Speech recognition on the speechdat-car database

Bimas Winahyu Aji

2008

Bimas Winahyu Aji. (2008). Speech recognition on the speechdat-car database. Master' s thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/40513>

<https://doi.org/10.32657/10356/40513>

SPEECH RECOGNITION ON THE SPEECHDAT-CAR DATABASE

Bimas Winahyu Aji

School of Computer Engineering

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

2008

Page intentionally left blank

To my parents and sisters.

Table of Contents

Table of Contents	iv
Table of Notations	vii
Abstract	xii
Acknowledgements	xiv
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	3
1.3 Contributions	4
1.4 Outlines	5
2 Related Works	6
2.1 Introduction	6
2.2 Baseline Recognizer	7
2.2.1 Feature extraction	8
2.2.2 Back-end SI-CHMM	18
2.3 Current State of the Art Techniques	33
2.3.1 Noise reduction and compensation techniques	34
2.3.2 Speaker adaptive acoustic modeling	37
2.3.3 Model adaptations	38
2.3.4 Observed data classification	39
2.3.5 Real time operation fixed-point processor	40
2.3.6 Fixed-point code design considerations	41

3	Extensions of Feature Extraction Module to Improve Robustness under Car Environment	44
3.1	Introduction	44
3.2	Feature Extraction Procedure	46
3.2.1	Noise Estimation	47
3.2.2	Noise Suppression Factor Calculation and Gain Smoothing . .	51
3.2.3	Convolutional Noise Reduction	53
3.3	End Point Detection	53
3.3.1	Band State	54
3.3.2	Global State	55
4	Development of a Speech Recognizer	58
4.1	Introduction	58
4.2	Development Environment	59
4.3	Fixed point code implementation	61
4.3.1	Dynamic vs Fixed Scaling Factor	61
4.3.2	Complex Math Functions	62
4.4	Outline of Development Process	63
4.5	Data Preparation	64
4.6	Training strategies	68
4.7	Refinement	71
4.7.1	Time-derivatives MFCCs	72
4.7.2	Multiple Mixtures	72
4.8	Evaluation	73
5	Experiments	75
5.1	Introduction	75
5.2	Data Selection	76
5.3	Baseline System Experiment	79
5.3.1	Design	79
5.3.2	Results	80
5.4	Isolated Word Recognition with HTK	83
5.4.1	Design	83
5.4.2	Results	85
5.5	Incorporation of Various Techniques	87
5.5.1	Experiments on Digital Differentiators for MFCCs' derivatives calculation	87
5.5.2	Experiments on Convolutional Noise Reduction Techniques . .	91

5.5.3	Experiments on Model Parameters Initialization and Number of Mixtures per State	96
5.5.4	Experiments on Additive Noise Reduction Techniques	99
5.6	Suggested System Configurations	110
6	Conclusions and Recommendations	112
	Bibliography	115

Table of Notations

- ASR – Automatic Speech Recognition
- BE – Back-End processing
- BEq – Blind Equalization
- CHMM – Continuous Hidden Markov Model
- Corr – Correlation, 2-D Correlation Coefficients
- CT – Close-Talk, Close-talk microphone as opposed to hands-free microphone
- DCT – Discrete Cosine Transform, Cepstral coefficient calculation module
- DD – Digital Differentiator, procedure needed to get derivatives of a signal
- DE – Deutsch, Deutsch speech database
- DFT – Discrete Fourier Transform
- DSP – Digital Signal Processing, Digital Signal Processor
- DTW – Dynamic Time Warping
- EPD – End-Point Detection
- FDNR – Frequency Domain Noise Reduction
- FE – Front-End processing, Front-end feature extraction
- FFT – Fast Fourier Transform

- FIR – Finite Impulse Response, a type of a digital filter
- FR – Framing
- GA – Gain Application, applying the calculated speech gain to the original signal
- GS – Gain Smoothing
- GSM – Global System for Mobile communications
- HAS – Human Auditory System
- HF – Hands-Free, Hands-free microphones located quite a distance from speakers
- HF-A – Hands-Free, Hands-free microphones located quite a distance from speakers, at the ceiling of the car near the A-pillar
- HF-B – Hands-Free, Hands-free microphones located quite a distance from speakers, at the ceiling of the car in front of the speaker behind the sunvisor
- HF-C – Hands-Free, Hands-free microphones located quite a distance from speakers, at the ceiling of the car over the mid-console (near the rear mirror)
- HM – High-mismatch data set, where training takes 75% data in low noise from CT microphone and testing uses 25% data in low noise and high noise data from HF and CT microphones
- HMM – Hidden Markov Model
- HTK – Hidden Markov Model Toolkit, a portable toolkit for building and manipulating hidden Markov models from Cambridge University Engineering Department
- IT – Italian, Italian speech database
- KHz – Kilo Hertz
- LDA – Linear Discriminant Analysis, a method used in statistics and machine learning to find the linear combination of features which best separate two or more classes of objects or events

- Lift – Liftering
- LMS – Least Mean Squares, a mathematical optimization technique
- Log – Logarithmic
- LTI – Linear Time Invariant
- MAP – Maximum a Posteriori estimation
- Method 1 NE – Noise estimation method that estimates and updates the noise spectrum only during the silent (e.g., during pauses) segments of the signal using a VAD
- Method 2 NE – Noise estimation that is obtained by recursive averaging of the noisy Mel-domain power spectrum using time-varying and frequency-dependent smoothing parameters, which are adjusted by speech-presence probability in individual Mel bins
- MFCC – MEL Frequency Cepstral Coefficients
- ML – Maximum Likelihood estimation, a popular statistical method used to calculate the best way of fitting a mathematical model to some data
- MLLR – Maximum Likelihood Linear Regression,
- MM – Medium-mismatch data set, where training uses 75% data in all driving conditions from CT microphone and testing uses 25% data in all driving conditions from HF and CT microphones
- MMI – Maximum Mutual Information
- MMSE – Minimum Mean Square Error, an estimator that minimizes the mean square error (MSE), which is a common measure of estimator quality
- MWE – Minimum Word Error
- NE – Noise Estimation
- NM – Noise Masking
- Norm – Normalization

- NR – Noise Reduction
- NSF – Noise Suppression Factor
- PDF – Probability Density Function
- PE – Pre-emphasis
- PMC – Parallel Model Combination
- PRIRR – Percentage Relative Improvement of Recognition Rate calculated as current recognition rate subtracted by the recognition rate of the reference system
- PSD – Power Spectral Density
- RASTA – Relative Spectral
- RMS – Root Means Square
- SAT – Speaker Adaptive Training
- SD – Speak Dependent
- SDCHMM – Subspace Distribution Clustering HMM
- SegErr – Segmental Relative Estimation Error
- SegSNR – Segmental SNR
- Sfr(n) – Speech signal data after framing (FR) at index n
- SI – Speaker Independent
- SI-CHMM – Speaker Independent CHMM
- Sin(n) – Speech signal data input at index n
- SKM – Segmental K -mean
- SNR – Signal to Noise Ratio
- Spe(n) – Speech signal data after pre-emphasis (PE) at index n

- STFT – Short-Time Fourier Transform
- STM – STMicroelectronics Pte Ltd
- TDNR – Time Domain Noise Reduction
- VAD – Voice Activity Detection
- VFR – Variable Frame Rate
- VTLN – Vocal Tract Length Normalization
- WM – Weakly mismatch data set, where training uses 75% of data from both HF and CT microphones in all driving conditions and testing uses the rest 25% data from HF and CT microphones in all driving conditions

Abstract

The SpeechDat-Car database was built in the framework of the SpeechDat-Car project (European Union founded). The availability of the database allows the development of the speech recognition applications in mobile device under severe condition of car environment. A speaker-independent continuous Hidden Markov Model (CHMM) algorithm for speech recognition is to be produced in the project. To achieve this objective, the CHMM system adaptations and testings on the SpeechDat-Car database would be investigated. This is done by first understanding the algorithm and evaluating both the front end and back end of the system, and then, modifying the algorithm for obtaining better performance. As the final algorithm adopted would be ported to a selected DSP, as given in the system specification, the work also focuses on algorithms with low computational cost and memory consumption.

The work is factored into two stages. In the first stage, one database (one language only) was processed and an optimum parameter set for the CHMM algorithm was derived. This derivation of optimum parameter set also involved the work on the model refinement and recognition algorithm enhancement. In the second stage, the algorithms used in the first stage was tested and extended to other languages. Throughout the project development, the fixed point code is used instead of floating point code as the final CHMM algorithm adopted would be ported to a selected

DSP and it is essential to maintain precision accuracy at different sections of the code/algorithm.

We have investigated some techniques suitable for front-end feature extraction process as well as back-end training process. The most suitable Digital differentiators were investigated to get the 1st and 2nd derivatives of the extracted features. Blind Equalization and Relative Spectral filtering were tried to compensate distortion due to channel transmission. Additive noise reduction which strives for interesting performance/resource ratio was proposed. Selected noise estimation and noise suppression function (two important components of a noise reduction method) substantially improves the speech recognition results for noisy speech, without serious degradation of the clean speech results. Lastly, HMMs are, due to their Markov property, very sensitive to initial values. For this reason, two techniques on model initialization, namely segmental K -means and mixture splitting were further investigated.

By incorporating the techniques suitable for speech recognition as well as extensive experiments to get the best configurations, we could derive the optimum parameter set for both Italian and Deutch languages. A average of 90% recognition rate was achieved with acceptable memory consumption and computational cost. Moreover, all the algorithms used for recognizer have been coded in fixed point C language to facilitate easy porting into a selected DSP.

Acknowledgements

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. I want to thank STMicroelectronics Pte Ltd (S) for giving me permission to commence this thesis in the first instance, to do the necessary research work and to use the speech database. I have furthermore to thank Dr Anoop Kumar who gave and confirmed this permission and encouraged me to go ahead with my thesis. I am bound to the people in R and D department of STMicroelectronics Pte Ltd (S) for their stimulating support.

I would like to thank Dr He Yulan, my supervisor, for her many suggestions and constant support during this research. Her help, stimulating suggestions and encouragement helped me in all the time of research and writing of this thesis.

Of course, I am grateful to my parents for their patience and *love*. Without them this work would never have come into existence (literally).

Finally, I wish to thank the following: Fairoz Ahmed (for his friendship); Devta (for changing my life from worse to bad); Adi, Ridwan, Fitrah, Ari, Aldo, ... (for all the good and bad times we had together); *and* my sisters (because they asked me to).

Singapore
June 30, 2007

Bimas Winahyu Aji

Chapter 1

Introduction

The question

Does speech recognition technology has reached a level of maturity to meet user demands in many application?

has special meaning to the enthusiasts of robot R2D2 in the George Lucas classic *Star Wars* series of movies.

While we are far from achieving the goal of a general purpose speech recognizer, for niche applications such as mobile devices, automated call centers and consumer electronics products, speech recognition technology has met its user demands.

1.1 Motivation

Application in mobile devices such as cars represents an innovative area and an attractive platform for speech recognition based features since that will enable new in-car applications to develop while taking care of safety aspects. Today, law prescribes the use of hands free devices for drivers in many countries. In addition to that, voice

control together with hands free operation seems to be very cool and attractive. Terminal without any keypad may stand at the end of an evolution path where voice control is virtually the only method of user control.

Current products in the market use dynamic time warping (DTW)-based speaker-dependent (SD) technology since it is of low complexity and language-independent. However, before using the system the user is prompted for sample utterances of the words in a training phase. Besides, the distance measures used in DTW based speech modeling do not consider the feature distribution. Another disadvantage is that the temporal modeling is not very specific and often is only based on heuristic penalties.

Researchers currently focus their work on the use of a stochastic process model to learn acoustic knowledge from samples of speech. The most commonly used stochastic modeling approach is based on Hidden Markov Model (HMMs) where a set of underlying states are connected to form a Markov chain which is assumed to have generated the speech.

HMM based recognizers imply a higher implementation complexity and need appropriate speech databases in specific environments and in many languages. The system performs well as long as it follows well-established techniques in a silent environment. However, performance rapidly degrades in (more realistic) noisy environments. This work therefore focuses on a number of issues such as robustness in adverse car environments, computational costs, and end users' acceptance for further improvements of the HMM based speech recognition.

Robustness in adverse environments can be tackled by improving the front-end algorithm, while the computational costs must be tackled at all level in the recognition process. In the end, a product has to be launched to the end-user customers.

Therefore, convenience in using the system is also one of the system requirements. This can be achieved by implementing speaker-independent (SI) technology since it offers recognition of words without requiring training by the end-users.

When dealing with the noise robustness and computational costs of Automatic Speech Recognition (ASR), it is a common practice to analyze in two development levels: the algorithmic level and the architectural level. At the first level, resource consumption relies on the complexity of the front-end acoustic models and noise reduction methods as well as the back-end search algorithms. At the second level, it depends on the precision of data representation and the implementation cost of arithmetic operations. Therefore, the final algorithms adopted are written using fixed point arithmetic which strive for high-speed and low resource as they would finally be ported into DSP processors.

1.2 Objectives

The techniques described in previous section show the direction of this work. This work focuses on robustness in adverse car environments, low computational costs, and end users' acceptance for further improvements of the HMM based speech recognition. The efficiency and effectiveness of each technique would be further analyzed using performance measurements commonly used in the speech recognition field as well as proposed methods from other field such as image processing.

To achieve those objectives, there is a need developing a noise reduction module and to find a way to integrate this module into a speech recognizer. Therefore, the goals of the thesis project are:

- To conduct a literature study on noise reduction and automatic speech recognition in general and to determine how to develop an ASR-engine.
- To study the STMicroelectronics ASR, a toolkit that supports the development of automatic recognition systems based on Continuous Hidden Markov Model. It had to be determined how the toolkit works and how it can be utilized in the development of an ASR-system.
- To develop a Noise Reduction module that would be integrated into the ASR.
- To develop an ASR-engine that can be used for experiments and future projects.
- To analyze the performance of the whole system.

1.3 Contributions

In order to make speech recognition useful in mobile device, special problems related to implementation constraints and adverse conditions have to be solved. In this thesis, we presented the characteristics of the STMicroelectronics ASR (Automatic Speech Recognizer) solutions on some key areas. We introduced the ASR methods for robustness improvement concentrating on the feature extraction. Our test results on real-world data show the benefit of our methods in particular in the interesting cases of high noise and high mismatch between training and testing.

The use of HMM recognizers in mobile devices is a question of available resources. The ASR uses segmented utterances training because it results in high-quality HMMs and memory efficiency at the same time. According to our test results

using SpeechDat-Car database, this kind of training result in a recognizer that performs better while using fewer mixtures. By allowing the use of small models, this training increases the memory efficiency while still maintaining good recognition accuracy.

We also investigated the effect of porting the ASR from floating-point development platform to fixed-point implementation platform. Using the same databases, a high accuracy of our porting techniques was proven. With this accuracy, floating-point trained HMMs can be used in the fixed-point code without any impact on the recognition rate.

1.4 Outlines

The rest of the thesis is organized as follows: A review of the state-of-the-art techniques on speech recognition as well as a baseline speech recognition system that has been developed by STMicroelectronics and the speech corpus to develop the recognizer are given in Chapter 2. The possible extension of feature extraction module to improve the robustness of the baseline system under adverse condition of car environment is discussed in Chapter 3. The development of this speech recognizer is described in Chapter 4. A number of experiments that were conducted using the developed modules are described in Chapter 5. Summaries and future explorations are given in Chapter 6.

Chapter 2

Related Works

In this chapter we give an overview of a speaker-independent continuous Hidden Markov Model (SI-CHMM) algorithm for speech recognition of isolated words for medium vocabulary. Following that, the current trends in speech recognition are presented, both on the front-end feature extraction and on the back-end SI-CHMM. In Chapter 3 we extend the techniques introduced in this chapter by proposing refinements which strive for noise robust, high-speed and low resource algorithms.

2.1 Introduction

There are several motivations for attempting SI-CHMM of isolated word recognition for medium vocabulary. Speaker independence is required to obtain the end user's acceptance due to the absence of training by the end user. HMMs are capable of robust and succinct modeling of speech. Furthermore, efficient maximum-likelihood algorithms exist for HMM training and testing. Good word decoding leads to good sentence decoding, and the ability to recognize the word accurately will undoubtedly provide the basis for an accurate sentence recognizer. Based on the success or failure

of this study, we can predict whether large-vocabulary sentence recognition based on word HMMs is viable.

Also, by evaluating our system on a car environment-based database, we provide a benchmark that allows direct comparison between many approaches that are suitable to be used in the adverse environment. When dealing with the noise robustness in the adverse environment, it is a common practise to solve the problem in different stages of an ASR system. These approaches to noise robustness in different stages of an ASR system are noise reduction and compensation techniques (front-end stage), model adaptations, speaker adaptive acoustic modeling and observed data classification (back-end stage).

2.2 Baseline Recognizer

STMicroelectronics (STM) Singapore has been developing a speech recognition system which is to be ported to a selected DSP. Following the current trend, the selected DSP still uses fixed-point processors due to the high cost of using floating-point processors in mobile devices. Because of this, the burden of implementing any algorithms in DSP systems are still on the DSP programmers, which have no option but to use fixed-point coding. This means that the programmers must not only deal with mathematically sophisticated applications, but also deal with the errors introduced by performing these applications using reduced-precision arithmetic.

While it would be ideal if programmers could avoid using fixed-point arithmetic, as a practical consideration, they cannot. Fixed-point DSPs execute at gigahertz rates;

floating-point DSPs peak out in the 300-400 megahertz range. The reduced precision and range of fixed-point numbers means that fixed-point applications encounter stability problems more easily than floating-point applications, but both types of applications encounter them. Therefore, trade-offs have always to be made at every stage of the system between accuracy/stability and computational cost (speed and energy consumption).

The baseline system consists of three main modules: front-end feature extraction, back-end training of model parameters and back-end recognizer. As front-end feature extraction and back-end recognizer would be ported into a DSP processor, the complexity of the algorithms have to be taken into account besides the performance in term of word accuracy. Besides easing the porting task to a DSP processor, implementing the two modules in fixed-point arithmetic also helps estimating the computational costs of the algorithms in the DSP processor.

In the following we describe the speech recognition system, which is used as the baseline for the project.

