.10)$$

Estimation of the covariance transformation matrix is done by substituting the expression 6.2.11

$$\hat{\mu}_{m_r} = \mu_{m_r}, \quad \hat{\Sigma}_{m_r} = H_r \Sigma_{m_r} H_r^T \quad (6.2.11)$$

where H_r is another n by n matrix to be estimated, into the auxiliary function 6.2.2.

Using the fact that covariance matrices are diagonal, the auxiliary function becomes

$$Q(M, \hat{M}) = K + \sum_{r=1}^R \beta_r \log(c_{ri} a_{ri}^T) - \frac{1}{2} \sum_{j=1}^d a_{rj} G_r^{(j)} a_{rj}^T \quad (6.2.12)$$

where $\beta_r = \sum_{m_r=1}^{M_r} \sum_{t=1}^T L_{m_r}(t)$ and $A_r = H_r^{-1}$. a_{ri} is the i^{th} row of A_r , the 1 by n row vector c_{ri} is

the vector of cofactors of A_r , $c_{rij} = \text{cof}(A_{rij})$, and $G_r^{(i)}$ is defined as

$$G_r^{(i)} = \sum_{m_r=1}^{M_r} \frac{1}{\sigma_{m_r,i}^2} \sum_{t=1}^T L_{m_r}(t) (o(t) - \hat{\mu}_{m_r}) (o(t) - \hat{\mu}_{m_r})^T \quad (6.2.13)$$

Differentiating the auxiliary function with respect to A_r , and then maximizing it with respect to the transformed mean yields the following update

$$a_{ri} = c_{ri} G_r^{(i)-1} \sqrt{\left(\frac{\beta_r}{c_{ri} G_r^{(i)-1} c_{ri}^T} \right)} \quad (6.2.14)$$

This is an iterative optimization scheme as the estimate of row i is dependent on all the other rows (in that block) according to the definition of cofactor. More details on mean and variance adaptation are given in [69].

6.3 Regression Class Trees

In model adaptation, if the amount of adaptation data is small, a global transform could be used. All the components are transformed using the global transform. When more adaptation data are available, more transformations could be generated and each transformation is more specific than others to a class of Gaussian components. Regression class tree techniques group all the Gaussian mixture components into different classes (leaf nodes of the tree). Within each class the Gaussian components are close in acoustic space and therefore they could share a common transformation matrix. All the Gaussian components are assigned a class number (leaf node) using the Euclidean distance measure. The advantage of such a regression class tree is that any component

could be tied to a regression class even if there are no observation feature vectors in the adaptation data at all.

A simple example is given in Figure 6-1, in which there are 4 classes: 4, 5, 6 and 7. Solid circles and arrows indicate that there are enough data to calculate the transformation matrix, whereas the dotted arrows and circles indicate that there are insufficient data to generate the transformation matrix. However, the parent class has enough mixtures to generate a transform. In this case, there are three transforms generated and they are W_2 , W_3 and W_4 . Since both classes 6 and 7 have insufficient data, a common transform W_3 is used for these two classes. Likewise is for class 5 transformation. Class 4 has its own transformation as there are sufficient data to estimate the W_4 matrix. The following mapping rule applies:

$$\begin{aligned} W_2 &\Rightarrow \text{Class 5} \\ W_3 &\Rightarrow \text{Class 6 \& 7} \\ W_4 &\Rightarrow \text{Class 4} \end{aligned}$$

Figure 6-1: A simple example of regression class tree

During the adaptation, the transform matrix is generated given the regression class tree structure and the class assignment for all Gaussian components. These transformation matrixes will be used to transform the baseline HMM during the decoding process.

6.4 Model Adaptation process using HTK

All model adaptation experiments reported in this thesis were performed using HTK 3.3.

In this section, a brief description of the adaptation process is described.

First of all, a binary regression class tree of the current baseline HMM set is generated, which requires the statistics on occupation of each state in the HMM set. The statistics here refers to the number of frames assigned to each state which is saved in a file and used for evaluating the amount of training data for each state. The regression class tree is produced using centroid splitting algorithm and the number of leaf nodes (nodes 4, 5, 6 and 7) as shown in Figure 6-1 is predefined.

Next, the mean vector of each state is adapted in two steps: global mean adaptation and regress class tree based adaptation. In global mean adaptation, all the mixtures of the entire HMM set are transformed using a single transform. In the second round, which is tree based adaptation, more specific transformation matrices corresponding to each class are calculated. The process of variance adaptation is exactly the same as mean adaptation. The mean transform must precede the variance transform, which agrees with equation 6.2.1. So for variance transformation, the mean transformation is considered as its parent transformation.

The adaptation process could also involve speaker adaptation, in which all the mean and variance adaptation are carried over for each speaker. For adaptation to the environment, it is unnecessary to separate different speakers; only one set of transformation matrices is calculated for each test condition. For the adaptation to different speakers, a mask, which is based on the filenames of the adaptation speech files, could distinguish each speaker and determine which transform to use during decoding. Both adaptation and testing

speech files must use the same file naming convention which groups files by different speakers. Each speaker has his own transformation matrix set.

The last step of adaptation is decoding, in which the file name mask, the parent transformation matrix (mean transform in this case), the variance transformation matrix and the regression class tree are all supplied to the decoding process. The baseline model of the adaptation process is trained using clean speech data as described in Section 3.2.

Figure 6-2 illustrates the basic work flow of the model adaptation procedure.

