A Model-based Approach to
Vocal Tract Length Normalization

A thesis submitted to
The Hong Kong University of Science and Technology
in partial fulfillment of the requirements for
the Degree of Master of Philosophy in
Electrical and Electronic Engineering

By

CHAU Chun Keung

Department of Electrical and Electronic Engineering
B. Eng. (Elec)

August 2001
A Model-based Warping Approach to Vocal Tract Length Normalization

by

CHAU Chun Keung

Approved by:

[Signature]
Dr. Bertram SHI
Thesis Supervisor

[Signature]
Dr. Chi Ying TSUI
Thesis Examination Committee Member (Chairman)

[Signature]
Dr. Man Hung SIU
Thesis Examination Committee Member

[Signature]
Acting Dept Head
Prof. H.S. Kwok

Department of Electrical and Electronic Engineering
The Hong Kong University of Science and Technology
August 2001
A Model-based Warping Approach
to Vocal Tract Length Normalization

by
Chau Chun Keung
for the degree of
Master of Philosophy in Electrical and Electronic Engineering
at The Hong Kong University of Science and Technology in August 2001

ABSTRACT

We describe an architecture for speech recognition based interactive toys and discuss a strategy we have adopted to deal with the fact that the speech recognizer must deal with users whose age ranges from children to adults. Large variations in vocal tract length between children and adults can significantly degrade the performance of speech recognizers. We propose a model-based approach to vocal tract length normalization (VTLN), and compare it with the more common feature-based approach. Our results indicate that there is very little difference in performance between the two schemes. We also compare the performance of three warping factor detection schemes previously proposed for feature based VTLN, and demonstrate that these schemes are also effective for model based VTLN. Assuming enough memory is available to store pre-warped models, the computational cost of model warping is lower than that of feature warping. Under certain conditions, the computational cost
may also be lower even if the warped models must be computed dynamically. In addition, the model warping approach requires no changes to the standard Mel-Frequency Cepstral Coefficient (MFCC) front-end. This technique is well suited for embedded systems using DSP and/or custom VLSI chips.
To

my dearest father and mother
Acknowledgment

Without the intellectual, physical and emotional support from many individuals, not only this thesis would not have been possibly completed, but also I could not have enjoyed my time at this university, where I have been growing to be an educated as well as a cultivated person during the last five years.

Firstly, I would like to especially thank my thesis supervisor, Dr. Bertram Shi, not only his technical support, but also for creating this particularly stress-free environment which provides a large degree of freedom for me to enjoy my studies and my research work. He also always shares his valuable experience and his intellectual maturity with me that all these are very useful for my future career. Next, I would like to thank Dr. Jack Lau providing excellent facilities and environment. I would also like to thank Dr. Chi Ying Tsui and Dr. Man Hung Siu to evaluate my thesis.

I would like to extend my appreciation to my friends and my colleagues in Consumer Media Center. Special thanks go to my partners Jackson Lai and Newman Lau for sharing their experience in the Interactive Toy Project.

Finally special thanks should be given to my family and my friends from primary and secondary school for their endless support during my life of studies.
Contents

1 Introduction
   1.1 Motivation................................................................. 1
   1.2 Requirements on the speech recognizer............................ 2
   1.3 Outline........................................................................... 4

2 Implementation of Speech Recognizer using DSP.................. 6
   2.1 Personal Computer (PC) ................................................. 7
   2.2 Digital Signal Processor (DSP) ....................................... 8
      2.2.1 End-point Detection............................................... 10
      2.2.2 Feature Extraction.................................................. 10
         2.2.2.1 Pre-emphasis and Hamming Window....................... 11
         2.2.2.2 FFT and its Power Spectrum................................. 13
         2.2.2.3 Mel-scaled Filterbank......................................... 14
         2.2.2.4 Discrete Cosine Transform (DCT) ......................... 14
         2.2.2.5 Log................................................................. 15
         2.2.2.6 DSP Feature Extraction .................................... 16
   2.2.3 Pattern Classification.............................................. 17

3 Feature and Model Warping............................................. 20
   3.1 Feature-based VTLN..................................................... 21
   3.2 Model-based VTLN....................................................... 22
      3.2.1 Transformation the Static Coefficients...................... 23
      3.2.2 Transformation the Delta and Acceleration Coefficients... 24
3.3 Experimental Results ................................................................. 26

3.3.1 Experiment Setup ............................................................... 26

3.3.2 Evaluation of Feature and Model-based Approaches ............... 28

3.3.3 Previous Model-Warping Approach ...................................... 30

3.3.4 Evaluation of Previous and Proposed Model-based VTLN ....... 31

4 Warp Factor Detection ............................................................. 33

4.1 Maximum Likelihood Strategy (ML) ......................................... 33

4.2 Multiple-Pass Strategy (MP) .................................................. 34

4.3 Gaussian Mixture Model Strategy (GMM) ................................ 37

4.4 Experimental Results .......................................................... 40

5 Feature vs Model based VTLN in Computational Cost .................. 44

5.1 System Configuration ............................................................ 44

5.1.1 Input Signal and Feature Vector ....................................... 44

5.1.2 Configuration of Acrostic Model ....................................... 45

5.1.3 Warp Factor Detection .................................................... 45

5.2 Feature warping vs Model warping with infinite memory space .. 46

5.3 Feature warping vs Model warping with limited memory space .... 48

6 Conclusion .................................................................................. 52

Appendix A .................................................................................... 54

Bibliography .................................................................................. 58
List of Tables

