

Noise Robustness in Automatic Speech Recognition

Chia-Ping Chen

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

Doctor of Philosophy

University of Washington

2004

Program Authorized to Offer Degree: Electrical Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Chia-Ping Chen

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of Supervisory Committee:

Jeff Bilmes

Reading Committee:

Jeff Bilmes

Katrin Kirchhoff

Mari Ostendorf

Date:

In presenting this dissertation in partial fulfillment of the requirements for the Doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to Bell and Howell Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature_____

Date_____

University of Washington

Abstract

Noise Robustness in Automatic Speech Recognition

by Chia-Ping Chen

Chair of Supervisory Committee:

Professor Jeff Bilmes
Electrical Engineering

The issue of noise robustness in automatic speech recognition is of practical importance and largely unsolved. In this thesis, this problem is tackled from both perspectives of front-end speech features and back-end speech models. For the front end, a feature processing technique consisting of mean subtraction, variance normalization and ARMA filtering is investigated. Mathematical analyses are carried out for the distortion of speech features in the presence of additive and convolutional noises. Extensive experiments are conducted to see how to best use this front-end technique. It is experimentally verified to be extremely effective for the noisy-digit databases of Aurora. This performance gain is achieved without increasing the model parameters and computational cost. For the back end, a novel random variable called a feature selector is introduced into speech models to dynamically select a robust component feature to score, ignoring the others. The values of the feature selectors are based on either the energy or the spectral entropy of the signal. This back-end technique does not lead to significant performance gain with the investigated feature streams of MV and MVA features. Yet it is a novel scheme of integrating multiple information sources.

TABLE OF CONTENTS

List of Figures	v
List of Tables	vii
Chapter 1: Introduction	1
1.1 Statistical Automatic Speech Recognition Basics	1
1.1.1 Speech Features	2
1.1.2 Speech Models	3
1.1.3 Evaluation Methods	3
1.2 Current ASR Performance Levels	4
1.3 Overview of Noise Robustness in Speech Recognition	5
1.4 Thesis Overview	8
Chapter 2: Review of Noise-Robustness in ASR Systems	9
2.1 Research Background and Overview	9
2.2 Noise-Robust Front Ends	11
2.2.1 Kalman Filters	12
2.2.2 Spectral Subtraction	13
2.2.3 RASTA	13
2.2.4 Correlation Features	14
2.3 Noise-Robust Back Ends	15
2.3.1 Parallel Model Combination	16
2.3.2 Model Adaptation	17
2.3.2.1 Maximum Likelihood Linear Regression	17
2.3.2.2 Maximum A Posteriori	17

2.4	Techniques on Recent Noisy Speech Databases	18
2.4.1	Aurora 2.0	18
2.4.1.1	Tandem Acoustic Modeling	19
2.4.1.2	Feature Vector Selection	20
2.4.1.3	Missing Data Techniques	20
2.4.1.4	Spectral-Based Discriminant Features	21
2.4.2	Aurora 3.0	22
2.4.2.1	Gabor Features	23
2.4.2.2	Splice	24
2.4.2.3	Discriminatively Trained Auditory Features	25
2.4.2.4	Histogram Equalization	26
2.4.2.5	Eigenspace Normalization	26
2.5	Discussion	26
Chapter 3: Analysis of Feature Distortion		28
3.1	Common Types of Noise	28
3.2	Definition of the Used Features	29
3.3	Feature Distortion in the Presence of Noise	30
3.3.1	Distortion by Convolutional Noises	30
3.3.2	Distortion by Additive Noises	33
3.3.2.1	General Noise Level	35
3.3.2.2	Low Additive Noises	37
3.3.2.3	High Additive Noises	38
Chapter 4: MVA Feature Processing: Definition and Analysis		40
4.1	Definition of MVA	40
4.2	Analysis on MVA with Noisy Speech Features	42
4.2.1	An Example	42
4.2.2	Convolutional Noise and Mean Subtraction	42

4.2.3	Additive Noise and Variance Normalization	44
4.2.4	ARMA Filtering	44
Chapter 5:	Dynamic-Bayesian-Network ASR Systems	49
5.1	Introduction	49
5.2	Investigated Multiple-stream ASR Systems	51
5.2.1	Multiple Feature Streams	51
5.2.2	Dynamic Bayesian Networks and Graphical Models	51
5.2.3	Feature Selector and New Models	52
5.3	Implementation Specification	52
5.3.1	Component Feature Streams	52
5.3.2	GMTK-based ASR Systems	54
5.3.3	Observed Feature Selectors	56
5.3.3.1	Energy-based SNR Estimation	56
5.3.3.2	Entropy-based SNR Estimation	60
Chapter 6:	Experimental Results	65
6.1	MVA on the MFCCs	65
6.1.1	Front-End Configurations	65
6.1.2	Back-End Configuration	66
6.1.3	Results	66
6.1.3.1	Comparison of Different Front-End Configurations	67
6.1.3.2	Results of Multi-Domain Processing	70
6.1.3.3	Comparison with Designed Filters	72
6.1.3.4	Comparison with Data-Driven Filters	81
6.1.3.5	Comparison of ARMA Orders	82
6.2	MVA on Other Features	83
6.2.1	MFCC	86
6.2.2	LPC	87

6.2.3	LPC-CEPSTRA	88
6.2.4	Tandem	89
6.2.5	PLP	91
6.2.6	Modulation Spectrogram	92
6.2.7	Modulation Cross-Correlagram	92
6.2.8	RASTA	93
6.2.9	Comparison Across Feature Sets	94
6.2.10	Summary	96
6.3	DBN ASR Systems	97
6.3.1	Experimental Setup	97
6.3.2	Results and Discussion	97
6.3.2.1	Baseline	97
6.3.2.2	Energy-based Feature Selectors	98
6.3.2.3	Entropy-based Feature Selectors	99
6.4	Experiments on a Non-Stationary Noisy Database	100
6.5	Experiments on the Spine Database	104
6.5.1	Different Front-end Configurations	105
6.5.2	Different ARMA Orders	107
Chapter 7:	Conclusions	109
7.1	Summary	109
7.2	Future Work	110
Bibliography		112
Appendix A:	EM Algorithm in Maximum Likelihood Estimation	122
Appendix B:	Commonly Used Notations	123

LIST OF FIGURES

1.1	Diagram of ASR system in noisy environments	7
4.1	The Block diagram of MVA feature processing technique	41
4.2	The time sequences of cepstral coefficients and log energy	43
4.3	The frequency response of a second-order ARMA filter	46
4.4	The spectra of the time sequences of MFCC and log energy	47
5.1	The scopes of HMMs, BNs, DBNs and GMs	52
5.2	Graphical models of DBN	53
5.3	A GMTK-based DBN ASR system	55
5.4	Fitting SNR (in dB) values to the range of log energy.	59
5.5	The spectral entropy of clean and noisy speech examples	62
5.6	Mean-variance plot of the spectral entropy	63
5.7	Cluster points using different data sets	64
6.1	Block diagram of multi-domain processing	71
6.2	Word accuracies of Aurora 2.0 multi-train tasks with MVA processing in both log mel spectral and cepstral domains	73
6.3	Word accuracies of Aurora 2.0 clean-train tasks with MVA processing in both log mel spectral and cepstral domains	74
6.4	Word accuracies for Aurora 3.0 German tasks with MVA processing in both log mel spectral and cepstral domains	75
6.5	Word accuracies for Aurora 3.0 Danish tasks with MVA processing in both log mel spectral and cepstral domains	76

6.6	Word accuracies for Aurora 3.0 Finnish with MVA processing in both log mel spectral and cepstral domains	77
6.7	Word accuracies for Aurora 3.0 Spanish tasks with MVA processing in both log mel spectral and cepstral domains	78
6.8	Word accuracies of Aurora 2.0 multi-train tasks with the ARMA filter replaced by various designed FIR filters	79
6.9	Word accuracies of Aurora 2.0 clean-train tasks with the ARMA filter replaced by various designed FIR filters	79
6.10	Word accuracies for Aurora 3.0 German with the ARMA filter replaced by various designed FIR filters	80
6.11	Word accuracies for Aurora 3.0 Danish with the ARMA filter replaced by various designed FIR filters	80
6.12	Word accuracies of Aurora 2.0 multi-train tasks with various orders of the ARMA filter	83
6.13	Word accuracies of Aurora 2.0 clean-train tasks with various orders of the ARMA filter	84
6.14	Word accuracies of Aurora 3.0 German tasks with various orders of the ARMA filter	84
6.15	Word accuracies of Aurora 3.0 Danish tasks with various orders of the ARMA filter	85
6.16	Word accuracies of Aurora 3.0 Finnish tasks with various orders of the ARMA filter	85
6.17	Word accuracies of Aurora 3.0 Spanish tasks with various orders of the ARMA filter	86
6.18	The comparison of feature sets with the Aurora 2.0 multi-train tasks	94
6.19	The comparison of feature sets with the Aurora 2.0 clean-train tasks	95

LIST OF TABLES

1.1	Word error rates of ASR systems on selected tasks	4
2.1	Word accuracies of the defined baseline ASR system on Aurora 2.0	19
2.2	Word error rates of selected ASR systems on Aurora 2.0	19
2.3	Relative improvements of selected ASR systems on Aurora 3.0	23
4.1	The Euclidean distances between the spectra of clean and noisy speech	48
5.1	Linear fitting of the log energy and the SNR	59
5.2	Statistics of energy-based utterance-level environmental estimation	60
6.1	Four front-end configurations	66
6.2	Word Accuracies of Aurora 2.0 tasks with various configurations	67
6.3	Word accuracies of Aurora 3.0 tasks with various configurations	68
6.4	Word accuracies for Aurora 2.0 with data-driven filter	82
6.5	Evaluation on the MFCC features	87
6.6	Evaluation on the LPC features.	88
6.7	Evaluation on the LPC-Cepstral features.	88
6.8	Evaluation on the tandem features	90
6.9	Evaluation on the PLP features.	91
6.10	Evaluation on the MSG features.	92
6.11	Evaluation on the MCG features.	93
6.12	Evaluation on the RASTA features.	93
6.13	Relative improvements of MVA over RAW on all features.	96
6.14	Performance of baseline systems of the Aurora 2.0 tasks	98

6.15	Performance of baseline systems of Aurora 3.0 tasks	98
6.16	Performance of the Aurora 2.0 tasks with energy-based utterance-level feature selector	99
6.17	Performance of the Aurora 3.0 tasks with energy-based frame-level feature selector	100
6.18	Performance of the Aurora 2.0 tasks with entropy-based feature selector . . .	100
6.19	Performance of the Aurora 3.0 tasks with entropy-based feature selector . . .	101
6.20	Performances of non-stationary Aurora 2.0 noisy databases	103
6.21	Results of GMTK-based systems on a highly non-stationary Aurora 2.0 noisy database	104
6.22	Spine results: Different front-end configurations	106
6.23	Spine results: Different orders of ARMA	108

ACKNOWLEDGEMENTS

In current times it is virtually impossible to do anything without significant assistance in many ways from many of people. There are so many of them deserving my acknowledgment, and most of them I don't even know who they are (and they don't know who I am). They are the people that work hard everyday to maintain the environment enjoyable and workable. Therefore, first of all, I would like to thank them.

I would like to express deep gratitude to all the members of my PhD supervisory committee, who dedicated time in understanding what I have done and in giving me invaluable guidance for this work. Professor Bilmes, the chair of this committee and my advisor, definitely provided the most support and help. The other members are Professors Bashein, Hwang, Kirchhoff, Ostendorf and Gupta.

Regarding research, I have obtained much help from people in the SSLI Lab at the Electrical Engineering Department of the University of Washington. Professors Ostendorf, Bilmes and Kirchhoff started this lab in 1999, which has since that time earned its first-rate academic reputation. Professor Kirchhoff has given me much advice, particularly in the summer of 2001, when my advisor was away. Ivan, Becky, and Zak all helped breaking me in with detailed instructions when I first joined the lab in 2000. Ozgur has always been willing to help and discuss ideas. Also, I have had the pleasure of accepting (or rejecting) opinions and tips from other members of the lab: Scott, Costas, Karim, Gang, Arindam, Julie, Sarah, Chris, Mukund, Xiao, Jon, Dustin, Xin, Kevin, Jeremy, and Tim. I hope they have had similar pleasures. I would also like to thank Takahiro for introducing to me the Julius speech decoder during his visit to SSLI in 2002, and Harriet for her discussion of feature normalization.

The computing staff has been able to keep the computational facility up and running. Furthermore, they have been responsive throughout, and have once rescued my work when my home directory had a disk failure during the long weekend of the Martin Luther King's Day. Special thanks to Lee, Ian, and Haychoi.

Outside SSLI, I have benefited from a few correspondences with David Gelbart at ICSI,

Professor Ellis at Columbia University and Dr. Kingsbury at IBM. They all have designed robust speech features and I was very fortunate to discuss the ideas directly with them.

The following tools have been indispensable for the research in this dissertation: HMM Toolkit (HTK) by Cambridge University; Graphical Modeling Toolkit (GMTK) by University of Washington and IBM; SPRACHcore by the International Computer Science Institute (ICSI); Internet-based information search and retrieval, particularly IEEEExplore, Google, and Yahoo. Many thanks to the developers of these programs.

Not strictly about speech recognition and natural language processing, the following courses have been wonderful: Optimizations by Professor Rockafellar; Stochastic Modeling by Professor Besag; Causal Models by Professor Richardson; Computational Electromagnetics by Professor Tsang; Computer Networks by Professor Azizoglu. I thank them for making those subjects fascinating. At times these courses have provided refreshing elements for my mind.

Special thanks to the sport of badminton and those who show up regularly at IMA on Tuesdays, especially Bill, Mulu, Meher, Nick, and Steve. It has been such a healthy distraction from work since 2003.

Above all, I could not have done this without the love of my family, who have managed to bear my long absence from home.

DEDICATION

This dissertation is dedicated to my father and mother.

Chapter 1

INTRODUCTION

1.1 Statistical Automatic Speech Recognition Basics

An automatic speech recognition (ASR) system is a programmed machine that recognizes human speech. An accurate ASR system can provide an ideal human-machine interface, as speech is a fast and natural way for most people to express intentions. A short list of voice commands can reduce significantly the number of keystrokes or point-and-clicks, which require complicated hand-eye coordination. Other examples of ASR applications include automatic information retrieval from a database of audio files and automatic word spotting for surveillance-system archives. In these applications, enormous amounts of data are being handled and it is therefore essential to make the process automatic. For this purpose, an ASR system can be used to convert audio signals into queries to initiate the automatic process.

In *statistical ASR*, the human speech is represented as a stochastic process, for which an *acoustic model* is used to approximate the acoustic aspects (such as temporal and spectral patterns) and a *language model* is used to deal with the linguistic aspects (such as syntax and semantics) of speech. Acoustic models are often established in a *feature* space, where features are meant to be salient representations of speech signals for the purpose of recognizing the embedded linguistic targets. Language models are often built in the discrete space of word sequences, with the goal of assigning most of the probability mass to the well-formed and meaningful word sequences (i.e. syntactically and semantically correct sentences) while maintaining non-zero (albeit very small) probabilities to the ill-formed ones.

An ASR system thus includes a module of feature extraction in the front end and a module of speech models in the back end. The parameters in speech models are first trained

with training data and then used for test data. (cf. Figure 1.1.)

After an ASR system has been trained and tested, its performance can be evaluated by different performance measures based on the objective of the underlying application. The *word error rate*, which is defined as the percentage that the evaluated ASR system commits an error in the recognized words, is used in this thesis.

The following subsections cover further details and common examples of the features, models and evaluation of ASR systems.

1.1.1 *Speech Features*

A discrete-time representation of human speech usually contains at least 8000 samples per second, based on the common belief that most of the information in speech is contained in the band of 0 – 4000 Hz. In an ASR system, speech samples are further processed into speech feature vectors. Such signal processing often incorporates knowledge of articulation, audition and perception about human speech. After the extraction of features, a typical setting is to have 100 feature vectors per second, each containing tens of feature components. The actual rate and size may vary from system to system.

Ideally, the features should be discriminative with respect to different linguistic units and robust to variation within the same linguistic units. The design of speech features is a paramount problem for robust and accurate ASR systems, as there are many factors contributing to acoustic variation as well as acoustic confusion.

A handful of features has been commonly adopted in ASR systems. The linear prediction coefficients (LPC) [62] represent a simple all-pole model of vocal tract. The mel-frequency cepstral coefficients (MFCC) [21] are based on the spectral envelopes of speech and on knowledge of the auditory systems. Features based on artificial neural networks (ANN) simulate an information-processing paradigm used by the human brain with emphasis on discrimination.

1.1.2 Speech Models

The speech model of an ASR system can be decomposed into an acoustic model and a language model. An acoustic model is an approximation to the probability distribution of random speech features given the linguistic targets. For example, with the hidden Markov models (HMMs) [44], this probability (density) is given by

$$p(A|W) = \sum_{s \in \{S_W\}} p(s_1)p(o_1|s_1) \prod_{t=2}^T p(s_t|s_{t-1})p(o_t|s_t), \quad (1.1)$$

where $A = \{o_1 \dots o_T\}$ is a sequence of observed feature vectors, with o_t being the feature vector at time t ; $s = \{s_1 \dots s_T\}$ is a sequence of states, with s_t being the state at time t ; W is a sequence of words or phones, and $\{S_W\}$ is the set of state sequences compatible with W .

A language model is an approximation to the probability of random word (or other linguistic unit) sequences. For example, with an n -gram language model [19], the probability of a word sequence $W \triangleq (w_1 \dots w_N)$ is given by

$$p(W) = \prod_{i=1}^N p(w_i|w_{1:i-1}) = \prod_{i=1}^N p(w_i|w_{i-n+1:i-1}), \quad (1.2)$$

where $w_{i:j} \triangleq (w_i \dots w_j)$.

From an engineer's perspective, a practical speech model satisfies at least two criteria: it is a fair approximation and it can be implemented with reasonable computational resources. The popularity of HMMs and n -grams is mainly due to efficiency rather than accuracy. That said, they are often good starting points from which to build an ASR system, subject to further refinements if additional resources become available.

1.1.3 Evaluation Methods

The performance level of an ASR system is often quantified by the *word error rate* (WER), which is a measure of discrepancy between the true transcription T and the recognition output R . With respect to an optimal alignment of R with T , WER is defined as

$$E = \frac{S + I + D}{N} \times 100\%, \quad (1.3)$$

Table 1.1: Word error rates of ASR systems on selected tasks.

database	style	vocabulary	environment	data size	WER
Aurora 2.0	read	11	noisy	4 hours	14.0% [41]
Phonebook	read	600	clean	5 hours	5.6% [6]
SPINE	spontaneous	6k	noisy	17 hours	28.0% [31]
Switchboard	spontaneous	65k	telephone	265 hours	19.3% [38]

where S , I and D are respectively the numbers of substituted, inserted and deleted words, and N is the total number of words in T . Equivalently, the word accuracy rate A , defined as $A = 1 - E$, can also be used. Note that a word error rate can be greater than 100% if the recognition output abounds in insertion errors.

Generally speaking, the performance measure varies from one application to another. In the circumstances where exact matches are crucial, the *sentence error rate* (SER) is often adopted. Here SER is defined as the number of incorrectly recognized sentences divided by the total number of test utterances. The count of errors is incremented unless the recognition output of a sentence is identical to the true transcription. The circumstances where SER is appropriate include recognition of telephone numbers, ID numbers or addresses, for example.

1.2 Current ASR Performance Levels

The fundamental difficulty in ASR tasks lies in the vast variation of human speech. Gender, age, tone, accent, environment, style, and speaking rate, to name a few, all contribute. The ability of humans to adjust to such variation is truly remarkable. ASR systems, however, are still far behind in terms of this capacity. To get an idea of what kind of challenges confront the current ASR community, the performance levels of selected ASR tasks are given in Table 1.1.

Most of the errors committed by ASR systems can be attributed to the following reasons. First, the speech features may be not sufficiently discriminative (i.e. well-separated in the feature space) for the distinctive linguistic classes they are representing; or they are not

sufficiently robust to variations for the same linguistic content in differing environments. This poses the challenge of *discriminative and noise-robust features*. Second, the speech models may be inaccurate because the modeling assumption is erroneous or because the model parameters cannot be reliably estimated. This leads to the research of *model refinements, adaptations*, and of *new speech models*. Lastly, search errors may occur due to inexact decoding algorithms. This occurs when the search space is large and the constraint on computational resource is stringent. This leads to the research of *fast and accurate search methods*, especially in tasks of large vocabulary and long utterances.

1.3 Overview of Noise Robustness in Speech Recognition

Noise is everywhere! At home, it is the circulation of heated air in the winter and the sound of the air-conditioner motor in the summer, the ringing of the telephone and the humming of the vacuum. Outside, the planes, the buses, the sirens, the winds and the rains. In public buildings, people's chatting, the broadcasting and the background music. People usually can "suppress" the noise and focus on meaningful signals, thus maintaining the capability of speech recognition and understanding. Machines often cannot discard the interfering noise in the signal and their performance can thus be severely degraded, even to the degree that their use becomes impractical.

Consequently, the issue of noise-robustness must be solved for ASR systems to become widely deployed, no matter how hard this problem is. For example, if an ASR system can be robust to the noise inside a bus or a train, then people will start using it to better and more safely use their time of commuting, when using hands and eyes may be inconvenient.

The surge in the usage of mobile phones also indicates the potential benefit that can be generated by a viable noise-robust technique. One apparent application is to command cell phones via voice, when using hands is not a good option.¹ Here an entire ASR system is to be embedded in a handset and the limitation in computational resources, such as memory footprint and power consumption, becomes a major issue.

¹If the hands are busy, chances are that the brain is busy as well. Using a cell phone will distract the user from his/her current focus. Noise-robustness can possibly reduce the distraction per usage but it can also possibly increase the frequency of usage.

As another application, in order to access real-time information from *anywhere* at *anytime*, the information server needs to be automated (for the “anytime” part), and the accessing device needs to be mobile (for the “anywhere” part). An ASR system, with a front end embedded in a mobile phone, if sufficiently robust, can provide a solution to this future vision and thus bring convenience to human life. As mobile phones are often used in noisy environments such as streets, restaurants, airports, and shopping malls, the embedded system certainly has to be robust to these settings. The front-end robustness in the accessing devices can greatly reduce the burden of the recognizer embedded in the information server, which may also have additional noise-robust techniques implemented.

An overview of ASR in noisy environments is depicted in Figure 1.1. Note that the difference in environments, represented by $\{h(t), n(t)\}$ and $\{h'(t), n'(t)\}$ in the graph, of the training speech samples and testing speech samples is the main reason why this recognition task is difficult. This is the problem of *data mismatch*.

The problem of noise-robustness, as it stands today, is largely unsolved. There is simply no sound and general framework for this problem to be treated systematically and thoroughly. What makes it so difficult to combat noise in ASR? This question is partially answered by reviewing past efforts of the research community, by an analysis of the distortion by noise, as well as by the vast variety in the types of noise. This dissertation aims to contribute towards moving close to a general solution of this problem.

For maximum usability, a noise-robust ASR technique shall not rely on the availability of system resources. Specifically, a technique that does not use multi-channel signals, does not require matched training data for test data, does not require prior knowledge about the noise, does not degrade the performance level on clean data, and does achieve decent performance level with unseen noisy data, with fast speed and without high computational costs, is certainly desirable. The front-end technique investigated in this thesis is one of this kind.

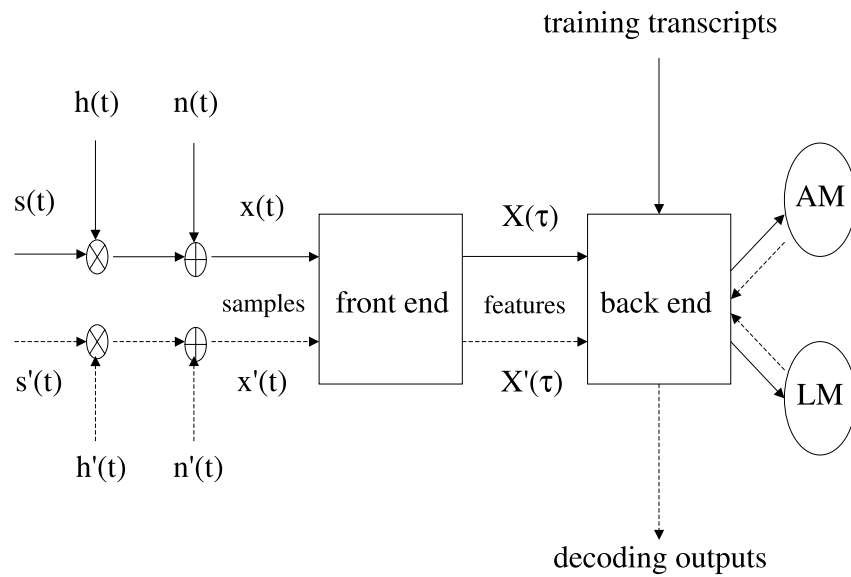


Figure 1.1: Diagram of ASR system in noisy environments. Here $s(t)$, $h(t)$, $n(t)$, and $x(t)$ represent respectively the speech signal, the convolutional noise, the additive noise, and the noisy speech signal of the training data. $X[\tau]$ is the extracted feature stream. t and τ are used to denote different sampling rates. Their counterparts of the test data are denoted by the same notation with additional prime. In the back end, “AM” is the acoustic model, and “LM” is the language model. The solid line is the flow of data when the system is being trained, while the dashed line is the flow when the system is being tested.

1.4 Thesis Overview

The goal of this thesis is to investigate novel schemes for noise-robustness in ASR systems. It starts with a comprehensive review of noise-robust techniques in Chapter 2. The review is followed by an analysis of the distortion of speech features in the presence of noise in Chapter 3. The proposed noise-robust schemes in the front end and the back end are described in Chapter 4 and Chapter 5. The experimental results on noisy speech databases are presented in Chapter 6. Finally, the conclusions are drawn in Chapter 7.

Chapter 2

REVIEW OF NOISE-ROBUSTNESS IN ASR SYSTEMS

This chapter opens with descriptions of the problems inherent in modern ASR systems, especially the acoustic mismatch. This is followed by a review of noise-robust techniques. These techniques are categorized in two main classes: in the front end or in the back end. In noise-robust front ends, one tries to extract front-end features that are robust to the corruption by noise. In other words, given a number of speech samples corresponding to the same underlying speech but produced in different environments, the extracted features from these speech samples should be similar to each other. In noise-robust back ends, one attempts to modify the speech models learned from the training speech samples in ways that compensate for the presence of different noise in the testing speech. The front-end features of clean and noisy speech need not be similar, but the difference must be captured and properly translated into model modifications. Details of these front-end/back-end techniques are further described in Sections 2.2 and 2.3, which provide background knowledge of noise-robustness research.

A specific review of techniques on recently released noisy databases is presented in Section 2.4. This review is valuable for the purposes of reference and placement of the proposed techniques in this thesis, as they are evaluated on the same databases.

2.1 Research Background and Overview

Simply put, an ASR system outputs the most likely hypothetical sentence W^* given the acoustic evidence A ,

$$W^* \triangleq \arg \max_W P(W|A) = \arg \max_W P(A|W)P(W), \quad (2.1)$$

where $P(A|W)$ is the probability given by the acoustic model and $P(W)$ is the probability given by the language model. If these models are accurate, and an admissible (i.e. free of

search errors) search algorithm is implemented, then the ASR problem is in principle solved in the sense of minimum sentence error rate.

A main issue here is that the acoustic model (as well as the language model, which will not be addressed further) learned from the training data is only an approximation for the test data. Such inaccuracy in the acoustic model for test data has to do with three somewhat separate issues. First, there may be *inaccuracy* in the assumed acoustic model. The ubiquitous hidden Markov models with state-dependent observational Gaussian (mixture) density functions belong to this category. Even asymptotically, i.e., with an unlimited amount of training data, the inaccuracy of an assumed model (with a fixed structure and number of parameters) leads to a non-zero KL-divergence [20] between the learned probability distribution and the true distribution. An account of the capacity/limitation of hidden Markov models is given in [7] and a general framework of segment modeling is given in [66]. Second, the issue of *data paucity*, leading to the result that the learned model is different from the asymptotic model, further compromises the accuracy of the acoustic model. Although parameter-tying schemes for robust estimation of model parameters [65, 81] can alleviate the problem of data paucity, the learned model is still not ideal. Lastly, the issue of *mismatch* makes the acoustic model of the test data different from that of the training data and thus can simply invalidate the model learned from training data.

The main difficulty in dealing with noise in ASR systems is the aforementioned acoustic mismatch, as the presence of noise corrupts speech samples and distorts speech features. To systematically counter the corruption and distortion in the presence of environmental noise has been the focus of research on noise robustness. In the case of corruption in speech samples, techniques such as the Kalman filter [67] and microphone arrays [77] can be used, in certain senses of optimality, to recover the clean speech samples. In the case of distortion of speech features, either sets of noise-robust features have been proposed or enhancement techniques have been adopted to restore the clean speech features. The former includes such features as RASTA-PLP [39], visual features [69] and cross-correlation features [9], while the latter includes such techniques as spectral subtraction [14], cepstral mean normalization [3, 30] and norm equalization [46]. The distortion in speech features can

be translated into deformation of the acoustic space, to which the models can be accordingly adjusted. Techniques stemming from such considerations to bridge the gap between training and testing acoustics include stochastic matching [70], parallel model combination [78, 33], and model adaptation [32]. Still other ways to achieve noise robustness can be generally called combination of multiple (complementary) information sources, where multiple feature streams are incorporated in the ASR systems. Such combination can be done in feature extraction [69], lattice re-scoring [27], or output generation [28, 60]. On small-vocabulary ASR tasks, feature combination (via discriminatively trained neural networks) has been experimented [26], while on medium- to large-vocabulary ASR tasks, combining hypotheses has been found to be effective [31, 74, 49], with noisy speech databases.