Figure 6-2: Model Adaptation Process Using HTK

6.5 MLLR for Speaker Adaptation

Firstly, MLLR was tested for speaker adaptation. We used the AURORA 4 database for our experiment. The initial model set was trained using all the clean data (7138 utterances). There are other 14 data sets and each set has 330 utterances from 8 speakers (each speaker records around 40 utterances). The 330 utterances in each set are divided into 2 parts: 164 utterances (20 utterances from each speaker) are used as the adaptation

data and the remaining 166 utterances are tested. These 166 utterances are the reduced test set recommended by the STQ AURORA DSR Working Group. There is no overlap between the adaptation data and the testing data. Since there are 8 speakers and 14 testing conditions, we adapt the model in two ways: first, adapt the model to the environment only, and second, adapt the model to both the environment and the speaker.

Table 6-1 lists the results for speaker and test environment adaptation. The speech feature used during the experiment is the MFCC_E_D_A (c0~c12, log energy with first and second derivatives). As the speech feature has a dimension of 39 by 1 and the transformation matrix is a square matrix, the size of the transformation matrices is 39 by 39. Baseline results are the results of the clean and multi training modes. The variance adaptation procedure is performed after the mean has been updated.

Table 6-1: Speaker adaptation on AURORA 4 (%)

Test set		A	B	C	D	Ave
Baseline (clean training mode)		87.85	45.84	50.57	30.29	42.51
Baseline (multi training mode)		79.00	74.75	52.45	55.76	65.32
Mean Adaptation	Adapt to the environment	90.02	73.97	66.41	56.62	67.14
	Adapt to both speaker and environment	92.19	78.78	87.85	69.77	76.52
Variance adaptation	Adapt to the environment	89.06	78.62	79.23	64.87	73.51
	Adapt to both speaker and environment	90.39	82.81	89.58	75.12	80.54

The table shows that under the same test condition, variance adaptation is better than mean adaptation except for clean test data (set A). This verifies that Equation 6.2.1 is correct. In both mean and variance adaptation, adapting to both the speaker and the environment is much better than adapting to the environment only. Intuitively, the HMM set which captures both speaker and environment characteristics is expected to perform better.

All the adaptation results are better than the multi training mode baseline result. Because in the multi training mode, the training data consist of utterances under 6 different types of noise and it does not take the characteristics of the secondary microphone into consideration. The system is not trained to match any particular test condition. Whereas with the adaptation, the entire HMM set is tailored to fit a particular test condition (noise condition and speaker variations). The speaker independent system is converted into a speaker dependent and task specific system. The adaptation also takes account of channel distortion at the secondary microphone. This is why the adaptation results are much better than those of the multi training mode.

In addition to speaker adaptation, model adaptation is also found to be a good approach to compensate for environment noise. When an ASR system trained with clean speech data is used under adverse acoustic conditions, the mismatch due to the noise results in poor performance. One way to mitigate the degradation is to train the ASR system with noisy data directly. However, collecting a large amount of noisy data for training is troublesome and expensive. Model adaptation could solve this problem. By using a negligibly small amount of noisy speech features, the clean trained model set could be transformed as if it were trained with noisy features. Model adaptation for reverberant noise and additive noise respectively will be examined in Sections 6.6 and 6.7.

6.6 Model Adaptation for Reverberant Speech

In this section we investigate the model adaptation approach for robust speech recognition under reverberant conditions. Similar work has been reported in [25]; however, the experimental results reported are obtained using digit sequences only. In

this section, large vocabulary continuous speech recognition results have been shown. A slightly different conclusion from that in [25] has been drawn.

The database for this experiment is the Wall Street Journal (WSJ) database, which is also the database used to develop the AURORA 4 task. The utterances at 16 kHz sampling rate are selected. The same set of training data and testing data as in AURORA 4 are selected. The training data consist of 7138 utterances from 83 speakers and testing data consist of 166 utterances from 8 speakers.

To generate the reverberant speech, impulse responses at various lengths are required. All the impulse responses used in our experiment were generated using Roomsim [31], a MATLAB simulation tool box for room acoustics. The impulse responses were designed to match the characteristic of a 5m x 4m x 3m room. Both the microphone and the receiver were placed 0.75m from the two walls at the opposite ends. The absorption coefficients of the ceiling, floor and walls were adjusted until the desired reverberation time is approximately reached. The 6 impulse responses have RT60 times from 0.1s to 0.6s at a step size of 0.1s. RT60 time is the time required for reflections of a direct sound to decay by 60 dB below the level of the original sound. Figure 6-3 shows the energy decay of these 6 impulse responses. Reverberant speech is then generated by convolving a clean utterance with one the six impulse responses.

A small portion of clean training utterances (i.e. a few from each speaker) were selected as the adaptation data. They were also convolved with impulse responses at various reverberation times. The number of adaptation utterances varies from 82 (one from each speaker) to 1478 (18 utterances from each speaker). In the experiment, the effectiveness of adaptation could be analyzed by varying the amount of adaptation data.

Table 6-2 presents the reverberant speech recognition results with MLLR model adaptation. The baseline (MFCC_E_D_A) result was obtained using clean speech training and reverberant speech testing. The adaptation results were obtained by adapting the clean trained model with 551 utterances at various reverberation times measured in RT60 time. If no reverberation is present (0ms in Table 6.2) in test set, the recognition accuracy is 87.85%. As the RT60 time gets longer, the recognition accuracy drops. When the reverberation level is above 200ms, the accuracy starts to drop rapidly.