2.2.1 Feature extraction

There are three main objectives of feature extraction: To capture essential information for speech recognition; To compress information into a manageable form; To remove irrelevant information to recognition, such as, background additive noise, long-term channel transmission characteristics and speaker-specific information. The feature extraction module in the system uses the mel frequency cepstral coefficients (MFCCs) method, which is motivated by properties of human auditory system's response.

The module, showed in Figure 2.1, processes the input signal, pre-emphasizing it

Figure 2.1: Front-end feature extraction processing.

and blocking it into frames of 32 ms duration (with 25% frames overlapping). For each frame, the Fourier transform is taken. The log amplitudes of the spectrum obtained are then mapped onto the Mel scale, using triangular overlapping windows. The Discrete Cosine Transform of the list of Mel log-amplitudes is taken as if it were a signal, resulting in 12 MFCCs.

The basic main path of the front-end processing is discussed as follows:

Pre-emphasis (PE) The source-filter model is the basis for many of the techniques used to extract acoustic information from the speech signal. It consists of a uniform tube with a sound source at one end to represent vocal cords and open at the other to represent lips. In this model, the excitation source is assumed to be linearly separable from the transmission characteristics of the vocal tract, which are represented by a quasi-time-invariant filter. The speech waveform itself is then assumed to be the output of this filter in response to the excitation source, which is either a quasi-periodic pulse generator (voiced sounds), a random-noise generator (unvoiced sounds) or a mixture of both (voiced fricatives).

Figure 2.2: Illustration of pre-emphasis on speech spectrum.

According to the source-filter model there is an overall -6 dB/octave trend as frequency increases. This is a combination of a -12 dB/octave trend due to the voiced excitation source and +6 dB/octave trend due to radiation from the mouth. This means that, for each doubling in frequency, the signal amplitude, and hence the measured vocal tract response, is reduced by a factor of 16. It is therefore desirable to compensate for the -6 dB/octave roll-off by pre-processing the speech signal to give a +6 dB/octave lift in the appropriate range so that the measured spectrum has a similar dynamic range across the entire frequency band. This is referred as **pre-emphasis**. Figure 2.2 shows the effect of the filter on typical speech spectrum.

The pre-emphasis is done on 4 frames/window at a time using a first order high pass filter. In the time domain, the following equation is used:

$$S_{PE}(n) = S_{in}(n) - 0.95S_{in}(n-1), \quad (2.2.1)$$

where $S_{in}(n)$ is the bit stream input signal value at index n . The pre-emphasized

signal $S_{PE}(n)$ is first normalized before being sent to the next block for frame formation.

Framing (FR) The speech is not a stationary signal and we prefer not to represent the entire signal by just one Fourier transform. Therefore the signal is divided into successive overlapping frames. The sampling rate is 8 KHz. The frame size and frame shift are chosen to be 15 ms and 8 ms respectively. Thus, every 15 ms we get a new chunk of signal composed by 64 overlapping samples and 120 new samples. As the FFT algorithm requires to work with data size of power of two, the 184 samples are formed into one frame by padding the rest with zero to get 256 samples per frame. See Figure 2.3 for illustration.

To every framed signal $S_{fr}(n)$, a windowing function is applied to suppress the effect of discontinuities at frame edges. The system uses commonly used window for *narrow band application* called *Hamming* window. Hamming function is moderate window in terms of resolution characteristics. Excellent resolution characteristics is a good choice for signals of comparable strength such as the pre-emphasized signals (low dynamic range). This will improve the sensitivity of the analysis, that is, if the input waveform contains random noise close to the signal frequency, the response to noise, compared to the signal, would be lower than with a lower-resolution window. In other words, the ability to find weak signals amidst the noise is improved by a low-dynamic range window. Below is the equation for Hamming window.

$$w(n) = 0.54 - 0.46 * \cos\left(\frac{2\pi n}{N-1}\right) \quad (2.2.2)$$

Figure 2.3: Framing illustration.

The windowed frame is first transformed into frequency domain to get the power spectral density.

Power Spectral Density (PSD) The Discrete Fourier Transform (DFT) is employed to analyze the frequencies contained in the sampled signal. The DFT can be computed efficiently in practice using a Fast Fourier Transform (FFT) algorithm. We use two twin FFTs of the 4 windows to save some computation costs, and we actually obtain as results real and imaginary parts of the FFT. Thus, the computation of the PSD becomes straightforward.

$$FFT_bin_k = \sum_{n=0}^{FFTL-1} S_{fr}(n).e^{-jnk\frac{2\pi}{FFTL}} \quad (k = 0, \dots, FFTL - 1). \quad (2.2.3)$$

Because $S_{fr}(n)$ is real-valued signal, FFT_bin_k is symmetric. So, only items for $k = 0, \dots, \frac{FFTL}{2}$ are used for further processing. The squared value of the FFT magnitude corresponds to the PSD.

$$\begin{aligned} PSD_bin_k &= \left| \sum_{n=0}^{FFTL-1} S_{fr}(n).e^{-jnk\frac{2\pi}{FFTL}} \right|^2 \\ &= Re\{FFT_bin_k\}^2 + Im\{FFT_bin_k\}^2, \quad (k = 0, \dots, FFTL - 1) \end{aligned} \quad (2.2.4)$$

The algorithm used to compute the FFT is two twin FFTs so that 4 frames can be processed at one time. The idea behind the twin FFT computation is that we are able to perform the FFT of a complex signal as easily as the FFT of a real signal. But all signals we are dealing with are real. In order to save some computation (the twiddle factors), we can do a twin FFT instead of two real FFTs. As the FFT is a

linear operation, we can send inputs of the FFT algorithm a complex signals which are built as follow:

Let $x_1[n]$ and $x_2[n]$ be our two real signals of the frame size (256 data points), we build $x[n]$ as

$$x[n] = x_1[n] + i.x_2[n] \quad for \quad 0 \leq n \leq N - 1 \quad (2.2.5)$$

then if we perform the FFT:

$$X[k] = X_1[k] + i.X_2[k] \quad (2.2.6)$$

we can then express x_1 and x_2 as:

$$x_1[n] = \frac{x[n] + x^*[n]}{2} \quad and \quad x_2[n] = \frac{x[n] - x^*[n]}{2i} \quad (2.2.7)$$

knowing that $DFT(x^*[n])$ is $X^*(N - k)$, we have:

$$X_1[k] = \frac{1}{2}[X[k] + X^*[N - k]] \quad and \quad X_2[k] = \frac{1}{2i}[X[k] - X^*[N - k]] \quad (2.2.8)$$

We use a radix-2, Decimation in Time implementation of the FFT. It uses the symmetry and periodicity properties of the twiddle factor to simplify the computation.

The information contained in the frequency-dependent representation PSD_{bin_k} is then regrouped in a *Mel* band representation, which is based on Human Auditory System.

Mel band regrouping (MEL) To approximate the human auditory system's response more closely than the linearly-spaced frequency bands obtained directly from the FFT, the frequency bands are positioned logarithmically on the *mel* scale. The mel scale, proposed by Stevens, Volkman and Newman in 1937 is a perceptual scale of pitches judged by listeners to be equal in distance from one another. The reference point between this scale and normal frequency measurement is defined by equating a 1000 Hz tone, 40 dB above the listener's threshold, with a pitch of 1000 mels. Above about 500 Hz, larger and larger intervals are judged by listeners to produce equal pitch increments. As a result, four octaves on the hertz scale above 500 Hz are judged to comprise about two octaves on the mel scale.

The useful frequency band lies between $f_{start} = 0$ Hz and $f_s/2 = 8$ KHz in which f_s corresponds to the index value $k = FFTL$ in the FFT calculation. This band is divided into $Q = 30$ channels equidistant in the Mel frequency domain. Each channel has triangular-shaped frequency window. Consecutive channels are half-overlapping, uniformly spaced before 1 KHz and logarithmic scale after 1 KHz. See Figure 2.4 for illustration.

The output of the Mel-band filtering Mel_bin_i is the weighted sum of the PSD spectrum values PSD_bin_k in each band calculated as follows:

$$Mel_bin_i = \sum_{k=0}^{FFTL/2} W_{i,k} \cdot PSD_bin_k, \quad (2.2.9)$$

where $W_{i,k} = Mel_weightage$, $i = channel_index = 1, \dots, Q$ and $k = PSD_bin_index$.

The Mel filter bank processing is illustrated in Figure 2.5. In short, the spectrum PSD is first subjected to the bank of filters according to Mel scale. The filter output

Figure 2.4: Triangular filters suggested by Davis and Mermelstein (1980) for transforming the output of a PSD transform onto a mel scale in both bandwidth and spacing.

is the sum of its filtered spectral components.

Cepstral coefficient calculation (DCT) Because the log power spectrum is real and symmetric function, inverse Fourier transform to obtain cepstrum (an anagram of spectrum) reduces to a Discrete Cosine Transform (DCT).

$$C_j = \sum_{i=1}^Q \log(\text{Mel_bin}_i + 16) \cdot \cos\left(\frac{j}{Q}(i - 0.5)\right) \quad (j = 0, \dots, 12) \quad (2.2.10)$$

The DCT has the property to produce highly uncorrelated features. This improves the diagonal assumption of HMM modeling which simply uses linear combinations of Gaussian density with diagonal covariance matrices. The transformation also makes

Figure 2.5: Mel filter bank processing.

Figure 2.6: Property of cepstrum.

the information that is of phonetic significance become concentrated in the lower-order terms. Therefore filtering the cepstrum (a process usually referred as *liftering*) can be applied to remove certain components. The liftering used in the given algorithm is implemented as follows:

$$dp_lift_fix[k] = 8 + 4.Q.\sin(\frac{i+1}{Q}) \quad (2.2.11)$$

Each ceptrum C_j is then normalized with the root-mean-square (RMS) of the coefficients computed on each frame and sent to the back-end module.

2.2.2 Back-end SI-CHMM

The SI-CHMM module is based on word-level modeling units. Each acoustic-word unit is modeled with left-to-right CHMMs or Bakis model, i.e, the underlying state sequence has the property that as the time increases the state index increases. Clearly the left-right type of HMM has the desirable property that it can readily model signals whose properties change over time, e.g., speech. As speeches are continuous signals (or vectors), it is advantageous to use HMMs with continuous observation densities. The system uses a Gaussian density as the form of model probability density (pdf) to ensure that the parameters of the pdf can be re-estimated in a consistent way.

Introduction Hidden Markov Models are based on the well-known Markov chains from probability theory that deal with a class of random processes having one-step memory. That is, they have the so-called Markov property: the probability that the system will be in a certain state at the next time step only depends on the current state. A Markov chain can be viewed as a finite state process with a probabilistic

transition function. In the Markov model each state corresponds to one observable event. But this model is too restrictive, for a large number of observations the size of the model explodes, and the case where the range of observations is continuous is not covered at all. Therefore the Hidden Markov concept extends the model by making the observation a probabilistic function of the state, this allows more freedom to the random process while avoiding a substantial complication to the basic structure of Markov chains. The resulting model is a doubly embedded stochastic process in which the underlying stochastic process is not directly observable but can be observed only through another set of stochastic processes that produce the sequence of observations. The Hidden Markov Model is defined as follows:

1. The number of states N .
2. The number of distinct observation symbols M .
3. An output alphabet $\mathbf{O} = O_1, O_2, \dots, O_M$.
4. A state space $\mathbf{Q} = q_1, q_2, \dots, q_n$.
5. A probability distribution of transitions between states $\mathbf{A} = a_{ij}$ where $a_{ij} = P(q_{t+1} = j | q_t = i) \quad 1 \leq i, j \leq N$
6. An observation symbol probability distribution $\mathbf{B} = b_j(k)$ in which $b_j(k) = P(o_t = o_k | q_t = j) \quad 1 \leq k \leq C, \quad 1 \leq j \leq N$
7. The initial state distribution $\pi = \pi_i$ where $\pi_i = P(q_1 = i) \quad 1 \leq i \leq N$
8. To indicate the complete parameter set of a model the notation $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ is used.

The observation probability distributions are usually all defined on the same domain, so in theory each state is capable of generating every possible observation symbol.

HMM generator of observations Given appropriate values of N , M , $\mathbf{A}$, $\mathbf{B}$, π and an alphabet the HMM can be used as a generator to give an observation sequence $\mathbf{O} = o_1 o_2 \dots o_T$ by performing the following steps:

1. Choose an initial state $q_1 = i$ according to the initial state distribution π .
2. Set $t = 1$.
3. Choose $o_t = O_k$ according to $b_j(k)$.
4. $t = t + 1$, transit to a new state $q_{t+1} = j$ according to a_{ij} .
5. Return to step 3 if $t < T$.

But if recognition is to be performed with a Hidden Markov Model the observation sequence is already available and we want to know how well a given model matches a given observation sequence, that is we want to know the probability that the observation sequence was produced by the model λ .

Now, when we have an observation sequence and know which model is most likely to generate the sequence, it is not clear how the model generates that sequence, because the underlying state sequence is hidden. Since practically any state sequence could generate each observation sequence there is no correct state sequence to be found here. All we can do is to try to solve this problem as best as possible and seek the most likely state sequence given the observation sequence and the model.

Another important issue that has to be addressed is how to obtain and optimize the model parameters to maximize $P(\mathbf{O}|\lambda)$. This is called the training problem. The next three sections discuss each of these fundamental questions concerning Hidden Markov Models in turn.

Forward algorithm When we consider a fixed state sequence $\mathbf{q}$ of length T , the probability of the observation sequence given the model λ and sequence $\mathbf{q}$ is stated by

$$P(\mathbf{O}|\mathbf{q}, \lambda) = \prod_{t=1}^T P(O_t|q_t, \lambda) = b_{q_1}(O_1)b_{q_2}(O_2)...b_{q_T}(O_T) \quad (2.2.12)$$

(Assuming statistical independence of observations).

There are N^T possible state sequences $\mathbf{q}$, each with probability:

$$P(q|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \dots a_{q_{T-1} q_T} \quad (2.2.13)$$

The probability of the observation sequence $\mathbf{O}$ given the model is now obtained by taking the joint probability of 2.2.12 and 2.2.13 and summing over all possible state sequences, giving

$$P(O|\lambda) = \sum_{\forall \mathbf{q}} P(\mathbf{q}|\lambda) \prod_{t=1}^T P(O_t|q_t, \lambda) \quad (2.2.14)$$

$$= \sum_{q_1, q_2, \dots, q_N} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \dots a_{q_{T-1} q_T} b_{q_T}(O_T) \quad (2.2.15)$$

As can be seen from equation 2.2.15 this calculation of $P(\mathbf{O}|\lambda)$ involves on the order of $2TN^t$ calculations since at every time $t = 1, 2, \dots, T$ there are N^t possible state

sequences and for each such state sequence about $2T$ calculations are required. So this calculation is computationally infeasible. Therefore a more efficient procedure is needed. Fortunately, such a procedure exists; it is called the forward procedure. For this procedure the probability at $\alpha_t(i)$ defined as the probability of being in state i at time t and having observed the partial observation sequence $o_1o_2o_t$ so far, given the model λ .

$$\alpha_t(i) = P(O_1O_2...O_t, q_t = i | \lambda) \quad (2.2.16)$$

Now $\alpha_t(i)$ can be computed inductively, as follows:

1. Initialization

$$\alpha_1(i) = \pi_i b_i(O_1) \quad 1 \leq i \leq N \quad (2.2.17)$$

2. Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(O_{t+1}) \quad 1 \leq t \leq T-1, \quad t \leq j \leq N \quad (2.2.18)$$

3. Termination

$$P(O | \lambda) = \sum_{i=1}^N \alpha_T(i) \quad (2.2.19)$$

Step one initializes the forward probabilities as the joint probability of state i and initial observation o_1 . The induction step follows from the fact that the product $\alpha_i(t) a_{ij}$ is the probability of the event that $o_1o_2...o_t$ are observed and state j is reached at time $t+1$ via state i at time t . Summing this product over all the N possible states, $i \ 1 \leq i \leq N$, at time t results in the probability of j at time $t+1$ with all of the accompanying previous partial observations. Now all that is left to do is to take account for observation o_{t+1} in state j , this is done by use of the probability $b_j(O_{t+1})$.

Viterbi algorithm Finding the most likely state sequence $q_1 q_2 \dots q_T$ given an observation sequence $\mathbf{O} = o_1 o_2 \dots o_T$ boils down to maximizing $P(\mathbf{q}|\mathbf{O}, \lambda)$ which is equivalent to maximizing $P(\mathbf{q}, \mathbf{O}|\lambda)$. Therefore we define $\delta_t(i)$ as the best score (highest probability) along a single path, at time t , which accounts for the first t observations and ends in state i .

$$\delta_t(i) = \max_{q_1, q_2, \dots, q_{t-1}} P(q_1 q_2 \dots q_{t-1}, t = i, o_1 o_2 \dots o_t | \lambda) \quad (2.2.20)$$

We can calculate $\delta_t(i)$ using a recursive procedure similar to the forward algorithm, but this time using a maximization over previous states instead of a summing procedure. The optimal path can be found by keeping track of the argument i that maximized $\delta_t(j)$ in equation 2.2.20.

1. Initialization

$$\delta_1(i) = \pi_i b_i(o_1) \quad 1 \leq i \leq N \quad (2.2.21)$$

$$\psi_1(i) = 0 \quad (2.2.22)$$

2. Recursion

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(o_t) \quad 2 \leq t \leq T, 1 \leq j \leq N \quad (2.2.23)$$

$$\psi_t(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] \quad (2.2.24)$$

3. Termination

$$\tilde{P} = \max_{1 \leq i \leq N} [\delta_T(i)] \quad (2.2.25)$$

$$\tilde{q}_T = \arg \max_{1 \leq i \leq N} [\delta_T(i)] \quad (2.2.26)$$

4. Path backtracking

$$\tilde{q}_t = \psi_{t+1}(\tilde{q}_{t+1}) \quad t = T-1, T-2, \dots, 1 \quad (2.2.27)$$

The Viterbi algorithm takes $O(N^2T)$ calculations. By taking the logarithms of the model parameters, the Viterbi algorithm can be implemented without the need for any multiplications.