Even for small-vocabulary ASR tasks, complicated systems have been attempted to achieve good performance. For example, in [26], a front end consisting of principle component analysis and a discriminative neural network applied to two types of speech features and a back end consisting of standard Gaussian mixture acoustic models are used. In [4], missing-data theory is used in identifying reliable features in the spectral-temporal domain. In [22], voice activity detector and variable frame rate techniques are used to drop noisy feature vectors to reduce the insertion errors. Such systems are computationally intensive as they often introduce additional parameters either in the feature extraction or in the speech models, and these parameters need to be learned from data samples. A fundamental issue here is the risk of over-training [13], which occurs when spurious model parameters are assumed in the model. Fine-tuning these parameters to match a particular set of data may result in performance degradation when used with unseen data.

2.2 Noise-Robust Front Ends

An ideal noise-robust front end extracts the same feature vectors irrespective of the environmental noises while preserving the information carried in the speech. A noise-robust technique in the front end can be applied at any stage, from raw speech samples to raw speech features, in the process of the audio signal processing. In this section, one technique restoring clean speech samples (Section 2.2.1), one technique modifying speech spectra

(Section 2.2.2), and two techniques generating noise-robust speech features (Sections 2.2.3 and 2.2.4) will be described.

2.2.1 Kalman Filters

A Kalman filter estimates the unknown clean signal from the observed noisy signal based on assumed stochastic models for signal and noise. In speech research, this filter is mainly used as an enhancement method for noisy speech. In [67], as an explicit example, the Kalman filtering method is studied and is shown to be significantly better than an alternative Wiener filtering method, in terms of the respective output SNRs of the processed signal by these filtering methods. Note that the minimum mean squared error criterion does not necessarily translate into the optimal quality for human perception (intelligibility) or the best accuracy for machine recognition.

The following formulation is excerpted from [55]. Let s_t be the clean signal at time t , then a first-order auto-regressive (AR) model for $\{s_t\}$ is

$$s_t = a_{t-1}s_{t-1} + u_{t-1}, \quad (2.2)$$

where $\{a_t\}$ is known and $\{u_t\}$ is a zero-mean uncorrelated sequence of random variables. Let x_t be the noisy signal obtained by corrupting s_t with additive noise n_t , which is also zero-mean and uncorrelated,

$$x_t = s_t + n_t. \quad (2.3)$$

and further assume that n_t is uncorrelated with u_t . It can be shown that the MMSE (minimum-mean-square-error) linear predictor of s_{t+1} , with $\{x_t|t = 0, \dots, t\}$ observed, is given by

$$\begin{aligned} \hat{s}_0 &= 0, \\ \hat{s}_{t+1} &= a_t \hat{s}_t + k_t(x_t - \hat{s}_t), \end{aligned} \quad (2.4)$$

where \hat{s}_{t+1} is the predictor of s_{t+1} and k_t is the Kalman gain which can be obtained by the following recursion:

$$\begin{aligned} E[\epsilon_0^2] &= E[s_0^2], \\ k_t &= \frac{a_t E[\epsilon_t^2]}{E[\epsilon_t^2] + E[n_t^2]}, \\ E[\epsilon_{t+1}^2] &= a_t(a_t - k_t)E[\epsilon_t^2] + E[u_t^2], \end{aligned} \quad (2.5)$$

where $\epsilon_t \triangleq s_t - \hat{s}_t$ is the prediction error. Note that this recursion requires a_t and the second moments of n_t and u_t .

The speech signal cannot be well approximated by a first-order AR process (2.2). This is not a handicap, as the above formulation can be extended to the appropriate orders for the signal in matrix forms. Rather, the difficulty is that the parameters $\{a_t\}$ are unknown and have to be estimated jointly with $\{s_t\}$.

2.2.2 Spectral Subtraction

Spectral subtraction [14] attempts to estimate the spectrum¹ of the additive noise and then subtract it from the spectrum of the noisy speech. Specifically, let $\{s_t\}$ be the speech signal, $\{n_t\}$ be the additive noise signal, $\{x_t\}$ be the noisy speech signal, and let $S^{(\tau)}[k], N^{(\tau)}[k]$ and $X^{(\tau)}[k]$ be the τ -th frame spectrum of $\{s_t\}, \{n_t\}$ and $\{x_t\}$, respectively. Then

$$x_t = s_t + n_t, \quad (2.6)$$

and

$$X^{(\tau)}[k] = S^{(\tau)}[k] + N^{(\tau)}[k], \quad (2.7)$$

or equivalently,

$$S^{(\tau)}[k] = X^{(\tau)}[k] - N^{(\tau)}[k]. \quad (2.8)$$

By (2.8), the speech spectrum $S^{(\tau)}[k]$ can be computed from $X^{(\tau)}[k]$ and $N^{(\tau)}[k]$, which can be estimated from speech and non-speech segments respectively.

The effectiveness of spectral subtraction depends on a reliable estimate of the noise spectrum, which in turn depends on the system's ability to distinguish between speech and non-speech segments. This method becomes more involved when the noise is highly non-stationary.

2.2.3 RASTA

The RelAtive SpecTrA (RASTA) filtering [39] is a technique applied to compressed critical-band power spectral envelopes. It is designed to remove the slow-varying environmental

¹The spectrum is the distribution of energy along the frequency axis. It is basically the square of the magnitude of a signal's Fourier transform.

variations and the fast-varying artifacts in speech processing. More specifically, the RASTA features are extracted through the following steps:

1. Compute the critical-band power spectrum².
2. For each band, transform the amplitude through a compressive mapping, e.g., taking the logarithm of base e .
3. Apply the RASTA filtering to the time sequence of each band.
4. Apply the inverse transform of the previous compressive transform.
5. Compute the linear predictive coefficients.

Which compressive (and the corresponding expansive) transform to use is a design choice and can depend on the type of noise. Incidentally, the RASTA filter, the z -transform of which is

$$H(z) = 0.1z^4 \frac{2 + z^{-1} - z^{-3} - 2z^{-4}}{1 - 0.98z^{-1}}, \quad (2.9)$$

has a frequency response resembling the characteristic curve of human sensitivity to frequency modulation [36].

2.2.4 Correlation Features

Speech, as a special set of audio patterns evolved for human communication, is different from other audio signals, such as bird chirps. Therefore, designing features based on the idiosyncratic statistics of human speech is potentially helpful to discriminate speech from other audio signals. The simplest statistics are the mean (first-order) and the correlation (second-order), so correlation features are among the simplest methods to exploit in the attempt to achieve this discrimination. Indeed, since the power spectrum is the Fourier

²The critical bands are discovered by Fletcher [29], who found that for a tone of a given frequency, the noise outside the critical band does not affect the perception of the tone by the listener. The critical-band power spectrum is obtained by convolving the power spectrum with the critical band filters.

transform of the auto-correlation sequence, the spectral-based features do utilize information in speech auto-correlation.

The ModCrossGram features, which are the short-time cross-correlation statistics between the spectral envelopes of different channels, extends the usage of signal statistics to cross-correlation. These features are extracted as follows [10]:

1. The cross correlation sequences between channel pairs are computed from the critical-band spectral envelopes with respect to a correlation window (of frames) and up to a maximum lag

$$R^{(\tau)l}[i, j] = \sum_{k=0}^{N-1} X^{(\tau+k)}[i]X^{(\tau+l+k)}[j]w_k, \quad l = 0, \dots, l_{max}, \quad (2.10)$$

where $X^{(\tau)}[i]$ is the i -th critical-band spectral magnitude at frame τ , w_k 's are windowing coefficients, N is the number of frames in a correlation window, and l_{max} is the maximum correlation lag.

2. For each channel pair, a linear least-square slope estimator is fitted to the lag index l , reducing $R^{(\tau)l}[i, j]$ to $r^{(\tau)}[i, j]$. Then a two-dimensional discrete cosine transform is applied to $r^{(\tau)}[i, j]$ and the low-order coefficients are retained as features.

Alternatively, the dimensionality reduction can be carried out by information-theoretic considerations. Here one calculates from data the mutual information between all pairs of original feature components and then uses only those pairs with top-ranking mutual information. That is,

$$\{(i_k, j_k, l_k), k = 1 \dots K\} = \arg K \max_{i, j, l} I(X^{(\tau)}[i]; X^{(\tau+l)}[j]),$$

where $\arg K \max_x F(x)$ denotes the K arguments with the K largest values in $F(x)$. The corresponding $R^{(\tau)l_k}[i_k, j_k]$ as defined in (2.10) can then be used as features. Redundancy in mutual information can be further taken into account in the selection scheme [10].

2.3 Noise-Robust Back Ends

Instead of trying to counter directly the distortion of speech features, a noise-robust back end attempts to change the speech model or modify the model parameters to compensate

for the presence of noise.

2.3.1 Parallel Model Combination

The technique of parallel model combination [78] essentially combines the speech and the noise models to account for observed features. That is,

$$\text{Observation Probability} = p(\text{observation} | M_S \otimes M_N), \quad (2.11)$$

where the subscripts S, N refer to the speech and the noise, and $M_S \otimes M_N$ represents symbolically the combined model for the noisy speech. The joint emitting density function, given the speech state j and the noise state v , is the convolution of the corresponding emitting density functions of j and v ,

$$p_{SN}(x_t | j, v) = \int_{-\infty}^{\infty} p_S(x | j) p_N(x_t - x | v) dx, \quad (2.12)$$

assuming that speech and noise are independent. To decode an observation feature sequence, the Viterbi algorithm is used to search for the optimal joint state sequence,

$$\phi_t(j, v) = \min_{i, u} \{ \phi_{t-1}(i, u) - \log [p_S(j | i) p_N(v | u) p_{SN}(x_t | j, v)] \}, \quad (2.13)$$

which can be further compacted to a speech state sequence.

Although parallel model combination is quite general in principle, in practice its effectiveness may not be optimal for the following reasons: First, the joint emitting density function (2.12) often does not have a closed form, so some sort of approximation has to be introduced. Secondly, the computational complexity in (2.13) is proportional to C^2 , where C is the cardinality of the joint state space. Since

$$C = C_S C_N, \quad (2.14)$$

the number of states in the noise model C_N is often designed to be small, limiting the modeling accuracy. Lastly, the model learned with training data is not likely to be useful for test data from a mismatched environment.

2.3.2 Model Adaptation

Model adaptation uses adaptation data to modify model parameters for a specific speaker or environment. A parametric transform is often used for fast and robust adaptation. Adaptation based on maximum likelihood linear regression (MLLR) and maximum a posteriori (MAP) criteria will be outlined next.

2.3.2.1 Maximum Likelihood Linear Regression

In MLLR [54] model adaptation, the model parameters are adapted via a linear transform in the parameter space. Adaptation can be supervised or unsupervised, with correspondingly labeled (transcribed) or unlabeled adaptation data, say D_a . The adaptation often starts with the model learned from the training data and then runs for a few epochs of iteration. The linear transform is determined by the auxiliary equation of the EM algorithm (A.1),

$$U_{MLLR} = \arg \max_U E[\log p(S, D_a | U\theta)], \quad (2.15)$$

where U is a matrix, S is the set of hidden variables and θ is the current model parameters. In the cases where the adaptation data are limited, the model parameters can be clustered into linear regression classes such that all parameters in the same class are transformed by the same adaptation matrix.

2.3.2.2 Maximum A Posteriori

In MAP model adaptation [34], the model parameters are treated as random variables with prior probability functions. Given adaptation data, the model is updated by maximizing the posterior probability,

$$\theta_{MAP} = \arg \max_{\theta} p(\theta | D_a) = \arg \max_{\theta} p(\theta, D_a) = \arg \max_{\theta} p(D_a | \theta) p(\theta). \quad (2.16)$$

When knowledge about θ is available, e.g., learned from the training data, it can be incorporated into the prior probability $p(\theta)$. Conjugate priors, which give rise to posteriors with the same parametric form (as the priors) [13], are often used for $p(\theta)$ to make the computation tractable.

2.4 Techniques on Recent Noisy Speech Databases

Thanks to the release of new noisy speech databases, many new techniques have recently been designed and implemented. These databases provide excellent platforms for studies of noise-robust techniques. Furthermore, special sessions in ASR conferences and workshops have been devoted to research on these new databases. In this section, selected ASR systems with novel techniques and great performances on these databases will be reviewed. As the proposed noise-robust techniques will be evaluated with these databases, this review will provide the background knowledge for this research.

2.4.1 Aurora 2.0

Aurora 2.0 [41] is a noisy speech database that consists of continuous utterances of digit strings. It has a very small vocabulary, and a typical utterance lasts no more than 3 seconds. The challenge of Aurora 2.0 is that the corrupting noises can be intense and mismatched. Specifically, clean speech is corrupted by being added to recorded noise at signal-to-noise ratios from 20 dB (almost clean) to -5 dB (almost all-noise), and may be further corrupted by convolutional noises. The recognition tasks include both matched and mismatched tasks. The multi-condition training set consists of both clean and noisy (5 – 20 dB) speech, while the clean training set consists of only clean speech. Test set A is composed of speech corrupted by the same additive noises as those used in the multi-condition training set; test set B is composed of speech with non-matched additive noises; and test set C is composed of speech corrupted by partially matched additive noises and by non-matched convolutional noises.

The performance of the baseline ASR system [41], which contains no specific noise-robust techniques, is duplicated in Table 2.1. Clearly, the performance degrades quickly as the noise level increases and as the environmental mismatch is introduced.

A special session on Aurora 2.0 was held in the September 2001 Eurospeech conference³ to compare noise-robust techniques from different research sites. Selected ASR systems with

³Two years later, in 2003, another evaluation on noisy Aurora databases was held in Eurospeech again, but the focus has shifted to Aurora 4.0, a large-vocabulary task.

Table 2.1: Word Accuracies (as percentages) of the defined baseline ASR system on Aurora 2.0.

	Multi-Train Tasks			Clean-Train Tasks		
	clean	0-20 dB	-5 dB	clean	0-20 dB	-5 dB
Test set A	98.5	87.8	24.6	99.0	61.3	7.9
Test set B	98.5	86.3	25.9	99.0	55.7	7.7
Test set C	98.5	83.8	21.6	99.1	66.1	11.5

Table 2.2: Word error rates of selected ASR systems on Aurora 2.0 averaged over 0 – 20 dB SNR test data.

	multi-train	clean-train
baseline system	13.6%	40.0%
tandem acoustic models	6.9%	N/A
feature vector selection	10.0%	16.3%
missing data technique	N/A	14.0%
spectral-based discriminative feature	10.0%	15.8%
MFCC-MVA features	8.0%	16.4%

noise-robust techniques are reviewed in the following sections. First, however, the word error rates of these systems are summarized in Table 2.2⁴ in order to show the performance level of systems with this database.

2.4.1.1 Tandem Acoustic Modeling

In tandem acoustic modeling [26], PLP-cepstral features are first processed by a 3-layer neural network. This neural network has an input layer of 39×9 units corresponding to 9 consecutive feature frames, a hidden layer of 480 units and an output layer of 24 units corresponding to 24 phone classes. It is trained with posterior phone-class probabilities as

⁴Note that we have included the performance of the MVA features, which will be described in Chapter 4.

targets. The 24 linear activations at the output units of the trained network are extracted as speech features. These features are further multiplied by a full-rank matrix derived by the principle component analysis [25]. Finally, the transformed features are used in a wholeword-HMM back end for training and recognition.

With the above features alone, the average WERs are 7.7%, 11.1% and 10.0% on 0 – 20 dB data of test set A, B and C respectively with the multi-train tasks. When the neural network based on PLP is combined with a similar neural network based on the modulation spectrogram, the average WERs are reduced to 6.2%, 7.9%, and 6.3% respectively [26].

2.4.1.2 Feature Vector Selection

Feature vector selection (FVS) [22] discriminates speech frames from non-speech frames. It drops the latter to reduce potential insertion errors.

One method to detect the speech/non-speech segments in a data stream is the voice activity detector (VAD). In [22], a frame is labeled non-speech and dropped if the SNR estimate is below a pre-defined threshold. Here the SNR estimate is determined by the difference between the current log energy and the long-term log energy. Another method to discard the non-speech frames is called the variable frame rate (VFR): a frame is discarded if the Euclidean distance between the delta-cepstral sub-vectors of the current frame and the previous frame is below an optimized threshold. In this scheme, the segments with only stationary noise are more likely to be discarded than frames with speech.

The WERs with VAD-based FVS are 10.0% for the multi-train and 16.3% for the clean-train on 0 – 20 dB test data. For VFR-based FVS, the WERs are 10.9% and 17.6%.

2.4.1.3 Missing Data Techniques

The technique of missing data theory (MDT) identifies reliable data in noise-corrupted data. The reliable data is treated as evidence while the rest is treated as missing (or hidden) data. The relationship between evidence and missing data is used to integrate away the missing data in order to marginalize the evidence probability for classification.

When speech is one of the sound sources, there are spectral-temporal regions (“evi-

dence”) that remain uncorrupted by noises and are consequently reliable for recognition. Spectral-temporal regions dominated by a single source can be grouped by patterns created by the dominant sound-producing process. One such pattern is harmonicity, that is, the energy of *voiced* speech will be organized around the harmonics of the fundamental frequency. Therefore, the existence of harmonic groups gives evidence for a dominant sound source [4]. When identification of harmonic groups fails, as in *unvoiced* speech or noise, a local SNR estimate can be a substitute labeling tool.

Combined with gender-dependent models (where the acoustic models for male and female speakers are distinct), the MDT-based systems achieve an average WER of 14.0% with clean-train tasks over 0 – 20 dB test data [4].

2.4.1.4 Spectral-Based Discriminant Features

The system in [5] integrates several techniques which independently achieve noise robustness. Specifically, it includes the following modules:

- data-driven temporal filter: A 101-point feature vector from the 7th mel band is used for linear discriminant analysis (LDA) [25]. After LDA, the leading discriminant vector is truncated to a 51-point causal vector and is used as the temporal filter for all mel bands.
- neural network VAD: A multi-layer neural network VAD is trained to discriminate speech and non-speech frames. It has an input layer of 54 units from 9 consecutive frames, a hidden layer of 50 units, and an output layer of 2 units corresponding to the silence class and the speech class.
- on-line mean and variance normalization: At frame t , the mean and variance estimates are updated if the current frame is not silent,

$$\begin{aligned}\mu_t &= \mu_{t-1} - \alpha(x_t - \mu_{t-1}) \\ \sigma_t^2 &= \sigma_{t-1}^2 - \alpha((x_t - \mu_t)^2 - \sigma_{t-1}^2),\end{aligned}$$

where x_t is the feature, μ_t and σ_t^2 are the mean and variance estimates. The normalized feature is simply

$$x'_t = \frac{x_t - \mu_t}{\sigma_t + \theta},$$

where $0 < \alpha < 1$ and $\theta \geq 0$ are empirically determined parameters.

- nonlinear discriminative transform: This is the same as with tandem acoustic modeling as outlined in Section 2.4.1.1.

The average WERs with this technique are 10.0% in the multi-train case and 15.8% in the clean-train case, over 0 – 20 dB test data.

2.4.2 Aurora 3.0

The Aurora 3.0 database [56, 76, 64, 58] consists of four languages: Danish, Finnish, German and Spanish. Utterances consisting of digit strings are recorded in cars using close-talking (CT) and hands-free (HF) microphones. Recordings occur in quiet, low noise, and high noise conditions. For each language, there are three tasks. The well-matched (WM) task is to train on 70% of both CT and HF recordings from all conditions and test on the remaining 30%. The medium-mismatched (MM) task is to train on HF recordings from quiet and low-noise conditions and test on HF recordings from high-noise condition. The highly mismatched (HM) task is to train on 70% of CT recordings from all conditions and test on 30% of the HF recordings from low and high noise conditions.

Like Aurora 2.0, Aurora 3.0 is a small-vocabulary database. Unlike Aurora 2.0, the noises in Aurora 3.0 are real-time and vary constantly over the course of the recordings. In addition, using HF microphones also introduces severe convolutional distortion.

A special session to evaluate novel noise-robust techniques on Aurora 3.0 database was held in the 2002 International Conference on Spoken Language Processing (ICSLP). Again, selected ASR systems will be reviewed in the following sections and respective performances are summarized in Table 2.3. ⁵

⁵Again we have included the relative improvement of MVA features. Note that the performance of MVA is remarkably good, regardless of the fact that only one single technique is implemented in the system.

Table 2.3: Relative improvements in word error rates over the baseline system of the selected ASR systems on Aurora 3.0.

	relative improvement
baseline system	0.0%
Gabor features	57.4%
Splice	47.2%
auditory features	38.4%
histogram equalization	30.5%
MVA features	47.7%

2.4.2.1 Gabor Features

In [52], Gabor features are derived from the two-dimensional (time and frequency) Gabor functions defined as

$$g(t, f) \triangleq n(t, f) e(t, f), \quad (2.17)$$

where

$$n(t, f) \triangleq \frac{1}{2\pi\sigma_f\sigma_t} \exp \left[\frac{-(f - f_0)^2}{\sigma_f^2} + \frac{-(t - t_0)^2}{\sigma_t^2} \right] \quad (2.18)$$

is Gaussian, and

$$e(t, f) \triangleq \exp [i\omega_f(f - f_0) + i\omega_t(t - t_0)] \quad (2.19)$$

is sinusoidal. The parameters of the Gabor functions are $\sigma_t, \sigma_f, t_0, f_0$, and ω_t, ω_f . Respectively, (t_0, f_0) parameterize the location, (σ_t, σ_f) parameterize the spread, and (ω_t, ω_f) parameterize the angular frequency of the Gabor functions. The product of a given Gabor function and the two-dimensional representation of speech produces a Gabor feature.

The Gabor functions used in [51] are further constrained by

$$\omega_t \sigma_t = \pi, \quad \omega_f \sigma_f = \pi \quad (2.20)$$

and

$$n(t, f) = 0, \text{ if } t \notin [-2\sigma_t, 2\sigma_t] \text{ or } f \notin [-2\sigma_f, 2\sigma_f], \quad (2.21)$$

so that each Gabor function is localized and has a fixed number of oscillations. Note that when σ_t vanishes, ω_t is not constrained by (2.20) and the corresponding Gabor feature is a purely spectral-domain processing on the original two-dimensional representation. Likewise, when $\sigma_f = 0$, the corresponding Gabor feature is purely temporal. Consequently, as special cases of the Gabor features correspond to extracting pure temporal information and pure spectral information from the two-dimensional representation of speech, this is arguably more general than methods that are based on strictly spectral cues or temporal cues. In practice, the set of Gabor features has been determined by a grid search over ω_t and ω_f [51].

On Aurora 3.0, an ASR system combining Gabor features and Qualcomm-OGI-ICSI features and using noise reduction and frame-dropping techniques yields a performance improvement of 57.4% averaged over all tasks and languages [52].

2.4.2.2 Splice

The Splice algorithm [24] learns a joint model of clean and noisy cepstral vectors from two channels of training data, clean and noisy. Specifically, let x and y be the clean and the noisy cepstral vectors and let s represent an index to the environment where the speech takes place. The model assumes that y is normally distributed given s , and that x is normal-distributed given s and y ,

$$\begin{aligned} p(y|s) &= N(y; \mu_s, \Sigma_s), \\ p(x|y, s) &= N(x; y + r_s, \Gamma_s). \end{aligned} \tag{2.22}$$

It follows that the joint probability density of (x, y) is given by

$$p(x, y) = \sum_s p(s, x, y) = \sum_s p(s)p(y|s)p(x|y, s). \tag{2.23}$$

The model parameters $\{\mu_s, \Sigma_s, r_s, \Gamma_s\}$ and $\{p(s)\}$ can be learned from training data.

The MMSE estimate of x from y is given by

$$\hat{x}(y) = \arg \min_{z(y)} E(x - z(y))^2 = E[x|y] = E[E[x|y, s]|y] = y + \sum_s p(s|y)r_s, \tag{2.24}$$

where $p(s|y) = \frac{p(y|s)p(s)}{\sum_{s'} p(y|s')p(s')}$.

The original splice algorithm depends on the availability of stereo (two-channel) data, which can be expensive to collect. A more general framework which learns the bias (r_s) via ML criterion and thus does not require stereo data during training has been reported to achieve better results with the Aurora 2.0 database [79].

When combined with the time-domain smoothing, blind equalization, and ideal end-pointing techniques, Splice yields an overall improvement of 47.2% over the baseline system on Aurora 3.0 [24].

2.4.2.3 Discriminatively Trained Auditory Features

The extraction of discriminative auditory features [59] is similar to the MFCCs, with the exception that the coefficients of the binning filters centered at mel-frequencies are discriminatively trained rather than fixed. These so-called auditory filters are equally spaced in the Bark-scale. The shapes of these filters are constrained to be smooth and bell-like. Specifically, the normalized filter coefficients $\{w_i; i = -L, \dots, 0, \dots, L\}$ are required to satisfy

$$w_i = \begin{cases} 1 - F(\sum_{j=1}^i H(\delta_j)), & i > 0, \\ 1, & i = 0, \\ 1 - F(\sum_{j=i}^{-1} H(\delta_j)), & i < 0, \end{cases} \quad (2.25)$$

where $H(x)$ is the exponential function, $F(x)$ is the sigmoid function, and δ_i is the difference between adjacent filter coefficients,

$$\delta_i = \begin{cases} w_i - w_{i-1}, & i > 0, \\ w_{i+1} - w_i, & i < 0. \end{cases}$$

The training of the model parameters now includes learning these normalized filter coefficients. The discriminative training uses the minimum classification error criterion in parameter estimation [45].

With Aurora 3.0, this technique yields an overall improvement of 38.4%.

2.4.2.4 Histogram Equalization

The histogram equalization [71] works as follows. Let x and y respectively represent the observations of two different conditions A and B . Based on the histograms of the data, a data point x in A is mapped to the point y in B with the same percentile. That is,

$$x(y) = C_X^{-1}(C_Y(y)), \quad (2.26)$$

where $C_Z(\cdot)$ denotes the cumulative histogram of samples of random variable Z . With this mapping, the models trained by data x can be used for data y .

This technique is applied in the cepstral domain, yielding an overall improvement of 30.5% with Aurora 3.0 [71] when it is combined with spectral subtraction and frame-dropping techniques.

2.4.2.5 Eigenspace Normalization

The eigenspace of a matrix is spanned by its principle axes. In [80], the cepstral feature vectors are represented in the eigenspace of the sample covariance matrix. In this space, the mean subtraction and variance normalization are applied to the data. That is, along each principle axis the data has zero mean and unit variance. Finally, the normalized vectors in the eigenspace are inversely transformed to the original space for the ASR back end.

This method is evaluated with the Aurora 3.0 database. It shows an overall improvement of 6.48% relative to the *advanced* front-end baseline, when combined with utterance-level frame dropping [80].

2.5 Discussion

ASR systems based on multiple techniques are very common for the purpose of noise-robustness. That is, novel techniques based on simple and sound ideas are often combined with other “standard” techniques to achieve better performance. Such combination is often justified by the performance gain rather than by theoretical considerations. It can, however, become difficult to compare different novel techniques when they are combined with different subsidiary techniques. In addition, efforts to tune the parameters in ASR systems are critical

for optimal performances. Differences in tuning criteria and data can lead to significant performance variation and thus further increase the difficulty of comparison. Note that these tuning details have been deliberately left out here to avoid obscuring the main ideas. Those who are interested and ready to delve into technicality are encouraged to go directly to the cited references.

From the Tables 2.2 and 2.3, it is noted that the MVA does not yield the best performance in the tables but it ranks *second*. However, in many, if not all, of the systems, the performance gain can not be attributed solely to the novel techniques, but is rather a combination of multiple techniques that requires significant development efforts to tune certain operational parameters. In contrast, the MVA is unique in the sense that it can be applied out of the box. In other words, the potential gain in combining other techniques have not been fully exploited. In this thesis, this technique is deliberately considered on its own to maintain its strength in simplicity and effectiveness.

Chapter 3

ANALYSIS OF FEATURE DISTORTION

As is pointed out in the preceding chapter, many modern noise-robust techniques for ASR systems are justified on practical rather than theoretical grounds. While it is important to know how a given technique works, it is even more important to understand why it does. Therefore, an analysis that motivates the investigated techniques in this thesis will be presented in this chapter. Specifically, the analysis is on the feature set of MFCCs and the log energy in the presence of additive and convolutional noise at different noise levels. The feature set and noise will first be defined, followed by an analysis of feature corruption by noise.¹

3.1 Common Types of Noise

Commonly encountered noise can be categorized into the additive noise and the convolutional noise. By definition, additive noise affects the speech signal by an additive term

$$x(t) = s(t) + n(t), \quad (3.1)$$

where $\{s(t)\}$ is the clean speech, $\{n(t)\}$ is the additive noise, and $\{x(t)\}$ is the noisy speech. Convolutional noise, on the other hand, affects the speech signal by a convolution

$$x(t) = s(t) * h(t) \triangleq \sum_{\tau} s(\tau)h(t - \tau), \quad (3.2)$$

where $\{h(t)\}$ is the impulse response of a given environment. Here the environment is assumed to be time-invariant, but note that otherwise $x(t)$ is simply the sum of impulse responses at each instant weighted by the speech signal.

¹This analysis of feature distortion here and MVA processing in the next chapter has appeared in a submission to the IEEE Transactions on Speech and Audio Processing. The manuscript is currently accepted pending mandatory revisions.

In a real situation, both types of noise are present and therefore the noisy speech signal includes both terms

$$x(t) = s(t) * h(t) + n(t). \quad (3.3)$$

Equation (3.3) can be regarded as a special subset of the general distortion, which maps the clean speech into noisy speech,

$$x(t) = \mathcal{F}(s(t)), \quad (3.4)$$

where \mathcal{F} is generally a non-linear mapping. However, this subset, of additive and convolutional noises, approximates quite accurately typical noise.

Another type of distortion happens not during the *transmission* of source samples but rather during the *production* of source samples. For example, under noisy environments, the Lombard effect [48] on articulation, such as elongating the durations of vowels and tilting the spectrum toward the high-frequency end, alters the acoustics. That is, the realization of speech by a speaker under a noisy condition, $\{\tilde{s}(t)\}$, is a distorted version of the realization of speech by the same speaker in a clean environment, $\{s(t)\}$. Note that the Lombard effect cannot be approximated accurately by (3.3). Note also that the presence of noise can have a subtle influence not only acoustically but also linguistically, as the noise can break the speaker's concentration.