3.1 The best performance (word accuracy) obtained on the test set........... 29
3.2 The word accuracy obtained on the test set at warp factor=1.2......... 32
4.1 Performance of three warp factor detection approaches............... 41
4.2 No. of operation for the 3 warp factor detection approaches......... 42
5.1 Overall Computational Cost for Feature and Model Warping ........... 47
5.2 Approximated no. of flops for 1 frame feature warping............... 48
5.3 Approximated no. of flops for 13 MFCC model warping............... 49
5.4 Approximated no. of flops for 13 derivatives MFCC model warping.... 50
5.5 Approximated no. of flops for 13 accelerations MFCC model warping.. 50
List of Figures

2.1 System Overview of DSP implementation.......................... 6
2.2 Graphical User Interface for Keyword Selection....................... 8
2.3 Flow Diagram for Speech Recognition................................ 9
2.4 Block Diagram of Feature Extraction.................................. 11
2.5 Plots of Real values of Log₂, Fast Log₂ and Taylor Series........... 16
2.6 Keyword Model with filler model........................................ 17
2.7 Dynamic Programming for Viterbi Algorithm............................ 18
3.1 Frequency mapping for feature warping................................ 21
3.2 Model Warping and “Backward-Forward Operation”..................... 23
3.3 Recognition Accuracy vs Warp Factor for Feature and Model Warping... 28
3.4 Recognition Accuracy vs. Warp Factor for Model Warping using Normalized and Unnormalized Models.................................................. 28
4.1 ML Strategy for (A) Feature Warping and (B) Model Warping........ 34
4.2 MP Strategy for (A) Feature Warping and (B) Model Warping.......... 36
4.3 GMM Strategy for (A) Feature Warping and (B) Model Warping........ 39
5.1 Feature Warping Procedures.................................................. 46
I. Graphical User Interface of PC Program..................................... 54
II. Program Flow of DSP Implementation...................................... 55
Chapter 1

Introduction

1.1 Motivation

Speech recognition technology enables us to build interactive systems which respond to users spoken input. We are developing an interactive toy which enables users to influence the flow and outcome of stories being told by the toy. Stories written for the toy contain choice points, at which the user can choose from multiple possible branches. Different choices will lead the user down different paths, resulting in different story outcomes. At each choice point, the toy uses speech recognition to identify the particular branch the user wishes to follow. Our interactive toy consists of three parts: the story telling engine, the speech recognition engine, and the authoring environment.

The story telling engine is responsible for telling a multi-path story. A multi-path story is a story which contains branching points where the plot of the story may change based on user input. In our system, user input is spoken and recognized via the speech recognition engine. Once a branching point is reached, the story telling engine sends a list of keywords to the speech recognition system. The speech recognition system records the speech of the users, spots the keywords in the utterance and returns the recognition results to the story telling engine. The story telling engine will then dynamically synthesize the story content according to the results obtained by the speech recognition system until another branch point is reached.

The authoring environment provides authors with a graphical user interface for
structuring their content to represent a desired flow for content delivery by the interactive toy. It also provides a platform for manipulating the materials. The branches of story flow and the expected users’ responses are decided by authors at this stage. The expected users’ responses are treated as keywords. The output of this environment is a database which contains the text of the story, multimedia content, as well as the branching structure. This database is all that is required by the story telling engine to tell the given story. Then, the stories can be shared and distributed over Internet in this format.

1.2 Requirements on the speech recognizer

The speech recognition system takes the list of expected keywords from the story telling engine and determines which of the keywords was spoken by the user. Although the number of keywords at any particular branching point is limited, for maximum flexibility, we must be able to handle a large number of possibilities. Therefore, we use phoneme based HMM models. To form a keyword, we can concatenate the phoneme-based models. Thus, the number of models is minimized.

To enable users to interact more naturally with the toy, the speech recognizer must be able to handle input which contains other words in addition to one of the expected keywords. In addition, it must also be able to recognizer when the utterance contains none of the expected keywords. Keyword spotting enables users to speak commands or requests without concern to the exact syntax of the request, as long as their utterance contains one or more of the keywords which the system has been designed to detect. A filler model is used to do the keyword spotting in order to extract the main theme of the speech. As users respond to the speech recognizer with a whole
sentence with natural way, we need a method to extract the keyword. Thus, filler model is a method to achieve the goal. Keyword spotting techniques are important to the interactive toy project because during playing, users will not know exactly what keywords they allowed to speak. Therefore, we need to extract the keywords from users’ speech. In order to implement keyword spotting, a filler model [1], [2] and [3] usually used to absorb the non-keyword part of the utterance. A filler model can be trained by any speech data. Data contained keyword speech can also used for training filler model and the structure of filler model can be single state with multiple Gaussian mixture [1].

The first two issues requirements above were addressed in the first version of the speech recognition engine [21]. In this thesis, we addressed two enhancements or modifications to the speech recognition engine. The first was porting the speech recognition from the PC platform to a DSP platform. The second was the development, implementation and evaluation of a model-based approach to vocal tract length normalization to deal with the wide variations in vocal tract length from the expected users of the interactive toy.

The first version of the interactive toy used a speech recognition engine implemented on the PC. However, the PC has disadvantages for toy applications, such as its high cost and the inconvenience and time associated with booting up the machine before the interactive toy can be run. A dedicated hardware implementation would eliminate these problems. As a first step towards this dedicated implementation, we ported the speech recognition onto a Digital Signal Processors (DSP). The implementations using the personal computer and that using the DSP are quite different because computer generally has large memory storage and high computational power; while
the DSP has high computational power, but the memory storage is limited to cut down the cost. Thus, some stages of speech recognition needed to be optimized to exploit the properties of DSP, as discussed in the Chapter 2.