Table 6-2: Evaluation of MLLR model adaptation at different reverberant time (%)

RT60 time	0ms	100ms	200ms	300ms	400ms	500ms	600ms
baseline	87.85	85.05	75.48	46.8	26.82	21.44	6.37
Adapt to 0.1s	-	85.19	78.05	56.73	35.64	34.99	19.77
Adapt to 0.2s	-	83.20	79.93	66.48	50.98	52.34	38.05
Adapt to 0.3s	-	77.23	76.83	74.62	62.28	55.95	47.33
Adapt to 0.4s	-	69.10	72.89	70.53	67.29	60.63	52.59
Adapt to 0.5s	-	61.79	71.38	65.97	62.98	63.72	54.14
Adapt to 0.6s	-	51.55	66.30	64.27	62.21	62.54	56.61

Table 6-2 also indicates that model adaptation achieves significant improvement, particularly under long reverberation times. At RT60 of 600ms, the accuracy could be improved by more than 50%. At each reverberation time, the best result (highlighted in bold) is obtained only when the clean model set is adapted using data of the same RT60 time. This observation is different from [25] in which TI-DIGIT is used. Even the adaptation data are not at the same reverberation level as the test data, and the recognition accuracy still rises a lot, because the characteristics of the background noise is captured by acoustic models.

Figure 6-3: Relative energy decay of impulse responses from RT60 0.1s to 0.6s

In Table 6-3, the number of adaptation utterances increases from 82 to 1478 in 7 steps, and to produce the best result, the adaptation and testing data always have the same reverberation time. In reality, the adaptation data should be transcribed and recorded under the same noise environment as the testing data. It is interesting to note that when

the number of adaptation utterance ranges from 82 (1% of training data) to 1478 (21% of training data), the increment in the overall accuracy is quite negligible. When the number of utterances exceeds 551, no evident improvement has been observed. This result verifies that even a small amount of adaptation data could help to improve the system's performance. Each adaptation speech utterance is phonetically balanced, and therefore even a small amount of data would cover the entire phone set in the HMM definition. Further increase in adaptation data dose not necessarily provide extra information on the distribution of speech features under the noisy environment.

Table 6-3: Evaluation of model adaptation at different amount of adaptation data (%)

RT 60 time	100ms	200ms	300ms	400ms	500ms	600ms	Ave
baseline	85.05	75.48	46.8	26.82	21.44	6.37	43.70
Adapt to 82 sentences	83.46	77.75	71.86	63.61	59.08	50.83	67.80
Adapt to 135 sentences	84.09	78.82	74.88	65.64	61.03	54.03	69.70
Adapt to 278 sentences	85.52	79.93	75.29	65.89	63.02	54.84	70.70
Adapt to 551 sentences	85.19	79.93	74.62	67.29	63.72	56.61	71.20
Adapt to 797 sentences	85.08	79.71	74.84	67.62	63.20	56.28	71.10
Adapt to 1058 sentences	84.86	79.96	75.14	67.70	63.98	56.61	71.40
Adapt to 1478 sentences	85.19	80.66	75.21	68.29	63.31	56.91	71.60

Table 6-4: The Euclidean distance between mean vectors of each monophone in testing and adaptation speech features

Speech features	Monophones in the speech utterance						
	IH	S	EH	D	N	ER	L
Clean - Test	3.8813	4.0949	3.9277	4.4077	5.0506	3.9439	2.9047
Adapt(82) – Test	2.4487	2.4156	2.0559	3.1749	2.7907	3.5412	2.3806
Adapt(551) – Test	2.4073	2.2912	1.9786	3.6570	2.6159	3.1176	2.7963
Adapt(1478) -Test	2.3877	2.4083	1.8106	3.7484	2.5459	3.0328	2.5570

Table 6-4 lists some results of an experiment in which the closeness between the mean vectors of each monophone is measured in terms of Euclidean distance. The mean vector represents the center of each monophone class. First, all the speech vectors of each monophone are gathered in testing and adaptation utterances respectively. Next, the mean

is calculated for each monophone in testing, clean and adaptation data. The Euclidean distance between the mean vectors of the same monophone in testing, clean and adaptation data is then calculated. In the first row of Table 6-4, the distance between the clean and noisy testing speech is shown, and in the next 3 rows, the distances between the noisy testing speech and adaptation speech are listed. The differences among these 3 rows are the numbers of adaptation utterances used, i.e. from 82 to 1478 utterances. This Table clearly shows that the mean vectors of adaptation data and noisy testing data are much closer than that between noisy and clean speech. It also indicates that the amount of adaptation data does not affect very much the center (the mean vector) of each monophone class. This is why 82 utterances could achieve the same adaptation effectiveness as 1478 utterances.

6.7 Model Adaptation for Additive Noise and Channel Distortion

Model adaptation for reverberant speech was discussed in the last section. This section will focus on model adaptation for additive noisy speech. In ASR applications, additive noise appears more frequently than the reverberant noise and most of the environmental noise in ASR is classified as additive noise. Some additive noise adaptation experimental results have been presented in Section 6.5 (see “adapt to the environment” in Table 6-1). However, in that case, adaptation data were taken from the same set as the testing data and some speaker characteristics have also been incorporated while adapting to the environmental noise. This makes the final result not so convincing for analyzing the effect of environmental noise adaptation. In this section, the adaptation data and the testing data are independent of each other; they are produced in different environment although similar and with different speakers.