3.2 Definition of the Used Features

For completeness, the definition of the adopted speech feature set is given next. Including the basic feature derivation provides a background review, familiarizes the reader with notation, and supplies a handy reference for later analysis.

An MFCC feature vector at a given speech frame, $C \triangleq (C[1] \dots C[D])^T$, is defined by

$$C = G \log_e Q, \quad (3.5)$$

where D is the number of cepstral coefficients, $Q \triangleq (Q[1] \dots Q[J])^T$ is the mel-frequency power spectrum obtained by summing over the power spectrum scaled by mel-frequency

filter banks (also known as mel-binning), and G is the $I \times J$ matrix representing the discrete cosine transform

$$G_{ij} = \sqrt{\frac{2}{J}} \cos\left(\frac{\pi i}{J}(j - 0.5)\right), \quad i = 1 \dots I, \quad j = 1 \dots J. \quad (3.6)$$

MFCC features are often enhanced with the zeroth cepstral coefficient $C[0]$, defined by

$$C[0] = \sum_{j=1}^J \sqrt{\frac{2}{J}} \log_e Q[j], \quad (3.7)$$

or the log energy ξ , defined by

$$\xi = \log_e \left(\sum_{n=0}^{N-1} x[n]^2 \right), \quad (3.8)$$

where N is the number of speech samples in an analysis window (including the zero-padding).

3.3 Feature Distortion in the Presence of Noise

Next, the distortion of the MFCCs and the log energy feature under noisy environments will be expressed in terms of the clean speech signal and the noise signal. This analysis will eventually lend a theoretical justification to the investigated techniques of this thesis, which will be introduced in Chapter 4.

3.3.1 Distortion by Convolutional Noises

In this section, the distortion of speech features caused by a convolutional noise is analyzed from scratch. This analysis is similar to an argument given in [3], that the (linear-prediction) cepstral mean subtraction leads to parameters that remain invariant in the presence of time-invariant convolutional noise. An analysis, based on a vector Taylor series approach, of the cepstral distortion due to additive noise and convolutional noise has been given in [63]. The analysis given here is detailed, and special caution is taken in the operations of binning and taking logarithms. As this analysis keeps track of the distortion in each domain, it is valuable and informative in appraising a potential signal processing technique.

In the presence of convolutional noise, recalling that a convolution in the time domain is equivalent to a multiplication in the frequency domain, the power spectra of $\{x(t)\}$, $\{s(t)\}$ and $\{h(t)\}$ are related by

$$P_x[k] = P_s[k]P_h[k], \quad (3.9)$$

where $P_x[k] = |X[k]^2|$, with $X[k]$ being the discrete Fourier transform of $x[n]$. Here the subscript indicates the source that alone produces the corresponding term. Following (3.5), the i -th cepstral coefficient of x is

$$C_x[i] = \sum_{j=1}^J G_{ij} \log_e \left(\sum_{k=0}^{N-1} F_{jk} P_x[k] \right), \quad (3.10)$$

where F_{jk} is the gain of the j -th mel-frequency filter at the k -th (Fourier) frequency. Note that for each j , F_{jk} is zero for the majority of k 's. In general, C_x and C_s are not simply related in terms of h . However, if it is assumed that P_h is flat within the passband of each mel-frequency filter, then the summation inside the parenthesis in (3.10) can be approximated by

$$\begin{aligned} \sum_{k=0}^{N-1} F_{jk} P_x[k] &= \sum_{k=0}^{N-1} F_{jk} P_s[k] P_h[k] \\ &\approx P_h[k_j] \sum_{k=0}^{N-1} F_{jk} P_s[k], \end{aligned} \quad (3.11)$$

where $P_h[k_j]$ is a representative energy level of $\{h(t)\}$ in the passband of filter j . Note that this assumption does not rule out a possible large variation of P_h among different bands of mel-frequency filters, but only requires that within each band the variation be insignificant. This assumption tends to be true in the passband of a well-designed transmission equipment but is unlikely to be justifiable in the case of reverberation, since multiple paths/reflections from source to receiver render the frequency response to have peaks and valleys [50].²

²In Aurora 2.0, the two types of convolutional noise [41] have smooth frequency response.

With the approximation of (3.11), (3.10) can be simplified by

$$\begin{aligned}
C_x[i] &= \sum_{j=1}^J G_{ij} \log_e \left(P_h[k_j] \sum_{k=0}^{N-1} F_{jk} P_s[k] \right) \\
&= \sum_{j=1}^J G_{ij} \left(\log_e P_h[k_j] + \log_e \left[\sum_{k=0}^{N-1} F_{jk} P_s[k] \right] \right) \\
&= \sum_{j=1}^J G_{ij} (\log_e P_h[k_j] + \log_e Q_s[j]) \\
&= B_h[i] + C_s[i],
\end{aligned} \tag{3.12}$$

where $B_h[i] \triangleq \sum_{j=1}^J G_{ij} \log_e(P_h[k_j])$. The noisy feature of $C[0]$ can be similarly simplified by

$$\begin{aligned}
C_x[0] &= \sum_{j=1}^J \sqrt{\frac{2}{J}} (\log_e P_h[k_j] + \log_e Q_s[j]) \\
&= B_h[0] + C_s[0].
\end{aligned} \tag{3.13}$$

Therefore the difference between the i -th (0-th included) MFCC feature of the noisy and the clean speech, $B_h[i]$, does *not* depend on the speech $\{s(t)\}$ but only on the noise $\{h(t)\}$. Clearly, if the convolution is fixed, then the bias added to speech features is also constant and thus removable. This conclusion supports the investigated method of mean subtraction in Section 4.2.2.

On the other hand, the distorted log energy feature of ξ becomes

$$\begin{aligned}
\xi_x &= \log_e \left(\sum_{n=0}^{N-1} x[n]^2 \right) \approx \log_e \left(\frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2 \right) \\
&= \log_e \frac{1}{N} + \log_e \left(\sum_{k=0}^{N-1} P_s[k] P_h[k] \right),
\end{aligned} \tag{3.14}$$

where Parseval's theorem³ is applied and “ \approx ” indicates possible approximations due to techniques such as pre-emphasis and non-rectangular windows. In (3.14), the summation

³Parseval's theorem: let $X[k]$ be the discrete Fourier transform of $x[n]$, then

$$\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2$$

in the argument of logarithm is over the entire spectral range and P_h cannot be pulled out of the parenthesis. The bias,

$$\xi_x - \xi_s = \log_e \left(\frac{\sum_k P_s[k] P_h[k]}{\sum_k P_s[k]} \right), \quad (3.15)$$

depends on *both* the noise $\{h(t)\}$ and the speech $\{s(t)\}$.

3.3.2 Distortion by Additive Noises

Next, the distortion of the MFCCs and the log energy feature caused by additive noises will be analyzed. The following argument may provide a perspective on the problem before algebraic details. The MFCCs are Fourier coefficients of the log-mel spectral sequence. For example, $C[0]$ is the average of this sequence, and $C[n]$ represents the amplitude of the n -th harmonic in this sequence. These coefficients equivalently reflect the spectral shape of a speech window. Difference in the *formant* positions, which represent peaks in the spectrum and physically stem from the configuration of the articulatory system, between different phonemes renders different patterns in the cepstral domain. When speech is mixed with noises from random sources, the overall spectrum tends to be flattened, leading to a shrinkage of the cepstral coefficients (except for $C[0]$).

In order to analyze the distortion of speech features in the presence of additive noises, equation (3.1) is rewritten as

$$x(t; \gamma) = s(t) + n(t; \gamma) = s(t) + \gamma n_0(t), \quad (3.16)$$

where an additive noise is now written as $n(t; \gamma) \triangleq \gamma n_0(t)$, with $n_0(t)$ summarizing the characteristics of a given noise, apart from a noise level parameterized by γ . The following analysis first deals with just the noise signal alone and then returns to the noisy speech signal.

The log mel-spectra of $n(t; \gamma)$ and $n_0(t)$ are related by

$$\begin{aligned} \log_e Q_{n(\gamma)}[j] &= \log_e \left(\sum_{k=0}^{N-1} F_{jk} \cdot (\gamma^2 P_{n_0}[k]) \right) \\ &= 2 \log_e |\gamma| + \log_e \left(\sum_{k=0}^{N-1} F_{jk} P_{n_0}[k] \right) \\ &= 2 \log_e |\gamma| + \log_e Q_{n_0}[j], \end{aligned} \quad (3.17)$$

where $Q_{n(\gamma)}$ and Q_{n_0} are the mel-frequency spectra of $n(t; \gamma)$ and $n_0(t)$, respectively. Since $\sum_j G_{ij} = 0 \quad \forall i \neq 0$, it follows that

$$\begin{aligned} C_{n(\gamma)}[i] &= \sum_{j=1}^J G_{ij} (2 \log_e |\gamma| + \log_e Q_{n_0}[j]) \\ &= C_{n_0}[i], \quad i = 1 \dots I, \end{aligned} \quad (3.18)$$

where $C_{n(\gamma)}$ and C_{n_0} are the cepstra of $n(t; \gamma)$ and $n_0(t)$, respectively. The zeroth cepstral coefficients of n_0 and n_γ are related by

$$\begin{aligned} C_{n(\gamma)}[0] &= \sum_{j=1}^J (2 \log_e |\gamma| + \log_e Q_{n_0}[j]) \\ &= 2\sqrt{2J} \log_e |\gamma| + C_{n_0}[0], \end{aligned} \quad (3.19)$$

so a change in the signal level shifts the value of $C[0]$ by a level-dependent constant. For the feature of log energy,

$$\xi_{n(\gamma)} = 2 \log_e |\gamma| + \xi_{n_0}, \quad (3.20)$$

which has a form similar to (3.19). Note that while the MFCC features (excluding $C[0]$) are invariant with respect to signal level, $C[0]$ and ξ are not. This justifies the common practice of signal normalization which fixes the terms related to γ in (3.19) and (3.20).

For noisy speech as given in (3.16), the power spectrum is

$$\begin{aligned} P_{x(\gamma)}[k] &= |S[k]^2| + 2\gamma |S[k]N_0[k]| + \gamma^2 |N_0[k]^2| \\ &= P_s[k] + 2\gamma |S[k]N_0[k]| + \gamma^2 P_{n_0}[k], \end{aligned} \quad (3.21)$$

where $P_{x(\gamma)}$, P_s , and P_{n_0} are respectively the power spectra of $x(t; \gamma)$, $s(t)$ and $n_0(t)$. It follows, as mel-binning is a linear operation, that

$$Q_{x(\gamma)}[j] = Q_s[j] + 2\gamma Q_1[j] + \gamma^2 Q_{n_0}[j], \quad (3.22)$$

where $Q_{x(\gamma)}$, Q_s , and Q_{n_0} are respectively the power spectra of $x(t; \gamma)$, $s(t)$ and $n_0(t)$, and $Q_1[j]$ is defined by $Q_1[j] \triangleq \sum_{k=0}^{N-1} F_{jk} |S[k] N_0[k]|$. The distortion of the mel-spectrum consists of the last two terms on the right-hand side of (3.22): a first-order term that is proportional to γ and depends on both the noise and the speech signal, and a second-order term that is proportional to γ^2 and depends only on the noise. In the following sections, the general condition in which no assumptions about noise level γ are made will be considered first. This is followed by analyses in the special cases of low noises ($|\gamma| \ll 1$) and high noises ($|\gamma| \gg 1$) where distortion terms can be simplified through approximation. In particular, it will be shown that in high-noise cases, the distortion of cepstral features consists of noise-dependent bias terms and a magnitude shrinkage (cf. equations (3.33) and (3.34)).

3.3.2.1 General Noise Level

In this section no assumptions are made about γ , the noise level, in the analysis. For the MFCCs it follows from (3.5) and (3.22) that

$$\begin{aligned}
C_{x(\gamma)}[i] &= \sum_{j=1}^J G_{ij} \log_e Q_{x(\gamma)}[j] \\
&= \sum_{j=1}^J G_{ij} \log_e (Q_s[j] + 2\gamma Q_1[j] + \gamma^2 Q_{n_0}[j]) \\
&= \sum_{j=1}^J G_{ij} \log_e \left(Q_s[j] \left(1 + 2\gamma \frac{Q_1[j]}{Q_s[j]} + \gamma^2 \frac{Q_{n_0}[j]}{Q_s[j]} \right) \right) \\
&= C_s[i] + \sum_{j=1}^J G_{ij} \log_e \left(1 + 2\gamma \frac{Q_1[j]}{Q_s[j]} + \gamma^2 \frac{Q_{n_0}[j]}{Q_s[j]} \right) \\
&= C_s[i] + \delta C_{x(\gamma)}[i], \quad i = 1, \dots, I,
\end{aligned} \tag{3.23}$$

where

$$\delta C_{x(\gamma)}[i] \triangleq \sum_{j=1}^J G_{ij} \log_e \left(1 + 2\gamma \frac{Q_1[j]}{Q_s[j]} + \gamma^2 \frac{Q_{n_0}[j]}{Q_s[j]} \right) \tag{3.24}$$

is the distortion, which depends on both the speech $s(t)$ and the noise $n(t; \gamma)$. For $C[0]$, similar to (3.23), it can be shown that

$$\begin{aligned} C_{x(\gamma)}[0] &= \sum_{j=1}^J \sqrt{\frac{2}{J}} \log_e Q_{x(\gamma)}[j] \\ &= C_s[0] + \delta C_{x(\gamma)}[0], \end{aligned} \quad (3.25)$$

where

$$\delta C_{x(\gamma)}[0] \triangleq \sum_{j=1}^J \sqrt{\frac{2}{J}} \log_e \left(1 + 2\gamma \frac{Q_1[j]}{Q_s[j]} + \gamma^2 \frac{Q_{n_0}[j]}{Q_s[j]} \right). \quad (3.26)$$

For the log energy ξ , the corrupted feature value is

$$\begin{aligned} \xi_{x(\gamma)} &= \log_e \left[\sum_{n=0}^{N-1} (s(n) + \gamma n_0(n))^2 \right] \\ &= \log_e \left[\sum_{n=0}^{N-1} (s(n)^2 + 2\gamma n_0(n)s(n) + \gamma^2 n_0(n)^2) \right] \\ &= \log_e \left(\sum_{n=0}^{N-1} s(n)^2 \right) \\ &\quad + \log_e \left(1 + 2\gamma \frac{\sum_n n_0(n)s(n)}{\sum_n s(n)^2} + \gamma^2 \frac{\sum_n n_0(n)^2}{\sum_n s(n)^2} \right) \\ &= \xi_s + \delta \xi_{x(\gamma)}, \end{aligned} \quad (3.27)$$

with distortion

$$\delta \xi_{x(\gamma)} \triangleq \log_e \left(1 + 2\gamma \frac{\sum_n n_0(n)s(n)}{\sum_n s(n)^2} + \gamma^2 \frac{\sum_n n_0(n)^2}{\sum_n s(n)^2} \right). \quad (3.28)$$

It is seen from (3.24), (3.26) and (3.28) that the distortion caused by a general additive noise depends on the speech (s), the noise type (n_0), and the noise level (γ). Therefore, it is difficult to design a general scheme to process the corrupted features of the noisy signals to be close to those of the clean speech signals.

If information about the noise n_0 can be obtained, such as in the case where both clean and noisy samples of the same utterance are available, then it is possible to design linear or non-linear mappings between clean and noisy signals and use such a relationship in processing. However, such an approach will not be effective in mismatched conditions. Furthermore, such an approach can be expensive, in terms both of data acquisition and

achieving desired model accuracy. This is quite contrary to the investigated technique in this thesis, which attempts to achieve noise-robustness without this ineffectiveness or expense.

3.3.2.2 Low Additive Noises

With $|\gamma| \ll 1$, the second-order term in the distortion of MFCC in (3.24) can be neglected and the distortion can be approximated by

$$\begin{aligned} \delta C_{x(\gamma)}[i] &\approx \sum_{j=1}^J G_{ij} \log_e \left(1 + 2\gamma \frac{Q_1[j]}{Q_s[j]} \right) \\ &\approx 2\gamma \sum_{j=1}^J G_{ij} \frac{Q_1[j]}{Q_s[j]} \\ &= 2\gamma C_{e1}[i], \quad i = 1, \dots, I, \end{aligned} \quad (3.29)$$

where $C_{e1}[i] \triangleq \sum_{j=1}^J G_{ij} \frac{Q_1[j]}{Q_s[j]}$ and $\log_e(1+x) \approx x$ when $|x| \ll 1$. Similarly, the distortion of $C[0]$ in (3.26) becomes

$$\begin{aligned} \delta C_{x(\gamma)}[0] &\approx \sum_{j=1}^J \sqrt{\frac{2}{J}} \log_e \left(1 + 2\gamma \frac{Q_1[j]}{Q_s[j]} \right) \\ &\approx 2\gamma C_{e1}[0], \end{aligned} \quad (3.30)$$

where $C_{e1}[0] \triangleq \sum_{j=1}^J \sqrt{\frac{2}{J}} \frac{Q_1[j]}{Q_s[j]}$. The distortion of the log energy feature in (3.28) can also be approximated by

$$\begin{aligned} \delta \xi_{x(\gamma)} &\approx \log_e \left(1 + 2\gamma \frac{\sum_n n_0(n)s(n)}{\sum_n s(n)^2} \right) \\ &\approx 2\gamma \frac{\sum_n n_0(n)s(n)}{\sum_n s(n)^2} \\ &= 2\gamma e^{-\xi_s} \sum_n n_0(n)s(n), \end{aligned} \quad (3.31)$$

since $\frac{1}{\sum_n s(n)^2} = e^{-\log_e(\sum_{n=0}^{N-1} s(n)^2)} = e^{-\xi_s}$, $\gamma^2 \approx 0$, and $\log_e(1+x) \approx x$. Being proportional to $e^{-\xi_s}$, the bias $\delta \xi_{x(\gamma)}$ is small when the speech energy is high. This is a desired property and is the reason that the log energy feature works better than $C[0]$ if no enhancement or feature processing schemes are adopted. This point is verified in the experiments by

comparing the results of using respectively $C[0]$ and ξ as the enhancing component for the MFCC features when the noise is relatively low (cf. Section 6.1.3.1).

3.3.2.3 High Additive Noises

With $|\gamma| \gg 1$, (3.22) can be approximated by

$$Q_{x(\gamma)}[j] \approx \gamma^2(Q_{n_0}[j] + \frac{2}{\gamma}Q_1[j]). \quad (3.32)$$

Therefore, the MFCCs are approximately

$$\begin{aligned} C_{x(\gamma)}[i] &\approx \sum_{j=1}^J G_{ij} \log_e \left(\gamma^2(Q_{n_0}[j] + \frac{2}{\gamma}Q_1[j]) \right) \\ &= \sum_{j=1}^J G_{ij} \log_e \left(\gamma^2 Q_{n_0}[j] \left(1 + \frac{2}{\gamma} \frac{Q_1[j]}{Q_{n_0}[j]} \right) \right) \\ &\approx \sum_{j=1}^J G_{ij} \left(2 \log_e |\gamma| + \log_e Q_{n_0}[j] + \frac{2}{\gamma} \frac{Q_1[j]}{Q_{n_0}[j]} \right) \\ &= C_{n_0}[i] + \frac{2}{\gamma} C_{e2}[i], \quad i = 1, \dots, I, \end{aligned} \quad (3.33)$$

where $C_{e2}[i] \triangleq \sum_{j=1}^J G_{ij} \frac{Q_1[j]}{Q_{n_0}[j]}$. For $C[0]$,

$$\begin{aligned} C_{x(\gamma)}[0] &\approx \sum_{j=1}^J \sqrt{\frac{2}{J}} \log_e \left(\gamma^2(Q_{n_0}[j] + \frac{2}{\gamma}Q_1[j]) \right) \\ &\approx 2\sqrt{2J} \log_e |\gamma| + C_{n_0}[0] + \frac{2}{\gamma} C_{e2}[0], \end{aligned} \quad (3.34)$$

where $C_{e2}[0] \triangleq \sum_{j=1}^J \sqrt{\frac{2}{J}} \frac{Q_1[j]}{Q_{n_0}[j]}$. The cepstrum is mainly determined by the noise $n_0(t)$, and the contribution from speech signal $s(t)$ is *inversely* proportional to γ through the term of C_{e2} . The distortion in the cepstral features is not merely a bias term. That is, although one can apply mean subtraction to eliminate $C_{n_0}[i]$ (and $2\sqrt{2J} \log_e |\gamma|$) in equations (3.33) and (3.34), the remaining term is different from that of clean speech, due to the scale change of γ^{-1} and the fact that C_{e2} is different from C_s .

The approximation is different for the log energy,

$$\begin{aligned}
\xi_{x(\gamma)} &= \log_e \left[\sum_{n=0}^{N-1} (s(n) + \gamma n_0(n))^2 \right] \\
&\approx \log_e \left[\sum_{n=0}^{N-1} (2\gamma n_0(n)s(n) + \gamma^2 n_0(n)^2) \right] \\
&= \log_e \left[\gamma^2 \left(\sum_{n=0}^{N-1} n_0(n)^2 \right) \left(1 + \frac{2 \sum_n n_0(n)s(n)}{\sum_n n_0(n)^2} \right) \right] \\
&\approx 2 \log_e |\gamma| + \xi_{n_0} + \frac{2 \sum_n n_0(n)s(n)}{\sum_n n_0(n)^2} \\
&= 2 \log_e |\gamma| + \xi_{n_0} + \frac{2}{\gamma} e^{-\xi_{n_0}} \sum_n n_0(n)s(n),
\end{aligned} \tag{3.35}$$

where ξ_{n_0} is the log energy of $n_0(t)$. The only term in (3.35) that contains information about speech is the last one, while the two previous terms can be removed by mean subtraction.

The difference between this term and the clean speech feature is

$$\begin{aligned}
&\log_e \left(\sum_n s[n]^2 \right) - \frac{2}{\gamma} e^{-\xi_{n_0}} \sum_n n_0(n)s(n) \\
&= \log_e \left(\frac{\sum_n s[n]^2}{e^{\frac{2}{\gamma} e^{-\xi_{n_0}} \sum_n n_0(n)s(n)}} \right),
\end{aligned} \tag{3.36}$$

which is, however, difficult to untwine.

Therefore, additive noise does not contribute a simple bias that depends solely on the noise characteristics. Instead, the distortion is jointly determined by both the speech and the noise in complicated ways. In the low-noise case, the distortion is small and it is shown that the log energy is less susceptible to such additive noise than is $C[0]$. In high-noise cases, after removing the terms from noise signals, the variance normalization can be applied to counter the effect of diminished feature magnitudes.

Chapter 4

MVA FEATURE PROCESSING: DEFINITION AND ANALYSIS

Based on the distortion of the MFCC features in the presence of additive and convolutional noises as analyzed in the previous chapter, a feature processing technique consisting of mean subtraction, variance normalization, and ARMA (Auto-Regression and Moving-Average) filtering in the cepstral domain for noise-robust feature extraction is proposed. This technique is named MVA processing. In this chapter, the MVA processing is defined. Its effect is first illustrated by an example, followed by an analysis of MVA-processed MFCC features in the presence of additive and convolutional noises.

4.1 Definition of MVA

Let $C^{(\tau)}$ be the raw feature vector at frame τ .¹ Mean subtraction (MS) is defined by

$$\bar{C}^{(\tau)} = C^{(\tau)} - \mu, \quad (4.1)$$

where \bar{C} is the mean-subtracted feature and μ is a mean vector estimated from data. Variance normalization (VN) is defined by

$$\hat{C}^{(\tau)}[d] = (\sigma^2[d])^{-1/2} \bar{C}^{(\tau)}[d], \quad (4.2)$$

where \hat{C} is the mean-subtracted and variance-normalized feature and $\sigma^2[d]$ is an estimate of the variance of the d th feature component. ARMA filtering is defined by²

$$\check{C}^{(\tau)} = \frac{\check{C}^{(\tau-m)} + \dots + \check{C}^{(\tau-1)} + \hat{C}^{(\tau)} + \dots + \hat{C}^{(\tau+m)}}{2m+1}, \quad (4.3)$$

¹Here the superscript (τ) is back as we are considering a time sequence.

²Note that here the coefficients in the AR and MA parts are the same and they sum to one. In general, the coefficients of an ARMA filter need not satisfy these constraints and can be any numbers. In other words, equation (4.3) is a special case.

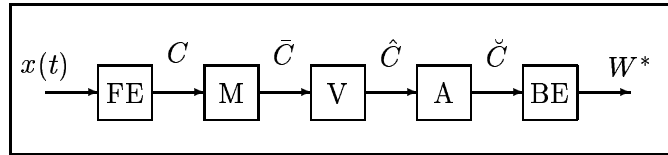


Figure 4.1: Block diagram of MVA feature processing technique. $x(t)$: speech waveform; FE: feature extraction; M: mean subtraction; V: variance normalization; A: ARMA filter; BE: back end; W^* : recognized text.

where \check{C} is the MVA processed feature and m is the order of the ARMA filter. The special case of $m = 0$ degenerates to no filtering.

The estimation of the mean μ and the variance σ^2 can be implemented in a number of different schemes. In a *per-side* estimation, these quantities are estimated by all the data in a conversation side [16]. Such an estimation can be quite robust if the communicating channel and background noise are stationary. In an *on-line* estimation [5], these quantities are estimated from only the samples currently available and do not depend on future observations. Such a scheme has low latency and is indispensable for real-time applications. Between these two extremes, a *per-utterance* estimation [3] is defined by

$$\mu = \frac{1}{T} \sum_{\tau=1}^T C^{(\tau)}, \quad (4.4)$$

and

$$\sigma^2[d] = \frac{1}{T} \sum_{\tau=1}^T (C^{(\tau)}[d] - \mu[d])^2, \quad (4.5)$$

where T is the number of frames in a given utterance.

A block diagram of the technique is depicted in Figure 4.1. Note that MVA feature processing is the same for every feature sequence in every utterance and is not trained for specific data. In other words, it is not tuned to particular types of noises.³

³This is an example of the principle of Occam's Razor, that one should not make more than the minimum assumption for a given problem. Tuning a method to specific types of noises amounts to making assumptions about the noise property, with the result that the method becomes inappropriate to different types of noise.

4.2 Analysis on MVA with Noisy Speech Features

The distortion of the MFCCs and the log energy feature under additive and convolutional noises has been derived in Section 3.3. In the present section, the focus is on how MVA can counter such distortion. First, an illuminating example is given to demonstrate the issues with the raw features in noisy environments and to show how such issues are substantially alleviated by MVA feature processing. This is followed by a further analysis on MVA-processed features in the presence of noise.

4.2.1 An Example

In Figure 4.2, the time sequences of $C[0]$, $C[1]$ and the log energy are plotted for the speech signals of the utterance of digit string 5376869 corrupted by different levels of additive subway noise (from the Aurora 2.0 database). Apparently, in this case the effect of the noise on the raw features (left column) is to change the level of the time sequences and to reduce the temporal variation. In contrast, this effect is largely eliminated with the MVA feature processing (right column). Specifically, one can see that MS and VN combine to bring the time sequences in different noise levels to the same relative level (via MS) and scale (via VN). After MS and VN, the feature time sequences in noisy speech show spurious spikes relative to the time sequences of clean speech, so applying the ARMA filter can smooth the sequences. While there is a risk in using visual inspection to deduce a processing methodology, MVA proves to be remarkably beneficial on system performance.⁴

4.2.2 Convolutional Noise and Mean Subtraction

In the presence of convolutional noise, it has been established in (3.12) and (3.13) that for the MFCCs the distortion is a bias *independent* of the speech. If the noise is stationary, e.g., when the acoustic environment is fixed, then it follows from (4.1), (4.4), (3.12) and (3.13)

⁴Results will be presented in Chapter 6.

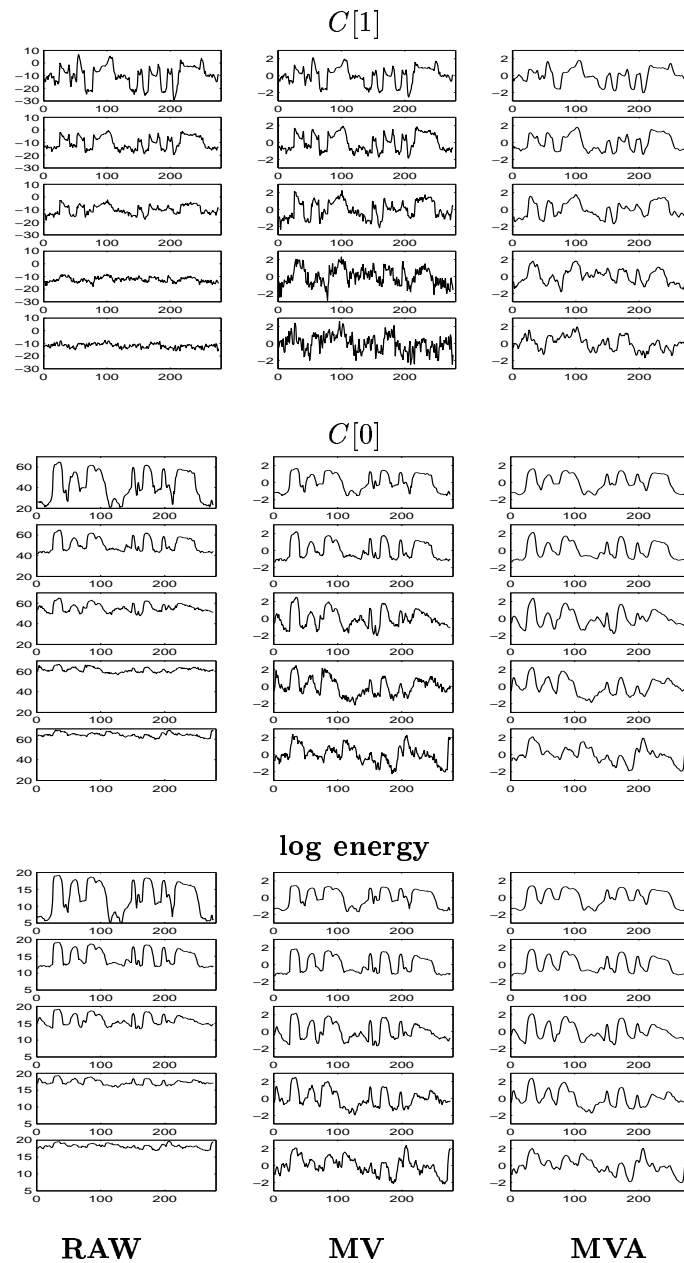


Figure 4.2: The time sequences of the cepstral coefficient $C[1]$ (top box), $C[0]$ (middle box), and log energy (bottom box), for the digit string 5376869 in clean condition (first row), 20 dB SNR (second row), 10 dB SNR (third row), 0 dB SNR (fourth row), and -5 dB SNR (last row), without feature processing (left column), with MS and VN (middle column), and with MVA feature processing (right column). The order of the ARMA filter used in this example is $m = 2$.

that for the MFCCs,

$$\begin{aligned}
\bar{C}_x^{(\tau)}[i] &= C_x^{(\tau)}[i] - \mu_x[i] \\
&= C_s^{(\tau)}[i] + B_h[i] - (\mu_s[i] + B_h[i]) \\
&= C_s^{(\tau)}[i] - \mu_s[i] \\
&= \bar{C}_s^{(\tau)}[i], \quad i = \mathbf{0}, 1, \dots, I.
\end{aligned} \tag{4.6}$$

That is, the mean-subtracted MFCCs are invariant in the presence of a stationary convolutional noise. With the per-utterance scheme, if the corrupting convolutional noise is stationary and spectrally smooth within the duration of the utterance, then MS is expected to work well, as features extracted from $x(t)$ are the same as those from $s(t)$.