The other main area we focused on was dealing with the fact that the recognizer must deal with a wide range of users, from children to adults. Variations in the vocal tract length can significantly degrade the performance of speech recognizers. The length of the vocal tract can vary by as much as 25% between speakers and the effect of this is to cause differences in the short-term spectrum of the same sound uttered by different speakers [4]. This variation can degrade the performance of speech recognizers. For example, models trained on adult data typically do not perform well with children [5]. In the second part of this thesis discusses our model-based approach to compensating for this problem. The contributions of the model based VTLN are which it can be applied only once, rather than to each incoming frame of speech. In addition, no changes or additions to an existing MFCC front end need be made. Thus, it is suitable for some computer-independent speech recognition systems, such as DSP [18] and VLSI chips [19], where the front end may be fixed in advance. With the warp factor detection methods, the whole model based VTLN can be run. It is especially suitable for the processors which have only limited memory and computational power because the computational cost of model based VTLN is low.

1.3 Outline

The remaining material of this thesis organized into four chapters.

Chapter 2 – Implementation of Speech Recognizer using DSP. In this chapter, we will
describe the implementation of speech recognizer using DSP. We will show the architecture of the DSP and how the speech recognition algorithms were modified to exploit the properties of the DSP.

Chapter 3 – Feature and Model Warping. We introduce the feature-based and model-based approaches to vocal tract length normalization in this chapter. Then, we will compare experimentally the result of feature warping and our proposed warping. Finally, we will compare the results of a previous approach to model warping and our proposed approach.

Chapter 4 – Warp Factor Detection. In this chapter, we will review the three feature-based warp factor detection methods. Then, we will show how these detection methods can be modified to work for model-based warp factor detection. Experimental results of warp factor detection methods of model warping will be shown.

Chapter 5 – The Computational Cost of Feature versus Model based VTLN. In this chapter, we compare the computational complexity of feature and model based VTLN using two systems for comparison. The first system is assumed to have enough memory space to store all the pre-calculated model coefficients. The second system is assumed to have limited memory space.
Chapter 2

Implementation of Speech Recognizer using DSP

The speech recognizers are often implemented using personal computers. However, the hardware and the software requirements are high. The hardware requirement of computer is a sound card, a high processing power CPU, and enough RAM and harddisk space to run and store the program; the software requirement of computer implementation is an expensive operation system such as Windows 98.

To reduce this cost, we investigated the implementation of a computer-independent speech recognizer using a digital signal processor (DSP). The DSP has limited resources, but the recognizer still can be run. In addition, there is no requirement for an expensive operating system and the system can be portable. However, the implementation of computer-based recognizer is different from that of DSP recognizer, which as only has limited processing power and memory storage. Thus, we need to carefully utilize the resources of DSP and use some different algorithms.

![Diagram](image)

*Figure 2.1: System Overview of DSP implementation*

Figure 2.1 shows the system overview of the DSP speech recognizer. All phoneme models, keyword phoneme sequences and the compiled program code are downloaded to the DSP from a PC. All the speech recognition procedures are done in
the DSP. The reasons of using PC for assistance and how we plan for the algorithms are mentioned in the sections below.

2.1 Personal Computer (PC)

The PC side provides a Graphical User Interface for users to input the keywords they want to be recognized and converts the keywords into phoneme sequences. Phonemes are basic elements to describe the pronunciation of keywords. Each keyword is described by the phoneme sequence, which is found by look up in a dictionary of keywords. Once the phoneme sequences are determined, the PC program then downloads the program code, all phoneme models and phoneme sequences of the keywords onto the DSP.

DSP then constructs Hidden Markov Models of the keywords by concatenating the HMM models for each of the phonemes into a keyword model according to its phoneme sequences. For example, the keyword “apple” is described as “ae”, “p” and “l” phoneme sequences. In our implementation, the phoneme sequences are represented as the indexes of models such as 2, 4, 6 where the number “2” means “ae” phone model in our phoneme models database.

The reason we use the PC to perform this step, rather than the DSP, is that the dictionary of keyword mapping to the phoneme sequence is quite large (~16Mbytes), corresponding to about 200,000 English words. If stored on the DSP, the dictionary along would take up most of the memory resources. While this limits the flexibility of the DSP recognizer because the vocabulary must be determined in advance, it is well suited to our application, since the keywords corresponding to a particular story or
application are defined during the authoring stage.

Figure 2.2: Graphical User Interface for Keyword Selection.

Figure 2.2 shows the Graphical User Interface of our PC program. Users can enter the list of desired keywords into the “Word List” window. The corresponding phoneme sequences determined by dictionary lookup are shown in the “Pronounce List” window. Once the button “DSP Process” being pressed, the program code, all phoneme models and phoneme sequences of the keywords are downloaded onto the DSP.

2.2 Digital Signal Processor (DSP)

The main job of the DSP is to perform the concatenation of keyword models from the phoneme models and to use these models to recognize a user’s utterance. We use the TMS320C31 floating point DSP to develop the recognizer.
The phoneme models are stored into the memory space. Given the phoneme sequences from PC, we can dynamically generate the recognition keyword models by combining the phoneme models. The total memory space of the C3X is 16M(million) 32-bit words. Program, data, and I/O space are contained within this 16M-word address space, allowing the storage of tables, coefficients, program code, or data in either RAM or ROM. In this way, memory usage is maximized and memory space allocated as desired. In our case of implementation, we use 1,350 words for the main program codes, 8,062 words for our tables and coefficients and 11,040 words for all of the 46 phoneme models.

The speech recognition algorithm we implemented can be divided into 3 parts: endpoint detection, feature extraction and pattern classification. Figure 2.3 shows the functional blocks and the program flow of our speech recognizer. In the following, we describe each of these steps in more detail, the capabilities of the DSP we exploited to optimize their execution, and the algorithmic approaches we used to overcome some of the limitations of the DSP.