We still used the Wall Street Journal (WSJ) database for our experiment. The clean data are the same as those used in the last section, i.e., 7138 clean training utterances from 83 speakers and 166 clean testing utterances from 8 speakers. Each clean test utterance was recorded using two microphones: a close-talking microphone and a secondary microphone. The secondary microphone was placed away from the speaker, introducing a distortion in the recording. In our experiment, both recorded data sets were used in order to examine the effectiveness of model adaptation in the presence of channel distortion.

To generate the noisy testing utterances, 4 types of noise have been selected from Noisex 92 database [48, 53]. They are: car noise, factory noise, babble noise and office noise. The FaNT – Filtering and Noise Adding Tool [26], was used to contaminate the clean speech. The FaNT is also the tool used to create AURORA 2 and AURORA 4 databases. The SNR (signal to noise ratio) of the generated noisy utterances is uniformly distributed between 5dB and 15dB. Two ITU recommended frequency characteristics, namely P.341 and G.712, are used when adding the noise to clean speech. Since the speech files were sampled at 16 kHz, they were first down sampled to 8 kHz and filtered with the G.712 characteristic to determine the weighting factor for noise addition. Next both speech and noise at 16 kHz were filtered with the P.341 characteristic followed by adding them at the desired SNR. With the 4 types of noise added to the clean test speech files recorded using two microphones, 8 noisy testing sets were created.

The adaptation data were generated in the same way as the testing data. The number of adaptation utterances was the same as that in the last section, i.e., from 82 to 1478. These speech files were also corrupted by noise. The SNR of the adaptation utterance was also

uniformly distributed between 5dB and 15dB. In the experiment, the clean trained HMM set was initially adapted with the same type of noise as the test data.

Table 6-5 presents the adaptation recognition results for the additive noisy speech from a close-talking microphone. The baseline result was obtained using clean training speech and no adaptation was employed during the decoding. If the test set is clean speech, the recognition accuracy at the current setting is 87.85% (not shown in the table). Due to the noise, the overall average accuracy drops to around 50%. Table 6-5 shows that significant improvement is achieved through model adaptation. Regardless of the amount of adaptation utterances, the overall accuracy rises by at least 25%. The amount of adaptation data does not affect the overall results too much even if only 1% (82 utterances) of the entire training set is used. When the number of utterances exceeds 278 (around 4%) no significant improvement has been observed.

Table 6-5: Adaptation to the additive noise for the close-talking microphone, testing and adaptation utterances has the same type of noise

noise	Car	Babble	Factory	office	Ave
baseline	57.41	47.07	42.62	50.9	49.5
adapt to 82 utterances	84.57	74.84	66.15	71.71	74.32
adapt to 135 utterances	85.45	76.17	67.51	73.08	75.55
adapt to 278 utterances	86.34	78.05	68.36	73.81	76.64
adapt to 551 utterances	86.52	78.16	68.1	74.33	76.78
adapt to 797 utterances	87.18	78.08	69.02	74.33	77.15
adapt to 1058 utterances	86.96	78.23	69.5	74.33	77.26
adapt to 1478 utterances	86.63	78.23	68.8	74.25	76.98

In addition to the experiment shown in Table 6-5, cross-noise adaptation, in which the adaptation speech has different noise characteristics from the testing speech, was also tested. The results listed in Table 6-6 show some interesting findings: even if the baseline HMM set is adapted to different noise environments, under a new noise environment, it is still possible to have an improved performance. In Table 6-6, the HMM set adapted to

babble noise still performs better than the baseline case in the presence of car, factory and office noise, and likewise for factory noise adapted model in the presence of babble and office noise, as well as car noise adapted model in the presence of babble and office noise. This is because these 4 types of noise possess some common characteristics. Although the noise is absent in the adaptation speech, HMM could still capture certain features of that noise in another noise type, e.g. some characteristics of babble noise could be found in car, factory and office noise. Nevertheless, the effectiveness of the cross-noise adaptation is not as good as the same noise adaptation and in the worst case it degrades the recognition performance with respect to the baseline case.

Table 6-6: Cross-noise Adaptation to additive noise for the close-talking microphone, testing and adaptation utterances has different types of noise

noise	Car	Babble	Factory	office
baseline	57.41	47.07	42.62	50.9
adapt to 82 utterances (car noise)	-	59.41	36.68	52.45
adapt to 278 utterances (factory noise)	57.16	75.99	-	72.60
adapt to 551 utterances (babble noise)	74.99	-	55.83	51.57

Table 6-7 lists the adaptation recognition accuracy of the secondary microphone. The original HMM set is the same as in the last experiment, i.e., trained with clean speech from the close-talking microphone. The 4 test sets in this experiment are quite similar to the last experiment except that the test utterances were recorded using a secondary microphone which was apart from the speaker. If no noise is added, the clean testing set from the secondary microphone results in an accuracy of 50.57% (not shown in the table), 37% less than that with the close-talking microphone. The baseline result in Table 6-7 shows that the channel distortion will cause the overall performance to drop by 15% (49.5% to 34.1%).

Table 6-7 shows that in the presence of channel distortion, the adaptation could still maintain its effectiveness in increasing the overall recognition accuracy. By using different amounts of adaptation data, the overall average result may be raised by around 20%. As in Table 6-5, it is seen here that the number of adaptation sentences does not affect the overall results too much. When more than 551 utterances are used, no significant improvement is observed.