On the other hand, as noted in (3.15), the bias of the log energy feature caused by a convolutional noise depends on *both* the noise $\{h(t)\}$ and the speech $\{s(t)\}$. Even if the noisy is stationary, the bias changes with the acoustics of speech. Therefore, MS is not expected to be effective.

4.2.3 Additive Noise and Variance Normalization

In the situation of high additive noises, one can see from (3.33) and (3.34) that the distortion in the cepstral features is not merely a bias term. Therefore, even after applying the MS to eliminate $C_{n_0}[i]$ in equations (3.33) and (3.34), the remaining term $\frac{2}{\gamma}C_{e2}[i]$ is not the same as that of the clean speech, due to a scale proportional to γ^{-1} and due to the fact that C_{e2} is different from C_s . Further application of VN can mend the first mismatch (scale) but not the second ($C_{e2} \neq C_s$). Note that if VN is applied alone, $C_{n_0}[i]$ is not eliminated, so the processed features are not close to those from clean speech, and the performance is not expected to improve. Similar arguments apply to the log energy feature, as the distortion terms are similar. In summary, combining MS and VN can only partially counter the effects of high-level additive noises.

4.2.4 ARMA Filtering

Human speech is anatomically constrained. Essentially, the vocal tract system cannot change its configuration arbitrarily quickly. It has been argued and demonstrated that

crucial information in human speech is contained in the low-frequency⁵ part of temporal envelope of speech spectrum [72] or spectral modulation [37, 42]. Therefore, as MFCC obviously conveys speech information, the low-frequency parts of the MFCC time sequences are more important than the high-frequency parts.

To illustrate, an example is given in Figure 4.4. Here the log-scale magnitudes of the discrete Fourier transforms of the MFCC and log energy feature time sequences are plotted. Note that here the Nyquist frequency is 50 Hz, as there are 100 frames per second. From this figure, one can see that for the raw features, an increase in the noise level leads to a downward shift in the magnitude and a flatness in the shape of the spectrum. The MV features (i.e., features processed by mean subtraction and variance normalization) are able to counter the effect of downward shift, but this does not help the flatness issue. The MVA features, due to the emphasis in the low-frequency part and de-emphasis in the high-frequency part of the ARMA filter, are able to make the overall frequency plots of clean and noisy samples quite similar. The frequency response of the investigated second-order ARMA filter is shown in Figure 4.3. To be more quantitative, the Euclidean distances between clean and noisy spectral plots are calculated. These numbers are summarized in Table 4.1. One can see that for each feature, the Euclidean distance between a given noisy sample and the corresponding clean sample increases as the noise level increases (from 20 dB to -5 dB). One can also see that MV features significantly reduce the Euclidean distance relative to the raw features between a noisy sample and the corresponding clean sample. Finally, one can see that the application of ARMA filter further reduces the Euclidean distances. Note that Euclidean distance, while promising, does not necessarily transform into word error rate. Nonetheless, in the experiments shown in this thesis, additional ARMA filtering does lead to performance gain, as will be shown in Section 6.1.3.

⁵Note that here “low-frequency” does not mean the low physical frequency in the spectrum, but the low modulation frequency in the time sequence of spectrum.

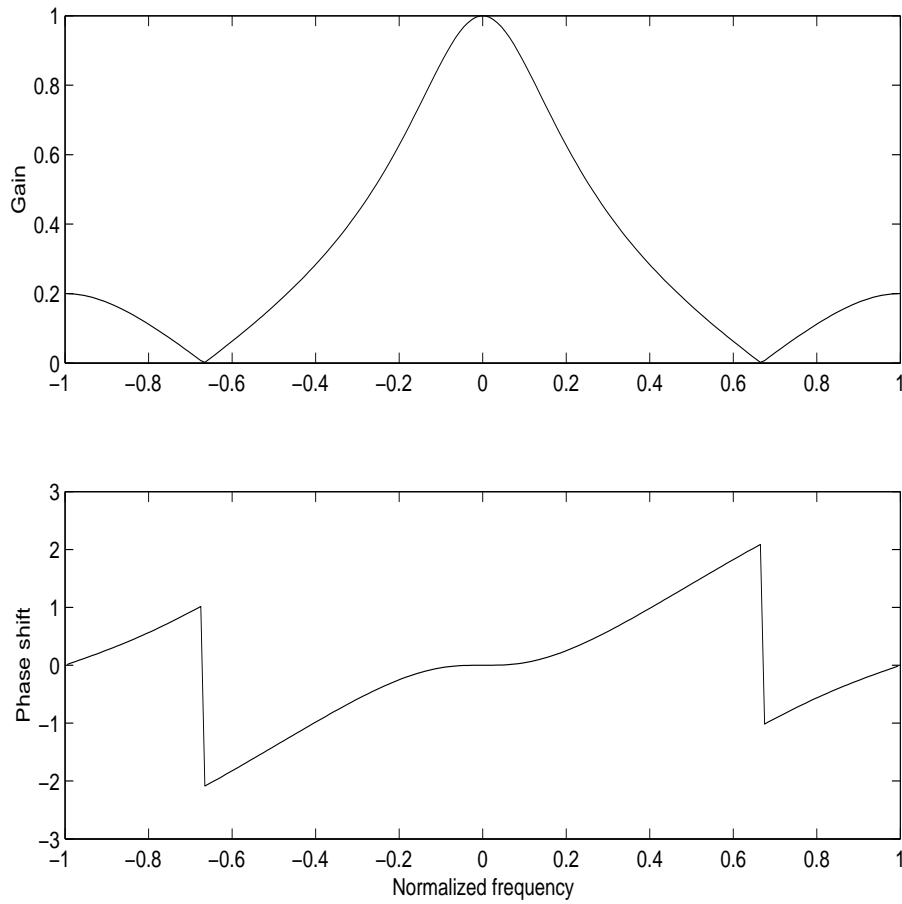


Figure 4.3: The frequency response of a second-order ARMA filter. Note that for this filter the gain in the low-frequency part is much higher than the gain in the high-frequency part.

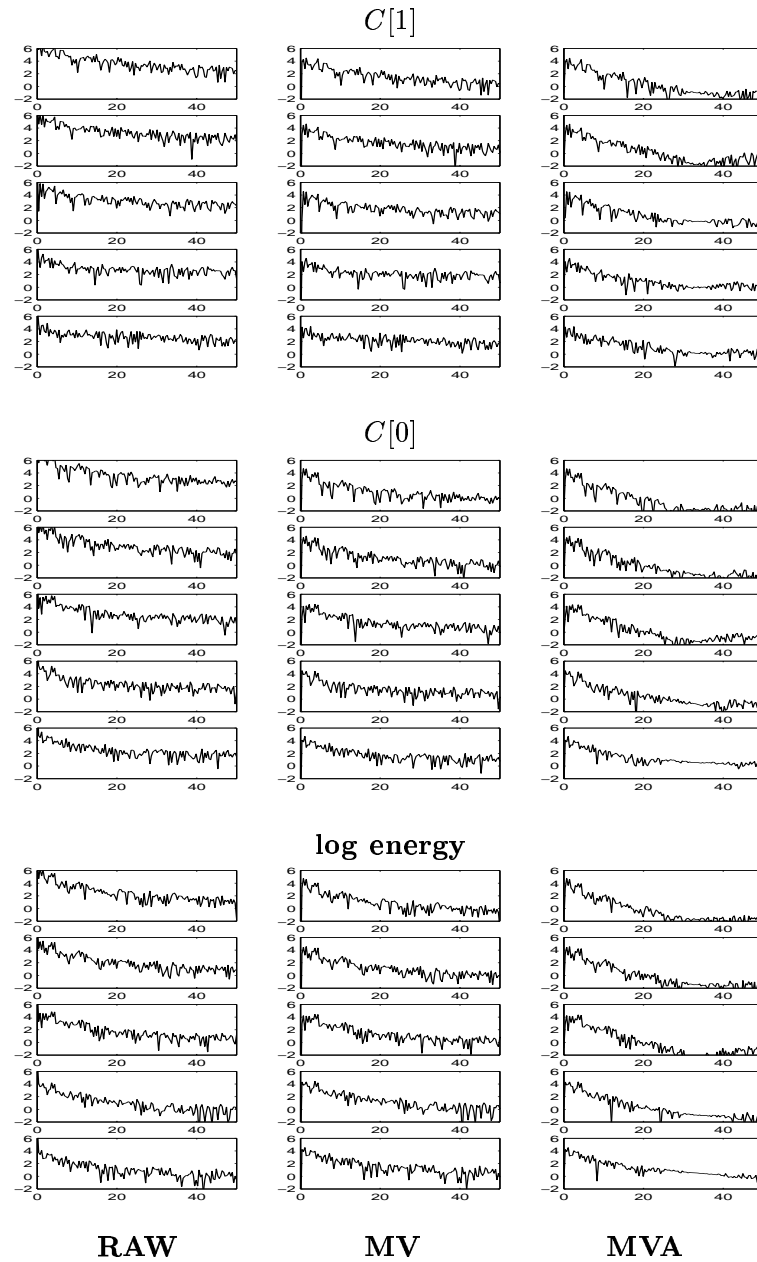


Figure 4.4: The frequency-domain plots of the time sequences of $C[1]$ (top box), $C[0]$ (middle box), and $\log \text{energy}$ (bottom box), for the digit string 5376869 in clean condition (first row) 20 dB SNR (second row), 10 dB SNR (third row), 0 dB SNR (fourth row), and -5 dB SNR (last row), without feature processing (left column), with MS and VN (middle column) and with MVA feature processing (right column). The x-axis is the frequency axis, ranging from 0 to 50 Hz (Nyquist frequency). The y-axis is the magnitude of spectrum in log scale. The order of the ARMA filter used here is $m = 2$. Note that these are the same examples as in Figure 4.2.

Table 4.1: The Euclidean distances in the frequency domain between the clean sample and the noisy samples. For example, the number 919 in position (RAW $C[1]$, 20 dB) is the Euclidean distance between the $C[1]$ features (without any feature processing) of clean and 20 dB samples. Here the distance is defined to be $[\sum_{i=1}^N (X_1(f_i) - X_2(f_i))^2]^{1/2}$, where f_i is the i -th discrete frequency, and $X_1(f_i)$ and $X_2(f_i)$ are the spectrum at f_i for the clean and noisy examples respectively. Note that here the original magnitude is used instead of the log scale in calculating the distance, unlike in Figure 4.4.

	20 dB	10 dB	0 dB	-5 dB
RAW $C[1]$	919	1339	1890	1922
MV $C[1]$	83	128	176	213
MVA $C[1]$	71	110	147	171
RAW $C[0]$	3116	4669	6219	7014
MV $C[0]$	69	142	161	219
MVA $C[0]$	61	132	148	210
RAW ξ	842	1278	1685	1918
MV ξ	74	137	151	202
MVA ξ	67	127	141	194

Chapter 5

DYNAMIC-BAYESIAN-NETWORK ASR SYSTEMS

In the previous chapter, the MVA technique in the front end of an ASR system is defined and analyzed. In this chapter, a different technique in the back end towards noise-robustness is introduced. This technique is essentially an extension of the basic HMM framework. Specifically, ASR systems with multiple feature streams and novel feature-selection schemes will be described.

5.1 Introduction

In a vanilla ASR system, often the front end creates a stream of feature vectors such as the MFCCs or the LPCs, and the back end embeds an acoustic model such as an HMM and a language model such as n -gram. While such simple systems can achieve satisfactory accuracies on ASR tasks with small vocabulary and clean speech, the performance degrades rapidly when the vocabulary sizes increase or when the acoustic environments become noisy or mismatched.

To stay with the framework of HMM and to cope with the challenges posed by a large vocabulary or diverse acoustic conditions, ingenious ideas have been invented to keep performance level of a state-of-the-art ASR system in pace with the task difficulties. For example, parameter tying for robust estimation and model adaptation to re-estimate model parameters to better fit the test/train data [32, 2] have helped to keep the acoustic model of HMM from being too simple to be adequate.

The popularity of HMMs in the ASR research community is also sustained with fundamental considerations universal to specific models. First, the decoding criteria of minimizing sentence error rate is challenged as the performances are often measured by word error rate. Combining multiple recognition hypotheses to directly minimize word error rate has been

implemented and has achieved significant improvements [61]. Second, it can be argued that different speech features unveil different facets of the information conveyed in speech waveforms, so their combination is potentially beneficial. For example, a system combining probabilistically the outputs from trained neural networks for two different features has resulted in performance gains in noisy environments [26].

It has been argued that HMMs have enormous modeling capacity in the sense that, with unlimited training data, the HMM framework is capable of modeling any random sequences of random observations by increasing the number of parameters infinitely [7]. In practice, however, both the data and computational resources (time and memory) are often quite limited. To best use the scarce resources to build up and train an ASR system, one might consider modeling the speech outside the domain of HMMs.

Using more refined models for speech to extend the basic HMM is an on-going trend. To begin, the multi-stream HMM [15] and factorial HMM [35, 57] stay in the framework of HMM while considering multiple feature streams and the detailed structures of the “meta-state” variables. Beyond HMM, the early research on the autoregressive HMM (AR-HMM) [47] and the recently proposed idea of the buried Markov models (BMM) [11] are exemplary works. Recently [82], a multi-stream asynchronous model with multiple state variables, each correlating to its neighbor, has been proposed. Note that these techniques can all be described in the language of graphical models, that (dependency) edges between variables are added to the simple HMM graph.

Methods of combining multiple feature streams are investigated in this chapter. Here, the combinations are implemented as distinct graphical models. The motivation is two-fold: (1) to exploit features that work well for various acoustic environments and (2) to investigate different models for the recognizer. The method is different from the aforementioned works in that it adds not only edges but also nodes to the graphical models of HMMs. The new nodes are in fact a sequence of nodes called the feature selectors, and the new edges are between the feature selectors and their adjacent nodes. These will be described in greater detail later in the appropriate contexts.

5.2 Investigated Multiple-stream ASR Systems

As different speech features convey different shades of information in the speech samples, an ASR system integrating multiple features has the potential to outperform other systems using single feature streams. In this work, the proposed ASR system has multiple feature streams, with an additional feature selector stream to dynamically select a feature according to certain rules. To illustrate, suppose that feature A is good for noisy environments and feature B is good for the clean environment. With the help of a noise level estimator, the system can select feature A when noise is high and use feature B otherwise. Thus, better performance than using feature A or feature B alone can be achieved.

5.2.1 Multiple Feature Streams

For clarity, the terminology will be defined first. A *feature stream* is a continuous sequence of feature vectors. A *uni-stream* (ASR) system uses a single feature stream. A *multi-stream* system uses multiple feature streams. A *component feature stream* is one of the feature streams in a multi-stream system. A *component feature* is a feature vector in a component feature stream. The *entire feature* refers to all component features. In this thesis, a *dual-stream* system with two component feature streams is considered. It should not be difficult to incorporate a novel component feature suited for a particular acoustic environment into a multi-stream system. Such integration can be realized by either increasing the number of component features or replacing one component feature with the new feature, and then modifying the speech model accordingly.

5.2.2 Dynamic Bayesian Networks and Graphical Models

A graphical model (GM) [53, 68] represents a collection of random variables as a graph, in which the nodes represent random variables and the edges represent the dependency relation between the random variables. Many aspects of ASR can be depicted as GMs [12]. Such representation has at least two advantages: 1) conditional independence relations can be read off the graph according to a handful of rules [68], and 2) inference algorithms are generally available for training and decoding. A Bayesian network (BN) is a directed graph

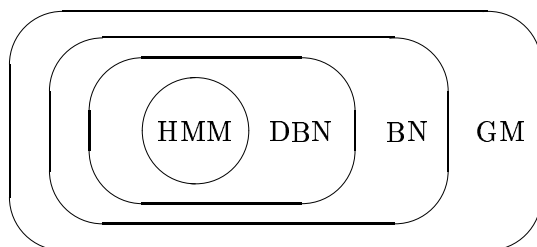


Figure 5.1: The scopes of HMMs, DBNs, BNs and GMs. One can see that GM represents the largest while HMM is the most limited in the these classes.

with no directed cycles. In a dynamic Bayesian network (DBN) [23], all edges point to the same direction of time to avoid directed cycles. For instance, HMM is a class of DBN. Altering a graph amounts to modifying the model it represents. For example, starting from an HMM graph, edges can be added to change the model to be a BMM, which is an instance of a DBN.

5.2.3 Feature Selector and New Models

The introduction of feature selectors (henceforth denoted by R) in the speech model is new. These random variables are used to choose one component feature from the entire feature observation to score a given frame. The value of this random variable can be determined by a noise estimator and is therefore observed.¹ In Figure 5.2, a baseline HMM graph and the proposed observed feature selector graph are depicted.²

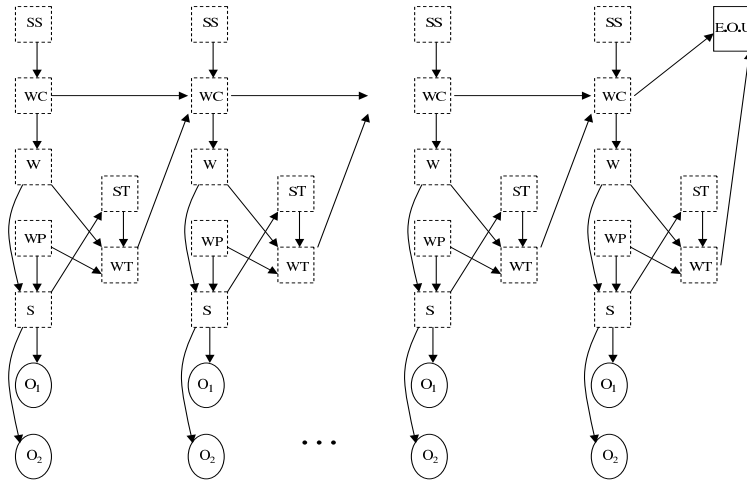
5.3 Implementation Specification

5.3.1 Component Feature Streams

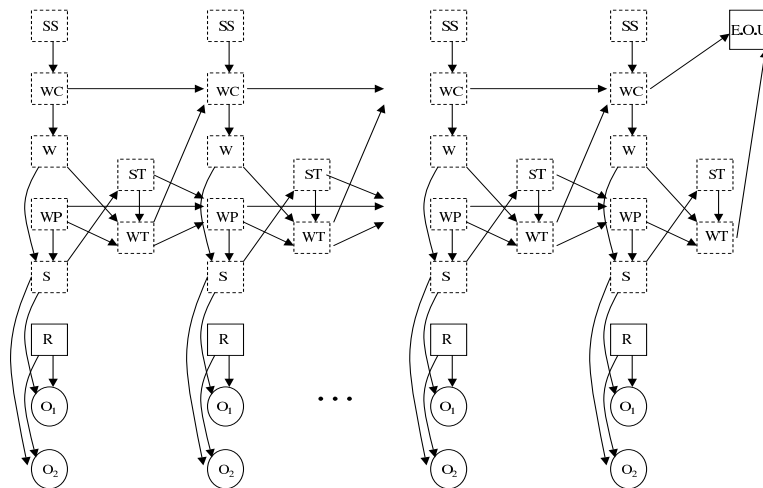
The choice of component features is crucial in a multi-stream system. The bottom line imperative is to choose component features such that each is appropriate for at least one

¹Alternatively, it can be assumed to be dependent on certain hidden nodes and is therefore hidden.

²Note that these are the training graphs. The corresponding decoding graphs have no Skip-Silences and Word-Counters and have edges between adjacent Words representing a bigram language model.



Baseline HMM



Proposed system with feature selectors

Figure 5.2: Graphical models of DBN. The top graph is the baseline HMM, while the bottom graph is the system with feature selectors. Discrete variables are represented by rectangles and continuous variables are represented by circles. A hidden variable is shown with a dashed boundary, while an observed variable has a solid boundary. The meaning of the variables are: SS = Skip-Silence, WC = Word-Counter, W = Word, WP = Word-Position, S = State, R = feature selector, O_1 = observation feature 1, O_2 = observation feature 2, ST = State-Transition, WT = Word-Transition, E.O.U. = End-Of-Utterance.

environment. Since this thesis deals with noise-robustness, one of the component features has to be robust to noise. In this regard, the MFCC features processed by MVA processing (referred to as MVA features) have been chosen, to be used in noisy environment. They have been shown to be very robust to noise with Aurora 2.0/3.0 tasks [17, 18]. The other component feature should complement the component feature of MVA and it will be used in a clean environment. Although MFCC features work fine in clean environments, they are not used in the proposed dual-stream system. This is because not every speech window can be clearly classified as clean or noisy. That is, some windows may be “slightly” noisy either because the noise level is low or because they span across noisy/clean segments of speech. These are exactly those which are prone to errors had the raw features been chosen. To accommodate such cases, the MV features are included instead. Thus the entire feature is composed of the MVA and MV features.

5.3.2 GMTK-based ASR Systems

The multi-stream ASR systems are implemented with the graphical model toolkit (GMTK) [8]. An outline of GMTK-based ASR systems is given in Figure 5.3. Dependency structures are stored in model structure files where a random variable is described as being hidden or observed, discrete or continuous, and by specifying its parents and the corresponding conditional probability table or parametric distribution. The training data set is used to train the model parameters, and the model is used to decode test data. An example of an HMM-based ASR system implemented with GMTK is described in [8].

The proposed system requires a little technical detail. When a component feature is chosen by the feature selector R , the non-chosen component feature needs to be ignored. In GMTK this is implemented as follows. A function named *unityScore* always returns 1 (or 0 in the log space) when it is called. Via a structure file, the system can be directed to use the *unityScore* for a component feature observation whenever it is not selected. To be more specific, the feature selector R acts as a switching parent³ to both O_1 and O_2 , and

³A switching parent of a random variable switches the probability distribution for the random variable. In particular, the random variable can have different local dependent structures as the value of its switching parents vary. Switching parents are different from conditional parents, which also affect the probability

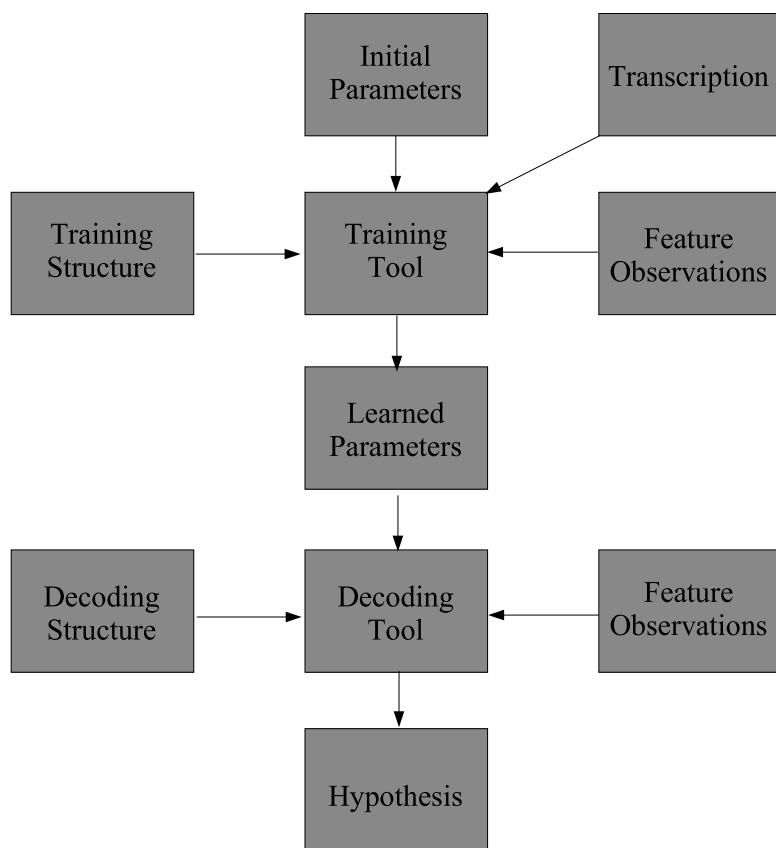


Figure 5.3: The GMTK systems. The major difference between this methodology and the traditional (HTK) ones is the introduction of input structure files. Structure files provide flexibility in implementing ASR systems when considering various speech models. The training and decoding tools are part of the graphical model toolkit.

one of the acoustic scores of O_1 and O_2 is ignored using `unityScore` based on the value of R . Thus the overall score of the entire feature observation in a given frame is from the selected component feature only. To be very specific, when R is 0, O_1 is scored based on the current state value while O_2 is scored by `unityScore`. When R is 1, O_2 is scored based on the current state value while O_1 is scored by `unityScore`. By definition, R is a switching parent of O_1 (and O_2), as its value determines the probability function used to score the random variable O_1 (and O_2).

5.3.3 Observed Feature Selectors

In the graphical model where the feature selectors are observed, the feature selector values need to be predetermined. These values can be decided by estimating the noise level of the acoustic environment. Two distinct time scales for such estimations are considered:

1. **utterance-level estimation:** In this case, a unique value is assigned to a given utterance. Equivalently, all frames of a given utterance are assigned the same value.
2. **frame-level estimation:** In this case, there is no unique value assigned to a given utterance. Equivalently, frames of a given utterance are assigned different values in general.

Note that the idea of frame-level environmental estimation makes practical sense only when the environment is constantly changing during the utterance of speech. In other words, it requires that the utterance is relatively long and the environment is relatively non-stationary.

In a system of two component features, the feature selection amounts to a binary classification. Two approaches for classifying clean and noisy speech are investigated in this work and will be described next.

5.3.3.1 Energy-based SNR Estimation

By comparing the waveforms of a pair of noisy and clean utterances, it is obvious that in a silent segment of the clean speech, the corresponding segment in the noisy speech has

distribution but do not change the local dependency structures.

a significant noise level. Such a difference can be exploited to estimate the noise level as follows.

Consider the case of additive noise corruption,

$$x^{(\tau)}(t) = s^{(\tau)}(t) + n^{(\tau)}(t), \quad (5.1)$$

where $x^{(\tau)}(t)$ is the t -th sample of the τ -th speech processing window. Summing over all samples of the τ -th window, one gets

$$\sum_{t=1}^N x^{(\tau)2}(t) \approx \sum_{t=1}^N s^{(\tau)2}(t) + \sum_{t=1}^N n^{(\tau)2}(t), \quad (5.2)$$

where it is assumed that $s(t)$ and $n(t)$ are uncorrelated, so $\sum_{t=1}^N s(t)n(t) \approx 0$ for large N . If $n(t)$ is stationary, so that $E\{\sum_{t=1}^N n^{(\tau)2}(t)\}$ is independent of τ , then

$$\sum_{t=1}^N n^2(t) \approx \min_{\tau} \sum_{t=1}^N x^{(\tau)2}(t). \quad (5.3)$$

In (5.3), the minimum occurs at a window contained in a non-speech segment, where the signal is dominated by the noise component. Therefore, the noise energy level can be approximated by the minimum of noisy speech energy in a given utterance.⁴

Summarizing the previous ideas, the basic steps for energy-based SNR estimation are:

- Extract the log energy features (as defined in (3.8)), $\{\xi^{(\tau)}, \tau = 1 \dots T\}$.
- Find the minimum value of the log energy feature streams $\xi_m \triangleq \min_{\tau} \xi^{(\tau)}$, which is an estimate of the log energy of the noise.
- For the frame-level estimate, compute the SNR for frame τ from ξ_m and $\xi^{(\tau)}$.
- For the utterance-level estimate, use the maximum value $\xi_M \triangleq \max_{\tau} \xi^{(\tau)}$ of an utterance.

⁴This method is more accurate with high SNR signal than low SNR signal. In low SNR case, the speech signal is essentially negligible and the minimum of signal energy tends to be an underestimate of the noise energy level.