Figure 2.3: Flow Diagram for Speech Recognition
2.2.1 End-point Detection

The endpoint detection is a process to detect the portions of the input recording which contain speech, rather than silence. Because the correct detection of speech affects the recognition rate, we use an algorithm described in [6] to detect the boundaries of the speech. It uses frame energy and zero crossing rate within each frame to distinguish between noise and speech.

We use the combination of frame energy and zero crossing rate to detect the start and the end of speech. We set thresholds on both measures. If the zero crossing rate is same or higher, but the frame energy is same or lower than the threshold, it is probably noise and no speech coming into the system. The speech is detected only if both the frame energy and zero crossing rate of that frame exceed pre-set thresholds.

Because the surrounding environment may change, we must decide the threshold during a calibration stage when the recognizer starts up. For our setting, the environment is evaluated for 2 seconds and the frame energy and the zero crossing rates of the environment are recorded. Then evaluated results are multiplied 1.5 to set the thresholds.

2.2.2 Feature Extraction

We use Mel-scaled Frequency Cepstral Coefficients (MFCC) as features. The speech signal is framed into windows of 400 samples. Each window is converted to 13 MFCCs. Then, the feature vectors are stored into the memory. Later, the MFCCs are used to generate the delta and acceleration coefficients. In total, a 39 dimensional feature vector is used for Pattern Classification.
This section reviews the feature extraction algorithm described in [7] and describes the implementation of the feature extraction. Figure 2.4 shows the main blocks of the feature extraction algorithm.

2.2.2.1 Pre-emphasis and Hamming Window

The pre-emphasis and hamming windows are the filters used to emphasize the speech data for feature extraction. The equations are shown in (2.1) and (2.2):

\[ H(z) = 1 - az^{-1}, a = 1 \]  
\[ h(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right), n = 0, \ldots, N-1 \]

Because the DSP core does not supply sine, cosine, logarithm or exponential functions, we must implement these algorithmically. One approach is by approximating them via Taylor series. However, we did not adopt Taylor series as many terms may be required to get an accurate estimation if the input range is large.

Instead, we used a "table-lookup" method. The coefficients are pre-calculated first and then stored into the memory. During runtime, we look up the table, read and apply the pre-calculated coefficients to the framed data. By using table-lookup method, we save the computational cost and the coefficients are more accurate than the values estimated by Taylor series.
From theoretical point of view, table-lookup method is good. However, there are some requirements of the hardware to archive it. Because the speech recognition performing in real time, the processing time is very important. For example, if memory access takes very long time, the overall processing time will take very long. Another example is related to representation of the coefficients. If they are not accurate, the whole calculation will be wrong. Therefore, a good architecture of DSP is a successful key of real time speech recognizer.

Our DSP is using floating-point processor. It has the following parts which enable the table lookup to be performed faster: 1. Floating-point/integer multiplier 2. Auxiliary Register Arithmetic Units and 3. Four internal buses.

**Floating-point/integer multiplier**

We need a precise floating format to represent those coefficients. The multiplier performs single-cycle multiplications on 32-bit floating-point values. The floating-point/integer multiplier is used to make the calculations of the feature-vectors more precise.

**Auxiliary Register Arithmetic Units (ARAU)**

Accessing data from memory is important for the calculation. Fast and efficient data access makes the calculation faster by reading two data from memory and performing multiplies and adds in one cycle. It is useful for the table look-up methods as the process is mainly reading coefficients from memory. Thus, the ARAU and Internal Buses are the key to enable those algorithms to be performed in real time.

Two auxiliary register arithmetic units (ARAU0 and ARAU1) can generate two
addresses in a single cycle. The ARAUs operate in parallel with the multiplier and ALU. They support addressing with displacements, index registers (IR0, IR1), and circular and bit-reversed addressing. Taking the table look-up method as an example, all the coefficients are stored in sequential order, then the coefficients can be accessed with addressing with displacements mode. If these coefficients are repeatedly accessed, the circular addressing mode can be used for efficient data accessing.

Four Internal buses

If the multiplies and adds can perform in a single cycle, the performance after using table look-up method and the performance of other computational function blocks increase. The DSP has four internal buses, CPU1, CPU2, REG1, REG2 carry to operands from memory and two operands from the register file, allowing parallel multiplies and adds/subtracts on four integer or floating-point operands in a single cycle. For example, after reading two filter coefficients from memory, the coefficients can be directly applied to the 2 signal data in one cycle with the parallel processing. The overall processing time is half of the system which without this property.

2.2.2.2 FFT and its Power Spectrum

The input sequence of DFT is \( \{u(n), n = 0, \ldots, N - 1\} \) and the equation is

\[
U(k) = \sum_{n=0}^{N-1} u(n) e^{\frac{2\pi in}{N}}, i = 0, \ldots, N - 1
\]

(2.3)

In practice, the Fast Fourier Transform (FFT), a fast version of DFT is used to minimize computation load. The table look-up method is also applied here to generate the sine and cosine components of the complex exponential.

The Auxiliary Register Arithmetic Units and four Internal buses of the DSP enable
the FFT to run faster. The ARAUs allow circular and bit-reversed addressing and the internal buses allow parallel operations in one clock cycle. This makes the Radix2 algorithm of FFT performing faster because Radix2 algorithm accesses the data in bit-reversed addressing mode and DSP supports parallel multiplies and adds/subtracts after accessing the data.

The power spectrum \( \{S_i(k)\} \) is given by,

\[
\{S_i(k) = |U(k)|^2, k = 0, ..., N - 1 \}
\]

(2.4)

2.2.2.3 Mel-scaled Filterbank

The “binning” is performed according to the Mel-scaled filterbank. Binning, putting the relevant data into a “bin”, is a kind of grouping. Binning also weights the data according to the filterbank weighting coefficients. Table look-up is used to store the filterbank coefficients. The ARAUs can be exploited to perform the grouping efficiently, as the addressing modes can access the data easier.