The next section describes a way of estimating the parameters of an HMM, but the Viterbi algorithm can also be used to find an (initial) estimate of the models state distribution parameters, given an observation sequence $\mathbf{O} = o_1o_2o_T$ and a particular HMM whose parameters are to be estimated. The procedure that accomplishes this is called Viterbi alignment. It consists of two steps. In the first step the Viterbi algorithm is used in the normal way, most likely sequence of states $\mathbf{q} = q_1q_2q_n$ that generated $\mathbf{O}$ is found. In the second step once again the concept of an HMM as generator of speech vectors is used. Each state q_i is thought of as having generated the corresponding subset $\mathbf{O}_i$ from the observation sequence. Now $\mathbf{q}$ effectively segments $\mathbf{O} = O_1||O_2||\dots||O_n$. These subsets can then be used to calculate the state means and variances.

Baum-Welch algorithm The third problem of HMMs is to determine a method to obtain or adjust the model parameters $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$. We would like to have a method that has the following maximum likelihood property:

$$\hat{\lambda} = \arg \max_{\lambda} P(\mathbf{O}|\lambda) \quad (2.2.28)$$

$\hat{\lambda}$ would allow the HMM to account best for the observed data $\mathbf{O}$. What we really want is for $\hat{\lambda}$ to account for as yet unseen data, so the best we can do here is to use a

training set that is as representative as possible. There is no known way to analytically solve for the model parameter set that maximizes the probability of the observation sequence. But we can choose $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ such that its likelihood $P(\mathbf{O}|\lambda)$ is locally maximized using a sample of the type of data the model is supposed to generate. This can be done by using an iterative procedure known as the *Baum-Welch* algorithm or Forward-Backward algorithm. To get to this re-estimation algorithm we first have to define a backward variable $\beta_t(i)$ analogues to the forward variable $\alpha_t(i)$ from section 2.2.2.

$$\beta_t(i) = P(O_{t+1}O_{t+2}\dots O_T | q_t = i, \lambda) \quad (2.2.29)$$

That is the conditional probability that $o_{t+1}o_{t+2}o_T$ are observed and the system starts in state $\mathbf{q}_j$ at time t , given the model λ . $\beta_t(i)$ can be computed inductively:

1. Initialization

$$\beta_T = 1 \quad 1 \leq i \leq N \quad (2.2.30)$$

2. Induction

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \quad 1 \leq i \leq N \quad (2.2.31)$$

The initialization step arbitrarily defines $\beta_t(i)$ to be 1 for all i .

Now we can define the probability of being in state i at time t given the entire observation sequence and the model as:

$$\gamma_t(j) = \frac{\alpha_t(j) \beta_t(j)}{P(\mathbf{O}|\lambda)} \quad (2.2.32)$$

And the probability of being in state i at time t and state j at time $t + 1$, given the model and the observation sequence by:

$$\xi(i, j) = \frac{P(q_t = i, q_{t+1} = j, \mathbf{O} | \lambda)}{P(\mathbf{O} | \lambda)} \quad (2.2.33)$$

$$= \frac{\alpha_t(i) a_{i,j} b_j(o_{t+1}) \beta_{t+1}(j)}{P(\mathbf{O} | \lambda)} \quad (2.2.34)$$

If $\xi_t(j)$ is summed over the time index t , the expected number of times that state i is visited is obtained or equivalently the number of transitions made from state i . Similarly, summation of $\xi_t(i, j)$ over t can be interpreted as the expected number of transitions from state i to state j . Now, using the concept of counting event occurrences, we can estimate a_{ij} as the expected number of transitions from state i to state j normalized by the expected number of transitions from state i :

$$\tilde{a}_{i,j} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad (2.2.35)$$

And similarly, $b_j(k)$ can be estimated by dividing the expected number of times in state j at which symbol o_k was observed by the expected number of times the system is in state j :

$$\tilde{b}_j(k) = \frac{\sum_{t=1, o_t=o_k}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)} \quad (2.2.36)$$

The initial state distribution for state j is simply equal to the expected frequency with which state j is visited:

$$\tilde{\pi}_j = \gamma_1(j) \quad (2.2.37)$$

Since the left hand side of these equations also appears on the right hand side we have to use an iterative procedure to improve the model parameters. After starting with an initial guess for $\mathbf{A}$, $\mathbf{B}$, and π , λ is used in place of λ in each iteration until the values stop changing within certain limits.

Up to this point the algorithms were explained terms of a finite alphabet of observation symbols and thus discrete probability functions could be used to generate these observations. But as mentioned earlier in speech recognition, continuous signal observation vectors are preferred. Older system, however, did use a limited codebook of observation symbols. During preprocessing vectors were then mapped to one of the observation symbols using a k-means segmentation. In modern state of the art systems the observation vectors are put directly into the Hidden Markov Models, therefore multivariate continuous probability density functions are needed to model the distribution of the observations. The form of the density functions then becomes

$$b_j(o_t) = N(o_t, \mu_j, U_j) \quad (2.2.38)$$

Where N is an elliptical symmetric density (e.g. Gaussian) with mean vector μ_j and covariance matrix U_j for state j . It can be shown that the re-estimation formulas for these parameters are of the form:

$$\mu_j = \frac{\sum_{t=1}^T \gamma_t(j) o_t}{\sum_{t=1}^T \gamma_t(j)} \quad (2.2.39)$$

$$U_j = \frac{\sum_{t=1}^T \gamma_t(j) (o_t - \mu_j)(o_t - \mu_j)'}{\sum_{t=1}^T \gamma_t(j)} \quad (2.2.40)$$

Figure 2.7: HMM-model topology.

HMM topology for isolated word recognition The topology of an HMM is designed using knowledge of the situation and to some extent, designer's intuition. The first decision to an HMM topology has been chosen in the project, that is the use of word as the HMM unit. In fact it is possible to use HMMs to model any unit of speech. Even if the speech units are poorly selected, HMMs have the ability to absorb the suboptimal characteristics within the model parameters; this might of course limit the performance of the system.

Word are chosen as they seem to be the most natural units to model, because these are what we want to recognize. The success of using word-level model is partly due to the fact that they are able to capture within-word phoneme co-articulation effects. However, as there are many unique words, training data are needed for each of these words, making this kind of system not easily extendible. So for large vocabulary natural speech recognition word units are not really an option. But for small well-defined vocabularies, for example a set of commands, they are well suited.

Usually left-to- right model topologies are used in which the number of states

depends on the number of phonemes in the word. One state per phoneme is a good rule of thumb. Figure 2.7 shows the model topology that we use in the project. The model has one state per phoneme, but provides transitions that skip the succeeding state. This model also has state dependent output probabilities.

CHMM training for isolated word recognition Good initial estimates of the parameters of $b_j(o_t)$ densities are essential for rapid convergence of the re-estimation formulas (Eq 2.2.39 and 2.2.40). Hence a procedure for providing good initial estimates of these parameters was devised as shown in Figure 2.8. The training procedure is a variant on the well-known K -means [36] iterative procedure for clustering data.

Following model initialization, the set of training observation sequences is segmented into states, based on the current model λ . This segmentation is achieved by finding the optimum state sequence, via the Viterbi algorithm, and then backtracking along the optimal path.

The result of segmenting each of the training sequences is, for each of the N states, a maximum likelihood estimate of the set of the observation that occur within each state j according to the current model. Because we are using continuous observation densities, a segmental K -means procedure is used to cluster the observation vectors within each state j into a set of M clusters (using a Euclidean distortion measure), where each cluster represents one of the M mixtures of the $b_j(o_t)$ density. From the clustering, an updated set of model parameters is derived as follows:

- c_{jm} = number of vectors classified in cluster m of state j divided by the number of vectors in state j

Figure 2.8: HMM training to estimate model parameters (after Rabiner et al. [35]).

- μ_{jm} = sample mean of the vectors classified in cluster m of state j
- U_{jm} = sample covariance matrix of the vectors classified in cluster m of state j .

Based on this state segmentation, updated estimates of the a_{ij} coefficients can be obtained by counting the number of transitions from state i to j and dividing it by the number of transitions from state i to any state (including itself).

An updated model λ' is obtained from the new model parameters, and the formal re-estimation procedure by Rabiner and B.H Juang [35] is used to re-estimate all model parameters, λ' . The resulting model is then compared to the previous model (by computing a distance score that reflects the statistical similarity of the HMMs). If the model distance score exceeds a threshold, then the old model λ is replaced by the new (re-estimated) model λ' , and the overall training loop is repeated. If the model distance score falls below the threshold, then model convergence is assumed and the final model parameters are saved.

CHMM testing for isolated word recognition CHMM testing/recognition takes two main steps in obtaining the scores (model likelihoods): Likelihood evaluation and Viterbi decoding. For each unknown word to be recognized, the processing as shown in Figure 2.9 must be carried out, namely, measurement of observation sequence $O = o_1 o_2 \dots o_T$, via a feature extraction procedure explained in Section 2.2.1; followed by calculation of model likelihoods for all possible models (V words to be recognized), $P(O|\lambda_v), 1 \leq V \leq V$; followed by selection of the word whose model likelihood is highest-that is, $v^* = \arg \max_{1 \leq v \leq V} [P(O|\lambda_v)]$.

Figure 2.9: HMM testing to recognize words.

Likelihood evaluation is a prerequisite for Viterbi decoding. In likelihood evaluation, the state likelihood $b_j(O_t)$ are computed for every feature vector O_t , $t=1$ to T at every state j , $j=1$ to N , of an utterance. The likelihoods are parameterized by Gaussian mixtures as follows:

$$b_j(O_t) = \sum_{i \in M_j} w_i N(O_t; \mu_i, \Sigma_i)$$

where M_j is a subset of Gaussians belonging to state j ; μ_i , Σ_i and w_i are the mean, covariance matrix and weight of the i^{th} Gaussian component respectively. Viterbi decoding, in which the best path score is used instead of the expectation of all the path scores, is chosen instead of full inference to reduce computation. It only requires on the order $V.N^2.T$ computations. For modest vocabularies, e.g., $V = 40$ words, with an $N = 10$ state model, and $T = 40$ observations for the unknown word, a total of 16×10^4 computations is required for recognition (where each computation is a multiply, and add, and a calculation of observation density, $b(\mathbf{o})$). Clearly this amount of computation is modest compared to the capabilities of most modern DSP chips.

2.3 Current State of the Art Techniques

In the past few years there has been a large and continuing increase in the number and range of products and services that incorporate speech technology. This increase in applications is due partly to the advances in the methods used in speech recognition, but also to the more general progress that has been made in computer technology.

This section only discusses the influence that the current methods made in speech

recognition field. The subsection division is based on specific areas in speech recognition that the researchers are currently working on. These are noise reduction and compensation techniques, model adaptations, speaker adaptive acoustic modeling, observed data classification and real time operation in fixed-point processor as well as fixed-point code design considerations.

2.3.1 Noise reduction and compensation techniques

When the surrounding environment is quiet, talking into good-quality close-distance microphone connected directly into an ASR system, the signal input to recognizer is very close to the output of the person's mouth. This ideal situation does not usually exist because the input signal has been corrupted in some way [18]:

There may be external noise present in the signal. Sources of environmental noise include computers and other office equipment, car engine, music, radio, and even other people talking in the background. A speech signal captured in these conditions will contain *additive noise*.

The uttered speech signal itself may undergo some spectral distortion during its transmission from the talker's mouth to the speech recognizer. Sources of such distortion include room reverberation, the microphone transducer and the transmission channel. The effect of these disturbances can be viewed as one of convolving the original speech signal with a filter representing the characteristic of the disturbance, and hence this type of corruption to speech signals is often referred to as *convolutional noise*.

Additive noise is easier to deal with in the linear spectral domain, where spectral

components due to the noise can be seen as being added to the components representing the speech. Convolutional noise on the other hand is easier to cope with in the log spectral or cepstral domain.

Techniques for handling either type of noise can be roughly divided into two categories [18]. The first category (*pre-processing*), encompasses methods that are applied purely at the level of the features, and includes both the selection of features that are inherently robust to corruption and the application of speech enhancement techniques to existing feature sets. Well-known techniques such as reinforced *spectral subtraction* [33], *cepstral channel normalization* or *cepstral mean normalization* [11], *relative spectral* [10] are of this category.

The second category (*noise compensation*) covers model-based methods whereby some way of dealing with the effect of noise is incorporated into the recognition process itself. Well-known methods such as *noise masking* [2] and *Parallel Model Combination* (PMC) [13] are of this category.

Feature-based techniques

Bernhard N. et al. [31] suggest two noise reduction methods, time domain noise reduction (TDNR) and frequency domain noise reduction (FDNR). Both methods try to reduce the effect of the additive noise.

The main idea of TDNR lies on the computation of a filter transfer function which is used to refine the estimate of noiseless signal spectrum. The impulse response of this filter transfer function is then computed, truncated and windowed. The noise-reduced signal is finally obtained by convolving the noisy input signal with the filter impulse response.

FDNR algorithm uses a modified maximum likelihood estimation [45], which relies heavily on estimated SNR values. To reduce recognition errors due to spectral distortion caused by badly estimated SNR values, FDNR uses smoothing in frequency domain (mainly) and in time domain. The frequency domain smoothing/filtering reduces the distortion in static cepstral coefficients while the time domain filtering reduces the distortion in the dynamic cepstral coefficients.

For convolutional noise, method such as blind equalization [22] was suggested. It relies on a LMS algorithm, which adjusts the cepstral coefficients according to the difference between the current cepstral vector and a reference cepstrum. Another well-known method is Relative Spectra (RASTA), in which a band-pass filter is used to remove mainly the DC value of the log filter bank magnitude. This DC value is believed to contain the stationary convolutive distortion introduced by the channel, e.g microphone, telephone line, which appears as an additive component in the logarithmic spectral energy domain.

It is well-known that noise reduction schemes such as mentioned above are beneficial in ASR to reduce training-test mismatch due to noise. However, a significant mismatch may still remain, especially in the non-speech portions of the signals. To reduce the impact of this mismatch, Johan de Veth et al. [8] suggest two methods for discarding nonspeech acoustic vectors at recognition time: variable frame rate (VFR) processing and voice activity detection (VAD). With VFR processing, an observation vector is discarded if it does not differ much from the previous observation vector. The VAD algorithm classifies an acoustic vector as speech or non-speech (noise) based on an estimate of SNR estimated from the difference between the short-term log energy and a long-term estimate of non-speech log energy. Vladj et al. [41] however suggest

to use frame attenuation instead of frame dropping as shown by their experiments.

Model-based techniques

Most noise compensation techniques have been limited to the log energy spectral domain, whereas recognition is more normally carried out in the cepstral domain, which is more accurate.

Mellor et al. suggest a noise masking technique [3] used in a cepstral transform domain, providing performance which is not significantly different from masking in the log energy domain. After the log energy filter bank observation vectors are generated, a tracking noise estimate is maintained and masked before the transformation. The masking approach is to maintain a log energy spectral domain version of the models on which to carry out the masking; the masked model means can then be transformed for use in evaluation of the observation probabilities.

Parallel Model Combination [13] starts with a model of speech in quiet environment and some model of the noisy environment and then attempt to predict what will happen to the speech in this noisy environment. The success of this technique is dependent upon the accuracy of the predictive model that is used. For this reason, current state of the art is to use adaptive schemes whereby the new speech data of the new environment are used to adjust the parameters of the models to be more representative of those data. This is discussed in the next subsection, model adaptations.

2.3.2 Speaker adaptive acoustic modeling

One of the system requirement given by STM is that the ASR system has to work with speech from a wide variety of individuals without re-training. It can be done by

employing speaker normalization techniques such as *Vocal Tract Length Normalization* (VTLN) or *Maximum Likelihood Linear Regression* (MLLR).

Welling et al. [43, 42] suggest improvement on VTLN method by using single Gaussian density for selecting speaker-dependent scale factor in training. For testing, they suggest using non-normalized acoustic for the first pass (two-pass strategy). The main idea of VTLN is to normalize the frequency axis of the acoustic vectors for each speaker in the recognition process and thus to remove speaker dependent variability in the acoustic vectors.

MLLR-based speaker normalization was first introduced by Leggetter et al. in [23]. The main idea of MLLR is calculating a linear regression transformation matrices from the available adaptation data in order to maximize the likelihood of the adaptation data. Leggetter et al. only consider the mean vectors which is to be transformed, leaving the covariance matrices unchanged. The variance transform can be estimated separately from the mean transform or alternatively the system can be constrained so that the same transformation matrix is also applied to transform the covariance matrix [14].

2.3.3 Model adaptations

A major issue for ASR is coping with acoustic variability due to differences between different environments, different transmission channel, different talkers and even the same talker on different occasions. Techniques in noise reduction and speaker adaptive acoustic modeling can help to make machines more robust to such variation. However, further performance gain are possible by adapting the parameters of the models to provide a better match to those data.

We are more interested in *unsupervised* adaptation whereby the text of the adaptation data is not known. This is more difficult than the supervised one as recognition must proceed concurrently with adaptation.

Four well-known model adaptation methods are classical *maximum likelihood* (ML) [5, 19, 27] approach and maximum *a posteriori* (MAP) estimation [15], both categorized as *Bayesian* learning, maximum likelihood linear regression [14] and *speaker-adaptive training* (SAT) [12], both categorized as linear transform-based model adaptation.

For ML approach, given some training data Y , values are found for some model parameters θ in order to maximize $P(Y|\theta)$, the probability/likelihood of the data being generated by the model. However, ML estimates tend to be unreliable when the data are sparse. MAP estimation involves finding values of θ that maximize $P(\theta|Y)$. Using Bayes' rule, the MAP estimate of the model parameters θ :

$$\theta_{MAP} = \arg \max_{\theta} P(\theta|Y) = \arg \max_{\theta} P(Y|\theta)P(\theta)$$

where $P(\theta)$ is the *a priori* probability of the model parameters θ . MLLR, as has been described previously, estimates a linear transformation for the Gaussian distribution parameters in order to maximize the likelihood of the adaptation data. For SAT, the adaptation technique is incorporated within the training process as well as being used to adapt to individuals at recognition time.

2.3.4 Observed data classification

The training procedure is quite typical among the researchers. It usually starts with a set of monophones HMMs, using single-Gaussian output distribution. After all possible triphones are transcribed, similar states are tied. Finally, multiple-component

mixture models are trained using a classification procedure. The difference usually lies on the techniques used for the observations mixture clustering.

Two well-known mixture clustering are Segmental K -means (SKM) [20] and mixture splitting [29]. The baseline system uses Segmental K -means, a combination between Viterbi alignment procedure to identify the state-level segmentation and K -means clustering to divide a set of vectors into a specified number of classes in order to locally minimize some within-class distance metric. However, this approach requires the number of mixture components to be decided in advance. Mixture splitting method starts with a single-Gaussian model and incrementally increase the number of mixture components. This is done by duplicating the parameters of the original distribution and perturbing the means by a small amount in opposite directions.