Specifically, the SNR is related to the log energy by

$$\begin{aligned}
SNR &= 10 \log_{10} \frac{\sum_{t=1}^N s^{(\tau)^2}(t)}{\sum_{t=1}^N n^2(t)} \\
&\approx 10 \log_{10} \frac{\sum_{t=1}^N x^{(\tau)^2}(t) - \sum_{t=1}^N n^2(t)}{\sum_{t=1}^N n^2(t)} \\
&= 10 \log_{10} \frac{e^{\xi^{(\tau)}} - e^{\xi_m}}{e^{\xi_m}} \\
&= 10 \log_{10} \left(e^{(\xi^{(\tau)} - \xi_m)} \right) \left(1 - e^{-(\xi^{(\tau)} - \xi_m)} \right).
\end{aligned} \tag{5.4}$$

Using the identity⁵

$$\log(1 - t) = -\left(t + \frac{t^2}{2} + \frac{t^3}{3} + \dots\right),$$

and let $t = e^{-(\xi^{(\tau)} - \xi_m)}$, it follows that

$$SNR \approx 4.343 (\xi^{(\tau)} - \xi_m) - \delta, \tag{5.5}$$

where

$$\delta = 4.343 \left(t + \frac{t^2}{2} + \frac{t^3}{3} + \dots\right). \tag{5.6}$$

Therefore the SNR estimator varies approximately linearly with the log energy, apart from the deviation term δ defined in (5.6). Based on (5.4), a linear relation can be assumed,

$$\hat{\zeta} = f(\chi) = a \chi + b, \tag{5.7}$$

where $\hat{\zeta}$ is the estimated SNR value in dB, χ is the dynamic range of the ξ (log energy) sequence, and a and b are the parameters to be learned from training data.

To test the viability of this scheme, speech data from the utterance “5376869” corrupted by the subway noise from the Aurora 2.0 [41] database are processed according to the above scheme and the results are summarized in Table 5.1. From these numbers, one can fit an optimal linear relation between ζ (SNR) and χ (dynamic range of log energy), with parameters $(a, b) = (4.94, -18.34)$ (for the utterance-level estimation). The discrepancy between the linear model $\hat{\zeta}$ and the target value ζ is minute, as shown in Figure 5.4. Also note that the slope $a = 4.94$ is close to the theoretical value 4.343 given in (5.4) in ideal approximation. The advantage of the data-driven method is that it automatically handles the approximation and deviation.

⁵Taylor expansion of $\log(1 - t)$ at $t_0 = 0$.

Table 5.1: Using the range (χ) of the log energy to fit utterance-level SNR values for one utterance corrupted by various levels of additive noises in Aurora 2.0. Here ζ is the target value and $\hat{\zeta}$ is the linearly predicted value.

	clean	20 dB	10 dB	0 dB	-5 dB
min	5.00	11.38	13.32	15.80	16.85
max	19.12	19.12	19.12	19.29	19.71
χ	14.12	7.74	5.80	3.49	2.86
ζ	inf	20	10	0	-5
$\hat{\zeta}$	51.41	19.90	10.31	-1.10	-4.21

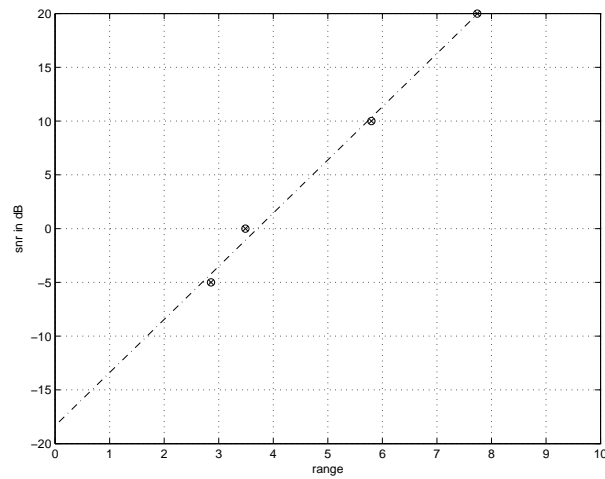


Figure 5.4: Fitting SNR (in dB) values to the range of log energy.

Table 5.2: Statistics of energy-based utterance-level environmental estimation. The numbers of noisy/clean frames are listed in the table.

	Test Set A	Test Set B	Test Set C
clean	0/702k	0/702k	0/351k
20 dB	100/702k	212/702k	11k/340k
15 dB	222k/481k	160k/542k	161k/190k
10 dB	659k/44k	541k/162k	318k/33k
5 dB	702k/0	689k/13k	347k/4k
0 dB	702k/0	700k/2k	351k/0
-5 dB	702k/0	701k/1k	351k/0

The same parameters have been used on the entire test data of Aurora 2.0. Numbers of frames labeled as 0/1 (noisy/clean) with utterance-level estimation based on a 10-dB threshold are summarized in Table 5.2. The proposed scheme works quite well for utterance-level SNR estimation. In particular, 100% clean data are labeled clean and 99.9% very noisy (-5 dB) data are labeled noisy. Note that there is a “grey-zone” with the medium (particularly 15 dB and 10 dB) noise levels, in the sense that the classification can go one way or the other. It is clear from this table that the energy-based environmental estimator works reasonably well.

5.3.3.2 Entropy-based SNR Estimation

As an information-bearing signal, human speech has spectral shapes and temporal patterns different from those of random noises. This distinct property of speech in contrast to random noises has been utilized in automatic end-pointing and speech/non-speech detection [73, 43, 1]. Here this property is used as a cue for noise/clean classification.

From elementary information theory [20], the entropy of a discrete random variable Z with a probability mass function $\{p_i \triangleq Pr(Z = z_i) \mid i = 1 \dots |Z|\}$ is given by

$$H(Z) \triangleq \sum_{i=1}^{|Z|} -p_i \log p_i, \quad (5.8)$$

where $|Z|$ is the cardinality of the support of Z . The entropy quantifies the disorder in the sense that the less uniform the distribution p_i , the lower the entropy $H(Z)$.

The spectral entropy used here is computed as follows: For a given speech window, the power spectrum is converted to a probability mass function by

$$p_i = \frac{1}{\mathfrak{C}} \log(1 + P[i]), \quad i = 0 \dots \frac{N}{2}, \quad (5.9)$$

where $P[i]$ is the power of the i th discrete frequency and \mathfrak{C} is a normalization constant given by

$$\mathfrak{C} \triangleq \sum_{i=0}^{\frac{N}{2}} \log(1 + P[i]). \quad (5.10)$$

The entropy of this probability mass function is computed and then normalized by a factor of $\log(\frac{N}{2} + 1)$ so that the resultant value is between 0.0 and 1.0.

Plots of the spectral entropy of clean and noisy speech samples are given in Figure 5.5. It can be seen that the mean of the entropy of clean speech is lower than that of noisy speech, as expected. More interestingly, the dispersion of entropy across clean speech frames is larger than that across noisy speech frames. An immediate thought is to exploit this temporal information to classify noisy and clean speech. To verify this idea, the mean and variance of the spectral entropy in a context window of one second surrounding a given frame is computed. The results are plotted in Figure 5.6. In this plot, one can see that although clean and noisy samples are not perfectly separated, the overlap is small.

Based on these findings, an environmental condition estimator, which will be used in the experiments in Section 6.3, can be implemented with the novel feature space of the mean and variance of frame-level spectral entropy in a context window. An unsupervised clustering algorithm based on the Euclidean distance is used to cluster the training data into two clusters, one for the clean data and one for the noisy data. Once the cluster points are computed, the test data are classified by the label of the nearest cluster point. These labels are exactly the environmental estimates. Note that this method does not depend on estimating SNR and applying a specific threshold.

A summary of cluster points using the data sets from the four languages of Aurora 3.0 [64] are summarized in Figure 5.7. In order to ensure that a data set includes both

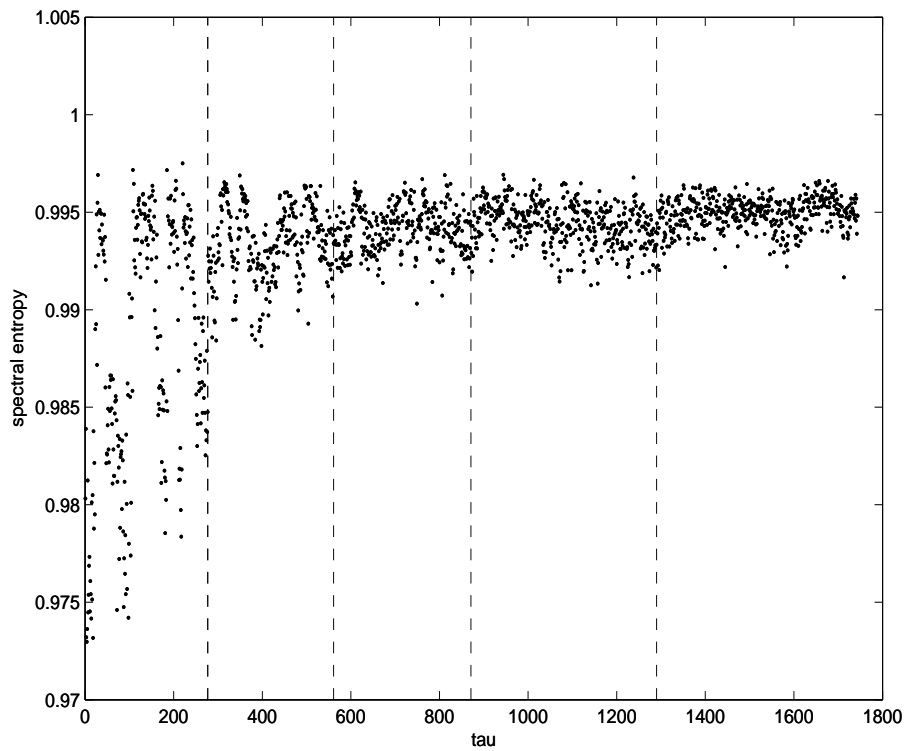


Figure 5.5: The spectral entropy of clean and noisy speech samples. Here each point represents a speech frame. The leftmost region is from clean speech examples (2 utterances). The remaining regions are from noisy speech (20, 15, 10 and 5 dB respectively, 2 utterances each condition). One can clearly see that the mean is lower and the variance is higher in the clean speech than in the noisy speech.

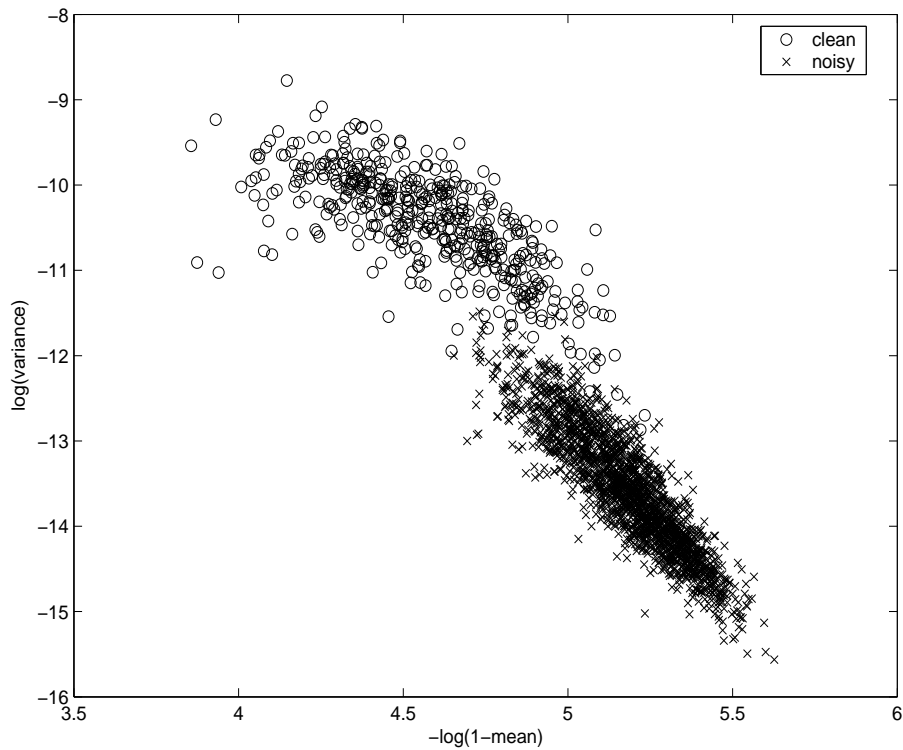


Figure 5.6: Mean-variance plot of the spectral entropy of clean and noisy speech samples. The context window used to calculate mean and variance is one second (100 frames). The upper-left loose cloud corresponds to samples from clean speech while the lower-right dense cloud corresponds to samples from noisy speech samples. In this plot, there are approximately 7000 points, with each point corresponding to a frame. Note that monotonic transforms of the mean and variance are used in plotting this graph. One can see that the clean and noisy samples fall into separate regions in this space.

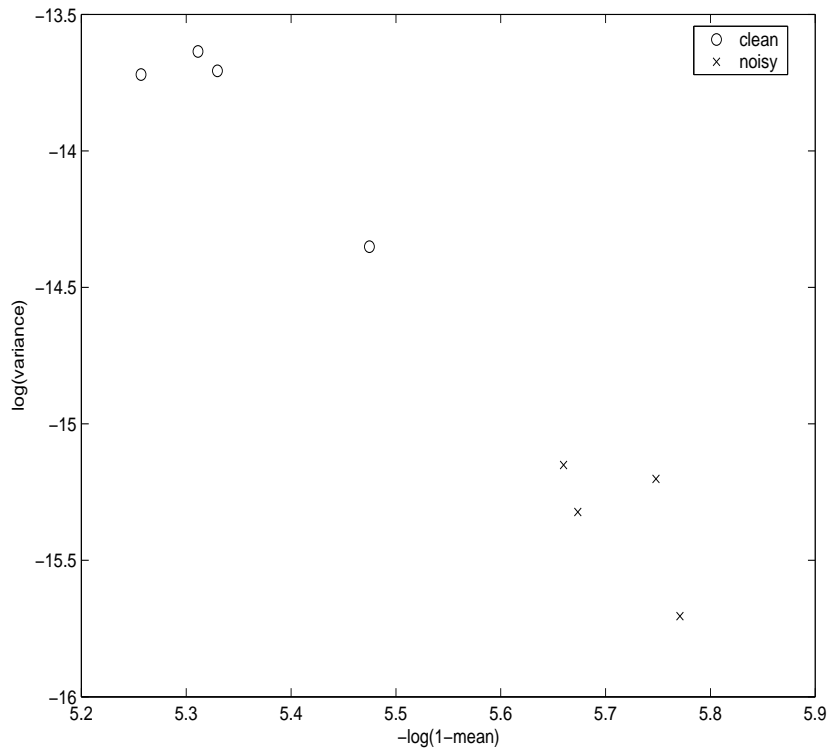


Figure 5.7: Cluster points in the feature space of the mean and variance of spectral entropy using data sets of different languages from the Aurora 3.0 database. One can see that the clean cluster points of different languages are quite close to each other, at the same time separated from the noisy cluster points. That is, the representation of clean/noisy data in this space is quite language-independent.

clean and noisy samples, the training data of well-matched tasks are used.⁶ A significant phenomenon is that the set of clean cluster points and the set of noisy cluster points are well separated. Therefore such representation is quite language-independent and one may exploit this fact to pool data from different languages during clustering.

⁶Again, the well-matched training data consists of speech data of all conditions.

Chapter 6

EXPERIMENTAL RESULTS

The experimental results of the proposed noise-robust schemes for ASR systems are presented in this chapter. These experiments are divided into several sets: MVA on MFCC-based feature sets (Section 6.1), MVA on different feature sets (Section 6.2), DBN-based multi-stream systems (Section 6.3), and the Spine experiments (Section 6.5).

6.1 MVA on the MFCCs

This section is organized as follows. The front-end configurations are defined in Section 6.1.1. The back end is defined in Section 6.1.2. Experimental results are presented in Section 6.1.3. First, the performances of different front-end MFCC configurations are compared in Section 6.1.3.1. Then the best one is used in subsequent investigation of different aspects of MVA. Specifically, the results of multi-domain processing are presented in Section 6.1.3.2, the results of designed filters are presented in Section 6.1.3.3, the results of data-driven filters are presented in Section 6.1.3.4 and the results of the ARMA filters of different orders are presented in Section 6.1.3.5.

6.1.1 Front-End Configurations

Four front-end configurations are evaluated in their combination with MVA. In Configuration 1, MVA is applied on the static features of the log energy ξ and the MFCCs of $C[1]$ to $C[12]$. The dynamic features (delta and delta-delta) are computed based on the MVA-processed static features. Configuration 2 is different from Configuration 1 in that $C[0]$ replaces ξ . The comparison between Configurations 1 and 2 is meant to verify the claim, as given in Section 4.2.2, that MVA is more effective when being applied on $C[0]$ than on ξ . Configuration 3 is different from Configuration 2 in that MVA feature processing is applied

Table 6.1: Four front-end configurations as the leaves of a decision tree. Q1: use ξ ? Q2: apply MVA only to the static features? Q3: band-limit the spectrum?

	Q1	Q2	Q3
Config. 1	yes	yes	yes
Config. 2	no	yes	yes
Config. 3	no	no	yes
Config. 4	no	no	no

to both the static and the dynamic features. The comparison between Configuration 2 and 3 tests whether it improves the performance to apply MVA on the dynamic features as well as the static features. Configuration 4, which is used in [17], is different from Configuration 3 in that it does not apply a band-limiting filter to the spectrum, that it has more mel-frequency filters, and that the mean subtraction is included in the configuration. The main differences between these configurations are summarized in Table 6.1.

6.1.2 Back-End Configuration

A fixed back end is used throughout the experiments in this section. A word is modeled by a whole-word HMM with 16 emitting states. Each of these emitting states has a 3-component Gaussian mixture observation density. The silence model has 3 emitting states, each having 6 Gaussian components. The short-pause model has only one emitting state, which is tied to the middle state of the silence model. This is the standard back-end configuration for the Aurora 2.0 tasks [41].

6.1.3 Results

Note that the *word accuracy rate* is used in presenting the performance level, while the *word error rate* is used in computing the relative performance improvement. This is standard for Aurora evaluation.

6.1.3.1 Comparison of Different Front-End Configurations

The first set of experiments is conducted to compare different front-end configurations specified in Section 6.1.1. The results are presented in Table 6.2 for Aurora 2.0 and in Table 6.3 for Aurora 3.0.

Table 6.2: Word Accuracies (as percentages) of Aurora 2.0 tasks with various configurations (1, 2, 3 and 4) and various feature processings (RAW, MV and MVA). The back end is fixed, with 39-dimensional feature vector, 16 states per word and 3 Gaussian components per state. The results are evaluated on the clean speech, noisy speech with SNR ranging from 0 to 20 dB, and very noisy speech with -5 dB SNR. Top: multi-train tasks. Bottom: clean-train tasks.

Aurora 2.0 multi-train									
	clean			0-20 dB			-5 dB		
	RAW	MV	MVA	RAW	MV	MVA	RAW	MV	MVA
1	98.6	97.9	97.9	85.8	88.1	89.3	23.3	23.3	30.1
2	98.5	98.3	98.1	85.6	90.3	91.0	24.2	35.0	39.4
3	98.5	98.6	98.5	85.4	91.2	92.0	22.4	36.9	42.0
4	98.4	98.6	98.7	88.3	91.0	91.9	25.3	35.1	41.6

Aurora 2.0 clean-train									
	clean			0-20 dB			-5 dB		
	RAW	MV	MVA	RAW	MV	MVA	RAW	MV	MVA
1	99.0	99.0	99.0	60.0	70.0	75.2	8.8	10.9	15.4
2	99.1	99.1	99.1	57.9	74.4	78.8	8.1	12.8	18.9
3	99.0	99.1	99.2	52.7	78.4	83.6	5.8	16.2	24.5
4	99.1	99.2	99.0	65.1	79.0	83.8	11.5	14.5	21.6

It is extremely clear that the application of MVA results in substantial improvements for noisy speech conditions in all inspected configurations, as one can see by comparing the RAW and the MVA columns in the tables. An important point here is that such improvement is dependent on the configuration. On Aurora 2.0, the relative improvement with 0 – 20 dB

Table 6.3: Word accuracies (as percentages) of Aurora 3.0 tasks (with end-pointed data) with various configurations (1, 2 and 3) and various feature processings (RAW, MV and MVA). WM: Well-Matched task; MM: Medium-Mismatched task; HM: Highly-Mismatched task.

German									
	WM			MM			HM		
	RAW	MV	MVA	RAW	MV	MVA	RAW	MV	MVA
1	91.5	94.2	94.7	81.5	87.7	87.0	74.1	86.8	88.4
2	92.1	94.9	95.0	81.5	87.8	88.6	74.5	89.2	90.8
3	91.2	94.8	95.3	79.9	87.8	88.3	71.2	88.3	90.3

Danish									
	WM			MM			HM		
	RAW	MV	MVA	RAW	MV	MVA	RAW	MV	MVA
1	86.1	90.0	90.9	67.7	76.6	79.7	39.9	57.2	67.3
2	87.5	92.1	92.3	68.4	80.2	82.5	39.7	72.1	77.2
3	84.5	91.5	92.2	63.3	79.7	80.5	32.3	71.4	76.9

Finnish									
	WM			MM			HM		
	RAW	MV	MVA	RAW	MV	MVA	RAW	MV	MVA
1	91.7	93.5	94.8	79.8	85.4	88.0	36.4	48.5	63.8
2	91.1	93.6	95.1	78.6	87.4	88.8	57.3	83.9	88.2
3	90.0	91.9	94.6	72.0	85.4	87.6	49.0	80.6	89.5

Spanish									
	WM			MM			HM		
	RAW	MV	MVA	RAW	MV	MVA	RAW	MV	MVA
1	93.1	95.1	95.1	83.1	90.1	91.6	52.8	79.3	82.7
2	92.7	95.6	95.8	86.0	93.9	92.2	43.0	87.2	87.6
3	90.9	95.3	95.8	83.3	91.1	92.5	42.9	86.5	87.6

test data using Configuration 3 is better than using Configuration 2 (45% vs. 38%), which in turn is better than using Configuration 1 (25%) in the multi-train case. Similarly, the relative improvements are 65%, 50% and 38% with Configurations 3, 2, and 1 respectively in the clean-train case. As is analyzed and hypothesized, configuration 2 (with $C[0]$ gives better results than configuration 1 (using ξ). Notice that using more mel-frequency bins and not band-limiting the spectrum does not lead to better performance, as revealed by the comparison between Configurations 3 and 4. Here the band-limiting makes the mel bins distribute over the specified band rather than over the entire band from zero to the Nyquist frequency.

On Aurora 3.0, in all WM, MM, and HM tasks, a similar pattern of performance and improvement persists with this real-world noisy speech database. Specifically, the overall relative improvement changes from 35% to 44% with WM tasks, from 40% to 44% with MM tasks and from 52% to 69% with HM tasks, when the front end changes from Configuration 1 to Configuration 2, i.e., from ξ to $C[0]$. Noteworthy is the extremely big gain in performance in the HM tasks, especially Finnish, with MVA on $C[0]$.

In summary, the following points can be made:

- With noisy test data, MVA improves the performance significantly with very little extra computational cost;
- With clean test data, MVA doesn't hurt the performance when using $C[0]$ (it does, however, hurt a little in the multi-train case using ξ).
- The improvement increases as the noise level and/or the degree of mismatch increases.
- MVA combined with $C[0]$ does lead to a better performance than with ξ , as theoretically predicted in Section 4.2.2.
- Comparing results of Configurations 2 and 3, one can see that for artificially corrupted noisy database of Aurora 2.0, applying MVA on both dynamic and static features is significantly better than applying MVA on static features alone. However, with the real-world noisy database of Aurora 3.0, the results are not so conclusive.

- Comparing results of the MV and MVA columns, one can see that ARMA filtering does provide substantial performance gain for the 0–20 dB test data (91.2% vs. 92.0% with multi-train; 78.4% vs. 83.6% with clean-train).
- Another interesting point is to compare the 0 – 20 dB results of the raw features with Configuration 1 and Configuration 2 (85.8% vs 85.6% with multi-train; 60.0% vs. 57.9% with clean-train). As is pointed out in Section 3.3.2.2, with low noise level and without any processing, ξ can outperform $C[0]$, which is indeed the case here.

In Table 6.3, the relative improvements of MVA over MV are quite different with different languages. In particular, Finnish has a greater gain than Danish, which in turn has a greater gain than German and Spanish. Such difference can be attributed to several issues. First, the collected data in different languages are corrupted by noise of different types and levels, and as will be shown in section 6.1.3.2, the effect of post-processing is more profound when the data is more corrupted. Another issue is the acoustic confusability of different linguistic targets (digits) with each language. As explained in section 6.1.3.5, more acoustical confusability leads to less improvement of ARMA filtering.

6.1.3.2 Results of Multi-Domain Processing

Temporal filtering techniques have been proposed in other domains as well, e.g., RASTA [39]. In order to better understand the merits of such processing in different domains, a generalization of MVA feature processing to a multi-domain processing scheme is investigated. Here the processing can be applied in either the log mel-spectral domain or the cepstral domain. Furthermore, MVA is decomposed into atomic MS, VN and ARMA modules and these modules can be applied in several orders. Figure 6.1 depicts this multi-domain processing system. Specifically, in each domain, the following enumerated processings are considered:

{

1 = N (no processing),

2 = M (MS),

3 = V (VN),

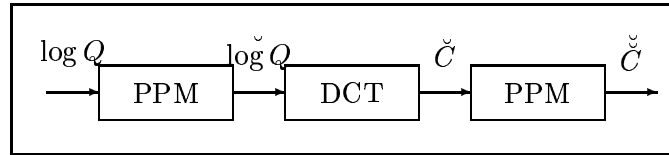


Figure 6.1: Block diagram of multi-domain processing. Here $\log Q$ is the log mel spectrum. PPM represents the processing module that can be switched to one of the options specified in Section 6.1.3.2. $\log \check{Q}$ is the processed log Q . DCT stands for discrete cosine transform, and \check{C} is the cepstral coefficients after multi-domain processing are performed.

4 = A (ARMA),
 5 = MV (MS followed by VN),
 6 = MA (MS followed by ARMA),
 7 = AM (ARMA followed by MS),
 8 = AMV (ARMA followed by MS followed by VN),
 9 = MVA (MS followed by VN followed by ARMA)
 };

The results of Aurora 2.0 and Aurora 3.0 are presented in Figures 6.2–6.7. In these figures, a multi-domain processing scheme is represented as the grid point (x, y) , where x indexes (as specified above) the processing in the log mel-spectral domain and y indexes the processing in the cepstral domain.

The following conclusions can be drawn from these charts:

- VN alone degrades the performance relative to the baseline, as there is a dip at the point $(1, 3)$, representing no processing in the log mel-spectral domain and VN in the cepstral domain. This has been pointed out in Section 3.3.2.3.
- For matched tasks, i.e., WM in Aurora 3.0 and multi-train in Aurora 2.0, variations in performance with respect to different processings are smaller than mismatched tasks. Such is not the case for mismatched tasks. This shows that the detail of signal processing is more relevant with eccentric data than with regular data.
- Comparing $(1, 9)$ and $(9, 1)$, one can see that MVA in the cepstral domain is better

than MVA in the log mel-spectral domain. Furthermore, the difference is bigger in mis-matched tasks than in matched tasks. It is thus crucial to apply MVA in the right domain in mismatched conditions. Note that the analysis in Chapter 4 considers MVA in the cepstral domain rather than in the log mel-spectral domain.

6.1.3.3 Comparison with Designed Filters

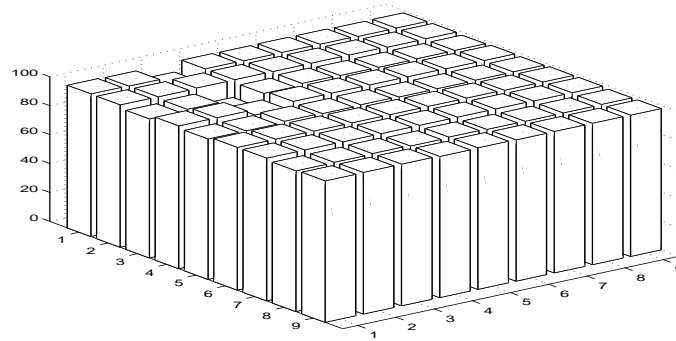
In this section, finite impulse response (FIR) filters are designed to replace the ARMA filter, in order to test whether there is an intrinsic advantage to the ARMA filter. These FIR filters are constrained to have similar complexity as the ARMA filters. In addition, they are designed to have the same normalized bandwidth, with passbands corresponding to low-pass, band-pass or high-pass filters. Specifically, the following cases are considered.