2.2.2.4 Discrete Cosine Transform (DCT)

A Discrete Cosine Transform (DCT) is applied to filterbank coefficients \( \{m(i)\}, i = 0, ..., M - 1 \). The transform is defined as follows:

\[
c(k) = \alpha(k) \sum_{i=0}^{M-1} m(i) \cos\left(\frac{\pi(2i + 1)k}{2M}\right), \quad k = 0, ..., M - 1
\]

(2.5)

where \( \alpha(0) = \sqrt{\frac{1}{M}} \), \( \alpha(k) = \sqrt{\frac{2}{M}} \), \( k = 1, ..., M - 1 \). The cosine terms are implemented by table look-up method.
2.2.2.5 Log

Although we can use Taylor series to approximate the log, it is suitable only in an appropriate range.

\[
\ln(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \frac{x^n}{n} \quad \text{where} \quad 2 \geq 1-x > 0
\] (2.6)

We can perform \(\log(x)\) in easier way by computing \(\log_2(x)\) using a property of the DSP’s floating point format, and then multiplying the base constant to convert to any other log base. To compute \(\log_2(x)\), we use the algorithm described in [10]. A floating point number \(X\) is represented using mantissa and exponent as follows:

\[
X = M*2^E
\] (2.7)

where \(M\) is the mantissa and the \(E\) is the exponent. By taking \(\log_2\) of this number \(X\), the log of number \(X\) becomes the sum of the exponent and mantissa in log form:

\[
\log_2(X) = E + \log_2(M)
\] (2.8)

Since \(E\) is in the exponent register, no calculation is needed and the value can be used directly as the part of estimated result. The remaining part is to estimate the \(\log_2(M)\).

If the value \(M\) is repeatedly squared, it becomes:

\[
X' = M^N \quad N=1,2,4,8,16\ldots
\] (2.9)

where \(X'\) can be represented using mantissas and exponent as:

\[
M^N = (M')*2^{E'}
\] (2.10)

where \(E'\) and \(M'\) are the new exponent and mantissa respectively which represent \(M^N\).

To approximate \(\log_2(M)\), we take the \(\log_2\) of both sides of (2.10) and divide by \(N\):

\[
\log_2(M^N) = \log_2 \left\{ (M')*2^{E'} \right\}
\]

\[
N*\log_2(M) = E' + \log_2(M')
\]

\[
\log_2(M) = \frac{(E' + \log_2(M'))}{N}
\] (2.11)

The final approximated result of \(\log_2(X)\) is equal to:

\[
\log_2(X) = E + E'/N
\] (2.12)
where the error $\log_2(M'/N)$ is very small when $N$ is large.

![Graph](image.png)

*Figure 2.5: Plots of Real values of Log₂, Fast Log₂ and Taylor Series*

The figure 2.5 shows that the approximation of $\log_2(X)$ using Fast Log₂ method with $N=128$ and Taylor Series with $N=7$ which take same operations in both cases. The two approaches can approximate correct values of $\log_2(X)$ in the range from 0.01 to 2. However, the Taylor Series diverges for input values larger than 2; while the Fast Log₂ method can still make the correct approximation. Thus, Fast Log₂ method is a more robust approach for different range of values.

### 2.2.2.6 DSP Feature Extraction

We compared the MFCC’s computed in the DSP by applying table-lookup method and the Fast Log₂ algorithm with the MFCC’s computed in the PC. The average value of MFCC is 14.475 and the maximum difference between outputs of the PC and DSP
is 0.05 for 5 seconds speech with 16k sampling rate.

2.2.3 Pattern Classification

The pattern classification stage takes the input observation (sequence of feature vectors) and finds the keyword model which best fits the observation. The fit is evaluated by finding the likelihood of the best path through a HMM network consisting of a filler models and the phoneme sequence representing the keyword, evaluated using the Viterbi algorithm.

With the filler model to absorb the non-keywords sentence shown in figure 2.6, we are able the extract the keyword. The filler model is trained from a large amount of unconstrained speech and represents the general speech properties. The structure of the models is:

![Figure 2.6: Keyword Model with filler model](image)

At Viterbi stage, the non-keyword speech has higher probability at the state of filler than at the state of keyword. The non-keyword speech is absorbed by the filler model. Without the filler model, non-keyword speech is introduces mismatch with the pure keyword model which degrades the recognition accuracy.

The steps of the Viterbi algorithm [8] are shown below:

Step 1: Initialization

\[ V_i(i) = \pi_i b_i(o_i) \quad 1 \leq i \leq N \]  

(2.13)
Step 2: Induction

\[ V_t(j) = \max_{i \in \mathbb{N}} \{ V_{t-1}(i) a_{ij} b_j(o_t) \} \quad 2 \leq t \leq T; \quad 1 \leq j \leq N \]  \hspace{1cm} (2.14)

Step 3: Termination

The keyword score = \( V_T(N) \)  \hspace{1cm} (2.15)

where \( \pi \) is the initial probability; \( a_{ij} \) is the transition probability from state \( i \) to state \( j \); \( b_j(o_t) \) is the output probability given \( o_t \) for the observation sequence. \( V_t(j) \) is the score at \( t \) frame at state \( j \). Each keyword has their own scores after run Viterbi. We select the keyword which has the highest score be the extracted keyword.

\[ \text{Figure 2.7: Dynamic Programming for Viterbi Algorithm} \]

The Viterbi computation is illustrated diagrammatically in Figure 2.7. The black circles mean the states. Because we use dynamic programming, at every immediate state, we only need to decide which path starting at that state has the highest
probability.