Table 6-7: Adaptation to the additive noise for the secondary microphone

noise	Car	Babble	Factory	office	Ave
baseline	35.37	32.98	30.33	37.72	34.1
adapt to 82 sentences	52.71	50.87	45.89	51.86	50.33
adapt to 135 sentences	54.25	52.97	48.8	53.81	52.46
adapt to 278 sentences	55.25	53.92	50.35	55.29	53.7
adapt to 551 sentences	56.35	54.73	51.27	56.02	54.59
adapt to 797 sentences	56.39	54.88	51.2	56.21	54.67
adapt to 1058 sentences	56.28	54.84	51.49	56.83	54.86
adapt to 1478 sentences	56.72	54.95	51.16	56.61	54.86

6.8 Summary

Experiments show that MLLR model adaptation is very effective for robust speech recognition. It could improve the system performance significantly under additive noise, microphone channel distortion and reverberant distortion. Through a series of experiments, we found that even when the adaptation speech and testing speech are corrupted with different types of noise, the recognition result could still be improved. Nevertheless, the best results are achieved when the adaptation data have the same noise characteristics as the test data. The other important feature in model adaptation is that the performance does not depend on the amount of adaptation data. Even a small number of noisy utterances could improve the recognition performance effectively.

The problem with the model adaptation method is that the adaptation data must be provided. Sometimes this kind of data which has to be transcribed may not be easily available. Furthermore, the adaptation will transform the model set into the adapted model set before the recognition, and therefore the decoding time is greater than that with conventional decoding.

Chapter 7 Conclusion and Future Work

Investigations into improvements that could be made to currently available robust ASR techniques have been reported in this thesis. It is well known that the major obstacle to robust speech recognition is the mismatch between the training and testing features, caused by environmental noise. All robust ASR techniques aim to reduce this mismatch by removing the effect of noise. Effective approaches considered in this study include robust feature extraction, feature enhancement and MLLR model adaptation.

MFCC, the most widely used feature extraction technique in robust speech recognition, has been considered in this thesis. In addition, it has been shown that the performance of the PAC derived feature is better than that of MFCC when non-speech like additive noise is present. However, when there is channel distortion or when low level non-stationary noise is present in the test environment, MFCC still performs better than the PAC derived feature. A good speech feature is expected to perform as well as MFCC under low noise conditions and better than MFCC under high noise conditions. Many proposed speech features improve ASR performance under high noise conditions, but they commonly suffer from degradation problems in clean and low noise conditions.

For the investigations reported in this thesis, AURORA 4 was used as the experimental framework for large vocabulary continuous speech recognition. With AURORA 4, the training procedure becomes crucial; it is much more complex than with AURORA 2, which involves only simple digit sequence recognition task. Factors affecting the performance of the system include the topology of the HMM, the state clustering

threshold, the number of Gaussian mixture components, the parameter tuning and the complexity of the language model.

Feature enhancement is an effective technique to improve the accuracy in robust ASR. Feature normalization and cepstral domain filtering are two of the techniques available for feature enhancement. In feature enhancement, Histogram Equalization (HEQ) has been shown to be more effective than the traditional Mean and Variance Normalization, because HEQ normalizes not only the first and second moments but also all the higher moments by equalizing the PDF of the speech feature components. Class based HEQ is an improved version in which HEQ is applied to each individual acoustic class. The experimental results show that by combining Class based HEQ and PCA (Principal Component Analysis) based feature compression, better performance is achieved than with the Advanced Front-end standard (ETSI ES 202 050) in an AURORA 4 task.

The ARMA, LDA (Linear Discriminant Analysis) and PCA (Principal Component Analysis) filters are basically cepstral domain filters and are all very effective in reducing the spectral variation and removing the noise components. In addition, the proposed entropy weighted multi-eigenvector filter and β -order multi-eigenvector filter, in which nonlinear weightings are attempted instead of the linear approach, have also been shown to outperform the original multi-eigenvector filter in the AURORA 4 and CENSREC-3 experimental frameworks. The plot of the filter response shows that the nonlinear weighting could increase the bandwidth of the filter in the cepstral domain. The idea of using entropy based weighting hinges on the fact that in the multi-dimensional feature space the direction having the most speech information should be the one with the highest entropy contribution. This idea is also verified to be correct by the experiment results.

It was found that the most distinctive feature among the above mentioned cepstral domain filters is the effective bandwidth in the modulation spectrum. The bandwidth determines the amount of the noisy components to be removed and the speech components to be retained, though it also determines the extent of distortions to the speech features at the same time. An optimum filter achieves a balance between noise reduction and feature smoothing. Furthermore, the effectiveness of cepstral domain filtering also depends on feature normalization, which is usually the stage before the filtering process. A combined effort of feature normalization and filtering results in an improvement of speech recognition. However, a good normalization technique may not work well if it is simply cascaded with a good cepstral filtering method. This is because the speech features might be over processed, and hence their inter-class variability is weakened. Instead, a weak feature normalization followed by strong cepstral filtering could possibly perform better. Such a combination may possibly avoid the conflict between suppressing noise components and damaging the speech features.

The essence of speech feature enhancement is to reduce the intra-class variability while increase the inter-class variability. However, this is not always easy to achieve at the same time. Though many speech feature enhancement techniques could enhance the inter-class variability, it also increases the intra-class variability. As a result, the overall recognition performance may not improve significantly, and under certain circumstances, it may even result in performance degradation. The complexity of speech feature space makes it difficult to achieve the two goals at the same time. This is also one of the reasons why some good algorithms are not working when they are cascaded together in an attempt to achieve robust speech recognition.