- {
- 1 = low-pass filter with passband [0.0, 0.6],
- 2 = high-pass filter with passband [0.4, 1.0],
- 3 = band-pass filter with passband [0.1, 0.7],
- 4 = band-pass filter with passband [0.2, 0.8],
- 5 = band-pass filter with passband [0.3, 0.9],
- 6 = RAW,
- 7 = MV,
- 8 = MVA
- };

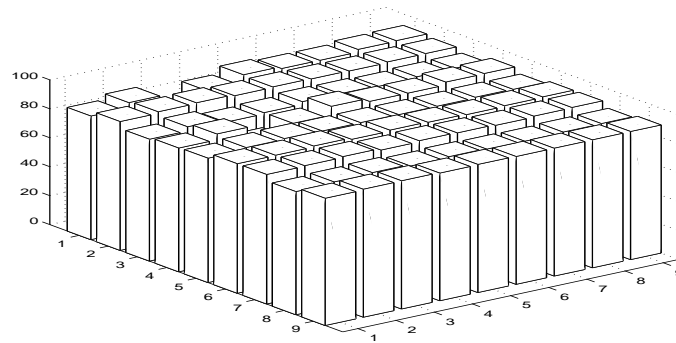
The results are presented in Figures 6.8–6.11. One can see that as the passband moves from the lower frequency to the upper frequency, the performance systematically degrades, confirming a common belief that important information of speech is in the lower frequency part of the cepstral vector time sequence [37]. Overall, the ARMA filter is still better than the best of the designed filters. Note that an ARMA filter can be implemented entirely in place (i.e., no memory needs to be allocated for temporary filter output).¹

¹An algorithm of ARMA filter that processes a time sequence x , assumed to be stored in a data array $x[0] \dots x[n-1]$, essentially updates the data array element from the 0-th to the $(n-1)$ -th. At each

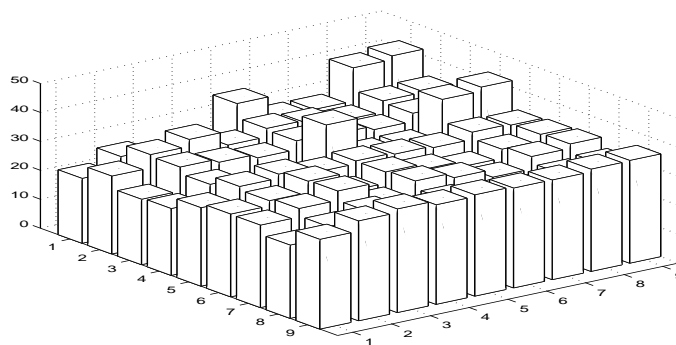
Aurora 2.0 multi-train tasks



clean test data



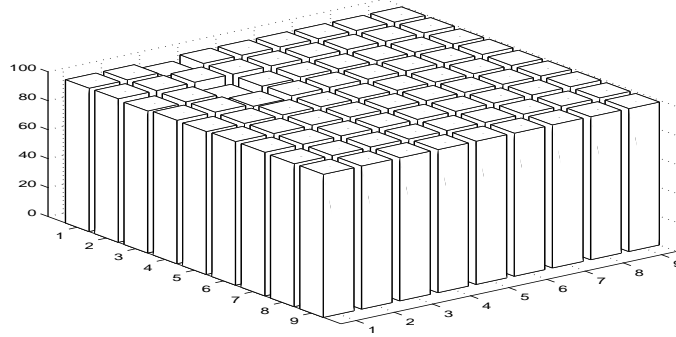
0 – 20 dB noisy test data



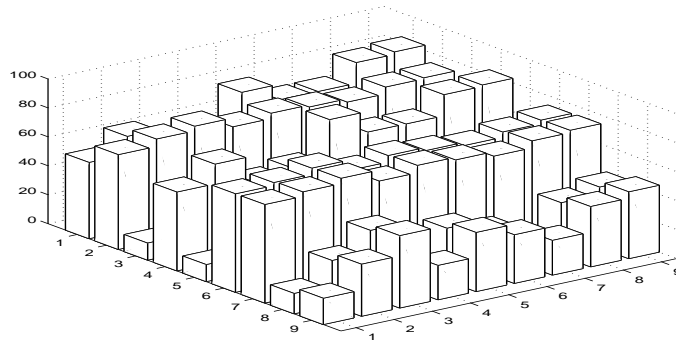
-5 dB test data

Figure 6.2: Word accuracies (as percentages) of Aurora 2.0 multi-train tasks with MVA processing in both log mel spectral and cepstral domains. The x-axis (going northwest-southeast) represents processing in the log mel-spectral domain, while the y-axis (going southwest-northeast) represents processing in the cepstral domain. The grid points on the x and y axes correspond to the processings defined in Section 6.1.3.2.

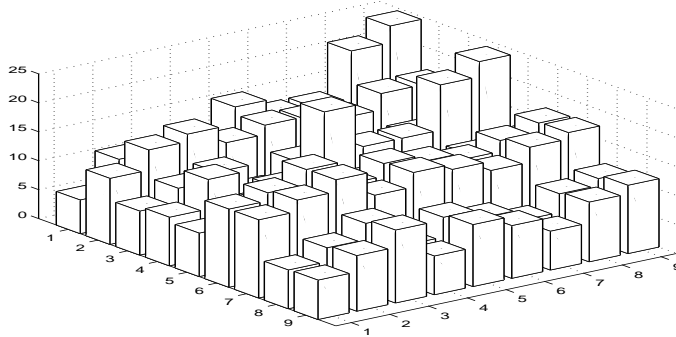
Aurora 2.0 clean-train tasks



clean test data



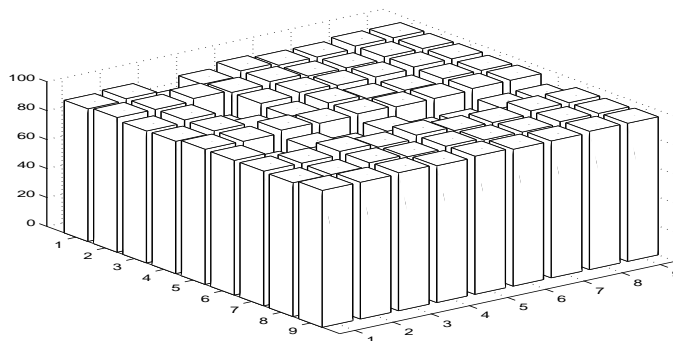
0 – 20 dB noisy test data



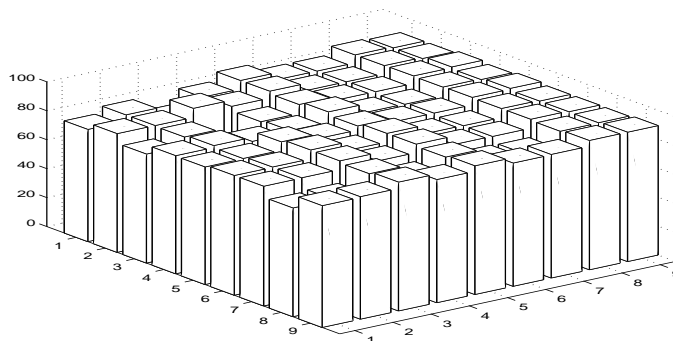
-5 dB test data

Figure 6.3: Word accuracies (as percentages) of Aurora 2.0 clean-train tasks with MVA processing in both log mel spectral and cepstral domains.

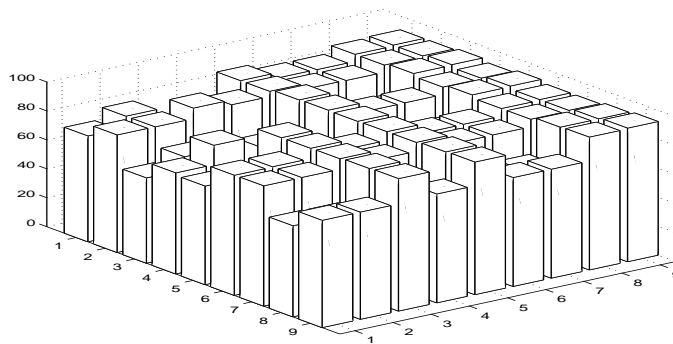
Aurora 3.0 German tasks



Well-Matched



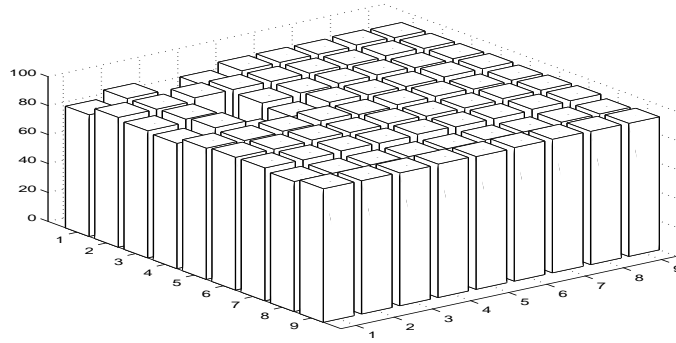
Medium-Mismatched



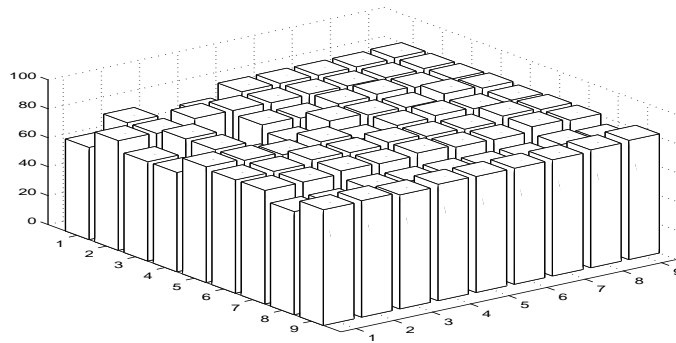
Highly-Mismatched

Figure 6.4: Word accuracies (as percentages) for Aurora 3.0 German tasks with MVA processing in both log mel spectral and cepstral domains.

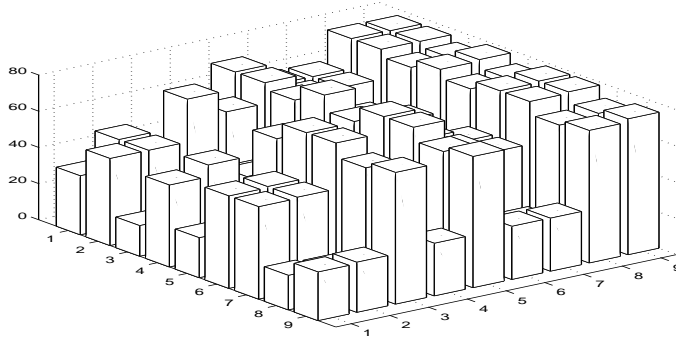
Aurora 3.0 Danish tasks



Well-Matched



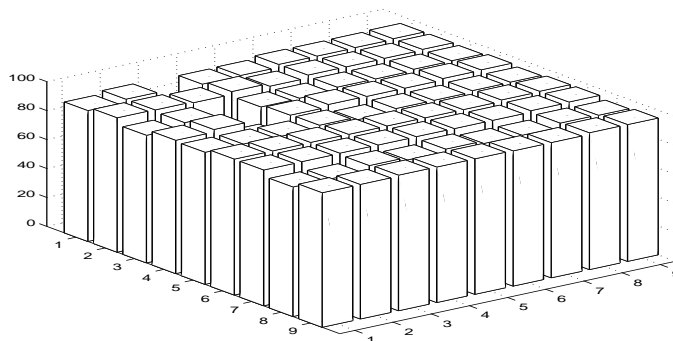
Medium-Mismatched



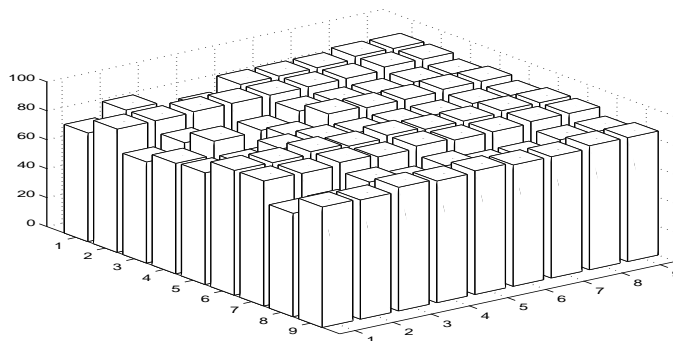
Highly-Mismatched

Figure 6.5: Word accuracies (as percentages) for Aurora 3.0 Danish tasks with MVA processing in both log mel spectral and cepstral domains.

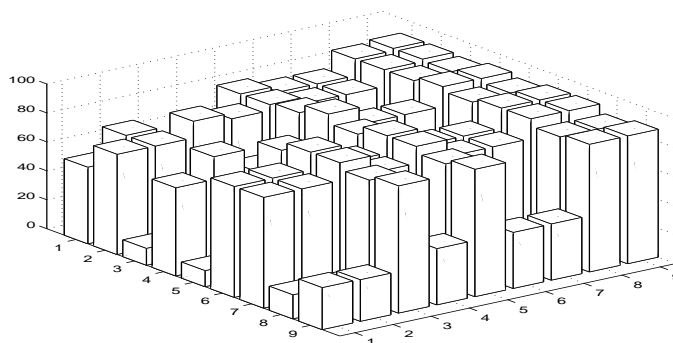
Aurora 3.0 Finnish tasks



Well-Matched



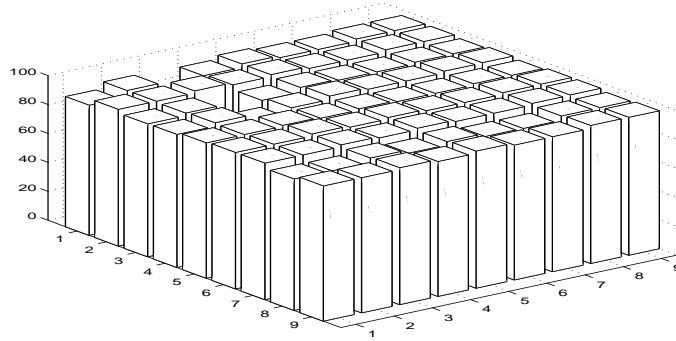
Medium-Mismatched



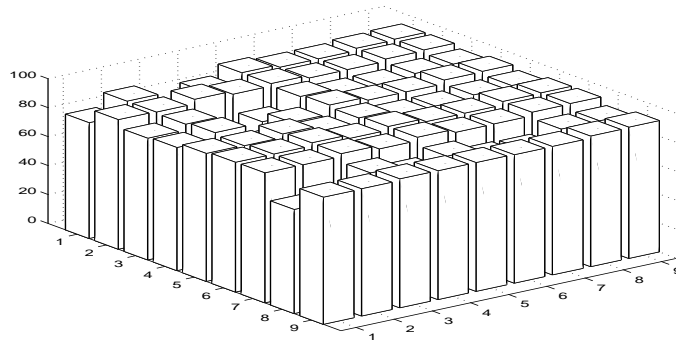
Highly-Mismatched

Figure 6.6: Word accuracies (as percentages) for Aurora 3.0 Finnish with MVA processing in both log mel spectral and cepstral domains.

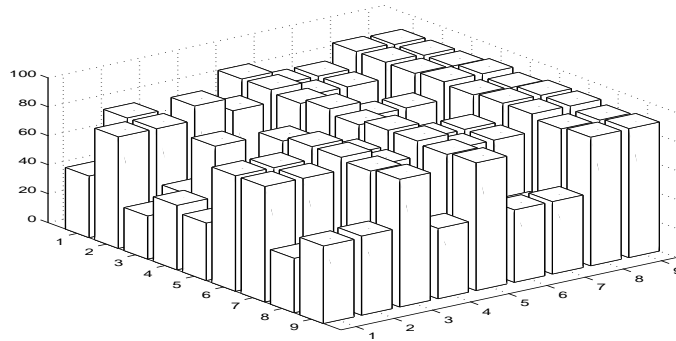
Aurora 3.0 Spanish tasks



Well-Matched



Medium-Mismatched



Highly-Mismatched

Figure 6.7: Word accuracies (as percentages) for Aurora 3.0 Spanish tasks with MVA processing in both log mel spectral and cepstral domains.

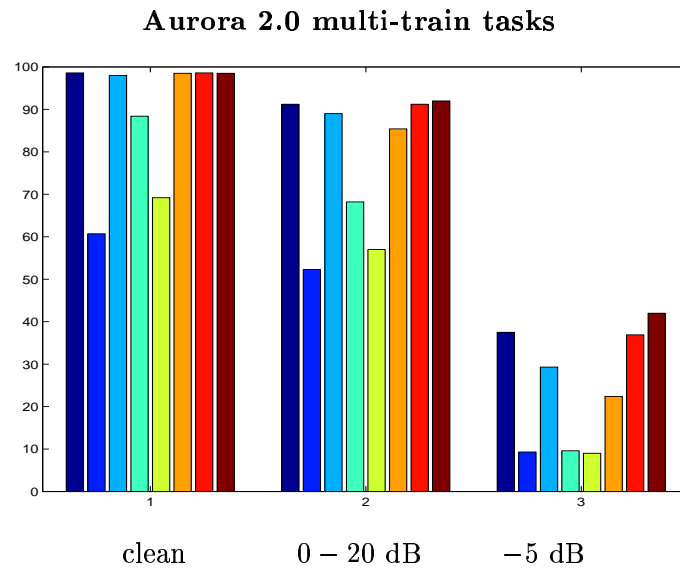


Figure 6.8: Word accuracies (as percentages) of Aurora 2.0 multi-train tasks with the ARMA filter replaced by various designed FIR filters. Group 1 is clean test data, group 2 is 0 – 20 dB test data, and group 3 is –5 dB test data. Within each group, the bars from left to right correspond to the items defined in Section 6.1.3.3.

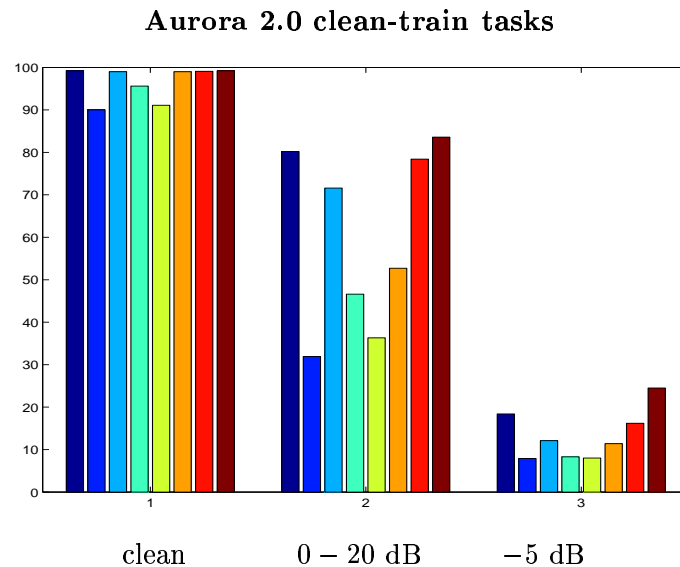


Figure 6.9: Word accuracies (as percentages) of Aurora 2.0 clean-train tasks with the ARMA filter replaced by various designed FIR filters. Same legend as in Figure 6.8.

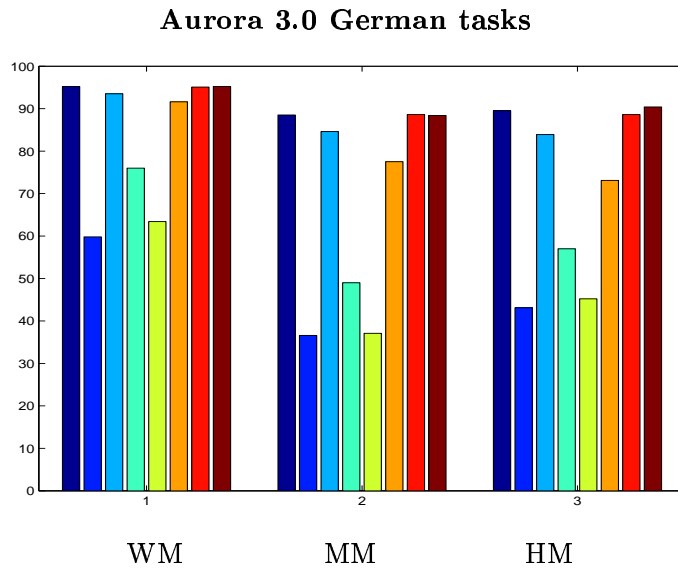


Figure 6.10: Word accuracies (as percentages) for Aurora 3.0 German with the ARMA filter replaced by various designed FIR filters. Group 1 is the WM task, group 2 is the MM task, and group 3 is the HM task. Within each group, the bars from left to right correspond to the items defined in Section 6.1.3.3.

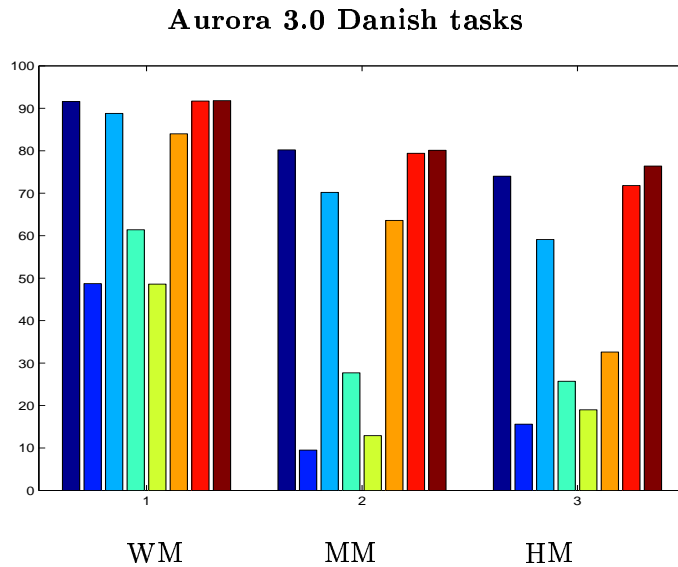


Figure 6.11: Word accuracies (as percentages) for Aurora 3.0 Danish with the ARMA filter replaced by various designed FIR filters. Same legend as in Figure 6.8.

6.1.3.4 Comparison with Data-Driven Filters

To further investigate the advantage of using the ARMA filter, a data-driven methodology is used to design the filter. To keep the same complexity as the second-order ARMA filter, a five-tap data-driven filter minimizing the squared error between the filtered noisy cepstrum and the clean cepstrum is applied to the feature sequences after the feature processing of MV. Explicitly, the solution of the following minimization problem is sought,

$$\min_a \sum_{u=1}^U \sum_{\tau=1}^{T_u} \left(X_u^{(\tau)} - \sum_i a_i Y_u^{(\tau+i)} \right)^2, \quad i = -2, \dots, 2, \quad (6.1)$$

where $a = \{a_i\}$ is the set of filter coefficients, U is the number of utterance pairs, T_u is the number of frames in utterance u , and $X_u^{(\tau)}$ and $Y_u^{(\tau)}$ are the feature vectors at frame τ in the pair of utterances corresponding to the clean and the noisy environments. If the following quantities are defined,

$$\begin{aligned} r_{XY}(i) &\triangleq \frac{\sum_u \sum_{\tau} X_u^{(\tau)} \cdot Y_u^{(\tau+i)}}{\sum_u \sum_{\tau} 1}, \quad i = -2, \dots, 2 \\ r_{YY}(i) &\triangleq \frac{\sum_u \sum_{\tau} Y_u^{(\tau)} \cdot Y_u^{(\tau+i)}}{\sum_u \sum_{\tau} 1}, \quad i = 0, \dots, 4, \end{aligned} \quad (6.2)$$

where r_{XY} is the cross-correlation of X and Y and r_{YY} is the auto-correlation of Y , then the solution of (6.1) is

$$a = R^{-1} * r_{XY}, \quad (6.3)$$

where R is the Toeplitz matrix formed from r_{YY} . The optimal 5-tap non-causal FIR filter thus found by the 8440 pairs of utterance of Aurora 2.0 training data is

$$a = \{0.1090, 0.1265, 0.4003, 0.1667, 0.0127\}.$$

Interestingly, the frequency response of this filter bears a strong similarity to the second-order ARMA filter, in that both are low pass filters and the main lobe of the frequency response extends from 0 to about 0.6 in the normalized frequency. Indeed, the case in which

position, the stored value is updated by the sum of elements in a filtering window, weighted by the filtering coefficients. This is in contrast to an FIR filter with the same number of coefficients, where a filtered element can be saved to the data array only after it goes out of the process window; the latter requires a temporary storage for processed output.

Table 6.4: Word accuracies (as percentages) for Aurora 2.0 with data-driven minimum-squared-error filter. DDT1 is the case where a single filter is derived and used for all cepstral components, while DDT2 is the case where each cepstral coefficient has its own filter. In other words, in DDT1 all components use the same filter, while in DDT2 different components use different filters.

	multi-train			clean-train		
	clean	0 – 20 dB	–5 dB	clean	0 – 20 dB	–5 dB
ARMA	98.5	92.0	42.0	99.2	83.6	24.5
DDT1	98.5	91.9	40.6	99.1	83.3	23.7
DDT2	98.5	91.8	40.2	99.1	82.8	23.1

each feature component has its own filter is also investigated, and these component-wise filters too have similar frequency responses. Experimental results with the data-driven filters are summarized in Table 6.4. The result is that the simple ARMA filter still outperforms the others.

6.1.3.5 Comparison of ARMA Orders

Also investigated is how the order of the ARMA filter, “ m ” in Equation (4.3), affects the performance, as there is an inherent trade-off in choosing the order. That is, a small m retains the short-term speech information but is vulnerable to noises, while a large m makes the processed features robust to noise corruption, but the short-term speech information is more likely to be lost. Intuitively, the most extreme cases of $m = 0$ or $m \gg 1$ will not have the best performance with noisy speech data, suggesting that the optimal order is a small positive integer.

The experimental results are summarized in Figures 6.12–6.17. From these figures, one can see that other than the very noisy test data in Aurora 2.0 and HM tasks in Aurora 3.0, the optimal ARMA order is a small integer. For the very noisy test data in Aurora 2.0 and HM tasks in Aurora 3.0, the improvement when the order is incremented quickly decreases beyond $m = 2$. It is thus verified that a small integer, $m = 2$, is a good choice for the order

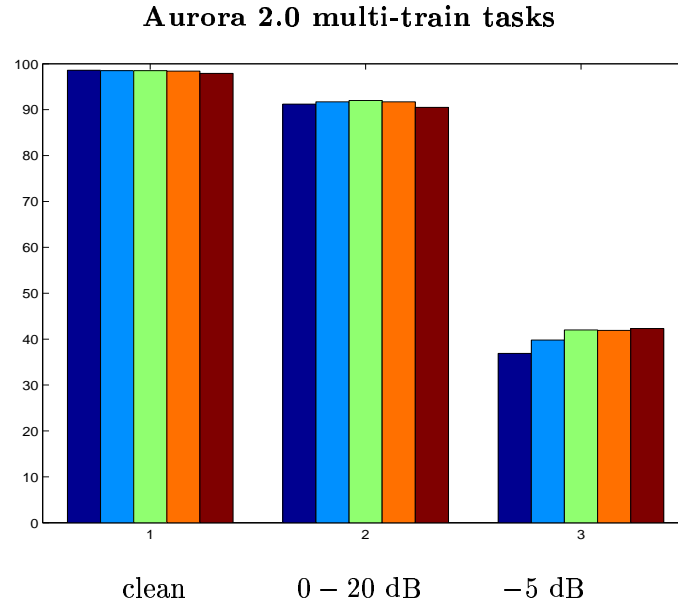


Figure 6.12: Word accuracies (as percentages) of Aurora 2.0 multi-train tasks with various orders of the ARMA filter. Within each group, the bars from left to right correspond to order 0, 1, 2, 3, 4 respectively.

of the ARMA filter.

Note that the performance ranking of different orders does depend on the tasks. When the training/test data are highly mismatched (such as the HM tasks of Aurora 3.0 and clean train task of Aurora 2.0) or when the test data is very noisy (such as the -5 dB SNR test set), the higher-order ARMA filters tend to be more effective than the lower order ones.

6.2 MVA on Other Features

The MVA processing on the MFCC-based front ends are intensively investigated in the last section. In this section, the focus is shifted to MVA processing rather than MFCC. In other words, to test the generality, MVA is blindly applied to distinctive feature sets, including auditory-based MFCC, articulation-based LPC, and perception-based PLP. In addition, feature sets derived from basic features such as RASTA, tandem, modulation-filtered spectrogram, and modulation cross-correlagram are also investigated. The goal is to find out characteristics of feature sets which do or do not combine well with the MVA

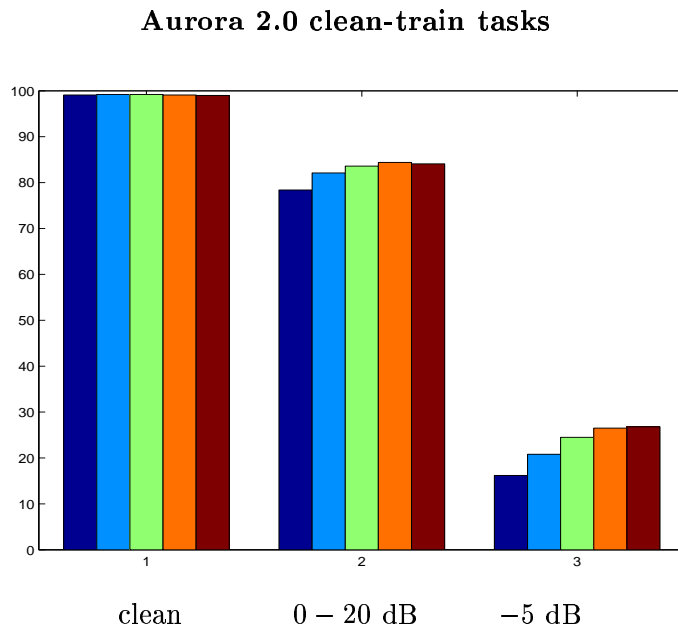


Figure 6.13: Word accuracies (as percentages) of Aurora 2.0 clean-train tasks with various orders of the ARMA filter. Same legend as in Figure 6.12.

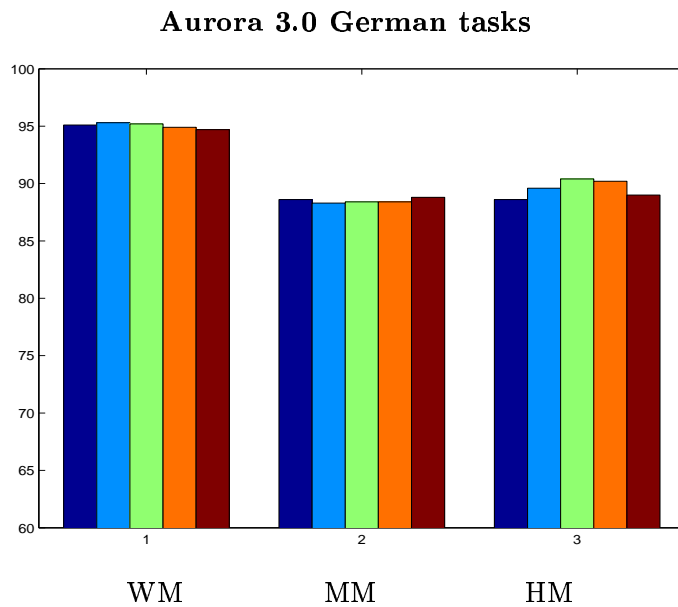


Figure 6.14: Word accuracies (as percentages) of Aurora 3.0 German tasks with various orders of the ARMA filter. Group 1 is the WM task, group 2 is the MM task, and group 3 is the HM task. Within each group, the bars from left to right correspond to orders 0, 1, 2, 3, 4 respectively.