Here, we apply a method to save memory usage. From the figure, we can see that only current frame scores are needed to decide the next path. Because we are only interested in the score of the best path, we do not need to perform backtracking. Thus, to save the usage of memory, we only store two columns of scores from every decision of path, rather than the whole Viterbi matrix.

Another method is used to save the computational cost. In the Viterbi algorithm, we always deal with the output probability:

\[
b_j(o_t) = \frac{1}{(2\pi)^n|\Sigma_j|} \exp\left[-\frac{1}{2}(o_t - \mu_j)'\Sigma_j^{-1}(o_t - \mu_j)\right]
\]  

(2.16)

Many implementations of the Viterbi algorithm work with the natural logarithm of the output probability, rather than the output probability directly, because it can help prevent under-flow errors due to multiplying all of the probabilities together.

\[
\ln b_j(o_t) = \ln\left\{\frac{1}{(2\pi)^n|\Sigma_j|} \exp\left[-\frac{1}{2}(o_t - \mu_j)'\Sigma_j^{-1}(o_t - \mu_j)\right]\right\}
\]  

(2.17)

In the single Gaussian mixture case, the log and the exponential function will cancel each other to form:

\[
\ln b_j(o_t) = K - \frac{1}{2}(o_t - \mu_j)'\Sigma_j^{-1}(o_t - \mu_j) \text{ where } K = \ln\left\{\frac{1}{(2\pi)^n|\Sigma_j|}\right\}
\]  

(2.18)

As our acoustic phoneme models use only single Gaussian output probabilities, we adopt the above equation and pre-calculate the K to eliminate the operations of taking log and exponential.
Chapter 3

Feature and Model Warping

The vocal tract is commonly modeled as a tube with varying cross section, which is excited either at one end or at a point along the tube [8] and [11]. Acoustic theory tells us that the transfer function of energy from the excitation source to the output can be described in terms of the resonances of the tube. These resonances are called formants for speech, and they represent the frequencies at which the tube passes the most acoustic energy from source to output. The vocal tract length has a large effect on the formant frequencies. When the vocal tract length gets longer, the positions of formants shift down and the shape of power spectrum will be compressed accordingly. Conversely, shortening the vocal tract shifts the formants upwards. This variation is a source of mismatch between the input utterance and the model during recognition.

Vocal tract length normalization (VTLN) seeks to remove this mismatch. This chapter describes two approaches to VTLN. The feature-based approach to VTLN modifies features extracted from the speech waveform so that they better match the model. The model-based approach modifies the model to better match incoming features. The feature-based approach is more commonly found in the speech recognition literature, however, we argue that the model-based approach is preferable.

In particular, this chapter will show that the performance of the approaches is nearly identical, despite the fact that the model based approach uses a coarser division of the frequency content of the signal. In additional, the performance of our proposed model-based approach is nearly identical to a previously proposed approach [16]. In a
later chapter, we will show that the model-based approach is preferable from the point of view of computation complexity.

3.1 Feature-based VTLN

The feature-based approach studied here was based on resampling the FFT derived spectrum. Frequencies \( f \) were mapped to new frequencies \( G(f) \) according to the equation below (1):

\[
G(f) = \begin{cases} 
\alpha f, & 0 \leq f < f_0 \\
\frac{f_{\text{max}} - \alpha f_0}{f_{\text{max}} - f_0} (f - f_0) + \alpha f_0, & f_0 \leq f < f_{\text{max}}
\end{cases}
\]  

(3.1)

where \( \alpha \) is the warping factor, \( f_{\text{max}} \) denotes the maximum signal bandwidth, and \( f_0 \) denotes the frequency at the band-edge. Figure 3.1 shows the mapping operation.

![Figure 3.1: Frequency mapping for feature warping](image)

Resampling is performed by computing a new spectrum, where the spectral amplitude at a particular frequency bin, \( f \), is equal to the spectral amplitude at the frequency \( G^{-1}(f) \) in the original spectrum. If \( G^{-1}(f) \) does not coincide with an exact bin
frequency, linear interpolation is used to estimate the spectral amplitude from the two nearest bins. For example, if the input frequency spectrum is more wideband than the average case due to a shorter vocal tract length, choosing $\alpha$ smaller than 1 will compress the input FFT spectrum. The new warped spectrum is then used to compute MFCC features as described in Chapter 2.

### 3.2 Model-based VTLN

This section describes the model-based approach to VTLN. The motivation for the model-based approach is that it has a reduced computational burden, as compared with feature based VTLN, while maintaining a similar accuracy. In addition, no changes or additions to an existing MFCC front end need be made.

In the model-based approach, we transform MFCC means according to a transformation, which is parameterized by the warping factor so that the model better matches the incoming speech. The basic idea of the transformation is to convert the MFCC’s back to the filterbank domain, where the coefficients are warp as with feature-based VTLN. Then, the warped filterbank coefficients are converted back to MFCC for recognition.
In particular, only the means of the acoustic models are transformed back to the filterbank domain \((b)\), then the filterbank coefficients \((b)\) are warped by interpolation according to the given warping factor for warping. The warped filterbank coefficients are then transformed to the MFCC domain. This operation is defined as a “backward-forward operation”. The derivatives and accelerations MFCC are warped in similar approaches. In the following, we describe the detailed model warping transformation model warping the means of the Static, Delta and Acceleration Coefficients [20].