Model adaptation is another category of robust speech recognition techniques. The experimental results obtained show that the improvement due to MLLR is significant. As long as the adaptation speech utterances are phonetically balanced, a small amount of noisy data could transform an ASR system trained on clean speech into a speaker dependent and task specific system. Further increase in the amount of adaptation data does not improve the recognition much. Although the best result is obtained using the adaptation data with the same noise type, cross adaptation was verified to be also effective in improving the system performance. MLLR is effective in increasing the recognition accuracy under reverberant noise conditions, additive noise conditions and channel distortion. The drawback of model adaptation is the long training and decoding time as well as the availability of extra adaptation speech data.

Although considerable effort has been made in the study of cepstral domain filtering techniques, the relationship between the speech and noise components in the cepstral domain is still unclear. This lack of sufficient knowledge is the main reason why only limited improvement has so far been achieved with the use of cepstral domain filtering techniques. Future work should therefore be concentrated on achieving a better understanding of the relationship between the noise and speech components in the cepstral domain.

The other problem encountered in robust speech recognition is the inconsistency of algorithm performance across different databases. It was found that some algorithms which were effective with AURORA 4 and CENSREC-3 did not work well with AURORA 2 and those which worked well with AURORA 2 were not so effective with AURORA 4 and CENSREC-3. One cause of this inconsistency is the differing

experimental frameworks of the databases, e.g., the nature of the utterance, the property of the background noise and the complexity of the language model. However the details are still not clear and further analysis is necessary. Even within a same database, the speech feature which works well under certain types of noise may not be effective under other types of noise. The goal for the future should be to arrive at some universal feature enhancement and extraction approaches which have stable performance, irrespective of the database used and noise type involved.

References

- [1]. Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Gareth Moore, Jukia Odell, Dava Ollason, Dan Povey, Valtcho Valtchev and Phil Woodland, *The HTK Book (for HTK Version 3.3)*, Cambridge University Engineering Department, Apr 2005.
- [2]. N. Parihar and J. Picone, "DSR Front End LVCSR Evaluation," AU/384/02, STQ Aurora Working Group, Dec 2002.
- [3]. G. Hirsch, "Experimental Framework for the Performance Evaluation of Speech Recognition Front-ends on a Large Vocabulary Task", AU/417/02, STQ Aurora DSR Working Group, Nov 2002.
- [4]. S.K Au Yeung, M. H Siu, "Improved Performance of AURORA 4 Using HTK and Unsupervised MLLR Adaptation", in *Proc. ICSLP 2004*, Korea, 2004.
- [5]. O. Viikki and K. Laurila, "Cepstral Domain Segmental Feature Vector Normalization for Noise Robust Speech Recognition," *Speech Commun.*, vol. 25, pp. 133-147, 1998.
- [6]. C. P. Chen, J. Bilmes and Daniel P. W. Ellis, "Speech Feature Smoothing for Robust ASR," in *Proc. ICASSP 2005*, Philadelphia, Mar 2005, pp. 525-528.
- [7]. N.C. Wang, J.W. Hung and L.S.Lee, "Data-driven temporal filters based on multi-eigenvectors for robust features in speech recognition," in *Proc. ICASSP 2003*, Hong Kong, Apr 2003, pp. 400-403.
- [8]. H. Hermansky, N.Morgan, "RASTA processing of speech", *IEEE Trans. Speech Audio Process.*, vol. 2, no. 4, pp. 578-589, Oct 1994.

- [9]. ETSI ES 202 050, “Speech processing, transmission and quality aspects (STQ), distributed speech recognition; Advanced front-end feature extraction algorithm; Compression algorithms,” v1.1.4, Nov 2005.
- [10]. S. Ikbāl, H. Misra and H. Bourlard, “Phase Autocorrelation (PAC) derived robust speech features,” in *Proc. ICASSP 2003*, Hong Kong, Apr 2003, pp.133-136.
- [11]. S. Young, “A review of large-vocabulary continuous-speech recognition,” *IEEE Signal Process. Mag.*, vol. 13, no. 5, pp. 45-57, Sep 1996.
- [12]. AURORA-J/CENSREC website: <http://sp.shinshu-u.ac.jp/CENSREC/>.
- [13]. S. Ikbāl, H. Hermansky and H. Bourlard, “Nonlinear spectral transformations for robust speech recognition,” in *2003 IEEE Workshop on Automatic Speech Recognition and Understanding*, Dec 2003, pp. 393-398.
- [14]. J.W. Hung, H. M Wang and L. S. Lee, “Comparative analysis for data-driven temporal filters obtained via principal component analysis (PCA) and linear discriminant analysis (LDA) in speech recognition,” in *Proc. Eurospeech 2001*, Denmark, Sep 2001, pp. 1959-1962.
- [15]. C. P. Chen, J. Bilmes and K. Kirchhoff, “Low-resource noise-robust feature post-processing on AURORA 2.0,” in *Proc. ICSLP 2002*, Denver, USA, Sep 2002, pp. 2445-2448.
- [16]. C. P. Chen, K. Filali and J. A. Bilmes, “Frontend post-processing and backend model enhancement on the AURORA 2.0/3.0 Databases,” in *Proc. ICSLP 2002*, Denver, USA, Sep 2002, pp. 241-244.