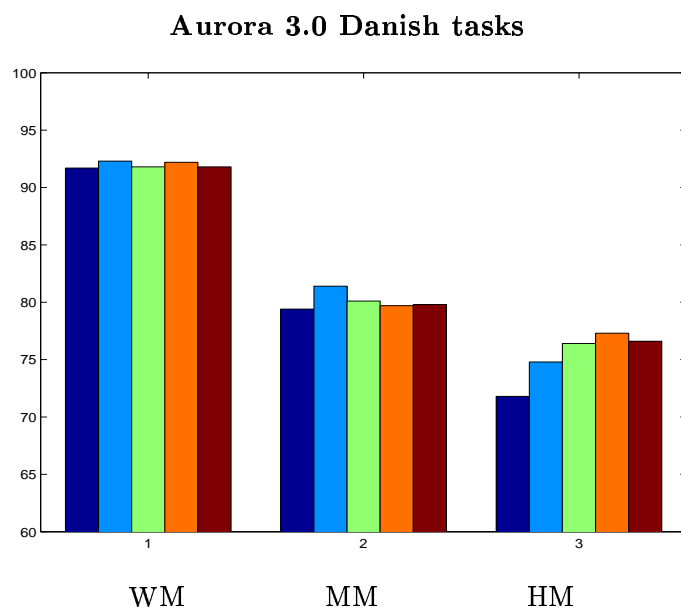


Figure 6.15: Word accuracies (as percentages) of Aurora 3.0 Danish tasks with various orders of the ARMA filter. Same legend as in Figure 6.14.

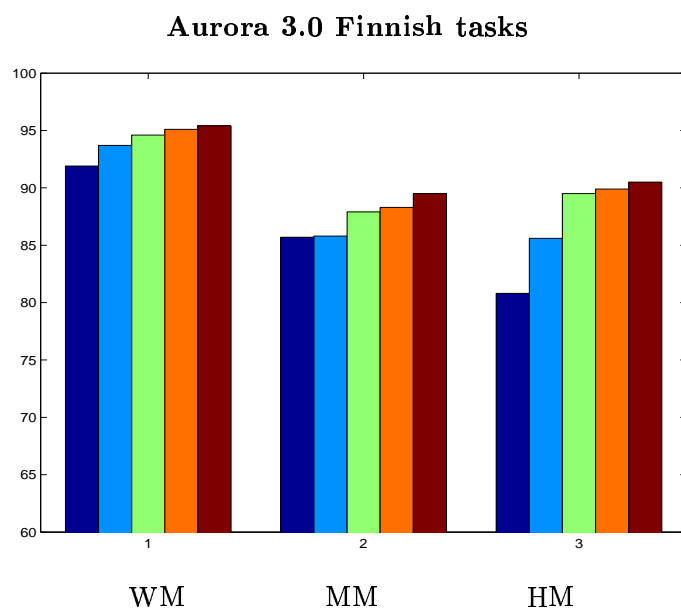


Figure 6.16: Word accuracies (as percentages) of Aurora 3.0 Finnish with tasks various orders of the ARMA filter. Same legend as in Figure 6.14.

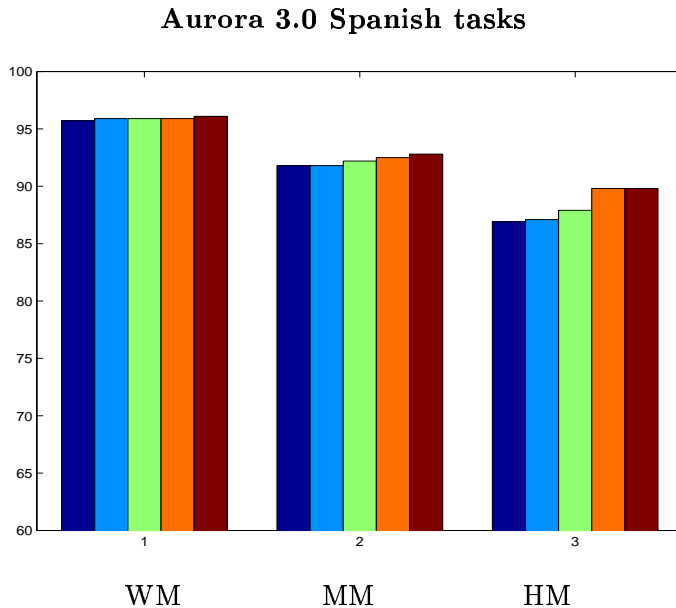


Figure 6.17: Word accuracies (as percentages) of Aurora 3.0 Spanish tasks with various orders of the ARMA filter. Same legend as in Figure 6.14.

processing.

In the experiments, the front end extracts features which may have different dimensions in different cases. The extracted features (referred to as RAW²) are subject to the processing of mean subtraction (M), followed by variance normalization (MV), followed by ARMA filtering (MVA). The back end uses whole-word HMMs, with 16 emitting states for a word model, 3 states for the silence model, and 1 state for the short-pause model. The observation density function for an emitting state is a Gaussian mixture with up to 16 components. Note that this back end is different from the one defined in 6.1.2, which uses 3 components per state.

6.2.1 MFCC

The first feature set that is tested with the combination of MVA is the MFCC. This is the most common feature set and can establish a performance reference to which other feature

²This is previously referred to as N, “no processing”.

Table 6.5: Evaluation on the MFCC features. The results are listed with respect to three properties: what training set is used (multi-train or clean-train), how noisy the test data is (clean, 0 – 20 dB or –5 dB), and what processing to the features is applied (RAW, M, MV or MVA). The numbers in the table are the word accuracy rates.

	multi-train			clean-train		
	clean	0-20dB	-5dB	clean	0-20dB	-5dB
RAW	99.21	88.25	22.27	99.65	54.28	7.23
M	99.48	91.64	30.69	99.72	68.58	3.94
MV	99.33	92.91	41.11	99.66	81.99	19.44
MVA	99.34	93.44	46.49	99.66	85.20	27.24

sets can be compared. The feature vector contains 12 MFCCs enhanced by the zeroth MFCC, along with the delta and delta-delta components.

The results with MFCC is presented in Table 6.5. One can see that MVA improves significantly over RAW. With the 0 – 20 dB noisy test data, MVA improves by 44.2% relative in the multi-train case and 67.6% in the clean-train case. Comparing MV and MVA, the ARMA filtering improves by 7.5% relative in the multi-train case and 17.8% in the clean-train case.

6.2.2 LPC

The LPCs represent the quasi-stationary process in a speech analysis window based on an all-pole model of articulation, while the MFCCs are based on the spectral information and human auditory knowledge (especially the mel-frequency). Therefore LPCs and MFCCs are quite different and there are no apparent reasons that they will be corrupted in similar ways with the presence of noise. It is not clear how noise influences the articulation and how the LPCs are corrupted. The feature set contains 12 LPC coefficients and the log energy, along with the delta and delta-delta components.

The results with the LPC features are summarized in Table 6.6. MVA again improves significantly over the RAW features. Specifically, it improves 26.9% in the multi-train case

Table 6.6: Evaluation on the LPC features.

	multi-train			clean-train		
	clean	0-20dB	-5dB	clean	0-20dB	-5dB
RAW	89.76	63.34	3.90	93.11	21.22	5.24
M	88.76	71.92	15.97	93.52	39.96	6.52
MV	87.19	71.56	1.10	93.39	39.69	4.53
MVA	87.14	73.21	6.06	93.33	42.17	0.94

Table 6.7: Evaluation on the LPC-Cepstral features.

	multi-train			clean-train		
	clean	0-20dB	-5dB	clean	0-20dB	-5dB
RAW	99.13	87.21	23.92	99.41	56.50	3.61
M	99.20	90.06	32.29	99.59	71.26	7.33
MV	98.88	90.12	31.70	99.52	80.39	17.52
MVA	98.86	90.71	36.43	99.46	82.61	23.97

and 26.6% in the clean-train case, with the 0 – 20 dB test tasks. Comparing MV and MVA, the ARMA filtering improves by 5.8% relative in the multi-train case and 4.1% in the clean-train case.

Overall, the performance level of the LPCs is not as well as that of the MFCCs. Furthermore, the improvement introduced by MVA over RAW features is not as significant as in the case of MFCC.

6.2.3 LPC-CEPSTRA

The observation that the LPCs do not perform as well as the MFCCs leads to evaluating the cepstral features of LPC, the LPC-CEPSTRA. The feature vector contains 12 LPC cepstral coefficients derived from 12 LPCs and the log energy, along with their delta and delta-delta.

The results with LPC-CEPSTRA are summarized in Table 6.7. MVA again improves significantly over the RAW features. Specifically, it improves 27.4% in the multi-train case and 60.0% in the clean-train case, with the 0 – 20 dB test tasks. Comparing MV and MVA, the ARMA filtering improves relatively 6.0% in the multi-train case and 11.3% in the clean-train case.

Overall, LPC-CEPSTRA performs substantially better than LPCs, although still not as good as MFCCs. Apparently, applying the discrete cosine transform (DCT) to the LPCs results in features more germane to the back-end configuration of HMM with Gaussian mixture densities. In addition, MVA shows a better relative improvement over RAW in the LPC-CEPSTRA than in the LPCs, especially in the (mismatched) clean-train case. That is, DCT leads to a better baseline and a better relative improvement.

6.2.4 Tandem

The tandem feature set is deemed one of the best feature sets with the Aurora 2.0 databases. It is included in the evaluation of MVA to see how a completely different feature set react to MVA processing. The feature vector is 24-dimensional corresponding to 24 phone classes, as explained in [26]. In order to extract the tandem features, two neural nets need to be pre-trained. Here both the case where the nets are trained by multi-train data and the case where the nets are trained by the clean-train data are investigated.³

The results with the tandem features are summarized in Table 6.8. The top part is the case where the neural nets are trained by the multi-train data set (referred to as Tand-M) and the bottom part is the case where the neural nets are trained by the clean-train data set (referred to as Tand-C).

Overall, with noisy test data, Tand-M performs better than Tand-C. This shows that the mismatch in the performance can be attributed, at least partially, to the trained neural nets with mismatched data. Furthermore, feature processing is capable of bridging such mismatch of performance. This is evidenced by the observation that the relative difference of the two cases is larger with the RAW features than with the processed features. Here the

³Thanks to Dan Ellis, the designer of the tandem feature set, for generating the RAW tandem features.

Table 6.8: Evaluation on the Tandem features. Top: Nets trained on multi-train data. Bottom: Nets trained on clean-train data.

	multi-train			clean-train		
	clean	0-20dB	-5dB	clean	0-20dB	-5dB
RAW	99.50	92.87	40.19	99.63	85.84	17.28
M	99.58	93.66	44.31	99.68	90.05	27.79
MV	99.50	93.69	44.48	99.65	90.78	32.06
MVA	99.57	93.68	44.61	99.67	91.15	35.33

	multi-train			clean-train		
	clean	0-20dB	-5dB	clean	0-20dB	-5dB
RAW	99.45	89.34	27.57	99.62	71.77	8.69
M	99.57	91.47	40.78	99.65	83.81	20.87
MV	99.51	91.39	40.87	99.68	83.15	17.14
MVA	99.55	91.25	40.60	99.64	83.41	21.68

Table 6.9: Evaluation on the PLP features.

	multi-train			clean-train		
	clean	0-20dB	-5dB	clean	0-20dB	-5dB
RAW	99.40	89.81	28.53	99.65	62.05	6.98
M	99.46	92.47	37.26	99.69	71.35	5.84
MV	99.29	93.01	44.07	99.61	83.71	21.48
MVA	99.28	93.20	47.66	99.67	85.68	28.40

improvement is mostly accounted for by M (mean subtraction). Finally, it is interesting to look at the case where the neural nets are trained by the clean-train data only (Tand-C) and thus learn no information about how noisy data is like. Although such neural nets are not expected to be robust to noise, it turns out that Tand-C nevertheless outperforms MFCC.

6.2.5 PLP

The design of the PLP features incorporates human auditory characteristics such as the equal-loudness curve and the power law of hearing. Note that the PLPs used here are based on the mel-frequency filter banks and are the cepstral coefficients (via HTK).

The results of the PLP features are summarized in Table 6.9. MVA still improves significantly over the RAW features. Specifically, it improves 33.3% in the multi-train case and 62.3% in the clean-train case, on the 0 – 20 dB test tasks. Comparing MV and MVA, the ARMA filtering improves relatively 2.7% in the multi-train case and 12.1% in the clean-train case. Here it is interesting to compare the MFCC and PLP features. Without any feature processing, the PLP is significantly better than the MFCC (89.81% vs. 88.25% in multi-train and 62.05% vs. 54.28% in clean-train). However, with MVA feature processing, the advantage of PLP either disappears or significantly decreases (93.20% vs. 93.44% in multi-train and 85.68% vs. 85.20% in clean-train).

Table 6.10: Evaluation on the MSG features.

	multi-train			clean-train		
	clean	0-20dB	-5dB	clean	0-20dB	-5dB
RAW	97.42	86.37	18.05	98.90	56.06	-2.19
M	97.39	86.82	25.70	98.86	66.41	6.29
MV	97.50	86.29	15.98	98.59	57.78	6.08
MVA	97.74	86.53	19.93	98.84	60.19	6.40

6.2.6 Modulation Spectrogram

The modulation-filtered spectrogram (MSG) [37] computes the 4-Hz spectral energy of filtered modulation amplitude of each critical band. The main idea is to “focus on the elements in the signal encoding phonetic information,” which change at a typical rate between 0-8 Hz corresponding to articulatory configuration. The signal processing in MSG has evolved from its original setting for improved performance [50]. Here the so-called “msg3” features, which are extracted by the the SPRACHcore software release from ICSI, are used.

The results of MSG features are presented in Table 6.10. Without any feature processing, the performance level is similar to that of MFCC. With feature processing, there are no significant performance gains. Note that in the processing of “msg3” features, an on-line normalization of mean and variance has already been implemented. Furthermore, the modulation amplitude filtering is essentially a hi-reject filtering which is similar to ARMA. Also note that to be used in an HMM recognizer the MSG features are often further processed by neural networks, which is not the case here.

6.2.7 Modulation Cross-Correlagram

The modulation cross-correlagram (MCG) [9] features are based on the cross-correlation of the magnitude sequences in different spectral channels. A two-dimensional DCT is further applied to the cross-correlation matrix and the lowest-order 6×6 sub-matrix of the DCT constitutes the final 36-dimensional feature vector.

Table 6.11: Evaluation on the MCG features.

	multi-train			clean-train		
	clean	0-20dB	-5dB	clean	0-20dB	-5dB
RAW	90.29	65.85	15.94	90.89	57.77	10.29
M	89.30	65.64	15.41	92.32	59.07	8.61
MV	91.00	66.20	14.36	93.44	60.98	7.40
MVA	90.73	66.16	15.62	93.02	60.88	9.99

Table 6.12: Evaluation on the RASTA features.

	multi-train			clean-train		
	clean	0-20dB	-5dB	clean	0-20dB	-5dB
RAW	99.50	90.95	30.97	99.72	61.59	7.93
M	99.52	92.28	34.06	99.76	73.16	5.70
MV	99.18	93.18	45.18	99.70	84.20	21.95
MVA	99.34	93.25	48.37	99.66	85.40	28.15

The results with MCG are presented in Table 6.11. In this case, the feature processing only introduces minor improvement over the raw features, if any.⁴

6.2.8 RASTA

Relative SpecTrA (RASTA) [39] is a filtering technique applied in a domain of the (compressed) critical-band spectral envelopes. It is designed to remove the slow-varying environmental variations and the fast-varying artifacts in speech processing. The 39-dimensional plain RASTA-PLP (instead of j-RASTA or log-RASTA) is used in this evaluation.

The results for the RASTAs are presented in Table 6.12. Without any feature processing, the performance level is better than that of MFCC in 0–20 dB test data, when one compares

⁴Note that in the experiments of clean-train with raw MCG features, on average there are only 7 components per mixture. Further splitting the components results in vanishing variances.

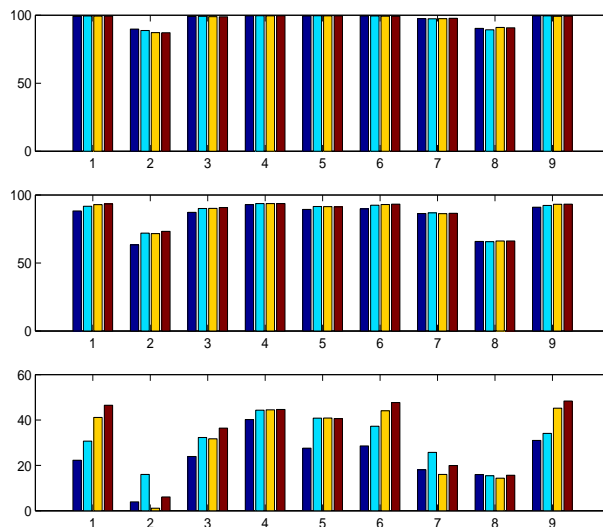


Figure 6.18: The comparison of feature sets with the multi-train tasks. The three blocks correspond to clean, 0 – 20 dB and –5 dB test data, respectively. Within each block, the abscissa is the enumerated feature set (1 = MFCC; 2 = LPC; 3 = LPC-CEPSTRA; 4 = Tand-M; 5 = Tand-C; 6 = PLP; 7 = MSG; 8 = MCG; 9 = RASTA), while the ordinate is the word accuracy rates. Within each feature set, the four bars in a bar group from left to right correspond to RAW, M, MV, and MVA.

RAW-RASTA and RAW-MFCC. With MVA feature processing, the performance level of RASTA is virtually the same as those of MFCC, when one compares MVA-RASTA and MVA-MFCC. Specifically, the ARMA filter introduces a significant performance boost with MFCC but only a minute gain with RASTA. In a sense the RASTA filtering has done the work for the ARMA filtering.

6.2.9 Comparison Across Feature Sets

In addition to the individually presented results, comparison across feature sets is conducted in this section, as the objective of this work is not only to investigate feature sets but also to identify the characteristics of the features that work well with MVA.

An overall summary over the experimented feature sets is presented in Figure 6.18 for the multi-train tasks and in Figure 6.19 for the clean-train tasks. These figures clearly show the absolute performance levels of different feature sets.

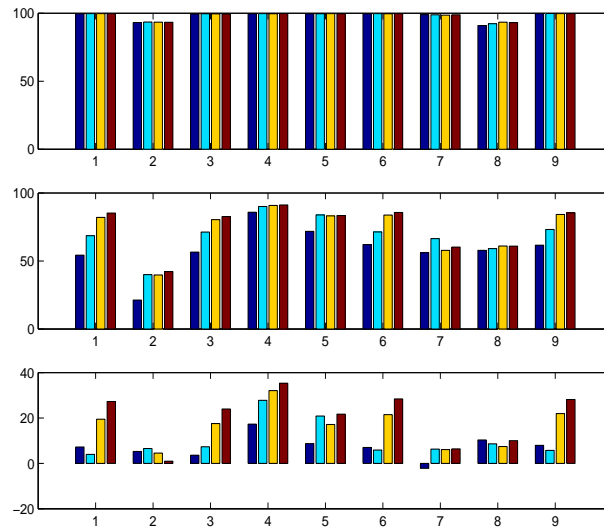


Figure 6.19: Same legend as the previous figure (Figure 6.18). The only difference is that the results presented here are those of the clean-train tasks.

Generally speaking, the performance rankings of feature sets in different tasks (i.e. different combinations of train/test data) are quite similar. The relative ranking is:

$$1 \approx 3 \approx 4 \approx 5 \approx 6 \approx 9 \succeq 7 > 8 \succeq 2$$

(using the same enumeration of features as in Figure 6.18).

The relative improvements of MVA over RAW are summarized in Table 6.13. Generally speaking, with clean test data, the performance may degrade with MVA in individual cases, but with noisy and/or mismatched data, the performance is boosted by MVA. The feature sets can be divided into three classes based on the performance gains on the 0 – 20 dB test data when MVA is applied as follows:

- **Feature sets with substantial performance gains.** The MFCC, LPC-CEPSTRA, PLP and RASTA features belong to this category. On average MVA achieves 63% relative improvements in clean-train and 33% in multi-train cases, on top of decent baseline results of the raw features.
- **Feature sets with medium performance gains.** The LPCs and Tandem features

Table 6.13: Relative improvements of MVA over RAW on all features.

	multi-train			clean-train		
	clean	0-20dB	-5dB	clean	0-20dB	-5dB
MFCC	16.46	44.17	31.16	2.86	67.63	21.57
LPC	-25.59	26.92	2.25	3.19	26.59	-4.54
LPC-C	-31.03	27.37	16.44	8.47	60.02	21.12
Tand-M	14.00	11.36	7.39	10.81	37.50	21.82
Tand-C	18.18	17.92	17.99	5.26	41.23	14.22
PLP	-20.00	33.27	26.77	5.71	62.27	23.03
MSG	12.40	1.17	2.29	-5.45	9.40	8.41
MCG	4.53	0.91	-0.38	23.38	7.36	-0.33
RASTA	-32.00	25.41	25.21	-21.43	61.99	21.96

(including Tand-M and Tand-C) belong to this category. On average MVA achieves 35% relative improvements in clean-train and 18% in multi-train cases.

- **Feature sets with minute or no performance gains.** The MCG and MSG belong to this category. On average MVA achieves 8% relative improvements in clean-train and 1% in multi-train cases.

6.2.10 Summary

In this section, the MVA feature processing scheme is applied to feature sets of different natures. The results show that MVA works well in the majority of the tested features, especially in highly noisy and/or mismatched data. It is also shown that the performance gain of certain noise-robust features can be improved by performing temporal integration in the final stage, as is done by MVA processing, effectively utilizing information from time spans longer than a typical analysis window. Here the working hypothesis is that *appropriately extracting information across multiple analysis windows is a fundamental property*

of noise-robust ASR systems, at least in small-vocabulary ASR tasks. This agrees with a recent research trend to integrate informations at different time scales [40]. MVA, unlike many other methods, does this in a simple and effective way, and is applicable to any feature extraction method.

6.3 DBN ASR Systems

The proposed noise-robust back ends and the corresponding implementations of multi-stream ASR systems have been described in Section 5.3. In this section, further details are elaborated and experimental results are presented.

6.3.1 Experimental Setup

As described in Section 5.3, the feature observation consists of two component feature streams – MV and MVA. The raw features used in the experiments are the MFCCs, along with the delta and delta-deltas (Configuration 3).

Whole-word models are adopted for the experiments. Each word model has 16 emitting states. The silence model has 3 emitting states. The short-pause model has only one emitting state (*not* tied to the middle state of the silence model). Each state-emitting density is a Gaussian mixture distribution of up to 16 Gaussian components.

During training, the parameters of the state-emitting density functions of both feature streams and the Markov chain transition probability matrix are jointly learned. To use all data during training, the conditional distributions of the feature observations are independent of the feature selector R . During testing, the feature selector is active in selecting the corresponding feature in a frame.

6.3.2 Results and Discussion

6.3.2.1 Baseline

The baseline systems in the evaluation of DBN-based systems are the uni-stream HMM-based systems, in which a component feature is used as the sole feature. In addition, the best-case-scenario performance for the utterance-level feature switching is computed by

Table 6.14: Performance of baseline systems of the Aurora 2.0 tasks.

	multi-train			clean-train		
	clean	0 – 20 dB	–5 dB	clean	0 – 20 dB	–5 dB
MV	99.18	92.84	41.66	99.70	82.62	20.60
MVA	99.24	93.47	47.30	99.66	85.02	25.66
oracle	99.44	94.59	51.62	99.75	86.63	29.05

Table 6.15: Performance of baseline systems of Aurora 3.0 tasks.

	Danish			Finnish			German			Spanish		
	WM	MM	HM	WM	MM	HM	WM	MM	HM	WM	MM	HM
MV	93.9	80.9	75.2	97.2	92.8	87.6	95.8	88.4	87.8	97.2	94.7	88.2
MVA	94.2	84.5	81.2	97.7	94.1	90.1	96.0	90.3	88.7	97.7	95.5	91.1
oracle	95.2	86.2	82.1	98.2	94.9	91.5	96.7	91.4	90.2	98.1	96.4	92.1

combining the hypotheses (the oracle) from these baseline systems. This is the empirical upper bound for the performance of an utterance-level feature switching ASR system.

The results of these baseline systems are given in Tables 6.14 and 6.15 for the Aurora 2.0 and the Aurora 3.0 tasks respectively. Apparently, the MVA features outperform the MV features in every task. Nonetheless, integrating both component features in an ASR system has a potential of 10 to 20% relative improvement over and above the MVA features.

6.3.2.2 Energy-based Feature Selectors

The results of the Aurora 2.0 tasks with the *utterance-level* energy-based feature selectors as described in Section 5.3.3.1 are summarized in Table 6.16. One can see that for clean test data the results of the utterance-level energy-based feature selector multi-stream system and the uni-stream MV system are identical, while for noisy test data the utterance-level energy-based feature selector multi-stream system has virtually the same performance as

Table 6.16: Performance of the Aurora 2.0 tasks with utterance-level energy-based feature selector. The threshold value used is 10, while the same $(a, b) = (4.9, 18.3)$ parameters as described in Section 5.3.3.1 are used.

	multi-train			clean-train		
	clean	0 – 20 dB	–5 dB	clean	0 – 20 dB	–5 dB
MV	99.18	92.84	41.66	99.70	82.62	20.60
MVA	99.24	93.47	47.30	99.66	85.02	25.66
energy	99.18	93.46	47.29	99.70	84.97	25.62

the uni-stream MVA system. This is not unexpected, as the utterance-level classification of clean/noisy speech works pretty well, as earlier shown in Table 5.2.

The performance of the Aurora 3.0 tasks with the *frame-level* energy-based feature selectors is summarized in Table 6.17. Here, unlike the utterance-level scheme, the results are almost identical to MVA results, with the only difference being 0.1% in the Danish WM task. A moment of thought uncovers the reason. The parameters (a, b) are derived based on the maximum and minimum values of an utterance. Therefore a frame is more likely to be labeled as noisy even if it is relatively clean. In other words, if 10 dB is used for the threshold, the frames in utterances with 10-dB SNR are almost all labeled as noisy, while about $\frac{3}{4}$ of the frames in utterances with 20-dB SNR are labeled as noisy. Therefore, because many frames are labeled as noisy, the performance of the dual-stream system is close to the MVA uni-stream system.

6.3.2.3 Entropy-based Feature Selectors

The entropy-based feature selectors, as described in Section 5.3.3.2, are exclusively frame-level. By design, the MV features are switched on when the environment is clean, while the MVA features are switched on when the environment is noisy.

The experimental results of the Aurora 2.0 tasks with entropy-based feature selector are summarized in Tables 6.18. In the clean-train case, feature switching gets the best out of the two, as the MV features outperform the MVA features with clean test data and the MVA

Table 6.17: Performance of the Aurora 3.0 tasks with frame-level energy-based feature selector. Same parameters as in Table 6.16 are used.

	Danish			Finnish			German			Spanish		
	WM	MM	HM	WM	MM	HM	WM	MM	HM	WM	MM	HM
MV	93.9	80.9	75.2	97.2	92.8	87.6	95.8	88.4	87.8	97.2	94.7	88.2
MVA	94.2	84.5	81.2	97.7	94.1	90.1	96.0	90.3	88.7	97.7	95.5	91.1
energy	94.3	84.5	81.2	97.7	94.1	90.1	96.0	90.3	88.7	97.7	95.5	91.1

Table 6.18: Performance of the Aurora 2.0 tasks with entropy-based feature selector. Note that these are frame-level.

	multi-train			clean-train		
	clean	0 – 20 dB	–5 dB	clean	0 – 20 dB	–5 dB
MV	99.18	92.84	41.66	99.70	82.62	20.60
MVA	99.24	93.47	47.30	99.66	85.02	25.66
entropy	99.18	93.46	47.30	99.70	85.02	25.66

features outperform MV features with noisy test data. In the multi-train case, however, as the MVA features outperform MV features in every kind of test data, feature switching is not able to improve performance.

The experimental results of the Aurora 3.0 tasks with the entropy-based feature selector are summarized in Tables 6.19. Note that in the Danish WM case, feature switching has a 0.2% (3.8% relative) improvement over the uni-stream MVA system.

6.4 Experiments on a Non-Stationary Noisy Database

The Aurora 2.0 database is quite stationary in the sense that the property of the background noise rarely changes drastically during the period of a typical utterance. A highly non-stationary noise is very difficult to conquer. Although MVA is successful in a quasi-stationary case, it may fail to be so when the environment is quickly changing. In fact, the

Table 6.19: Performance of the Aurora 3.0 tasks with entropy-based feature selector.

	Danish			Finnish			German			Spanish		
	WM	MM	HM	WM	MM	HM	WM	MM	HM	WM	MM	HM
MV	93.9	80.9	75.2	97.2	92.8	87.6	95.8	88.4	87.8	97.2	94.7	88.2
MVA	94.2	84.5	81.2	97.7	94.1	90.1	96.0	90.3	88.7	97.7	95.5	91.1
entropy	94.4	84.3	80.8	97.7	94.1	90.0	96.0	90.2	88.5	97.7	95.4	91.0

earlier analysis assumes the stationarity of noise, at least during a typical utterance. This limitation in the derivation has no implications for the effectiveness of MVA outside the assumption of stationarity.

In order to empirically inspect the effectiveness of MVA in a highly non-stationary noisy environment, a noisy speech database based on Aurora 2.0 is created as follows: For each data set, a designated noise at a prescribed noise level is added to the clean speech. The addition of noise signal is itself a 2-state Markov chain. That is, whether the current speech sample is to be added noise, given all the previous processed samples, depends only on whether the previous sample has been added noise. The resulting noisy speech can be written as

$$x(t) = s(t) + \sqrt{2}\gamma I(t)n(t), \quad (6.4)$$

where γ is determined by the noise level⁵ and $I(t)$ is an indicator random variable of whether $s(t)$ is corrupted. $I(t)$ is a Markov chain, whose state transition probability is

$$p(I(t) = 0|I(t-1) = 1) = p(I(t) = 1|I(t-1) = 0) \triangleq q, \quad (6.5)$$

where q is the probability of transition out of the current state. A symmetric transition matrix for the Markov chain is used so the stationary probability distribution is $(\frac{1}{2}, \frac{1}{2})$. The initial probability is chosen to be the stationary probability distribution. Thus, for a noisy utterance, half of the samples are corrupted and the other half remain uncorrupted, on average.