New proposed methods are usually evolved from these two methods. Messina et al. [28] propose a sequential clustering which clusters the training frames in the order in which they are available and according to the density to which they are associated. The difference with SKM is that the Viterbi-alignment is performed only once and not at each iteration.

It should be noted that all the above methods are only to initialize the Gaussian mixture distribution. The re-estimation method mostly still uses the well-known Baum-Welch/Expectation-Maximization algorithm [5].

2.3.5 Real time operation fixed-point processor

In addition to obtaining an acceptable level of recognition accuracy, computationally efficient implementation is needed in order to minimize the unnecessary delays. In CHMM-based recognizer, the computational cost is divided between likelihood

evaluation and search [46]. Likelihood evaluation refers to the computation of the observation probabilities of the HMM states; search refers to the search for the best word candidate or word sequence according to the observation. In the baseline system, these two computations are specially treated in the Likelihood evaluation module and Viterbi decoding module.

The likelihood evaluation can be reduced by using methods such as vector quantisation to pre-select those Gaussians which will sufficiently high likelihood [1], tying at different levels of hierarchical structure of HMMs [25], selecting only features with the highest discriminative power [32], evaluating the component likelihood of certain features and approximating the remaining features by prediction [26] or reducing the number of Gaussian components per state [39, 24].

The search space can be reduced by making approximations which allow alternative paths to be merged, for example by approximating the language model [17], or by limiting the number of new words through some form of look-ahead based on a fast-match pre-selection of possible followers [21] or by building automatic online phonetic lexicon which allows a minimal but unlimited lexicon [24].

Other practical considerations include memory usage efficiency [40], low-power feature extraction [9] and high accuracy fixed-point coding [16]. These are discussed in the following subsection, fixed-point code design considerations.

2.3.6 Fixed-point code design considerations

Embedding an HMM speech recognition engine in a mobile device requires to a drastic reduction in terms of memory and computation resource consumption. To deal with this problem, solutions are given which reduce the need of memory and power

consumption while still maintaining an acceptable recognition rate.

To reduce the memory consumption Varga et. al [40] propose a memory efficient HMM modeling. The solution consists of the usage of LDA in feature extraction, discriminative training and HMM parameter coding. The main purpose of LDA is to reduce the dimensionality of the feature vector. Besides that, it also allows to model the emission probabilities by Gaussian mean vectors with diagonal covariance matrices as well as to code the mean vector of each Gaussian by less than one byte per component. Discriminative training is a method to achieve high recognition rates with moderate amount of Gaussians. It uses a performance measure like the Minimum Word Error (MWE) for training instead of the well-known Viterbi based Maximum Likelihood training. The coding of the HMM parameters is done using so called Subspace Distribution Clustering HMM (SDCHMM) where Gaussians are represented by pointers to a codebook.

Delaney et. al [9] suggest methods to reduce power consumption while increasing the speed in feature extraction by using architectural as well as algorithmic optimizations. In architectural optimization, they substitute costly functions with the existing estimates such as the use of a linear combination of the real and imaginary parts of FFT to estimate its magnitude and simple bit manipulation to estimate $\log_2(x)$. In algorithmic optimization, the trigonometric functions used in the computation of FFT are pre-computed and stored in a lookup table, thus eliminating multiple function calls in the FFT loop.

Gong et. al [16] stress the importance of maintaining a high accuracy on fixed-point DSP at different sections of the code. In MFCC calculation, the numeric variables in the algorithm are normalized and represented in a suitable Q-point. Q-n

means there are n bits to the right of the decimal point. All variables before the log step are dynamic Q-point; all variables after that are fixed Q-point. This is because log operation reduces the dynamic range tremendously; it is possible to represent the range by a fixed Q-point 16-bit format after logarithmic operation. The $\log(x)$ operation is implemented by polynomial fit for $x = 0.5$ to 1.0 . The curve is very smooth, thus polynomial fit is very accurate. Their system implement Parallel Model Combination (PMC) for noise robustness. In PMC, the 2-base exponential, $y = 2^x$, is implemented by polynomial fit for $x = 0$ to 1 . It is first normalized to between 0 and 1, and then fed into the polynomial evaluation.

Considering the current STM algorithm as a baseline for the project and following the current state of the art in the speech recognition introduced in this chapter, we propose some changes for both the front-end and back-end. These would be discussed in the next chapter.

Chapter 3

Extensions of Feature Extraction Module to Improve Robustness under Car Environment

In this chapter we combine the discussion of noise reduction methods with their implementation considerations using fixed point arithmetics to show that for every computationally costly algorithms, there should be some approximations to replace the original methods such that the system would be more robust against noise while still consuming reasonable resources. This result might be generalized to either a deterministic or a random strategy of approximation algorithms.

3.1 Introduction

In this chapter we describe fixed-point implementation of *pre-processing* noise reduction techniques in *linear Mel spectral* domain for dealing with the additive noise and in *log cepstral* domain for tackling the convolutional noise. We start by briefly introducing a whole process of feature extraction with noise reductions incorporated. Sections 3.2 describes the noise reduction methods more specifically. Implementation

issues pertaining to fixed-point coding would be deferred in the next chapter, Section 4.3.

The specification covers computation of feature vectors from speech waveform sampled at 8 KHz sampling rate. The feature vectors consist of the EPD status and 12 static MFCCs, followed by their common norm (See Figure 3.1). In order to do experiments with the derivatives of 12 static MFCCs, they have to be calculated in the CHMM back-end module instead of the FE module in order to reduce the communication bandwidth between FE outputs and BE input.

In the proposed method, noise reduction is achieved by multiplying the smoothed Noise Suppression Factors (NSFs) with the target speech power spectrum to be enhanced in the Mel-scale filter bank domain. This offers a realization of error-robust spatial noise reduction with few computational complexities. In addition, we introduce noise estimation techniques in the Mel-scale filter bank domain, and utilize the estimated noise spectral for calculation of NSFs.

Furthermore, the recognition is influenced by the frequency response of the whole transmission channel from the speaker to the audio input of the recognizer. The term *convolutional* noise has been introduced in the previous chapter for this type of distortion. Several approaches are known to compensate this effect, which usually target the cepstrum domain. This domain is chosen because the linear functions such as simple bias removal are used to remove the convolutional noise, which has additive effect in the cepstrum domain.

However, it is well known that distortion caused even by additive noise only is highly nonlinear in the log-spectral or cepstrum domain. Therefore, a nonlinear transformation or compensation is more appropriate. Despite that, nonlinear functions are

not used to improve the robustness of the system in this work because of the trade off between the increase in robustness and its computational cost.

We also need to transmit the state of Voice Activity Detection (VAD) in order to test the recognizer in real life application. To achieve this, End-Point Detection module, is introduced. The global state of VAD in EPD module is dependent on the state in each band: if all bands but one has noise only, we will consider this as a speech frame, because one of the bands detected speech.

Followings are the details of additive and convolutional noise reductions as well as end-point detection which were introduced in the FE module as shown in Figure 3.1.

3.2 Feature Extraction Procedure

Noise robustness is solved in the feature extraction (FE) module since it would eliminate the need for additional application specific data collection or parameter compensation. Figure 3.1 shows the front-end ASR with noise reduction methods incorporated. It involves time domain processes of pre-emphasis, framing and windowing, Mel-frequency domain process of spectral smoothing and log spectral domain processes of convolutional noise reduction based on RASTA filtering or Blind Equalization as well as DCT based mel cepstrum feature extraction procedure.

Followings are the details of additive and convolutional noise reductions which were introduced in the FE module as shown in Figure 3.1. The estimation of clean Mel bins is done in NE, NSF, GS and GA sub-modules (Inside dashed box in Figure 3.1), which stand for Noise Estimation, Noise Suppression Factor, Gain Smoothing, and Gain Application respectively.

Figure 3.1: Robust feature extraction algorithm of a front-end ASR system.

3.2.1 Noise Estimation

The simplest approach was first used in the work which is to estimate and update the noise spectrum only during the silent (e.g., during pauses) segments of the signal using a voice-activity detection (VAD). Although such an approach might work satisfactorily in stationary noise (e.g., white noise), it will not work well in more realistic environments (e.g., in a car) where the spectral characteristics of the noise might be changing constantly. Hence there is a need to update the noise spectrum continuously over time and this can be done using noise-estimation algorithms.

The second noise estimate is obtained by recursive averaging of the noisy Mel-domain power spectrum using time-varying and frequency-dependent smoothing parameters, which are adjusted by speech-presence probability in individual Mel bins. Speech presence is controlled by the ratio of the noisy speech power spectrum to its

local minimum, which is updated continuously by averaging past values of the noisy spectra with a look-ahead factor. The local minimum estimation algorithm adapts very quickly to highly non-stationary noise environments such as in a car.

Update of estimated noise spectrum

Let the noisy speech signal in the time domain be denoted as

$$y(n) = x(n) + d(n) \quad (3.2.1)$$

where $x(n)$ and $d(n)$ denote clean speech and uncorrelated additive noise signals, respectively, where n is a discrete-time index. The observed signal $y(n)$ is divided into overlapping frames by the application of a window function and analyzed using the short-time Fourier transform (STFT). Specifically,

$$Y(k, l) = \sum_{n=0}^{N-1} y(n + lM)h(n)e^{-j(2\pi/N)nk}, \quad (3.2.2)$$

where k is the frequency bin index, l is the time frame index, h is an analysis window of size N , and M is the frame update step in time. The squared value of the STFT magnitude corresponds to the Power Spectral Density (PSD). The PSD bins are then linearly recombined for each mel-band to get new regrouping representations. Let the new information be denoted as $Mel(i, l)$ where i is the mel frequency bin index and l is the time frame index.

Given two hypothesis, $H_0(i, l)$ and $H_1(i, l)$, which indicate, respectively, speech absence and presence in the l th frame of the i th sub-band, we have

$$H_0(i, l) : Mel(i, l) = D(i, l) \quad (3.2.3)$$

$$H_1(i, l) : Mel(i, l) = X(i, l) + D(i, l) \quad (3.2.4)$$

where $X(i, l)$ and $D(i, l)$ represent the Mel bin information of the clean and additive noise signals, respectively.

For uncorrelated additive noise,

$$\text{Var}(\text{Mel}(i, l)) = \text{Var}(X(i, l)) + \text{Var}(D(i, l)), \quad (3.2.5)$$

where $\text{Var}(\cdot)$ denotes the variance of a random variable, indicating how its possible values are spread around the expected values. Where the expected value shows the location of the distribution, the variance indicates the scale of the values.

Let $\text{Var}(D(i, l)) = \lambda_d(i, l) = E[|D(i, l)|^2]$ denote the variance of the noise in the i th mel bin. Then a common technique to obtain its estimate during speech absence period ($\text{Var}(X(i, l)) = 0$) is to apply a temporal recursive smoothing to the noisy signals. In particular,

$$\begin{aligned} H'_0(i, l) : \hat{\lambda}_d(i, l+1) &= \alpha_d \hat{\lambda}_d(i, l) + (1 - \alpha_d) |\text{Mel}(i, l)|^2 \\ H'_1(i, l) : \hat{\lambda}_d(i, l+1) &= \hat{\lambda}_d(i, l), \end{aligned} \quad (3.2.6)$$

where $\alpha_d (0 < \alpha_d < 1)$ is a smoothing constant and H'_0 and H'_1 denote hypothetical speech absence and presence, respectively.

Let $p'(i, l) \triangleq P(H'_1(i, l) | \text{Mel}(i, l))$ denote the conditional signal speech presence probability. Then Eq 3.2.6 implies

$$\hat{\lambda}_d(i, l+1) = \bar{\alpha}_d(i, l) \hat{\lambda}_d(i, l) + (1 - \bar{\alpha}_d(i, l)) |\text{Mel}(i, l)|^2$$

where

$$\bar{\alpha}_d(i, l) \triangleq \alpha_d + (1 - \alpha_d) p'(i, l) \quad (3.2.7)$$

is a time-varying smoothing parameter. Thus, the noise spectrum is estimated by averaging past power spectral mel bin values, using a smoothing parameter that is adjusted by the signal presence probability.

Signal presence probability

Let the ratio of smoothed power spectrum of noisy speech and its local minimum be defined as

$$S_r(i, l) = P_{sth}(i, l) / P_{min}(i, l) \quad (3.2.8)$$

where P_{sth} denotes in-time-smoothed power spectrum computed using the following first-order recursive equation:

$$P(i, l) = \eta P(i - 1, l) + (1 - \eta) Mel(i, l) \quad (3.2.9)$$

and P_{min} keeps track the minimum values of the noisy speech computed using Doblinger's method of continuously averaging past spectral values as follows [37]:

$$\begin{aligned} \text{If } P_{min}(i - 1, l) < P_{sth}(i, l) \quad \text{then} \\ P_{min}(i, l) &= \gamma P_{min}(i - 1, l) + \frac{1 - \gamma}{1 - \beta} (P(i, l) \\ &\quad - \beta P(i - 1, l)) \\ \text{else } P_{min}(i, l) &= P_{sth}(i, l) \end{aligned}$$

where the experiment data showed η , γ and β to be 0.7, 0.998, 0.8 respectively.

The ratio $S_r(i, l)$ is compared with a frequency dependent threshold, and if the ratio is found to be greater than the threshold, it is taken as a speech-present mel bin else it is taken as a speech-absent mel bin. This is based on the principle that the power spectrum of noisy speech will be nearly equal to its local minimum when speech is absent. Hence the smaller the ratio in Eq (3.2.8) is, the higher the probability that it will be a noise-only region and vice versa. The speech-presence decision can be

summarized as follows:

$$\begin{aligned}
 &\text{If } S_r(i, l) > \delta(l) \quad \text{then} \\
 &\quad I(i, l) = 1 \quad \text{speech presence} \\
 &\text{else} \\
 &\quad I(i, l) = 0 \quad \text{speech absence}
 \end{aligned}$$

where $\delta(k)$ is the frequency-dependent threshold determined experimentally. From the above rule, the speech-presence probability, $p'(i, l)$, is updated using the following first-order recursion and smoothing in frequency:

$$\begin{aligned}
 p'(i, l) = \alpha_p p'(i-1, l) + (1 - \alpha_p) [0.1I(i, k-1) \\
 + 0.8I(i, k) + 0.1I(i, k+1)] \quad (3.2.10)
 \end{aligned}$$

where α_p is a smoothing constant. Note that the above equation exploits the correlation for speech presence in adjacent frames as well as in neighboring mel bins. The constants used in frequency smoothing are chosen empirically.

3.2.2 Noise Suppression Factor Calculation and Gain Smoothing

Two important steps are required for noise suppression:

1. Estimate the noise power spectrum in mel bin $N[i, l]$.
2. Calculate noise suppression factor $G[i, l]$.

Table 3.1: NSF Algorithms and their formulas.

Algorithm	Formula $G(i,k)$
Spectral Subtraction	$\sqrt{\frac{Mel(i,l) - \rho N(i,l)}{Mel(i,l)}} \quad \rho = 1 \text{ for Power subtraction}$
Magnitude subtraction	$1 - \sqrt{\frac{N(i,l)}{Mel(i,l)}}$
Wiener	$\frac{Mel(i,l) - N(i,l)}{Mel(i,l)}$
MMSE Approx. of Ephraim-Malah	$\sqrt{(\frac{\xi_i}{1+\xi_i})^2 + \frac{\xi_i}{(1+\xi_i)\gamma_i}}$

Noise suppression method such as Modified Maximum likelihood [45] which is represented by noise suppression functions $G[i, l]$ could be used.

$$G(i, l) = [\frac{1}{2} + \frac{1}{2} \sqrt{\frac{Mel(i, l) - N(i, l)}{Mel(i, l)}}] P(H_1 | Mel(i, l)) + G_{min}(1 - p[H_1 | Mel(i, l)]) \quad (3.2.11)$$

Various other algorithms and their related noise suppression functions are given in Table 3.1. such as Wiener filter, Spectral Subtraction, Magnitude subtraction, Decision directed method.

Note that in all those algorithms except for MMSE approximation of Ephraim-Malah algorithm, $G[i, l]$ is a function of $\frac{Mel[i, l]}{N[i, l]}$, i.e., temporal SNR of the input signal. For MMSE approximation of Ephraim-Malah algorithm, ξ_i and γ_i are interpreted as a priori SNR and a posteriori SNR respectively [44]. Ideally, when input signal SNR is high, it is most likely that speech is present. When original input SNR is low, it most likely corresponds to noise alone and maximum suppression factor should be applied in order to get a quiet natural noise background. The calculated noise suppression factor $G[i, l]$ is then smoothed in frequency and or in time by gain smoothing GS operation.

3.2.3 Convolutional Noise Reduction

For convolutional noise, Blind Equalization (BEq) [22] can be used. It relies on a Least Means Square (LMS) algorithm, which adjusts the cepstral coefficients according to the difference between the current cepstral vector and a reference cepstrum.

An alternative to BEq is Relative Spectra (RASTA), in which a band-pass filter is used to remove mainly the DC value of the cepstral coefficients. This DC value is believed to contain the stationary convolutional distortion introduced by the channel, e.g microphone, telephone line, which appears as an additive component in the the cepstral domain. The filtering in time is done using the following transfer function [10]

$$H(z) = \frac{0.2 + 0.1z^{-1} + 0.1z^{-3} - 0.2z^{-4}}{z^{-4}(1 - 0.94z^{-1})}. \quad (3.2.12)$$

Besides removing the convolutional noise, the filter will also smooth the fast spectral changes which are not likely related to phonetic properties.

3.3 End Point Detection

Five global state of the VAD are:

- 0 - WAIT_STATE
- 1 - STARTWORD_STATE
- 2 - INWORD_STATE
- 3 - MUC_STATE
- 4 - ENDWORD_STATE.

Figure 3.2: State diagram for each band.

The global state will depend on the state in each band: if all bands but one has noise only, we will consider this as a speech frame, because one of the bands detected speech.

3.3.1 Band State

Each band can be in one of the 3 following states: **WAIT**, **STARTWORD**, **INWORD**. Please refer to Figure 3.2. To know whether we are in a speech frame or a silence frame, we will compare the speech presence probability to the band's thresholds.