III

- [17]. ETSI ES 201 108, "Speech processing, transmission and quality aspects (STQ); Distributed speech recognition; Front-end feature extraction algorithm; Compression algorithms," v1.1.3, Sep 2003.
- [18]. CMU dictionary website: <http://www.speech.cs.cmu.edu/cgi-bin/cmudict?in=re>.
- [19]. Y. Chen and L. S. Lee, "Energy-based frame selection for reliable feature normalization and transformation in robust speech recognition," in *Proc. InterSpeech 2005*, Portugal, Sep 2005, pp. 385-388 .
- [20]. H. You, Q. Zhu and A. Alwan, "Entropy-based variable frame rate analysis of speech signals and its application to ASR," in *Proc. ICASSP 2004*, Montreal, Canada, May 2004, pp. 549-552.
- [21]. S. Tiberwala and H. Hermansky, "Multi-band and adaptation approaches to robust speech recognition," in *Proc. Eurospeech 1997*, Rhodes, Greece, 1997, pp. 2619-2622.
- [22]. O. Viikki, D. Bye and K. Laurila, "A recursive feature vector normalization approach for robust speech recognition in noise," in *Proc. ICASSP 1998*, Seattle, USA, May 1998, pp. 733-736.
- [23]. W. Zhu and D. O'Shaughnessy, "Log-energy dynamic range normalization for robust speech recognition," in *Proc. ICASSP 2005*, Philadelphia, USA, Mar 2005, pp.245-248.
- [24]. M. J. F. Gales, D. Pye and P.C. Woodland, "Variance compensation within the MLLR framework for robust speech recognition and speaker adaptation," in *Proc. ICSLP 1996*, Philadelphia, USA, Oct 1996, pp. 1832-1835.

- [25]. M. Toh, R. Togneri and S. Nordholm, "Combining MLLR adaptation and feature extraction for robust speech recognition in reverberant environments," in *Proc. SST2006*, Auckland, New Zealand, Dec 2006, pp. 88-93.
- [26]. FaNT website: <http://dnt.kr.hs-niederrhein.de/download.html>
- [27]. L. R. Rabiner, "A tutorial on Hidden Markov Models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257-286, Feb 1989.
- [28]. R. C. Gonzalez & R. E. Woods, *Digital Image Processing*, 2nd edition, Prentice Hall, New Jersey, 2002.
- [29]. Y. Suh, M. Ji and H. Kim, "Probabilistic class histogram equalization for robust speech recognition," *IEEE Signal Process. Lett.*, vol 14, no. 4, pp. 287-290, Apr 2007.
- [30]. J. F. Kaiser, "On a simple algorithm to calculate the 'energy' of a signal," in *Proc. ICASSP 1990*, Albuquerque, USA, Apr 1990, pp. 381-384.
- [31]. D. R. Campbell, K. J. Palomaki and G. J. Brown, "A MATLAB simulation of "shoebox" room acoustics for use in research and teaching," *Comput. Inf. Syst.*, vol. 9, no. 3, pp. 48-51, Oct 2005.
- [32]. H. Hirsch and D. Pearce, "The AURORA experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *Proc. ISCA ITRW ASR2000*, Paris, France, Sep 2000.
- [33]. R. G. Leonard, "A database for speaker-independent digit recognition," in *Proc. ICASSP 1984*, San Diego, USA, Mar 1984, pp. 328-331.
- [34]. C. Becchetti and L. P. Ricotti, *Speech Recognition Theory and C++ Implementation*, John Wiley & Sons, Chichester, England, 1999.

- [35]. R. Suoranta, K.-P. Estola, S. Rantala and H. Vaataja, "PDF estimation using order statistic filter bank," in *Proc. ICASSP 1994*, Adelaide, Australia, Apr 1994, pp. 625-628.
- [36]. Y. Suh and H. Kim, "Class-based histogram equalization for robust speech recognition," *ETRI Journal*, vol. 28, no. 4, pp. 502-505, Aug 2006.
- [37]. H. Hermansky, "Modulation spectrum in the automatic recognition of speech," in *IEEE Workshop Automatic Speech Recognition Understanding*, Santa Barbara, USA, Dec 1997, pp. 140-147 .
- [38]. J. W. Hung and L. S. Lee, "Data-driven temporal filters obtained via different optimization criteria evaluated on AURORA 2 database," in *Proc. ICSLP 2002*, Denver, USA, Sep 2002, pp. 441-444.
- [39]. F. Hilger and H. Ney, "Quantile based histogram equalization for noise robust large vocabulary speech recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol 14, no. 3, pp. 845 -854, May 2006.
- [40]. Á. de la Torre, A. M. Peinado, J. C. Segura, J. L. Pérez-Córdoba, M. C. Benítez and A. J. Rubio, "Histogram equalization of speech representation for robust speech recognition," *IEEE Trans. Speech Audio Process.*, vol. 13, no.3, pp. 355-366, May 2005.
- [41]. J. C. Segura, C. Benítez, Á. de la Torre, A. J. Rubio and J. Ramírez, "Cepstral domain segmental nonlinear feature transformations for robust speech recognition," *IEEE Signal Process. Lett.*, vol.11, no.5, pp. 517-520, May 2004.
- [42]. Y. Gong, "Speech recognition in noisy environments: a survey," *Speech Commun.*, vol. 16, no. 3, pp. 261-291, Apr 1995.