⁵To be exact, here $\gamma = 10^{-SNR/20}$, where SNR is the prescribed signal-to-noise ratio.

The processed speech samples thus consist of alternating corrupted and uncorrupted segments. The average length of these segments is determined by the Markov-chain transition probability matrix and the sampling rate. The dependence of the performance on this average length is investigated. The parameter being varied is q , where a small q corresponds to a long segment. The results are summarized in Table 6.20, where q is varied from 0.0001 (10000 samples per segment on average, 1250 ms with 8000 Hz sampling frequency) to 0.009 (111 samples, 13.9 ms).

Here one sees the divergent trends with the multi-train and the clean-train tasks. In the multi-train case, the performance reaches a peak when q is roughly 0.002 (500 samples, 62.5 ms). In the clean-train case, the performance degrades when q is increased from 0.0001 (10000 samples, 1250 ms).

From these results, one can see that the match of q in the train and test data plays a primary role in performance, but also that there is a secondary effect related to the match of two temporal parameters, the *fixed* speech window length and the *varying* segment length (controlled by q). Specifically, in the clean-train case, the segment length is infinity for the train data. Divorcing q from 0 for the test data thus increases the mismatch between the train and test data. In this case, the mismatch in q dominates the performance curve, so the performance is the best when q is the smallest. In contrast, in the multi-train case, the train and test data have the same matched q . When q is varied, the difference between the speech window length and the segmental lengths is varied. The results show that the best performance is achieved when the segment length is roughly 2 – 3 times the speech window length. This makes sense because if the segments get shorter, the corruption of speech samples in different speech windows will be likely to be different, leading to different patterns of the same linguistic targets.⁶

To further investigate the effectiveness of the MVA processing, the case of $q = 0.01$ is chosen, with 100 samples (12.5 ms) per segment on average, resulting in a highly non-stationary noise condition.

⁶Note that the limiting case of $q = 0$ does not reduce to the original database. In this case, half of the supposedly noisy utterances are clean and the other half are noisy (with the noise level doubled). These two components in the resultant database do not “average out.” This explains why the performances of low q are quite different from the original database.

Table 6.20: Results of non-stationary Aurora 2.0 noisy databases varying the average segmental length. The numbers in the leftmost column are the q values, where each q value is used in re-generating the entire database, with which ASR experiments are run. In the front end, the 39-dimensional raw feature vector of Configuration 3 is used. The back end is 16-state whole-word HMMs, with each state having 3 Gaussian components.

multi-train						
	Test Set A			Test Set B		
	clean	0 – 20 dB	–5 dB	clean	0 – 20 dB	–5 dB
0.0001	98.92	82.95	63.05	98.92	81.36	60.28
0.0002	98.99	83.49	64.47	98.99	82.82	62.25
0.0005	98.83	85.34	66.61	98.83	84.65	64.27
0.0010	98.75	86.48	67.41	98.75	86.38	65.60
0.0020	98.75	87.53	67.92	98.75	87.95	67.74
0.0050	98.55	85.07	60.23	98.55	86.70	58.67
0.0090	98.50	82.10	54.16	98.50	83.48	48.27
clean-train						
	Test Set A			Test Set B		
	clean	0 – 20 dB	–5 dB	clean	0 – 20 dB	–5 dB
0.0001	98.94	72.03	49.70	98.94	66.10	48.38
0.0002	98.94	70.49	45.85	98.94	64.70	43.01
0.0005	98.94	66.62	36.07	98.94	60.99	32.37
0.0010	98.94	62.81	26.67	98.94	57.25	21.10
0.0020	98.94	59.29	21.64	98.94	54.77	12.86
0.0050	98.94	57.01	16.75	98.94	53.43	8.44
0.0090	98.94	56.31	15.04	98.94	52.69	7.73

Table 6.21: Results of GMTK-based systems on the highly non-stationary Aurora 2.0 noisy database with $q = 0.01$ (very short segments).

multi-train						
	Test Set A			Test Set B		
	clean	0 – 20 dB	–5 dB	clean	0 – 20 dB	–5 dB
RAW	99.42	82.72	52.95	99.42	85.82	48.20
M	99.40	84.26	57.76	99.40	86.66	53.11
MV	99.18	85.38	61.20	99.18	88.05	56.38
MVA	99.29	84.01	58.47	99.29	86.92	54.93
clean-train						
	Test Set A			Test Set B		
	clean	0 – 20 dB	–5 dB	clean	0 – 20 dB	–5 dB
RAW	99.62	55.24	17.20	99.62	60.08	16.28
M	99.73	59.84	17.83	99.73	61.72	14.76
MV	99.66	73.89	36.60	99.66	77.07	31.79
MVA	99.70	65.52	23.23	99.70	67.89	20.37

The performances of this highly non-stationary case are summarized in Table 6.21. From this table, one can see that although MVA feature processing does improve the performance in the non-stationary cases, the improvement via MVA feature processing in the non-stationary cases is not as significant as in the stationary cases.

6.5 Experiments on the Spine Database

Spine (SPeech In Noisy Environments) [75] is a database recorded in noisy military environments (the military noises were played in the background during the recording of the speech). The Spine 1 database consists of 140 conversations in the training set and 120 conversations in the evaluation set. The Spine 2 database consists of 64 conversations in

the training set, 32 conversations in the development set, and 64 conversations in the evaluation set. In the Spine 2 evaluation, the use of the Spine 1 data for training is allowed (and this is often the case with participating sites). With that, the total amount of speech data for training is approximately 17 hours, and for the test data, about 2.4 hours.

The MVA technique is also evaluated on Spine. The Spine task has a larger vocabulary (5721 entries in the dictionary) than Aurora. In addition, the language-model aspect (using bigram) is more important in Spine than in Aurora (using word net). In addition, Spine consists of data recordings in environments which are different from those in Aurora. These environments are called *Bradley*, *car*, *carrier*, *F16*, *helo*, *office*, *quiet*, and *street*. Based on the author's subjective personal impression, *Bradley*, *carrier*, *F16* and *helo* environments have significant battle-field noise characteristic of a military setting in which some utterances have been yelled. In contrast, *car*, *street*, *office* and *quiet* environments do not have significant background noise but some utterances are quite unnatural in terms of the speaking rate (very slow and inconsistent) or volume (whispering). This leads to a large degree of pronunciation variation. The SNR is reportedly between 5 dB to 20 dB.

6.5.1 Different Front-end Configurations

The first set of experiments is designed to compare the use of the zeroth cepstral coefficient ($C[0]$) and the log energy (ξ). Different from the per-utterance scheme in the Aurora experiments, a *per-side* scheme is used in the Spine experiments. That is, the mean and variance of an entire conversation side are computed and then used in the mean subtraction and variance normalization. A second-order ARMA filter is optionally applied to the feature stream after the normalization.

The word error rates *without* any model adaptations (cf. Section 2.3.2) are presented in Table 6.22. One can see the advantage of using $C[0]$ over ξ from this table. In addition, the relative improvement of the processed features over the RAW feature is larger when $C[0]$ is used than when ξ is used.

Table 6.22: Spine results of different front-end configurations. Here the comparison is between using the log energy versus using the zeroth cepstral coefficient. The order of the ARMA filter in MVA is 2. Note that the numbers in the tables are the word error rates.

	using the log energy ξ								
	Bradley	car	carrier	F16	helo	office	quiet	street	Overall
RAW	63.9	47.1	46.0	69.5	65.6	47.5	54.5	53.9	54.7
M	54.9	40.7	38.7	59.4	60.9	42.7	49.2	48.1	48.1
MV	52.1	41.1	37.1	56.6	55.4	41.8	47.7	46.7	46.4
MVA	53.5	39.8	38.3	59.1	59.5	42.5	48.6	48.0	47.5

	using the zeroth cepstral coefficient $C[0]$								
	Bradley	car	carrier	F16	helo	office	quiet	street	Overall
RAW	66.2	45.5	46.1	69.4	69.0	47.1	52.3	50.8	54.2
M	53.6	39.1	39.5	57.7	61.5	41.6	47.8	45.5	47.0
MV	51.5	39.8	36.3	53.8	54.0	41.9	46.6	46.7	45.5
MVA	53.1	40.4	39.1	57.0	58.0	41.4	49.0	46.4	47.0

6.5.2 Different ARMA Orders

The second set of experiments is designed to compare different orders of the ARMA filters.

The results are summarized in Table 6.23. In this set of experiments, the best overall performance occurs when using ARMA filter of order 1. Compared to the MV processing, it is better in *carrier*, *F16*, *quiet* and *street*, and it is worse in *Bradley*, *car*, *helo* and *office*.

The difference in the effectiveness of the ARMA filtering in Spine and in Aurora deserves an explanation. The ARMA filtering basically smoothes the time sequences of each feature component. On the positive side, it can reduce the noise effect in the speech signal. But on the negative side, it can reduce the discriminativity of different speech targets (words). In a small-vocabulary and highly noisy database, such as Aurora, the advantage outweighs the disadvantage. In Spine, the discrimination is much more crucial, as there are more linguistic classes to discriminate. The results show that beyond the first-order, the ARMA filtering provides no performance gain.

Table 6.23: Spine results of different orders of ARMA. Here A_m denotes the ARMA filter of order m . The results have been decomposed into different environments. Again, the reported numbers are the word error rates.

	using the log energy ξ								
	Bradley	car	carrier	F16	helo	office	quiet	street	Overall
MV	52.1	41.1	37.1	56.6	55.4	41.8	47.7	46.7	46.4
MVA_1	52.0	40.2	38.1	57.4	57.1	40.6	48.2	46.6	46.4
MVA_2	53.5	39.8	38.3	59.1	59.5	42.5	48.6	48.0	47.5
MVA_3	52.1	41.6	41.8	61.6	60.0	43.0	51.8	49.1	49.0
	using the zeroth cepstral coefficient $C[0]$								
	Bradley	car	carrier	F16	helo	office	quiet	street	Overall
MV	51.5	39.8	36.3	53.8	54.0	41.9	46.6	46.7	45.5
MVA_1	51.8	39.4	36.6	57.0	55.7	41.9	48.7	46.2	46.2
MVA_2	53.1	40.4	39.1	57.0	58.0	41.4	49.0	46.4	47.0
MVA_3	51.8	40.9	39.7	59.2	58.6	43.2	50.2	47.3	47.8

Chapter 7

CONCLUSIONS

7.1 *Summary*

Due to the rapid growth of usage of mobile communication and computation devices, noise-robust ASR is becoming more and more important everyday, as mobile devices are often used outside the quiet environment of office or home. Furthermore, users may need to command a device via voice when their hands or eyes are occupied.

On a mobile device, due to limitation in size/weight and battery life, computational power is restricted. Therefore, a low-resource solution to the issue of noise-robustness is much desired. Such a solution should be effective, that is, not significantly compromising recognition accuracy. The MVA feature processing proposed in this thesis is such a technique.

Both the classic methods and modern techniques are described to the best of the author's comprehension and ability. This account is not meant to be complete but rather is meant to provide a necessary background for research of noise-robust ASR systems.

An analysis of the feature set of MFCC and its distortion under additive and convolutional noise is given in Chapter 3. This treatment continues through Chapter 4 on how the investigated technique, MVA, deals with such distortion.

Chapter 5 proposes a different approach for noise-robustness. The emphasis shifts from the front-end signal processing to the back-end models. Here, systems with multiple feature streams with feature selection are described in detail.

The experimental results are presented in Chapter 6. First, MVA is evaluated in multiple domains. In addition, the ARMA filter is compared to other designed or data-driven filters. Second, MVA is evaluated in combination of various speech feature sets. Third, the multiple-stream systems are implemented with two distinct feature selection schemes. Finally, the

effectiveness of MVA on a medium-size vocabulary (the Spine) task, with respect to different ARMA orders and different feature configurations, is evaluated.

Overall, MVA is quite successful in reducing the word error rates in noisy tasks as a feature processing technique. Furthermore, it achieves essentially the same performance level as other systems without consuming as many resources. Its effectiveness, however, does strongly depend on the used feature set, the vocabulary size, and the domain of application.

The idea of a feature selector in multiple-stream systems, on the other hand, does not lead to significant performance gain. This may be due to the fact that very similar feature streams are used, leading to failure to take advantage of cues for environmental condition from entropic or energetic sources of information. In hindsight, it is probably a better idea to choose the component features to be of a more distinct nature.

7.2 *Future Work*

This thesis work can be extended in several directions.

- **Temporal filters.** The inspected ARMA filtering is essentially temporal processing, with the coefficients of the temporal filter being set to uniform values irrespective of any data or learning algorithm. To generalize from the current special case, one may
 - assume a parameterized shape (such as a Mexican hat), learn a parameter-noise relation from training data, and then use this relation to determine the parameter for given test data.
 - assume general unknown coefficients and then learn these coefficients from training data based on certain criteria, such as principle component analysis or linear discriminant analysis.

However, one must be aware of the issue of data mismatch when adopting data-driven approaches. In this regard, the current MVA does not suffer from this problem. It cannot be over-trained to a particular data set since the coefficients are fixed.

- **Analysis.** The feature distortion is analyzed only for the MFCCs and the log energy, leading to the MVA technique. Yet, it has been shown that MVA also combines well with several other feature sets. For those features whose signal processing procedures do not include neural networks, it should not be difficult to analyze their distortions due to additive and convolutional noise.

- **Multi-stream system.** The used feature streams are quite similar in nature and the graphs are based on observed feature selectors. For future extension of the current work, one may
 - use feature streams of more different natures.
 - use more refined dependency patterns among feature selectors and other random variables in the graph, such as using phonetic information to help guide feature selectors.

BIBLIOGRAPHY

- [1] I. Abdallah, S. Montresor, and M. Baudry. Speech signal detection in noisy environment using a local entropic criterion. In *European Conference on Speech Communication and Technology (EuroSpeech)*, 1997.
- [2] T. Anastasakos, J. McDonough, R. Schwartz, and J Makhoul. A compact model for speaker adaptive training. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 1137–1140, 1996.
- [3] B. S. Atal. Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification. *Journal of the Acoustical Society of America (JASA)*, 55:1304–1312, 1974.
- [4] J. Barker, M. Cooke, and P. Green. Robust ASR based on clean speech models: An evaluation of missing data techniques for connected digit recognition in noise. In *European Conference on Speech Communication and Technology (EuroSpeech)*, pages 213–216, 2001.
- [5] C. Benitez, L. Burget, B. Chen, S. Dupont, H. Garudadri, H. Hermansky, P. Jain, S. Kajarekar, N. Morgan, and S. Sivasdas. Robust ASR front-end using spectral-based and discriminant features: experiments on the Aurora tasks. In *European Conference on Speech Communication and Technology (EuroSpeech)*, pages 429–432, 2001.
- [6] J. Bilmes. *Natural Statistical Models for Automatic Speech Recognition*. PhD thesis, U.C. Berkeley, Dept. of EECS, CS Division, 1999.
- [7] J. Bilmes. What HMMs can do. Technical Report UWEETR-2002-003, University of Washington, Dept. of EE, 2002.

- [8] J. Bilmes and G. Zweig. The graphical models toolkit: An open source software system for speech and time-series processing. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Orlando, June 2002.
- [9] J. A. Bilmes. Joint distributional modeling with cross-correlation based features. In *Proceedings of IEEE ASRU Workshop*, pages 148–155, 1997.
- [10] J. A. Bilmes. Maximum mutual information based reduction strategies for cross-correlation based joint distributional modeling. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1998.
- [11] J. A. Bilmes. Buried Markov models for speech recognition. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Phoenix, March 1999.
- [12] J. A. Bilmes. Graphical models and automatic speech recognition. In R. Rosenfeld, M. Ostendorf, S. Khudanpur, and M. Johnson, editors, *Mathematical Foundations of Speech and Language Processing*. Springer-Verlag, New York, 2002.
- [13] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.
- [14] S. Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on Acoustics, Speech and Signal Processing (ASSP)*, 27(2):113–120, 1979.
- [15] H. Bourlard, S. Dupont, and C. Ris. Multi-stream speech recognition. Technical Report IDIAP-RR 96-07, Dalle Molle Institute for Perceptual Artificial Intelligence, 1996.
- [16] O. Çetin, H. Nock, K. Kirchhoff, J. Bilmes, and M. Ostendorf. The 2001 GMTK-based SPINE ASR system. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, 2002.

- [17] C.-P. Chen, J. Billes, and K. Kirchhoff. Low-resource noise-robust feature post-processing on Aurora 2.0. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 2445–2448, 2002.
- [18] C.-P. Chen, K. Filali, and J. Billes. Frontend post-processing and backend model enhancement on the Aurora 2.0/3.0 databases. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 241–244, 2002.
- [19] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. Technical Report Tr-10-98, Center for Research in Computing Technology, Harvard University, Cambridge, Massachusetts, August 1998.
- [20] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.
- [21] S. B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing (ASSP)*, 28(4):357–366, August 1980.
- [22] J. de Veth, L. Mauuary, B. Noe, F. de Wet, J. Siemel, L. Boves, and D. Jouver. Feature vector selection to improve ASR robustness in noisy conditions. In *European Conference on Speech Communication and Technology (EuroSpeech)*, pages 201–204, 2001.
- [23] T. Dean and K. Kanazawa. Probabilistic temporal reasoning. *AAAI*, pages 524–528, 1988.
- [24] J. Droppo, L. Deng, and A. Acero. Evaluation of Splice on the Aurora 2 and 3 tasks. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 29–32, 2002.
- [25] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley and Sons, Inc., 2000.

- [26] D. Ellis and M. Gomez. Investigations into tandem acoustic modeling for the Aurora task. In *European Conference on Speech Communication and Technology (EuroSpeech)*, pages 189–192, 2001.
- [27] G. Evermann and P. C. Woodland. Large vocabulary decoding and confidence estimation using word posterior probabilities. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Istanbul, Turkey, 2000.
- [28] J. G. Fiscus. A post-processing system to yield reduced word error rates: Recognizer Output Voting Error Reduction (ROVER). In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, Santa Barbara, California, 1997.
- [29] H. Fletcher. Auditory patterns. *Rev. Mod. Phys.*, 12:47–65, 1940.
- [30] S. Furui. Cepstral analysis technique for automatic speaker verification. *IEEE Transactions on Acoustics, Speech and Signal Processing (ASSP)*, 29(2):254–272, April 1981.
- [31] V. Gadde, A. Stolcke, D. Vergyri, J. Zheng, K. Sonmez, and A. Venkataraman. Building an ASR system for noisy environments: SRI's 2001 SPINE evaluation system. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 1577–1580, 2002.
- [32] M. J. F. Gales. The generation and use of regression class trees for MLLR adaptation. Technical Report CUED/F-INFENG/TR263, Department of Engineering, University of Cambridge, 1996.
- [33] M. J. F. Gales and S. Young. An improved approach to the hidden Markov model decomposition of speech and noise. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 233–236, 1992.
- [34] J.L. Gauvain and C.H. Lee. Maximum a-posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE Transactions on Speech and Audio Processing (SAP)*, 2:291–298, 1994.

- [35] Z. Ghahramani and M. Jordan. Factorial hidden Markov models. *Machine Learning*, 29, 1997.
- [36] G. Green. *Temporal Aspects of Audition*. PhD thesis, Oxford University, 1976.
- [37] S. Greenberg and B. Kingsbury. The modulation spectrogram: in pursuit of an invariant representation of speech. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1647–1650, 1997.
- [38] T. Hain, P. Woodland, G. Evermann, and D. Povey. The CU-HTK march 2000 HUB5E transcription system. In *Proceedings of Speech Transcription Workshop*, 2000.
- [39] H. Hermansky and N. Morgan. RASTA processing of speech. *IEEE Transactions on Speech and Audio Processing (SAP)*, 2(4):578–589, October 1994.
- [40] H. Hermansky and S. Sharma. Temporal patterns (TRAPS) in ASR of noisy speech. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1999.
- [41] H. G. Hirsch and D. Pearce. The AURORA experimental framework for the performance evaluations of speech recognition systems under noisy conditions. In *ICSA ITRW ASR 2000*, September 2000.
- [42] Tammo Houtgast and Herman J. M. Steeneken. A review of the MTF concept in room acoustics and its use for estimating speech intelligibility. *Journal of the Acoustical Society of America (JASA)*, 77(3):1069–1077, March 1985.
- [43] L.-S. Huang and C.-H. Yang. A novel approach to robust speech endpoint detection in car environments. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2000.
- [44] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1997.

- [45] B-H Juang and S. Katagiri. Discriminative learning for minimum error classification. *IEEE Transactions on Signal Processing (SP)*, 40(12):3043–3054, December 1992.
- [46] B.-H. Juang and K. K. Paliwal. Hidden Markov models with first-order equalization for noisy speech recognition. *IEEE Transactions on Signal Processing (SP)*, 40(9):2136–2143, 1992.
- [47] B.-H. Juang and L.R. Rabiner. Mixture autoregressive hidden Markov models for speech signals. *IEEE Transactions on Acoustics, Speech and Signal Processing (ASSP)*, 33(6):1404–1413, December 1985.
- [48] J. C. Junqua. The Lombard reflex and its role on human listeners and automatic speech recognizers. *Journal of the Acoustical Society of America (JASA)*, 91(1):510–524, January 1993.
- [49] B. Kingsbury, G. Saon, L. Mangu, M. Padmanabhan, and R. Sarikaya. Robust speech recognition in noisy environments: The 2001 IBM SPINE evaluation system. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 53–56, 2002.
- [50] B. E. D. Kingsbury. *Perceptually Inspired Signal-processing Strategies for Robust Speech Recognition in Reverberant Environments*. PhD thesis, University of California, Berkeley, 1998.
- [51] M. Kleinschmidt. Spectro-temporal Gabor features as a front-end for automatic speech recognition. In *Forum Acusticum Sevilla 2002 Proceedings*, 2002.
- [52] M. Kleinschmidt and D. Gelbart. Improving word accuracy with gabor feature extraction. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 25–28, 2002.
- [53] S.L. Lauritzen. *Graphical Models*. Oxford Science Publications, 1996.

- [54] C.J. Leggetter and P.C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer Speech and Language*, 9:171–185, 1995.
- [55] A. Leon-Garcia. *Probability and Random Processes for Electrical Engineering*. Addison-Wesley, 1994.
- [56] B. Lindberg. Danish SpeechDat-Car Digits Database for ETSI STQ Aurora Advanced DSR, 2001.
- [57] B.T. Logan and P.J. Moreno. Factorial HMMs for acoustic modeling. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1998.
- [58] D. Macho. Spanish SDC-Aurora Database for ETSI STQ Aurora WI008 Advanced DSR Front-End Evaluation: Description and Baseline Results, 2000.
- [59] B. Mak and Y. Tam. Performance of discriminatively trained auditory features on aurora2 and aurora3. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 33–36, 2002.
- [60] L. Mangu, E. Brill, and A. Stolcke. Finding consensus in speech recognition: Word error minimization and other applications of confusion networks. In *Proceedings of the Third World Multiconference on Systemics, Cybernetics and Informatics joint with the Fifth International Conference on Information Systems Analysis and Synthesis*, volume 5, pages 246–252, 1999.
- [61] L. Mangu, E. Brill, and A. Stolcke. Finding consensus in speech recognition: Word error minimization and other applications of confusion networks. *Computer, Speech and Language*, 14(4):373–400, 2000.
- [62] J. D. Markel and A. H. Gray. *Linear Prediction of Speech*. Springer-Verlag, New York, 1976.

- [63] P. J. Moreno, B. Raj, and R. M. Stern. A vector taylor series approach for environment-independent speech recognition. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 733–736, 1996.
- [64] L. Netsch. Description and Baseline Results for the Subset of the Speechdat-Car German Database used for ETSI STQ Aurora WI008 Advanced DSR Front-End Evaluation, 2001.
- [65] J. J. Odell. *The Use of Context in Large Vocabulary Speech Recognition*. PhD thesis, University of Cambridge, England, 1995.
- [66] M. Ostendorf, V. Digalakis, and O. Kimball. From HMMs to segment models: A unified view of stochastic modeling for speech recognition. *IEEE Transactions on Speech and Audio Processing (SAP)*, 4(5):360–378, September 1996.
- [67] K. K. Paliwal and A. Basu. A speech enhancement method based on Kalman filtering. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 177–180, 1987.
- [68] J. Pearl. *Causality*. Cambridge, 2000.
- [69] G. Potamianos, C. Neti, G. Iyengar, and E. Helmuth. Large-vocabulary audio-visual speech recognition by machines and humans. In *European Conference on Speech Communication and Technology (EuroSpeech)*, pages 1027–1030, 2001.
- [70] A. Sankar and C.-H. Lee. A maximum-likelihood approach to stochastic matching for robust speech recognition. *IEEE Transactions on Speech and Audio Processing (SAP)*, 4(3):190–202, May 1996.
- [71] J. Segura, M. Benitez, A. Torre, and A. Rubio. Feature extraction combining spectral noise reduction and cepstral histogram equalization for robust asr. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, pages 225–228, 2002.

- [72] R. V. Shannon, F.-G. Zeng, V. Kamath, J. Wygonski, and M. Ekelid. Speech recognition with primarily temporal cues. *Science*, 270:303–304, 1995.
- [73] J.-L. Shen, J.-W. Hung, and L.-S. Lee. Robust entropy-based endpoint detection for speech recognition in noisy environments. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, 1998.
- [74] R. Singh, M. Seltzer, B. Raj, and R. Stern. Speech in noisy environments: Robust automatic segmentation, feature extraction and hypothesis combination. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 273–276, 2001.
- [75] Spine2: Speech in noisy environments. <http://elazar.itd.nrl.navy.mil/spine/index.html>.
- [76] STQ Aurora DSR Working Group. Availability of Finnish SpeechDat-Car database for ETSI STQ WI008 frontend standardisation.
- [77] T. M. Sullivan. *Multi-Microphone Correlation-Based Processing for Robust Automatic Speech Recognition*. PhD thesis, Carnegie Mellon University, 1996.
- [78] A. P. Varga and R. K. Moore. Hidden Markov model decomposition of speech and noise. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 845–848, 1990.
- [79] J. Wu and Q. Huo. Several hku approaches for robust speech recognition and their evaluation on aurora connected digit recognition tasks. In *European Conference on Speech Communication and Technology (EuroSpeech)*, 2003.
- [80] K. Yao, E. Visser, O.-W. Kwon, and T.-W. Lee. A speech processing front-end with eigenspace normalization for robust speech recognition in noisy automobile environments. In *European Conference on Speech Communication and Technology (EuroSpeech)*, 2003.

- [81] S. Young. Cepstral mean compensation for HMM recognition in noise. In *Proceedings of ESCA Workshop on Speech Processing in Adverse Conditions, Cannes-Mandelieu*, pages 123–126, 1992.

- [82] Y. Zhang, Q. Diao, S. Huang, W. Hu, C. Bartels, and J. Bilmes. DBN based multi-stream models for speech. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2003.

Appendix A

EM ALGORITHM IN MAXIMUM LIKELIHOOD ESTIMATION

In maximum likelihood parameter estimation, the objective is to find the model parameter values which maximizes data likelihood. The EM (Expectation-Maximization) algorithm is commonly used in such parameter estimation because the data likelihood is non-decreasing with iterations.

An auxiliary function is defined by

$$Q(\Theta, \Theta_0) = E[\log p(S, x|\Theta)] = \sum_s p(s|x, \Theta_0) \log p(s, x|\Theta), \quad (\text{A.1})$$

where Θ_0 is the current model parameter set, x is the (observed) data, and S is the hidden variable. This function is maximized with respect to the unknown, Θ . Suppose the solution in maximizing (A.1) is Θ' , then

$$Q(\Theta', \Theta_0) \geq Q(\Theta_0, \Theta_0). \quad (\text{A.2})$$

Furthermore,

$$\begin{aligned} \log p(x|\Theta') - \log p(x|\Theta_0) &= Q(\Theta', \Theta_0) - Q(\Theta_0, \Theta_0) + \sum_s p(s|x, \Theta_0) \log \frac{p(s|x, \Theta_0)}{p(s|x, \Theta')} \\ &\geq Q(\Theta', \Theta_0) - Q(\Theta_0, \Theta_0) \\ &\geq 0. \end{aligned} \quad (\text{A.3})$$

It follows that the data likelihood does not decrease with each maximization.

The complexity of EM algorithm depends on the complexity in finding the optimal solution in (A.1). The algorithm is very efficient when the solution has a closed form and it can be *computed* directly from data statistics and old parameters. In the cases where there is no closed-form solution, the computational complexity depends on the suitable numerical methods. Alternatively, one may update Θ which satisfies (A.2) and which can be obtained easily, although it may take more iterations to converge.

Appendix B

COMMONLY USED NOTATIONS

notation	description
t	continuous time
$s(t)$	clean speech signal
$n(t)$	additive noise signal
$h(t)$	convolutional noise signal
$x(t)$	noise-corrupted speech signal
j	index of a mel-frequency filter
J	number of mel-frequency filters
i	index of a static cepstral coefficient
I	number of static cepstral coefficients
d	index of a feature vector component
D	number of feature vector components
P	power spectrum
F	mel binning filter
Q	mel-frequency spectrum
G	matrix representation of discrete cosine transform
C	(static) cepstral vector
ξ	log energy
N	number of samples in a frame (including zero-padding)
n	index of discrete time in a frame
k	index of discrete frequency
τ	frame index
γ	noise level

VITA

Chia-Ping Chen grew up in the city of Kaohsiung, one of the largest harbors in Asia. He earned a bachelor's degree from the National Taiwan University and a master's degree from the National Tsing-Hua University, both in physics. In 1998, he began his graduate study in the Department of Electrical Engineering at the University of Washington, where he earned a master's degree in 2001 and a PhD in 2004.

Besides physics, communications, signal processing, and automatic speech recognition, he is also interested in machine learning, data mining, information search/retrieval, stochastic models, causal models, and convex optimizations.