The speech detector determines whether speech is present by comparing with a threshold value an average of a plurality of factors $th1(l)$ and $th2(l)$ associated with

respective frequency bins. Each $th1(l)$ and $th2(l)$ factor is the result of computing a first average of the Fourier components associated with that factor's associated frequency bin for samples that include those taken when the speech detector has indicated the presence of speech, computing a second average of Fourier components associated with that frequency bin for samples taken when the speech detector has indicated the absence of speech, and taking $th1(l)$ and $th2(l)$ as the ratio that the difference between the first and second averages bears to the first average by also giving bigger value to $th2(l)$ by a constant ratio.

3.3.2 Global State

The global state depends on the states of the different bands, and is determined according to the state machine drawn in Figure 3.3.

State of band i will be written as $s(i)$. Those states, as mentioned earlier, can be **WAIT**, **STARTWORD (ST)**, **INWORD (IN)**.

The actual system will operate in a real-environment depending on the global state. Look at Figure 3.4 below for illustration. As can be seen in the figure, when the beginning of a new word is detected, RECO.STARTED is sent, and the beginning mark stored. Later, when no vocal activity is detected, the ending mark is stored, but the module has to wait before sending RECO.ENDED, to be sure that there is not an intra-word silence. The time length of features sent to the decoder would then be the value of the maximum silence allowed into a word (MAX_LEN_HANGOVER) greater than the difference between the beginning and the ending marks.

Figure 3.3: Global state diagram for EPD module.

Figure 3.4: Illustration on the use of EPD status to use the ASR in real-life environment.

Chapter 4

Development of a Speech Recognizer

This chapter describes the design and construction of a speech recognizer. Firstly, the requirements for the system are stated and the development platform, tools as well as fixed-point coding issues are described. Next an outline of the steps needed to build a speech recognizer is given. Subsequent sections explain how these steps were performed. For each step the actions that were actually taken are explained to ensure that this description can be used as a guide for similar projects. Whenever appropriate some background theory is also explained, the emphasis here is on the differences between practice and the theory described in previous chapter. The final section gives an evaluation of the system.

4.1 Introduction

This chapter describes the design and construction of a speech recognizer capable of recognizing sounds recorded under car environment. There are many possible types of speech recognizers, ranging from task-specific word recognizers or word spotting

systems to large vocabulary recognizers for natural free speech. The system that was built here was intended to be task-specific word recognizers for mobile applications.

To meet these requirements the recognizer is designed to recognize medium vocabulary isolated word speech and is speaker independent. This way the system is not easily extendible. A different vocabulary would involve training some words and adding them to the model parameters storage. The emphasis of the system is more on the robustness of the recognizer with limited resources under severe condition of car environment. This recognizer would be implemented on a currently available fixed-point DSP chip. The training procedure to get estimated model parameters, however, can be run on a normal system with abundant resources.

Furthermore, the system should provide a baseline system for further research on the subject of speech recognition with low storage and computational complexity. Further improvement for the system to get the end user's acceptance can be done by building phoneme based recognizer to make sure that the systems vocabulary is not limited in any way to some fixed set of words. This way the system is easily extendible because a different vocabulary only involves adding some words to the pronunciation dictionary.

4.2 Development Environment

The system was developed on the Sun Microsystem platform using a variety of tools, programs and scripts. The first set of scripts that was written can be described as meta scripts, because they contain the calls to the different tools and provide these tools with their numerous attributes, scripts and with the output obtained from other

tools. This approach was taken to make reproduction of the entire process easy. This is not just useful for future attempts to build similar systems, but it also turned out to be a necessity for keeping the process manageable and for recovering from errors.

By far the largest set of tools and software libraries was taken from the STMicroelectronics ASR described in the previous chapters. To be specific, these tools consists of three main modules; Database extraction, Front-end feature extraction and Back-end CHMM training and testing modules.

In the Database extraction module, we have tools to prepare the database before the FE and CHMM BE processing. In this module, different scripts and softwares are devised in order to create database organization so that an easy files search is possible just by looking at the path and the file name. The front-end module consists of scripts and software to extract the features from the organized database and stored them into appropriate folders. The folders have a naming convention such that specific settings of the feature extraction process, such as whether noise reduction module is used, are reflected on the folders' name. Some other front-end configurations, e.g number of MFCCs per frame, number of samples per frame, are however written in a text file because these configurations would be read by other modules during run-time. This module is responsible for the speech features extraction which will be used by both the training and testing of the back-end processing.

The back-end module has three main submodules. The first module consists of scripts and software to train a speech recognizer to get the estimated model parameters. The second module is used to test the model parameters against unknown utterances. The last module is used mainly for the analysis of the recognizer's performance as well as for inserting, deleting word models or other forms of manual

modifications of the model parameters such as creating initial models.

4.3 Fixed point code implementation

In addition to obtaining an acceptable level of recognition accuracy, computationally efficient implementation is needed in order to minimize the power consumption as well as unnecessary delays. Fixed-point arithmetic uses scaled integers to perform basic math functions and strives for some approximations to perform complex ones. With these two approaches, efficient computations are achieved in the price of the acceptable accuracy.

4.3.1 Dynamic vs Fixed Scaling Factor

The scaling factor or location of the decimal point can be dynamic or fixed at design time and designated by Qn , where n is the number of bits to the right of the decimal. The basic rules of arithmetic still hold; adding two numbers requires lining up of the decimal points, multiplying two numbers with $b1$ and $b2$ decimal points yields a number with $b1 + b2$ decimal points.

The procedure taken strives for a high accuracy on fixed-point DSP at different sections of the code. This is achieved by having numeric variables normalized and represented in a suitable Q-point. With exception for SNR values in spectral averaging, all variables before the log step are dynamic Q-point; all variables after that are fixed Q-point. This is because log operation reduces the dynamic range tremendously; it is possible to represent the range by a fixed Q-point 16-bit format after logarithmic operation.

All types of SNR calculations and data representations in the noise reduction are represented in $Q8$ to anticipate the dynamic range of the SNR values. Dynamic Q -point is not used to reduce the memory consumption for storing the shift level and the computation needed for normalization.

4.3.2 Complex Math Functions

The feature extraction procedure contains logarithmic operations and many division operations. For logarithmic operations, approximation using Taylor's Equation was used.

$$\ln(1 - x) = -x - \frac{x^2}{2} - \frac{x^3}{3} \dots - \frac{x^n}{n} \quad (4.3.1)$$

where $-1 \leq x < 1$. Since the input to this operation, A , is an integer, the number is first scaled to an appropriate range,

$$A = B(1 - x)$$

where $B = 2^N$.

It leads to

$$\ln(A) = \ln(B) + \ln(1 - x) = N \ln 2 + \ln(1 - x).$$

Different values for N are chosen in different sections of the code depending on the dynamic range of the input variables.

Division is done in binary with right shifting and subtraction. It starts from the left of the number being divided as in decimal division. The answer is made up of 0s and 1s. A zero indicates that the division was not possible. A one indicates that the division is possible. The calculation of the next amount of the number to be divided occurs with the shift and subtraction step.

RASTA filtering and DCT coefficient calculations are relatively easy to do in fixed-point arithmetic, but taking the natural logarithm is a more difficult task. The algorithm to estimate $\log_2(x)$ using simple bit manipulation described in [6] was found faster than other methods of calculating logarithm. The $\ln(x)$ can then be determined by multiplying the result with a constant $\ln(2)$.

4.4 Outline of Development Process

As described in Section 2.2 of Chapter 2, a Hidden Markov Model based isolated word speech recognizer consists of two parts.

1. A preprocessing part, which extracts relevant features from the speech signal.
2. An acoustic model, which computes how a word string is likely to be pronounced.

This subdivision gives some clues on the decisions that have to be made and the steps that have to be performed to build a recognizer. In the case of the preprocessor decisions have to be made concerning the data format of the audio streams and on the type of preprocessor that is to be used. With regard to the acoustic model decisions have to be made concerning the unit of speech on model represents, the number of states in such a model, the way these states are connected and concerning the type of distribution functions. These acoustic models should then be trained and refined using a set of examples. So a training data set have to be prepared.

As shown in Figure 4.1 the overall development process can be divided into four stages. In the data preparation stage training data are collected and prepared and

Figure 4.1: The development process of a recognizer.

the design decisions concerning the models are taken. In the initial training stage, a simple, baseline speech recognition system is built. In the refinement stage this system is then incrementally refined to make it more advanced and robust. In the final stage the performance of the system is evaluated. The next four sections describe how these four stages were performed to build the speech recognizer.

4.5 Data Preparation

The data preparation phase involves a number of inter-related tasks. Obviously speech data is needed for training, testing and evaluating the models in the process, therefore the first step involved selection of an appropriate speech database. The selected utterances would be categorized according to certain grouping criteria, e.g based on gender and region of speaking. For each of these groups a textual transcription was created. Further a HMM topology was defined. The exact steps and their relations are now described in details.

Data selection

In order to build a robust medium vocabulary speech recognizer, it is crucial that all acoustic (word) models receive sufficient training examples, taking into account the many variations that can occur in speech, due to, for example, variations in speed of speech, different gender and different accents. It follows that a training corpus should be sufficiently large, consisting of speech samples spoken by men and women from different dialect regions.

Recording such a data set is a major undertaking involving sub-tasks like selection of participants, recording the data and the most time-consuming parts post-processing and transcribing the data. Fortunately, these speech corpora are now commercially available. One of the corpora available is the SpeechDat-Car database, which is used to select the training data for the recognizer as it contains speech samples that fulfill the requirements stated above. The speakers in the database were selected from all geographic regions. The ratio between male and female speakers is close to fifty-fifty.

Since the SpeechDat-Car database is recorded under car environment many samples are of poor quality, or contain background noise or even background speech. To ensure well-trained models that can be used for recognizing speech by an average person, only sessions that fit the following profile were used:

- Men and women from all regions.
- Utterances for training may contain background noise or background speech but they should still have high SNR. Mouth noises, like smacking, sniffing or loud breaths and verbal hesitations like uh are also allowed.

Manually selecting utterances that adhered to this profile is practically impossible.

The data in the SpeechDat-Car database is stored on few DVDs each with about 500 directories, called sessions, with one speaker per session. Information concerning the speakers and the utterances is listed in three large plain text format tables. No data selection tools whatsoever are included in the database.

To overcome this problem a script program was written, called `cepstrumDB.pl`, which automatically selects a subset of the database according to some selection criteria it is provided with. This program provides a command line interface. At this command line a number of options can be specified:

- A subset of the database that should be searched.
- A caller profile: gender, dialect region.
- A number of output options.

The script program creates a number of text files based on the selected options. These text files are needed by the ASR system tools in later steps. Furthermore the script creates a list of all the words in the selected set. It also creates a file containing all the transcriptions of the selected utterances indexed by the name of the utterances. The program also creates unique filenames for the selected utterances.

Two different data sets were extracted from the SpeechDat-Car database this way; A training set and an evaluation test set to evaluate the final performance of the system. Depending on the scenario being tested, the evaluation test set would contain, for example, different groups of people from other regions or different background environments.

Feature vectors

The first step in speech recognition is the extraction of feature vectors. Theoretically this can be done on the fly during training, as is the case during real-time recognition. But for training purposes it makes sense to do this once and for all and save the feature vectors in a file, as each utterance is processed a number of times during training.

Preprocessing of the data files was done in one stage. FE module is a tool for translating audio files to feature vector files using Mel scale cepstral coefficients. It takes as its parameters a list summing all the files it has to convert and the new directories they should be copied to and a script containing information on output formats. The filename list was created also by *cepstrumDB.pl*.

The choice of using MFCCs was based on literature and earlier experiments. We do not use other features on the project. Configurations such as each vector containing 12 cepstral coefficients with log energy, all scaled around zero by filtering all vectors using RASTA filter which resulted in 13 dimensional feature vectors, are written in the script arguments execution. For more or less constant configurations such as a sampling rate of 15 ms overlapping window of 8 ms are written in a text file, which we can modify manually.

HMM prototype

The last step in data preparation was the selection of a Hidden Markov Model topology for the acoustic models. Since a word based recognizer was built, a model represents a word. A topology consisting start and end states states and two emitting states per phoneme, using single Gaussian density functions, was chosen as a baseline. The states were connected in a left-to-right way, with maximum one skip transition.

The model as shown in Figure 2.7 is used.

This is a rather simple model topology, but earlier experiments with three or more state models showed that the two state per phoneme model performs as well, and in some cases even slightly better than the three state model, despite the fact that it has fewer parameters. These experiments showed that it works best to start with a simple model with only a few parameters and then increasing the complexity of the model as training progresses. So this is the approach that was also taken during this project.

4.6 Training strategies

The precise way in which the training tools are used depends on the type of HMM system to be built and the form of the available training data. In practice, however, HMM-based speech recognizers are either whole-word or sub-word. In our system, we use the whole-word HMM system. As the name suggests, whole word modeling refers to a technique whereby each individual word in the system vocabulary is modeled by a single HMM.

Whole word HMMs are most commonly trained on examples of each word spoken in isolation. This is also the technique used in the project. These training examples, which are often called tokens, have had leading and trailing silence removed so that then they can be input directly into the training tools without the need for any label information. To remove the leading and trailing silence, we use a Matlab script which shows each of the selected speech signals and is able to play the audio file. The segmentation of the training data is then done audio-visually using this Matlab script.

The most common method of building whole word HMMs is to firstly calculate initial parameters for the model and then refine the parameters using Baum-Welch re-estimation. Where there is limited training data and recognition in adverse noise environments is needed, so-called fixed variance models can offer improved robustness. These are models in which all the variances are set equal to the global speech variance and never subsequently re-estimated.

An HMM model can be initialized in two ways:

1. Directly reading a model from a file (ReadBootModel function in CHMM module)
2. Computing an initial model using the training data (ComputeInitialModel function in CHMM module).

In the project, we use the first method in a situation where the complete training database is not available in one time. That is to say we use different set of training database for successive training whereby in each training, the model parameters computed from previous training is used as initial model. We can also use this method in order to refine the model parameters, e.g increasing the number of mixtures by splitting the states in the previous model parameters from previous training.

The second method of initialization computes the initial model using Viterbi algorithm for splitting the frames into appropriate states and K-Means clustering method for further splitting the frames into the right mixture. In order to create a HMM definition, it is first necessary to produce a prototype definition. The function of a prototype definition is to describe the form and topology of the HMM, the actual numbers used in the definition are not important. Hence, the vector size should be specified and the number of states chosen. The allowable transitions between states

should be indicated by putting non-zero values in the corresponding elements of the transition matrix and zeros elsewhere. The rows of the transition matrix must sum to one except for the final row which should be all zero. Each state definition should show the required number of mixture components. All mean values can be zero but diagonal variances should be positive and covariance matrices should have positive diagonal elements. All state definitions can be identical.

Having set up an appropriate prototype, a HMM can be then be initialized. The basic principle of initialization method depends on the concept of a HMM as a generator of speech vectors. Every training example can be viewed as the output of the HMM whose parameters are to be estimated. Thus, if the state that generated each vector in the training data was known, then the unknown means and variances could be estimated by averaging all the vectors associated with each state. Similarly, the transition matrix could be estimated by simply counting the number of time slots that each state was occupied.

The above idea is implemented by an iterative scheme as explained in Section **CHMM training for isolated word recognition** in Chapter 2. This process requires some initial HMM parameters to get started. To circumvent this problem, the procedure starts by uniformly segmenting the data and associating each successive segment with successive states. Of course, this only makes sense if the HMM is left-right. If the HMM is ergodic, then the uniform segmentation can be disabled and some other approach taken.

Figure 4.2: System refinement procedure.

4.7 Refinement

The basic system has a number of shortcomings. In the first place, all of the models have the same shape. Further, the system does not take into account linguistic effects like coarticulation. By using word units, the coarticulation has been roughly modeled. However, it might not be sufficient. Finally, the system does not make up for possible unbalances in the training data, some models may receive much training, while others may receive only little training because they only have a small number of examples. There are a number of ways to refine a speech recognizer system: mixture component splitting, the use of 1st and 2nd derivatives of MFCCs, generalized parameter tying and data driven or tree based clustering. To improve the system a combination of several of these techniques was applied.

Figure 4.2 shows the sub-phases that were performed in this phase. Time-derivatives of MFCCs models were created to capture the spectrum relationship between the

adjacent frames. Multiple mixture models were created to make up for model inaccuracies. These model sets were merged and the resulting model was fine-tuned to find the optimal combination of acoustic models.

4.7.1 Time-derivatives MFCCs

The performance of a speech recognition system can be greatly enhanced by adding time derivatives to the basic static parameters. To better represent temporal variations in the speech signal, higher-order time derivatives (or simply, delta parameters for first derivatives, delta-delta for second derivatives) are used. However, the performance improvement by using these derivatives should be justified because the memory consumption and computational cost would also increase.

In the next chapter we discuss the various techniques to effectively calculate the derivatives. Discussion is also given to better judge the use of these components in the system.

4.7.2 Multiple Mixtures

The baseline system uses Gaussian distributions to model the *observation functions*, but as a matter of fact these do not adequately reflect the speech process. They are not capable of modeling the variations in speech that are due to, for example, the difference in pitch between male and female voices. To overcome this problem more realistic density functions have to be found.

Alternatively this can be achieved by using a large number of states based on simple densities like the Gaussian density. The latter option implies that an appropriate topology for the word models has to be found, which is not an easy task. The

first option can be achieved by a range of techniques from multivariate statistics, but an interesting solution lies in the use of Gaussian mixture densities, which can approximate any continuous probability density function in the sense of minimizing the error between two density functions. The Gaussian mixture density function is composed by taking the superposition of a number of Gaussian densities each with its own mean and variation and its own mixture weight.

Section 5.5.3 discusses various techniques to effectively split the Gaussian distributions.

4.8 Evaluation

Once the test data have been processed by the recogniser, the next step is to analyze the results. The *Score Extraction Tool* submodule is provided for this purpose. The tool compares the transcriptions output by the recognizer with the original reference transcriptions and then outputs various statistics such as recognized words, active vocabulary, etc.

In attempting to evaluate the performance, we distinguish between two levels of specificity:

- **Criterion:** We are interested in evaluating:
 1. Vocal Activity Detection (VAD): The detection of speech activity is based on the Noise Suppression Factors (NSF) on each Mel band from noise reduction module. Whether or not the noise reduction module is used to purify the signals, these NSFs are still calculated in order to facilitate the VAD.