- [43]. M. Fujimoto, K. Takeda, and S. Nakamura, "CENSREC-3: An evaluation framework for Japanese speech recognition in real driving car environments," *IEICE Trans. Inf. & Syst.*, vol. E89-D, no. 11, pp. 2783-2793, Nov 2006.
- [44]. L. Rabiner and B-H. Juang, *Fundamentals of speech recognition*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1993.
- [45]. L. Rabiner and R. Schafer, *Digital processing of speech signals*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1978.
- [46]. S. Furui, *Digital speech processing, synthesis, and recognition*, Marcel Dekker Inc., New York, 2001.
- [47]. F. Jelinek, *Statistical Methods for speech recognition*, MIT press, London, England, 1997.
- [48]. A. Varga, H. Steeneken, M. Tomlinson and D. Jones, "The NOISEX-92, study on the affect of additive noise on automatic speech recognition," *Technical report*, DRA Speech Research Unit, Malvern, England, 1992.
- [49]. X. Wang, Y. Dong, J. Hakkinen, and O. Viikki, "Noise robust Chinese speech recognition using feature vector normalization and higher-order cepstral coefficients," in *Proc. WCC 2000 - ICSP 2000*, Beijing, China, Aug 2000, vol 2, pp. 21-25.
- [50]. O. Viikki and K. Laurila, "Noise robust HMM-based speech recognition using segmental cepstral feature vector normalization," *ESCA-NATO workshop on robust speech recognition for unknown communication channels*, 1997, pp. 107-110.

VII

- [51]. N. Morgan and H. Bourlard, "Continuous speech recognition, an introduction to the hybrid HMM/connectionist approach," *IEEE Signal Process. Mag.*, vol. 12, no. 3, pp. 24-42, May 1995.
- [52]. H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *J. Acoust. Soc. Am.*, vol. 87, no. 4, pp. 1738-1752, Apr 1990.
- [53]. A. Varga and H. J. M. Steeneken, "Assessment for automatic speech recognition: II. NOISE-92: A database and an experiment to study the effect of additive noise on speech recognition systems," *Speech Commun.*, vol. 12, no. 3, pp. 247-251, Jul 1993.
- [54]. S. Ikbāl, H. Misra, H. Bourlard and H. Hermansky, "Phase autocorrelation (PAC) features in entropy based multi-stream for robust speech recognition," in *Proc. ICASSP 2004*, Montreal, Canada, May 2004, vol. 1, pp. 205-208.
- [55]. A. M. Toh, R. Togneri and S. Nordholm, "Investigation of robust feature for speech recognition in hostile environments," in *Proc. 2005 Asia-Pacific Conference on Communications*, Perth, Australia, Oct 2005, pp. 956-960.
- [56]. H. Misra, S. Ikbāl, H. Bourlard and H. Hermansky, "Spectral entropy based feature for robust ASR," in *Proc. ICASSP 2004*, Montreal, Canada, May 2004, vol. 1, pp. 193-196.
- [57]. B. J. Shannon and K. K. Paliwal, "MFCC computation from magnitude spectrum of higher lag autocorrelation coefficients for robust speech recognition," in *Proc. ICSLP 2004*, Korea, Oct 2004, pp. 791-794.

- [58]. B. J. Shannon and K. K. Paliwal, "Influence of autocorrelation lag ranges on robust speech recognition," in *Proc. ICASSP 2005*, Philadelphia, US, May 2005, pp. 1545-1548.
- [59]. L. S. Lee and J. W. Hung, "Optimization of temporal filters for constructing robust features in speech recognition," *IEEE Trans. Audio Speech Lang. Process.*, vol. 14, no. 3, pp. 808 -831, May 2005.
- [60]. C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," *Comput. Speech Lang.* vol. 9, no. 2, pp. 171-185, Apr 1995.
- [61]. W. K. Liu and P. Fung, "MLLR-based accent model adaptation without accented data," in *Proc. ICSLP 2000*, Beijing, China, Oct 2000, vol.3, pp. 738-741.
- [62]. Y. R. Oh, J. S. Yoon and H. K. Kim, "Acoustic model adaptation based on pronunciation variability analysis for non-native speech recognition," *Speech Commun.*, vol. 49, no. 1, pp. 59-70, Jan 2007
- [63]. G. Zavagliakos, R. Schwartz and J. McDonough, "Maximum a posteriori adaptation for large scale HMM recognizers," In *Proc. ICASSP 1996*, Atlanta, GA, 1996 pp. 725–728.
- [64]. X. Xiao, E. S. Chng and H. Z. Li, "Normalizing the speech modulation spectrum for robust speech recognition," in *Proc. ICASSP 2007*, Honolulu, USA, Apr 2007, pp. 1021 -1024.
- [65]. H. Misra, S. Iqbal, S. Sivasubramanian and H. Bourlard, "Multi-resolution spectral entropy feature for robust ASR," in *Proc. ICASSP 2005*, Philadelphia, US, Mar 2005, pp. I253-I256.

IX

- [66]. R. W. Johnson and J. E. Shore, "Which is the better entropy expression for speech processing: -SlogS or logS," *IEEE Trans Acoust Speech Signal Process*, vol. ASSP-32, no. 1, pp. 129-137, Feb 1984.
- [67]. X. Xiao, E. S Chng and H. Z. Li, "Evaluating the temporal structure filtering techniques on Aurora-4 task," in *Proc. InterSpeech*, Antwerp Belgium, Aug 2007.
- [68]. M. Welling, "Fisher Linear Discriminant Analysis," *a Classnotes in Machine Learning*, <http://www.cs.huji.ac.il/~csip/Fisher-LDA.pdf>
- [69]. M. J. F. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Comput. Speech Lang.* vol.12, no.2, pp. 75-98, Apr 1998.