3.2.1 Transforming the Static Coefficients

The first step in transforming the means of the static coefficients is to compute the filterbank outputs corresponding to the static means by inverting the log and DCT operations. Any high order MFCC coefficients discarded during feature extraction are replaced by zero-padding. From these filterbank outputs, the transformation tries to predict the filterbank outputs which would result if the incoming speech were frequency warped as follows. For each filterbank \(i\),
1. Let \( f \) be the center frequency corresponding to that filterbank, and compute the frequency which would warp to the filterbank center via equation (1), \( f_{\text{warp}} = G^{-1}(f) \).

2. Let \( f_{\text{warp}} \) be the largest filterbank center frequency smaller than \( f_{\text{warp}} \) and \( f_{\text{warp}} \) be the smallest filterbank center frequency larger than \( f_{\text{warp}} \) and let \( b_{\text{warp}} \) and \( b_{\text{warp}} \) be the corresponding outputs derived from the MFCC means.

3. Use linear interpolation to calculate the new filterbank output, \( b_{\text{new}} \):

\[
b_{\text{new}} = b_{\text{warp}} \frac{f_{\text{warpl}} - f_{\text{warp}}}{f_{\text{warpl}} - f_{\text{warp}}} + b_{\text{warp}} \frac{f_{\text{warp}} - f_{\text{warp}}}{f_{\text{warp}} - f_{\text{warp}}}
\]

(3.2)

The new MFCC means are then obtained by applying the log and DCT operations to this new set of filterbank outputs.

Note that in this case, if the incoming speech is more wideband than the average, corresponding to a shorter vocal tract length, a value of \( \alpha \) larger than one will shift the model to match the incoming speech. Thus, the warping factor used in the model-based transformation is the inverse of that used in the feature-based transformation. In the following, we will reference all results to the warping factor used in the model-based transformation, with the understanding that this warping factor is inverted to apply feature-based VTLN.

### 3.2.2 Transforming the Delta and Acceleration Coefficients

The delta coefficients can be transformed in a similar manner. Note that the static MFCC mean \((m)\) is obtained from binned spectrum \((b)\) by the equation

\[
m = C \ln b
\]

(3.3)
where $C$ denotes the matrix used to compute the DCT. Differentiating both side of this equation with respect to time, we obtain the relationship between the delta MFCC coefficients and the delta filterbank outputs,

$$\Delta m = \frac{C\Delta b}{b} \quad (3.4)$$

Reorganize the terms, the equation becomes:

$$\Delta b = bC^{-1}\Delta m \quad (3.5)$$

where $C^{-1}$ is the invert DCT. The warping process applied to the delta filterbank outputs is the same as that applied to the static filterbank outputs. The new delta MFCC coefficients are obtained by applying equation (3.4).

Similarly, for the acceleration coefficients, we differentiate equation (3.3) twice to obtain

$$\Delta^2 m = C \left[ \frac{\Delta^2 b}{b} - \left( \frac{\Delta b}{b} \right)^2 \right] \quad (3.6)$$

Reorganize the terms, the equation becomes:

$$\Delta^2 b = \left[ \Delta^2 mC^{-1} + \left( \frac{\Delta b}{b} \right)^2 \right] b \quad (3.7)$$

This equation enables us to obtain the acceleration coefficients for the filterbank output, which are similarly warped and transformed back to the MFCC domain to obtain the new model parameters.

Note that the frequency information used in the model-based approach is much coarser than that used in the feature based approach. In the feature based approach, the entire FFT spectrum (e.g. 256 points) is resampled. In the model-based approach, on the filterbank outputs are resampled (e.g. 26 points). Thus, intuitively, we might
expect that the performance of the feature-based approach to be better than that of the model based approach. Surprisingly, our experimental results indicate that there is little difference in performance. From the results shown in next chapter, the performance of feature-based approach is similar to that of model-based approach.

3.3 Experimental Results

In these experiments, we are interested in comparing the capability of the two approaches in compensating for the difference in vocal tract length between the adults used in training and the children used in testing. Because the warping factor used strongly affects the performance of a VTLN algorithm, rather than trying to identify a particular warping factor, which might favor the feature or the model based approach, we present our results as a graph of accuracy versus warping factor. In particular, each point represents the accuracy resulting from choosing the same warping factor for all the data or models.

3.3.1 Experimental Setup

In the experiments, we used feature vectors with 39 elements consisting of 12 MFCC with 1 frame energy and delta and acceleration coefficients computed from 26 filterbank coefficients. Sampling rate was 20k, frame size was 400 samples and the frame shift was 8ms. The 11 digit models used were 9 state left-to-right HMM models with a single mixture component. The silence model trained had 3 states, 4 mixtures and loop back path.

To evaluate two warping methods, we built 2 sets of adult-speaker connected digit model from the TI-digits database: an unnormalized model (UM) and a normalized
model (NM). All 25 men and 25 women in the training set were used in training, 3750 utterances in total. The UM were obtained by training directly on TI-digits database.

To train the NM, we grouped the training data according to gender. Then, for each gender we applied the model warping described in the previous section to the MFCC features of the training data with a set of warping factors, from 0.88 to 1.12 with the step size 0.04. We tested the warped training data using the unnormalized model and chose the warping factor that gave the highest accuracy for each gender as a gender-dependent warping factor, and used the corresponding set of warped training data to train the normalized models. The adult models were tested on the children’s testing set of the TI-digits corpus (both 25 boys and 25 girls, totally 3750 utterances), in order to evaluate the effectiveness of the feature and model-based VTLN algorithms.
3.3.2 Evaluation of Feature and Model-based Approaches

Figure 3.3: Recognition Accuracy vs Warp Factor for Feature and Model Warping

Figure 3.4: Recognition Accuracy vs. Warp Factor for Model Warping using Normalized and Unnormalized Models
<table>
<thead>
<tr>
<th>VTLN used</th>
<th>Unnormalized Model</th>
<th>Normalized Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>92.34%</td>
<td>89.10%</td>
</tr>
<tr>
<td>Model-based</td>
<td>96.33%</td>
<td>96.59%</td>
</tr>
<tr>
<td>Feature-based</td>
<td>96.37%</td>
<td>96.62%</td>
</tr>
</tbody>
</table>

Table 3.1: The best performance (word accuracy) obtained on the test set

Figure 3.3 compares the word accuracy as a function of warping factor for the model and feature-based approaches when tested with the unnormalized models. The performance of the two approaches is nearly identical. Note that the optimal global warping factor is about 1.25, which is consistent with data [8], indicating that the third formant in children is approximately 1.25 times that of adults.