2. Recognition Accuracy with/out VAD: Recognition of unknown utterances against available word models.

- **Measure:** We are to record specific property of system performance, in an attempt to get at the chosen criterion:

1. *Complete Detection*: Number of detected word divided by number of word
2. *Recognition Rate without VAD*: Number of recognized word divided by number of word
3. *Recognition Rate with VAD*: Number of recognized word divided by number of detected word.
4. *Percentage Relative Improvement of Recognition Rate*: Current recognition rate subtracted by the recognition rate of the reference system.

Chapter 5

Experiments

Previous chapters have explained various techniques to improve the baseline speech recognition system. To test the effectiveness of those additional processes in the speech recognizer, experiments were done by integrating those techniques in the system and analyzing the results whether those techniques achieved the objectives. Analysis was further done by finding ways to improve the techniques to better suit the application given the said environment.

5.1 Introduction

As described in Chapter 3, the use of noise reduction in speech recognition may result in more robust systems. The problems that have to be solved in the integration of noise reduction module include the additional computational cost and levels of reliability. A number of noise reduction techniques for speech was described in Chapter 3. To test the recognition using these noise reductions in real car environment, a set of experiments were conducted with SpeechDat-Car database. The results of these

experiments are described in this chapter. The goal of these experiments was to integrate the most suitable noise reduction technique to be used under car environment as well as the best configurations in the front-end feature extraction and back-end training and testing. In the next section a description of the data set is given, followed by the description of the baseline recognition system that was used to compare with the new system. Subsequently, the designs of experiments, results and analysis will be described. And finally, conclusions will be drawn.

5.2 Data Selection

The SpeechDat-Car database [7, 4] was built in the framework of the SpeechDat- Car project (European Union founded). It contains the speech data in ten languages (nine European languages plus American English) recorded in the car environment. For each language, 600 sessions from at least 300 speakers were recorded. The vocabulary size is however quite small as there are only 30 to 40 words and most words are digits and commands. The availability of the database allows the development of the speech recognition applications in mobile telephone communications and voice-operated services.

The database were recorded under one of the following conditions [30]: car stopped with motor running (*stop*), driving in the town traffic (*town*), driving on low speed on rough road (*low*), driving on high speed on good road (*high*). The recordings are made with or without additional environmental noise due to air-conditioning, open windows, etc. (*noisy*) and with or without the car radio on (*radio*).

A session consists of 125 items, including isolated words, spelled words, connected digit sequences, phonetically rich utterances, continuous speech, etc. For all the four

different input channels (close talk and three far electret condenser microphones) the recordings were realized with 16 KHz sampling frequency and 16 bit accuracy. At the same time and additional AKGfar-microphone, connected to a mobile telephone equipment, allowed remote recording (at 8 KHz/8bit A-law compression) of speech signals transmitted through the GSM telephone network.

Thus, two types of recordings compose the database. The first type consist of wideband audio signals recorded directly in the car and the second type is composed by a GSM signal transmitted from the car and recorded simultaneously in a far-end. In this project, we only used the first one. This mobile recording platform records the signals from four high quality audio channels. For this purpose, four microphones were used, a close-talk microphone, and 3 far-talk microphones placed at different locations in the car. The positions for the far-talk microphones or hands-free microphones are:

- A: at the ceiling of the car near the A-pillar
- B: at the ceiling of the car in front of the speaker behind the sunvisor
- C: at the ceiling of the car over the mid-console (near the rear mirror)

The mobile platform stores the recorded signals as sequences of 16bit, 16 kHz uncompressed and multiplexed. Channels are sequentially multiplexed in short unsigned.

In this project, training and testing data are provided for both Italian (IT) and Deutsch (DE) languages. All sessions have been recorded with native speakers using both a close talk (CT) microphone and hands-free (HF) microphones in cars under different driving conditions (See Table 5.1).

Table 5.1: Database background environment.

Index	Driving Condition	Condition
1	0 km/h, engine on	Quiet
2	40-60 km/h	Low noise
3	40-60 km/h, window open	Low noise
4	100-120 km/h	High noise
5	100-120 km/h, music on	High noise

The selected utterances from the database are organized into three different matching conditions:

- **Weakly mismatch condition** (WM) where training uses 75% of data from both HF and CT microphones in all driving conditions and testing uses the rest 25% data from HF and CT microphones in all driving conditions.
- **Medium-mismatch condition** (MM) where training uses 75% data in all driving conditions from CT microphone and testing uses 25% data in all driving conditions from HF and CT microphones.
- **High-mismatch condition** (HM) where training takes 75% data in low noise from CT microphone and testing uses 25% data in low noise and high noise data from HF and CT microphones.

Table 5.2 shows the statistic of 75% utterances of selected words found in the database using Medium-mismatch condition criterion. These utterances are used for training to get the word model parameters.

Table 5.2: Statistic of selected utterances.

Language	Training set	Testing set (per channel)
IT	3,176 utterances from 23 words	993 utterances from 23 words
DE	1,675 utterances from 13 words	555 utterances from 13 words

5.3 Baseline System Experiment

5.3.1 Design

As starting point in the experiments, the left-to-right HMM model with features extraction without any noise reduction was used. This particular system was chosen because it does not contain any complicated structures like multiple mixtures, making the word model as small as possible. Further the modest results achieved by this model leave room for gains in performance. On the other hand its simple structure also limits the performance that can be reached by this model, therefore great leaps in performance are not to be expected, it is the relative performance that is of interest here. Below are the setup configurations for both the front-end and back-end processing.

FE feature extraction Process: No noise reduction; Features: 12 static MFCCs + Energy.

BE training Process: K -means segmentation for model initialization; HMM model: Left-to-right model with the number of states corresponding roughly to the number of sounds (phonemes) within the word (each word model has the same number of states), continuous observation densities with one mixture per state (single Gaussian), diagonal covariance matrices.

BE testing Viterbi algorithm to compute likelihood evaluation.

Figure 5.1: Front-end feature extraction process.

Figure 5.2: Back-end training process.

Figure 5.1, 5.2, 5.3 illustrate the baseline system configurations of the front-end, back-end training and testing (recognition) respectively.

5.3.2 Results

To provide a baseline system to which the proposed systems could be compared, the left-to-right single mixture model system was trained using IT WM, MM and HM training sets. Tables 5.3, 5.4, 5.5 show the recognizer's performance on the respective test sets.

Table 5.4 shows that the result in CT without VAD is the same or better than

Figure 5.3: Back-end testing process.

with VAD. This is expected because without VAD means that human intervention is needed to specify the starting and ending of each word utterance by manually see the speech signal display and listen to the word uttered.

While with VAD means we use the VAD algorithm to send RECO_STARTED flag to the front-end module and the beginning mark stored. Later, when no vocal activity is detected, the ending mark is stored, but the module has to wait before sending RECO_ENDED, to be sure that there is not an intra-word silence. This procedure was explained in Section 3.3.2.

There could be a difference in the starting and ending marks between with and without VAD due to the performance of the VAD algorithm. The VAD algorithm could be late in sending out the RECO_STARTED flag because the starting phoneme of the word has energy which is about the same as signal without speech activity: thus, no speech activity is detected.

For the sending of RECO_ENDED, we have specified maximum hangover just to

Table 5.3: Weakly mismatching IT data set; Test results (Exp. 5.3).

Channel	Complete Detection (%)	Recognition Rate without VAD (%)	Recognition Rate with VAD (%)
CT	95.0	96.5	96.5
HF-A	92.2	75.2	73.8
HF-B	93.2	72.4	68.6
HF-C	92.0	78.6	75.5

be sure that low energy phoneme in the end of a word is not cut. But there is nothing we can do on if the phoneme is on the starting word.

Table 5.4 also shows that result in HF-A is worse than result in CT. This is also expected because the database used for this experiment are Medium-mismatch condition. It means that the training uses 75% data in all driving condition from CT microphone which is close-talk microphone and testing uses 25% data in all driving condition from HF and CT microphones. HF microphones were located quite a distance from the speaker. Thus, they will pick up higher noise level as compared to CT microphone.

Therefore, there are two reasons why the result in HF-A is lower than in CT. Firstly, the speech signals in CT are used in training as well as in testing. We didn't use signals which come from HF microphone for training. Secondly, signals from HF microphones contain relatively higher level of noise (lower SNRs) as compared to signals from CT microphone.

Table 5.4: Medium mismatching IT data set; Test results(Exp. 5.3).

Channel	Complete Detection (%)	Recognition Rate without VAD (%)	Recognition Rate with VAD (%)
CT	95.0	98.7	98.7
HF-A	92.2	64.2	63.8
HF-B	93.2	62.6	61.9
HF-C	92.0	68.2	67.5

Table 5.5: Highly mismatching IT data set; Test results(Exp. 5.3).

Channel	Complete Detection (%)	Recognition Rate without VAD (%)	Recognition Rate with VAD (%)
CT	94.5	98.5	98.5
HF-A	90.2	54.2	53.4
HF-B	92.5	53.4	51.9
HF-C	91.0	58.4	57.2

5.4 Isolated Word Recognition with HTK

5.4.1 Design

The HTK toolkit was used to implement a word-based HMM recognizer. The purpose of using HTK toolkit with SpeechDat-Car database is to test whether our implementation is correct. If the implementation using STM system results in comparable performance on the HTK engine, then it will show that the implementation is correct.

The HTK toolkit was used to implement the isolated word recognizer with whole word HMM models. In the experiment, there were 23 HMM models trained to recognize the 23 Italian words. Each model had the number of states corresponding roughly to the number of sounds (phonemes) within the word. Each model had a

single Gaussian mixture with diagonal covariance matrices. In the training phase of each word, every training utterance was segmented into equal lengths and then initial model parameters were estimated. This is done using HINIT tool. HINIT can be used to provide initial estimates of whole word models in which case the observation sequences are realizations of the corresponding vocabulary word. The process is started by performing a uniform segmentation of each training observation and for mixture Gaussians, the vectors in each uniform segment are clustered using a modified K-Means algorithm.

In combination with the tool HINIT for estimating initial values mentioned earlier, HREST allows isolated word HMMs to be constructed from a set of training examples using Baum-Welch re-estimation. Model parameters were re-estimated repeatedly until the estimates were unchanged or the maximum number of iterations were reached. One final point that should be mentioned is that the computation of the forward and backward probabilities in Baum-Welch re-estimation involves taking the product of a large number of probabilities. In practice, this means that the actual numbers involved become very small. Hence, to avoid numerical problems, the forward-backward computation is computed in HTK using logarithmic arithmetic.

The Viterbi algorithm was applied in the testing phase to determine the most likely model that best matched each test utterance. There is no HTK tool which implements Viterbi algorithm directly. Instead, a tool called HVITE is provided which along with its supporting libraries, HNET and HREC, is designed to handle continuous speech. Since this recognizer is syntax directed, it can also perform isolated word recognition as a special case.

Below are the setup configurations for both the front-end and back-end processing.

FE feature extraction Process: No noise reduction; Features: 12 static MFCCs + Energy.

BE training Process: Uniform segmentation of each training observation for model initialization, model parameters re-estimation using Baum-Welch algorithm; HMM model: Left-to-right model with the number of states corresponding roughly to the number of sounds (phonemes) within the word (each word model has the same number of states), continuous observation densities with one mixture per state (single Gaussian), diagonal covariance matrices.

BE testing Viterbi algorithm to compute likelihood evaluation.

5.4.2 Results

To provide a baseline system verification to which the STMicroelectronics system could be compared, the left-to-right single mixture model system was trained using IT WM, MM and HM training sets. Tables 5.6, 5.7, 5.8 show the recognizer's performance on the respective test sets.

Comparing Tables 5.3, 5.4, 5.5 with Tables 5.6, 5.7, 5.8 respectively show that the STMicroelectronics baseline system results in comparable performance with that of HTK engine. The small difference might be attributed to the fact that STMicroelectronics system uses fixed point algorithm in the front end feature extraction as compared floating point feature extraction usage by HTK engine.

Table 5.6: Weakly mismatching IT data set; Test results(Exp. 5.4).

Channel	Complete Detection (%)	Recognition Rate without VAD (%)	Recognition Rate with VAD (%)
CT	95.1	96.7	96.6
HF-A	92.3	75.4	73.9
HF-B	93.4	72.5	68.7
HF-C	92.1	78.7	75.7

Table 5.7: Medium mismatching IT data set; Test results(Exp. 5.4).

Channel	Complete Detection (%)	Recognition Rate without VAD (%)	Recognition Rate with VAD (%)
CT	95.0	98.7	98.7
HF-A	92.3	64.3	63.8
HF-B	93.3	62.7	62.0
HF-C	92.1	68.3	67.6

Table 5.8: Highly mismatching IT data set; Test results(Exp. 5.4).

Channel	Complete Detection (%)	Recognition Rate without VAD (%)	Recognition Rate with VAD (%)
CT	94.6	98.5	98.5
HF-A	90.3	54.3	53.6
HF-B	92.6	53.6	52.1
HF-C	91.2	58.8	57.5

5.5 Incorporation of Various Techniques

In this section, we would discuss various techniques that might improve the performance of the baseline system. The performance of these techniques were compared with that of the baseline system. If improved performance was reported, the technique would be suggested. On the other hand, if poorer performance was reported, the technique would be further investigated.

5.5.1 Experiments on Digital Differentiators for MFCCs' derivatives calculation

Design

For the baseline system, the features extracted from speech signals used as inputs to CHMM training are only static MFCCs. In order to improve the performance of the system, the 1st (delta) and/or 2nd (acceleration) order(s) MFCCs are to be used as well.

The baseline system uses a simple 2-term Digital Differentiator (DD) to get the delta and acceleration MFCCs as shown by the following difference equation.

$$y[n] = x[n] - x[n - 1] \quad (5.5.1)$$

However, it turned out that this DD did not sufficiently predict the 1st and 2nd order derivatives of the MFCCs with SpeechDat-Car database. As a result, the model parameters which were trained using the static MFCCs and their derivatives did not converge within maximum iterations (specified to be 100) of re-estimation procedure during the training phase. This resulted in poorer recognition rate as shown in Table

Table 5.9: Medium matching IT data set; Test results with incorporation of the delta MFCCs with DD of Eq. 5.5.1(Exp. 5.5.1).

Channel	Complete Detection (%)	Recognition Rate without VAD (%)	Recognition Rate with VAD (%)
CT	95.0	95.2	95.2
HF-A	92.2	54.4	52.8
HF-B	93.2	52.4	50.9
HF-C	92.0	57.6	55.8

5.9. Compare with Table 5.4.

For this reason, four versions of digital differentiators which originate from several filter design techniques were tried. Digital Differentiators mentioned below use maximum five terms only in order to conserve the memory consumptions.

1. 5-term non-causal, non-recursive LTI filter which is used in Hidden Markov Model Toolkit (HTK) has the following difference equation:

$$y[n] = 0.2 * x[n + 2] + 0.1 * x[n + 1] - 0.1 * x[n - 1] - 0.2 * x[n - 2] \quad (5.5.2)$$

2. 5-term non-causal, non-recursive LTI filter which is designed using Linear-phase FIR filter design using least-squares error minimization from MATLAB toolbox has the following difference equation:

$$y[n] = -0.0833 * x[n + 2] + 0.666 * x[n + 1] - 0.666 * x[n - 1] + 0.0833 * x[n - 2] \quad (5.5.3)$$

3. 5-term non-causal, non-recursive LTI filter which is designed using Parks-McClellan optimal equiripple FIR filter design from MATLAB toolbox has the following

difference equation:

$$y[n] = -0.6547 * x[n+2] + 1.3900 * x[n+1] - 1.3900 * x[n-1] + 0.6547 * x[n-2] \quad (5.5.4)$$

4. 5-term non-causal, non-recursive LTI filter which is approached using a tapered window because truncation of the impulse response (the coefficients in the LTI filters) produces ripples particularly near sharp discontinuity. Using the Hamming window, the LTI filter has the following difference equation:

$$y[n] = -0.5 * x[n+2] + x[n+1] - x[n-1] + 0.5 * x[n-2] \quad (5.5.5)$$

The same formula is applied to the delta coefficients to obtain acceleration coefficients. Below are the setup configurations for both the front-end and back-end processing.

FE feature extraction Process: No noise reduction; Features: 12 static MFCCs + static Energy + 12 delta MFCCs + delta Energy + 12 acceleration MFCCs + Acceleration Energy.

BE training Process: K -means segmentation for model initialization; HMM model: Left-to-right model with the number of states corresponding roughly to the number of sounds (phonemes) within the word (each word model has the same number of states), continuous observation densities with one mixture per state (single Gaussian), diagonal covariance matrices.

BE testing Viterbi algorithm to compute likelihood evaluation.

Table 5.10: Percentage Relative Improvement of Recognition Rate (PRIRR) without VAD over the baseline system using various DD methods(Exp. 5.5.1).

Channel	DD Eq. 5.5.2	DD Eq. 5.5.3	DD Eq. 5.5.4	DD Eq. 5.5.5
CT	0	0	0	0
HF-A	2.3	2.1	1.8	2.2
HF-B	2.4	2.3	2.0	2.2
HF-C	2.3	2.2	1.8	2.3

Results

The new four DDs resulted in converged model parameters within eight to ten iterations. As compared to the re-estimation with static MFCCs, there is an increase of three to five iterations. This is expected and still reasonable as we are now training using more features.

To provide easy comparison with the baseline system, Table 5.10 shows the Percentage Relative Improvement of Recognition Rate (PRIRR) without VAD over the baseline system (Table 5.4) using IT MM data set. As shown in the table, there is only slight improvement in the recognition rate by introducing the 1st and 2nd derivatives of MFCCs. Furthermore, we have to use more memory space and computational power to calculate the derivatives. Also, the model parameters storage would increase by minimum threefold per word model.

Figure 5.4: Procedure to get and use reference cepstrum for Blind Equalization.

5.5.2 Experiments on Convolutional Noise Reduction Techniques

Design

Two techniques of convolutional noise reduction were tried: Blind Equalization (BEq) and Relative Spectra (RASTA) filtering. BEq relies on a Least Means Square (LMS) algorithm, which adjusts the cepstral coefficients according to the difference between the current cepstral vector and a reference cepstrum with a fixed step size. This constant reference cepstral signal is representative, in the digital speech signal and in the set of cepstral vectors, of the long-term cepstrum of the speech signal. The reference cepstrum was obtained from the database, as described in the following paragraph. Please refer to Figure 5.4.

As shown in Figure 5.4, the system comprises at least one module 1 for transformation of the digital speech signal $sn(t)$ into a set of cepstral vectors $Cn(k)$ representative of the digital speech signal over the given horizon and, in particular, for the window of rank n of the observed digital speech signal. The aforementioned module 1 is followed by a module 2 that generates a reference cepstrum representative, for each of the cepstral vectors $Cn(k)$, of the long-term cepstrum of the speech signal. The reference cepstrum enables a reference cepstrum $Rn(k)$ representative of the long-term cepstrum of the speech signal to be associated with each cepstral vector $Cn(k)$, for each of these vectors. It will therefore be understood that the value of the reference cepstrum can advantageously be approximated by a time constant. However, the distribution $Rn(k)$ of this value for each component $Cn(k)$ of rank k of the cepstral vectors can differ as a function of the rank k under consideration. However, in a particular, non-limiting embodiment, the reference cepstrum $Rn(k)$ had a constant value that was identical for each component, or corresponding cepstral coefficient $Cn(k)$ of rank k , the component constituting each cepstral vector $Cn(k)$.

With regard to the module 2 that generates the reference cepstrum, it is stated that, for a set of cepstral vectors $Cn(k)$ wherein $k \in [1, M]$, the reference cepstrum $Rn(k)$ is representative, for each of the cepstral vectors, of the long-term cepstrum of the speech signal. It will therefore be appreciated that the database constituting the reference cepstrum generator 2 can be organised so as to transmit the reference cepstrum representative of the long-term cepstrum of the speech signal as a function of the index k designating the component of the cepstral vector $Cn(k)$. Figure 5.5 shows the pseudo-code to compensate the MFCC values using BEq method.

```
blind_equalization(ARG)
{
    stepSize = 0.01;
    for (i = 0; i < 12; i++) {
        mfcc[i] = subtract(mfcc[i], bias[i]);
        temp[i] = subtract(mfcc[i], refCep[i]);
        temp[i] = multiply(stepSize, temp[i]);
        bias[i] = add(bias[i], temp[i]);
    }
    refCep[12] = {const1, const2, ..., const12}
```

Figure 5.5: Blind equalization pseudo-code for compensating 12 MFCCs.

The RASTA filtering is done using the following transfer function [10]

$$H(z) = \frac{0.2 + 0.1z^{-1} + 0.1z^{-3} - 0.2z^{-4}}{z^{-4}(1 - 0.94z^{-1})}. \quad (5.5.6)$$

In time domain, it is translated into the following equation

$$\begin{aligned} Con(k, l) = & 0.2Log(k, l) + 0.1Log(k, l - 1) + 0.1Log(k, l - 3) \\ & - 0.2Log(k, l - 4) + 0.98Con(k, l - 1). \end{aligned} \quad (5.5.7)$$

Below are the setup configurations for both the front-end and back-end processing.

FE feature extraction Process: BEq or RASTA filtering; Features: 12 static MFCCs + static Energy.

BE training Process: K -means segmentation for model initialization; HMM model: Left-to-right model with the number of states corresponding roughly to the number of sounds (phonemes) within the word (each word model has the same number of states), continuous observation densities with one mixture per state (single Gaussian), diagonal covariance matrices.

BE testing Viterbi algorithm to compute likelihood evaluation.

Results

The main contribution of convolutional noise is the channel transfer function, i.e the microphone characteristics. In order to test the effectiveness of the techniques in removing the convolutional noise, MM data set were used because there is a difference in the type and placement of microphone used between the training and testing procedure.

The RASTA filtering is done using the following transfer function [10]

$$H(z) = \frac{0.2 + 0.1z^{-1} + 0.1z^{-3} - 0.2z^{-4}}{z^{-4}(1 - 0.94z^{-1})}. \quad (5.5.6)$$

In time domain, it is translated into the following equation

$$\begin{aligned} Con(k, l) = & 0.2Log(k, l) + 0.1Log(k, l - 1) + 0.1Log(k, l - 3) \\ & - 0.2Log(k, l - 4) + 0.98Con(k, l - 1). \end{aligned} \quad (5.5.7)$$

Below are the setup configurations for both the front-end and back-end processing.

FE feature extraction Process: BEq or RASTA filtering; Features: 12 static MFCCs + static Energy.

BE training Process: K -means segmentation for model initialization; HMM model: Left-to-right model with the number of states corresponding roughly to the number of sounds (phonemes) within the word (each word model has the same number of states), continuous observation densities with one mixture per state (single Gaussian), diagonal covariance matrices.

BE testing Viterbi algorithm to compute likelihood evaluation.

Results

The main contribution of convolutional noise is the channel transfer function, i.e the microphone characteristics. In order to test the effectiveness of the techniques in removing the convolutional noise, MM data set were used because there is a difference in the type and placement of microphone used between the training and testing procedure.

Table 5.11: PRIRR without VAD over the baseline system using BEq and RASTA methods(Exp. 5.5.2).

Channel	BEq	RASTA
CT	0	0
HF-A	9.9	10.0
HF-B	10	9.8
HF-C	10.0	10.2

The relative performance improvement over the baseline system is shown in Table 5.11. As shown in the table and compared with Table 5.3, the performance of the recognizer in both situations for HF microphones are about the same. This fact shows that with less training data set which only comes from CT microphone, we can get similar performance with a recognizer which is trained using different types of microphone (CT and HFs) if we were to incorporate some sort of channel normalization to the MFCC values. Moreover, the performance for CT microphone data set is not decreased as shown in row Channel CT from both tables.

Table 5.11 also shows that RASTA filtering gives superior performance as compared to Blind Equalization method. This might be because the time-varying environment is correctly predicted by the RASTA filter which uses assumption that modulation of components of speech basically (more than 95 %), are concentrated in a range from 1 to 16 Hz. It is caused by quantity of syllables uttered by one person per one second. Thus, the noise varied with frequency out of range, can be removed with the filtration of modulation spectrum.

BEq method, on the other hand, uses constant reference cepstrums which are adapted using small portion of the database and used during training and testing. This small portion of the database is selected from various driving condition and

different types of microphones with the hope that it would be able to normalize cepstrums from time-varying car environment.

However, while the use of RASTA filtering instead of BEq increased the recognition rate by at least 9.8%, BEq might still be preferred since RASTA filtering increased the execution time and memory consumption as discussed in Section 5.5.4 later.

5.5.3 Experiments on Model Parameters Initialization and Number of Mixtures per State

Design

Two mixture clusterings for Model Parameters Initialization are tried: a. Segmental K -means (SKM) [20] and b. Mixture splitting [29]. Segmental K -means is a combination between Viterbi alignment procedure to identify the state-level segmentation and K -means clustering to divide a set of vectors into a specified number of classes in order to locally minimize some within-class distance metric. However, this approach requires the number of mixture components to be decided in advance.

Mixture splitting method starts with a single-Gaussian model and incrementally increase the number of mixture components. This is done by duplicating the parameters of the original distribution and perturbing the means by a small amount in opposite directions. The splitting procedure is as follows:

- In each state, find the mixture candidate to be split by looking at one with the highest variance. The mixture would be split if the variance value is higher than specified threshold.

- Duplicate all the mixtures' parameters in each state plus the selected mixture to be split with the means perturbed in opposite direction by the width of the original mixture's variance.
- Continue the next iteration. The last iteration of model initialization occurs when either the condition for a mixture to be split is not met or the specified number of mixture is reached.

Below are the setup configurations for both the front-end and back-end processing.

FE feature extraction Process: RASTA filtering; Features: 12 static MFCCs + static Energy.

BE training Process: K -means segmentation for model initialization; HMM model: Left-to-right model with the number of states corresponding roughly to the number of sounds (phonemes) within the word (each word model has the same number of states), continuous observation densities with 1, 4, 16, 64, 128 mixtures per state, diagonal covariance matrices.

BE testing Viterbi algorithm to compute likelihood evaluation.

Results

In order to evaluate the two techniques above as well as the influence of HMM model size on the recognition rate, we test 5 configurations from 1 to 128 Gaussian per state. For the front-end configuration, RASTA filtering from the previous experiment was used. However, only static MFCCs and the energy were used as the features. The performance improvement uses Table 5.4 as reference because IT MM data set was used in this experiment.

Table 5.12: PRIRR without VAD with respect to the baseline system (Table 5.4) by increasing number of mixtures per state using segmental K -means(Exp. 5.5.3).

Mixtures per state	CT	HF-A	HF-B	HF-C
1	0	10.0	9.8	10.2
4	0.3	15.0	15.2	15.2
16	0.4	16.4	16.5	16.2
64	0.4	16.6	16.7	16.3
128	0.4	16.7	16.9	16.3

Table 5.13: PRIRR without VAD with respect to the baseline system (Table 5.4) by increasing number of mixtures per state using mixture splitting(Exp. 5.5.3).

Mixtures per state	CT	HF-A	HF-B	HF-C
1	0	10.0	9.8	10.2
4	0.3	14.9	15.2	15.0
16	0.4	16.2	16.4	16.2
64	0.4	16.3	16.5	16.3

For the segmental K -means approach, we got exactly the number of mixtures in each state. On the other hand, we can only put the maximum number of mixtures for the mixture splitting method. Table 5.12 and 5.13 show, as expected, that increasing the number of components in the model increases the recognition rate. However, using the higher order models only increases the recognition rate by small amount (less than 0.2%) compared to using middle-size order ones.

Table 5.13 also states a fact that the mixture splitting method continue to increase the number of mixture up to 64 components by state though specified to have 128 components. Another interesting fact is that the performance of the two methods is comparable for 1 to 16 mixtures per state. Also, it seems that 16 mixtures per state obtain the more interesting performance/resource ratio for the targeted application.

5.5.4 Experiments on Additive Noise Reduction Techniques

Design

Two main criteria are considered to evaluate the effectiveness of noise reduction methods; analysis using 2-D correlation coefficient method, and recognition rate. To evaluate the effectiveness of fixed-point code optimization, two criteria are used; performance in terms of CPU cycles per frame and memory consumption.

In additive noise reduction, there are two important components that we conduct experiments on: noise estimation and noise suppression factor. As mentioned in Section 3.2.1, the noise estimation methods could be a simple method (method 1) which is to estimate and update the noise spectrum only during the silent (e.g., during pauses) segments of the signal using a voice-activity detection (VAD) or a more involved one (method 2) which is obtained by recursive averaging of the noisy Mel-domain power spectrum using time-varying and frequency-dependent smoothing parameters, which are adjusted by speech-presence probability in individual Mel bins.

For the noise suppression functions, five methods e.g Modified Maximum Likelihood, Wiener filter, Spectral Subtraction, Magnitude subtraction, Decision directed method were tried. Please refer to Table 3.1 for the complete equations. Standard 12 MFCCs normalized in 16 bits and the energy in 16 bits were used as feature vectors of 13 dimensions. These features are also filtered using RASTA method. Each word model was trained as a left-to-right topology with the number of states corresponding roughly to the number of sounds and 1 maximum skipping among states by using K-means segmentation for model initialization and Baum-Welch algorithm for re-estimation of the model. Output distribution probabilities were modeled by means of mixtures of 4 Gaussian components only to test the effectiveness of the NR

methods as different background noise might be tolerated by the Gaussian mixture distributions.

Below are the setup configurations for both the front-end and back-end processing.

FE feature extraction Process: One of the additive NR technique + RASTA filtering; Features: 12 static MFCCs + static Energy.

BE training Process: K -means segmentation for model initialization; HMM model: Left-to-right model with the number of states corresponding roughly to the number of sounds (phonemes) within the word (each word model has the same number of states), continuous observation densities with 4 mixtures per state, diagonal covariance matrices.

BE testing Viterbi algorithm to compute likelihood evaluation.

Results on Noise Estimation Techniques

The two noise estimation algorithms mentioned in Section 3.2.1 are evaluated and compared. The assessment is based on an objective improvement in the segmental SNR and subjective study of speech spectrum.

The noise signal used in the evaluation is taken from speech utterances of hands-free (HF) microphones in cars under different driving conditions. Considering speech utterances of close-talk (CT) microphone as relatively clearer than those of HF microphones, we would be able to get the real noise signal by treating CT speech utterances as noiseless speech signals. The speech signal is sampled at 8 KHz and degraded by the various noise sources depending on driving conditions with segmental SNR in

Table 5.14: Segmental relative estimation error for various noise types from different driving conditions(Exp. 5.5.4).

Methods	Index 1	Index 2	Index 3	Index 4	Index 5
Method 1	0.249	0.264	0.254	0.233	0.257
Method 2	0.113	0.118	0.124	0.143	0.242

various degree of dB. The segmental SNR is defined by [38]

$$SegSNR = \frac{10}{|\mathcal{L}|} \sum_{l \in \mathcal{L}} \log \frac{\sum_i |X(i, l)|^2}{\sum_i |D(i, l)|^2} \quad (5.5.8)$$

where $\mathcal{L}$ represents the set of frames that contain speech, and $|\mathcal{L}|$ its cardinality. The spectral analysis is implemented with Hamming windows of 256 sample length (15 ms) and 120 samples frame update step.

A quantitative comparison between method 1 and 2 noise estimation methods is obtained by evaluating the segmental relative estimation error in various driving conditions. The segmental relative estimation error is defined by

$$SegErr = \frac{1}{L} \sum_{l=0}^{L-1} \frac{\sum_i [\hat{\lambda}_d(i, l) - \lambda_d(i, l)]^2}{\sum_i \lambda_d^2(i, l)} \quad (5.5.9)$$

where $\lambda_d(i, l)$ is the ideal noise estimate, $\hat{\lambda}_d(i, l)$ is the noise estimated by the tested method, and L is the number of frames in the analyzed signal. Table 5.14 presents the results of the segmental relative estimation error achieved by method 1 and 2 estimators for various noise types from different driving conditions (Please refer to Table 5.1 for the meaning of each index). It shows that method 2 obtains significantly lower estimation error than method 1.

The segmental relative estimation error is a measure that weighs all frames in a uniform manner, without a distinction between speech presence and absence. In

Table 5.15: Segmental SNR improvement for various noise types from different driving conditions(Exp. 5.5.4).

Methods	Index 1	Index 2	Index 3	Index 4	Index 5
Method 1	8.09	6.46	4.85	3.45	3.25
Method 2	8.50	6.61	4.93	3.45	3.35

practice, the estimation error is more consequential in frames that contain speech, particularly weak speech components, than in frames that contain only noise. We therefore examine the performance of the estimation methods when integrated into speech noise reduction system. The two estimators are then evaluated both objectively using an improvement in segmental SNR measure, and subjectively by analyzing the spectral tracks.

Table 5.15 summarizes the results of the segmental SNR improvement for various noise types from different driving conditions. Method 2 estimator consistently yields a higher or equal improvement in the segmental SNR, than method 1, under all tested environmental conditions. The fact that the benefit is greater for low input SNR implies that weak speech components are better preserved when noise is estimated by method 2. This is conformed by a subjective study of spectral tracks.

Another major advantage of method 2 noise estimation method is its tracking capability under non-stationary noise environments. In noise reduction applications, this quality is often not fully appreciated when considering the average improvement in the segmental SNR, since variations in the statistics of the noise are usually sparse. However, a frame-by-frame trace of the improvement in the segmental SNR, as illustrated by Figure 5.6 reveals that the effectiveness of method 2 is particularly notable during alteration in noise characteristics.

We provide qualitative comparisons between the two noise estimation algorithms mentioned in Section 3.2. The simple approach (method 1) estimates and updates the noise spectrum only during the silent using a VAD. Hence the adaptation time of the noise estimate depends on the VAD time to detect a non-speech frame. For non-stationary noise conditions where the noise power varies very slowly over time, method 1 and method 2 have the same adaptation time. But for increasing noise levels of non-stationary noise, the adaptation time might be slightly more than 30 frames for method 1 whereas for method 2 it is only 10 frames. Figure 5.6 shows a comparison between method 1 and method 2 in a situation where there is a sudden increase in noise power level. From the figure we can see that method 1 takes more than 30 frames to update the noise spectrum, whereas method 2 takes only 10 frames to update to the higher noise floor.

Results on Noise Suppression Functions

We provide quantitative comparisons between several noise suppression function algorithms mentioned in Section 3.2; Analysis using 2-D correlation coefficient method, and recognition rate.

Analysis using 2-D Correlation Coefficient The 2-D correlation coefficient is calculated as follows:

$$r = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{(\sum_m \sum_n (A_{mn} - \bar{A})^2)(\sum_m \sum_n (B_{mn} - \bar{B})^2)}},$$

where A and B are the MFCCs signals from close-talk and distant microphone respectively with m and n represent the frame index and MFCC index respectively. $\bar{A}$ and $\bar{B}$ mean the average of the matrix elements of signals A and B.

Figure 5.6: Comparison between the noise spectrum (for MFCC spectrum index = 1) estimated using Method 1 and Method 2 algorithms for an utterance corrupted by noise and multi-talker babble.

Figure 5.7: MFCCs tracking values between close talk (blue) and distant microphone (red) without NR (a) and with NR (b).

Figure 5.7 shows the tracking of MFCC values for MFCC index $n = 1$ from Channel CT and Channel HF-A signals during speech utterance. Figure 5.7 (a), (b) show plots of MFCCs with background noise without noise reduction and the enhanced MFCCs with noise reduction, respectively. The plots as well as their correlation coefficients show the effectiveness of the noise reduction in bringing the two different signals, which come from the same utterance, more similar.

While the increase in correlation coefficient is a suggestive measure in front-end performance of an ASR system, the final measurement is still in terms of improvement in recognition rate.

Table 5.16: PRIRR without VAD with respect to the baseline system (Table 5.4) by incorporating Method 2 NE, Modified Maximum Likelihood NSF and RASTA filtering.

Code version	CT	HF-A	HF-B	HF-C
Fixed NR	0.2	17.0	17.2	17.2
Float NR	0.2	17.9	18.1	18.0

Analysis using Recognition Rate In this section, we provide the result from the experiment design in terms of recognition accuracy. This experiment incorporated into the front-end processing the additive noise reduction using method 2 noise estimation and Modified Maximum likelihood as the noise suppression function as well as RASTA filtering. It achieved considerable improvement recognition rate for HF microphones on average of 18% by floating-point version and 17.1% by fixed-point one as compared to the same setup without the additive and convolutional NR (Table 5.16). The difference in recognition rate between the floating-point and fixed-point version is mainly due to some approximation functions in fixed-point NR and partly due to accuracy and saturation problems in representing the variables used. It should be noted that the presented approach is optimal from a speech recognition rate perspective because it substantially improves the speech recognition results for noisy speech, without serious degradation of the clean speech results. The degradation of the clean speech results could be seen by comparing Table 5.12 with mixture row 4 and column CT with Table 5.16 column CT.

Table 5.17 shows the comparisons among different noise suppression functions. The result was taken using Method 2 as the noise estimation procedure. The table shows that the Modified Maximum Likelihood gives more promising result compared

Figure 5.8: Comparison of the 2-D correlation coefficients among different Noise Suppression Factors.

to others. This might be because of the use of G_{min} as a delimiter which gives minimum noise suppression factor in the low SNR section. Other methods gave recognition rate close or even lower than the baseline system with RASTA filtering. This might stem from the fact that the algorithms introduce distortions that damage speech features important for speech recognition, though it has been reported that the algorithms performed well for speech enhancement. In other words, Modified Maximum Likelihood method can bring the two signals from CT and HF-A closer (similar to one another). Please refer to Figure 5.8. From the figure, we can easily see that most of the 2-D correlation coefficients from Modified Maximum Likelihood have higher values than that of other methods.

Table 5.17: PRIRR without VAD with respect to the baseline system (Table 5.4) by incorporating Method 2 NE, RASTA filtering and various NSFs.

Channel	Modified Maximum Likelihood	Wiener filter	Spectral Subtraction	Magnitude Subtraction	MMSE Approx. of Ephraim-Malah
CT	0.3	0.2	0.2	0.2	0.3
HF-A	17.0	16.0	15.2	14.8	15.3
HF-B	17.2	16.2	15.4	15.0	15.5
HF-C	17.2	16.5	15.4	15.0	15.4

Analysis of Memory and Computational Resources Skyeye [34] is used to simulate the performance of the system in an embedded computer system. It simulates EP7312 with ARM720T processor which is specially designed for ultra-low-power applications such as PDAs, smart cellular phones. Results for processing one frame (32ms) of speech are shown in Table 5.18. Baseline source code contains no noise reduction methods implemented in fixed-point. Floating code contains the noise reduction method using method 2 noise estimation and modified maximum likelihood described in this thesis implemented in floating-point. Finally, the fixed-point code contains the noise reductions method implemented in fixed-point. For each version of the code, the performance (in CPU cycles) and the memory consumption (in bytes) are shown.

Table 5.18 shows that with fixed-point implementation, the CPU cycles is reduced by 30% and memory consumption is reduced by 58% as compared to float NR. With this performance, a 90 MHz EP7312 with ARM720T processor needs 44 ms to process 23 ms samples with 8 KHz sampling frequency. Though by incorporating noise reduction, the number of CPU cycles and the memory consumption are increased by nearly twice compared to the baseline system without noise reduction, it is still

Table 5.18: Performance and memory consumed per frame of speech.

Source code version	Resources	
	CPU cycles	Memory bytes
baseline	3M	13802
float NR	9M	55926
fixed NR	6M	23474

acceptable according to the DSP chip design requirement.

Results on Gain Smoothing Techniques

The calculated noise suppression factor $G[i, l]$ could be smoothed in frequency and in time by gain smoothing GS operation. For smoothing in frequency, two methods shown in the Equations 5.5.10 and 5.5.11 below are tried.

$$GS(k) = \beta G(k) + (1 - \beta) ave_{k=M(k')} (G(k)) \quad (5.5.10)$$

$$GS(k) = \beta G(k) + (1 - \beta) med_{k=M(k')} (G(k)) \quad (5.5.11)$$

For Eq 5.5.10, the gain smoother produces a smoothed gain $GS[k]$ by averaging the center bin with its neighboring bins using a window function. This method focuses on reducing the level of musical noise which is occurred by impulsive spectral components whose frequency is random in consecutive time domain.

Eq 5.5.11 uses the median function instead of the normal averaging to avoid the mis-decision of the average value due to absolutely big or small value in the specified averaging region. This method focuses on the clearness of the speech spectrum which is needed in differentiating one phoneme with another phoneme in speech recognition.

In our experiments, the good result can be obtained if we set β to 0.5 and 0.2 for method 1 and 2 respectively. These two number were used in the previous experiment on Noise Suppression Functions.

For smoothing in time, first-order recursion of the in-frequency smoothed gain values with the following equation is used:

$$GS(k, l) = \alpha GS(k, l) + (1 - \alpha)GS(k, l - 1) \quad (5.5.12)$$

In our experiments, the good result can be obtained if we set α to 1. This means the in-time gain smoothing that depends on previous gain values $GS[i, l - 1]$ is not suitable to use. Two justifications for not using in-time gain smoothing are : (1) It might blur the distinction between consecutive phonemes. (2) Memory consumption increases to store the old values.

5.6 Suggested System Configurations

As mentioned in Section 1.2, this work focuses on robustness in adverse car environments, low computational costs, and end users' acceptance for further improvements of the HMM based speech recognition. Based on all the experiments conducted, we propose the system uses these following configurations:

FE feature extraction Process: Additive NR with method 2 NE and Modified Maximum Likelihood NSF + RASTA filtering; Features: 12 static MFCCs + static Energy.

BE training Process: K -means segmentation for model initialization; HMM model: Left-to-right model with the number of states corresponding roughly to the

Table 5.19: Medium matching DE data set; Test results.

Channel	Complete Detection (%)	Recognition Rate without VAD (%)	Recognition Rate with VAD (%)
CT	97.2	94.6	91.7
HF-A	97.4	85.7	83.4
HF-B	97.2	77.8	77.3
HF-C	97.6	80.2	79.6

number of sounds (phonemes) within the word (each word model has the same number of states), continuous observation densities with 4 mixtures per state, diagonal covariance matrices.

BE testing Viterbi algorithm to compute likelihood evaluation.

Using above configurations, DE database was then used to confirm that the configurations were also suitable for other languages with different characteristics. Table 5.19 shows the performance of the recognizer using IT MM data set with the suggested system configurations. The table shows better performance for CT and HF-A data set. For HF-B and HF-C data set, however, we could only get on average 79 % recognition rate without VAD and 78.45 % with VAD. Overall, the system performance between the two languages are comparable. Further improvement, however, could be envisaged by incorporating techniques which are specific to certain language. By exploiting the characteristics of the language, the noise reduction technique would be more effective and efficient in removing the noise and preserving speech information.

Chapter 6

Conclusions and Recommendations

The first part of this thesis focuses on the general theory of automatic speech recognition and the development of an automatic medium vocabulary speech recognizer in particular. The theory of speech recognition, presented in chapter 2, is a theory of mathematical nature; most textbooks on this subject concentrate mainly on the Hidden Markov Model and its dynamic programming algorithms. More practical aspects like the advantages or disadvantages of certain model topologies or the steps involved in data preparation and training are seldom covered. Many questions concerning these issues came up during the development of the speech recognizer described in chapter four and five. The first issue that has to be addressed is data preparation. This part of the process took much more time than initially expected. Data had to be selected that met the criteria posed for the recognizer. As huge database is used for training of speaker independent speech recognizers, a tool had to be created that could perform automatic data selection. One of the decisions that has to be made during data preparation is the topology of the acoustic models. In experiments with the SpeechDat-Car speech corpus we found that number of states corresponding roughly to the number of sounds (phonemes) within the word (each word model

has the same number of states) had better performance than if we were to assign different number of states for each word depending on the number of phonemes in the word. Since this means less parameters as well as lower computational cost for the recognizer, it was chosen for this project. During training it was found that it is important to create good initial models as HMMs are, due to their Markov property, very sensitive to initial values. Errors in the initial model set a limit to the performance of the final system. Therefore, manual segmentation was first done on the the SpeechDat-Car data by creating new transcriptions and copy them in new separate directories. Training of a model set on this segmented data may result in a recognizer that performs better while using for example fewer mixtures. The model parameters were then initialized using segmental K -means and re-estimated using *Baum–Welch* procedure.

Analysis have also been made pertaining to the memory consumption and the computational cost of the algorithm. Through simulation of the algorithm on a standard embedded system configuration, according to the DSP chip design requirement it is still acceptable to incorporate the selected algorithms.

The system can cope with car noises and this system is speaker independent. It has been tested with vocabularies of medium size (twenty three words). The overall results of the final system are quite good, more than ninety eight percent of close-talk words are recognized correctly. While for hands-free microphones, we could get on average eighty two percent recognition rate.

We have also tested the suggested configurations on another language which has different characteristics. Overall, the system performance between the two languages are comparable. Further improvement, however, could be envisaged by incorporating

techniques which are specific to certain language. By exploiting the characteristics of the language, the noise reduction technique would be more effective and efficient in removing the noise and preserving speech information.

By comparing the performance among different microphones which have different SNR values, the errors that occur typically involve really very low SNR utterances because the features might be damaged by the noise suppression factor. This kind of errors could be reduced by a more sensitive noise reduction technique that optimize the parameters of the noise suppression rule. The optimization rule should maximize an objective function that is closely related to speech recognition accuracy. Method such as maximum mutual information (MMI), which is well known to be correlated with speech information might be used.

Bibliography

- [1] Bocchieri E. A, *A study of the Beam Search algorithm for large vocabulary continuous speech recognition and methods for improved efficiency*, Proc. of Eurospeech (Berlin), 1993, pp. 1521–1524.
- [2] Mellor B. A and Varga A.P, *Noise masking in the MFCC domain for the recognition and noise speech recognition*, Proc Inst Acoustic Autumn Conf on Speech and Hearing, vol. 14, 1992, pp. 361–368.
- [3] ———, *Noise masking in a transform domain*, Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing, vol. 2, 1993, pp. 87–90.
- [4] C. Draxler G. Richard K. Choukri J. Allen A. Moreno, B. Lindberg and S. Euler., *SpeechDat-Car: A large speech database for automotive environments*, In Second International Conference on Language Resources and Evaluation (2000).
- [5] L. E. Baum, *An equality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes*, Inequalities, vol. 3, 1972, pp. 1–8.
- [6] Jack W. Crenshaw, *Math toolkit for real time programming*, CMP Books, Lawrence, Kansas, 2000.
- [7] SpeechDat-Car Database., <http://www.speechdat.org/sp-car>.

- [8] Johan de Veth, Laurent Mauuary, Bernhard Noe, Febe de Wet, Juergen Siemel, Louis Boves, and Denis Jouver, *Feature vector selection to improve ASR robustness in noisy conditions*, Proc. of Eurospeech, 2001, pp. 201–204.
- [9] B. Delaney, M. Hans, T. Simunic, and A. Aquaviva, *A lowpower, fixed-point front-end feature extraction for a distributed speech recognition system*, 2001.
- [10] Hermansky H. et al., *Compensation for the effect of the communication channel in the auditory-like analysis of speech(RASTA-PLP)*, Proc. of Eurospeech, 1991, pp. 1367–1370.
- [11] Rosenberg A. et al., *Cepstral channel normalization techniques for HMM-based speaker verification*, Proc. of Intl. Conf. on Spoken Language Processing, 1994, pp. 1835–1838.
- [12] M.-W Feng, R. Schwartz, F. Kubala, and J. Makhoul, *Iterative normalization for speaker-adaptive training in continuous speech recognition*, Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing, vol. 1, May 1989, pp. 612–615.
- [13] M. Gales and S. Young, *Robust continuous speech recognition using parallel model combination*, IEEE Trans. on Speech and Audio Processing, Sep. 1996, pp. 352–359.
- [14] M.J.F. Gales, *Maximum likelihood linear transformation for HMM-based speech recognition*, Computer Speech and Language, vol. 12, 1998, pp. 75–98.
- [15] J. L. Gauvain and C. H. Lee, *Maximum a posteriory estimation for multivariate gaussian mixture observations of Markov chains*, IEEE Trans. on Speech and Audio Processing, vol. 2, Apr. 1994, pp. 291–298.
- [16] Yifan Gong and Yu-Hung Kao, *Implementing a high accuracy speaker-independent continuous speech recognizer on a fixed-point DSP*, Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing **6** (2000), 3686–3689.

- [17] Murveit H, Monaco P, Digalakis V, and Butzberger J, *Techniques to achieve on accurate real-time large vocabulary speech recognition system*, Proc Human Language Technology Workshop (Plainsboro NJ, Morgan Kaufman Publishers Inc), Mar. 1994, pp. 393–398.
- [18] John Holmes and Wendy Holmes, *Speech synthesis and recognition*, 2nd ed., ch. 11, pp. 11–12, Taylor and Francis, New York, 2001.
- [19] B. H. Juang, *Maximum-likelihood estimation for mixture multivariate stochastic observations of Markov chains*, AT&T Techn. J, vol. 64, 1985.
- [20] B.-H. Juang and L.R. Rabiner, *The segmental K-means algorithm for estimating parameters of hidden Markov models*, IEEE Transactions on Signal Processing, vol. 38, Sep. 1990, pp. 1639–1641.
- [21] Bahl L, Gopalakrishnan PS, and Kanevsky DS Nahamoo D, *A fast admissible method for identifying a short list of candidate words*, Computer Speech and Language, vol. 6, 1992, pp. 215–224.
- [22] Mauuary L., *Blind equalization in the cepstral domain for robust telephone based speech recognition*, Proc. EUSIPCO, IX European Signal Processing Conference (Rhodes, Greece), vol. 2, Sep. 1998, pp. 359–363.
- [23] C.J. Leggetter and P.C. Woodland, *Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models*, Computer Speech and Language, vol. 9, 1995, pp. 171–185.
- [24] C. Levy, G. Linares, P. Nocera, and J.-F. Bonastre, *Reducing computational and memory cost for cellular phone embedded speech recognition system*, Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing, vol. 5, May 2004, pp. V – 309–12.

- [25] Xiao Li, *High-Speed, Low-Resource, Automatic Speech Recognition System*, Master's thesis, University of Washington, 2003.
- [26] Xiao Li and J. Bilmes, *Feature pruning in likelihood evaluation of HMM-based speech recognition*, IEEE Automatic Speech Recognition and Understanding Workshop, 2003, pp. 303–308.
- [27] L. R. Liporace, *Maximum-likelihood estimation for mixture multivariate observations of Markov sources*, IEEE Trans. Inform. Theory, vol. IT-28, Sep. 1982, pp. 729–734.
- [28] R. Messina and D. Juvet, *Sequential clustering algorithm for Gaussian mixture initialization*, Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing, vol. 1, May 2004, pp. I – 833–6.
- [29] C. Montacie, M.-J. Caratay, and C. Barras, *Mixture splitting technique and temporal control in a HMM-based recognition system*, Proc. of Intl. Conf. on Spoken Language Processing, vol. 2, Oct. 1996, pp. 977–980.
- [30] A. Moreno, B. Lindberg, and C. Draxler, *SPEECHDAT-CAR. A large speech database for automotive environments*, Intl. Conf. on Language Resources and Evaluation (Athens, Greece), 31 May - 2 June 2000.
- [31] Bernhard No, Jrgen Sienel, Denis Juvet, Laurent Mauuary, Louis Boves, Johan de Veth, and Febe de Wet, *Noise reduction for noise robust feature extraction for distributed speech recognition*, Proc. of Eurospeech (Aalborg, Denmark), 2001, pp. 433–436.
- [32] J. Nouza, *On the speech feature selection problem: Are dynamic features more important than the static ones?*, Proc. of Eurospeech, 1995, pp. 919–922.
- [33] S. Ogata and T. Shimamura, *Reinforced Spectral Subtraction method to enhance speech signal*, IEEE Catalogue No.01CH37239, 2001, pp. 242–245.

- [34] Skyeye project, <http://www.skyeye.org/index.shtml>.
- [35] L. R. Rabiner, *A tutorial on Hidden Markov Models and selected applications in speech recognition*, Proc. of the IEEE, vol. 77, Feb. 1989, pp. 257–286.
- [36] L. R. Rabiner, J. G. Wilpon, and B. H. Juang, *A Segmental k-means training procedure for connected word recognition*, AT&T Technical Journal, 1986, pp. 21–40.
- [37] S. Rangachari and P.C. Loizou, *A noise-estimation algorithm for highly non-stationary environments*, Speech Communication, 2006, p. 220231.
- [38] T. Barnwell S. Quackenbush and M. Clements, *Objective measures of speech quality*, Englewood Cliffs, NJ:Prentice-Hall, 1988.
- [39] K. Shinoda and K. Iso, *Efficient reduction of Gaussian components using MDL criterion for HMM-based speech recognition*, Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing, vol. 1, May 2002, pp. 869–1 – 869–7.
- [40] I. Varga, S. Aalburg, B. Andrassy, S. Astrov, J.G. Bauer, C. Beaugeant, C. Geissler, and H. Hoge, *ASR in mobile phones-an industrial approach*, IEEE Trans. on Speech and Audio Processing **10** (2002), no. 8, 562–569.
- [41] B. Kaciv Z. Horvat B. Vlaj, D. Kotnik, *Usage of frame dropping and frame attenuation algorithms in automatic speech recognition systems*, EUROCON 2003. Computer as a Tool. The IEEE Region 8, vol. 2, Sep. 2003, pp. 149–152.
- [42] L. Welling, S. Kanthak, and H. Ney, *Improved methods for Vocal Tract Normalization*, Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (Phoenix, AZ), vol. 2, Apr. 1999, pp. 761–764.
- [43] L. Welling, H. Ney, and S. Kanthak, *Speaker adaptive modeling by Vocal Tract Normalization*, IEEE Trans. on Speech and Audio Processing, vol. 10, Sep. 2002, pp. 415–426.

- [44] S.J. Wolfe, P.J.; Godsill, *Simple alternatives to the ephraim and malah suppression rule for speech enhancement*, Statistical Signal Processing, Proceedings of the 11th IEEE Signal Processing Workshop, 2001, pp. 496 – 499.
- [45] J. Yang, *Frequency domain noise suppression approaches in mobile telephone systems*, Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing, vol. 2, Apr. 1993, pp. 363–366.
- [46] S. Young, *Large vocabulary continuous speech recognition: A review.*, Proc. of IEEE, Dec. 1995.