Figure 3.4 compares the performance of the model-based approach on the normalized and unnormalized models. The performance using the normalized models is clearly much more dependent upon accurate estimation of the proper warping factor, than the unnormalized models.

Table 3.1 shows that the performance of the model and the feature based approaches is approximately the same, if the best warping factor is used. There is a slight reduction in the error rate, about 7%, with the use of the normalized models. This is consistent with results reported in [5] using a feature based approach. Thus, it appears that applying VTLN can improve the performance of both feature and model-based VTLN.
3.3.3 Previous Model Warping Approach

In this section, we describe a previously proposed approach [16] to model-based VTLN and highlight the similarities and differences between the two approaches. The results of this approach and our proposed approach are compared experimentally in next section.

Similar to our proposed approach, the previous approach also performs “backward-forward” operations as described in last section, although the implementation is different. In the previous approach, the warped means ($\tilde{c}$) is warped from the mel cepstrum $c$:

$$\tilde{c} = C \cdot \log \{ T \cdot \exp(C^{-1} \cdot c) \}$$  \hspace{1cm} (3.8)

where $C$ is denoted as the DCT transform; $T$ can be expressed as the following matrix:

$$T = F \cdot W \cdot A$$  \hspace{1cm} (3.9)

where $W$ is performing warping in matrix form whose elements are:

$$w(i, j) = \begin{cases} 1, & \text{if } i = \text{round}(w \cdot j) \\ 0, & \text{otherwise} \end{cases}$$  \hspace{1cm} (3.10)

$F$ is the matrix storing the binning coefficients ($w_{i,j}$) of each filterbank channels where $i$ is the filterbank channel and $j$ is the index of power spectrum component. $A$ is a matrix such that $F \cdot A = I$ and

$$F_{n \times N_{w \cdot n}} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,N_1} & 0 & \cdots & 0 \\ 0 & \cdots & w_{2,1} & \cdots & w_{2,N_2} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & \cdots & w_{n_1,1} & \cdots & w_{n_1,N_{n_1}} \end{bmatrix}$$  \hspace{1cm} (3.11)

Thus, the means (MFCC) of models are warped in this way. The derivatives and accelerations MFCC are transformed in similar way which shown below.
The warped derivatives MFCC ($\Delta \tilde{c}$) is warped from the MFCC ($c$) and the original MFCC($\Delta c$):

$$\Delta \tilde{c} = C \cdot \log \{T \cdot \exp(C^{-1} \cdot c^{(i)})\} - C \cdot \log \{T \cdot \exp(C^{-1} \cdot (c^{(i)} - \Delta c^{(i)})\}$$  \hspace{1cm} (3.12)

For the warped accelerations MFCC ($\Delta^2 \tilde{c}$) is transformed using original MFCC, the derivatives MFCC and the accelerations MFCC ($\Delta^2 c$) as below:

$$\Delta^2 \tilde{c} = C \cdot \log \{T \cdot \exp(C^{-1} \cdot c)\} - 2 \cdot C \cdot \log \{T \cdot \exp(C^{-1} \cdot (c - \Delta c)\}$$

$$+ C \log \{T \cdot \exp(C^{-1} \cdot (c - 2\Delta c + \Delta^2 c)\}$$ \hspace{1cm} (3.13)

The similarity between two approaches is that both are performing "backward-forward" operation. It transforms the original MFCC back to the filterbank coefficients, performs warping transformation and then transforms the warped filterbank coefficients to the MFCC domain. The difference is the warping transformation. Our transformation is similar to the feature warping approach performing interpolation, estimates the warped component by two neighborhoods. The previous model warping uses equation (3.10), where the input spectrum is reconstructed by assigning all of the energy for each filterbank to the spectral bin corresponding to the center frequency.

3.3.4 Evaluation of Previous and Proposed Model-based VTLN

To evaluate these two approaches of model-based VTLN, the trained adult models were tested on the children’s testing set of the TI-digits corpus (both 25 boys and 25 girls, totally 3750 utterances), in order to evaluate the effectiveness between two approaches. As [16] mentioned, the analysis of vocal tract length of adult and children using the mean third format (F3) [11] for TI-digits database have this relationship:

$$\text{Warp factor}=F3,\text{children}/F3,\text{adults}=1.2$$ \hspace{1cm} (3.14)

This means the warp factor to transform adult models to children models is 1.2. Thus,
we apply the estimated warp factor 1.2 to both previous and proposed model-based approaches to test with the children’s testing set of the TI-digits. The results is shown as follows:

<table>
<thead>
<tr>
<th>VTLN used</th>
<th>Model warped @ 1.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>92.34%</td>
</tr>
<tr>
<td>Previous</td>
<td>96.25%</td>
</tr>
<tr>
<td>Proposed</td>
<td>96.28%</td>
</tr>
</tbody>
</table>

Table 3.2: The word accuracy obtained on the test set at warp factor=1.2

From Table 3.2 shown, the performance of both methods using warp factor 1.2 is similar. As conclusion, the warping operations both are perform in the filterbank domain and the transformations are similar to give our similar results.
Chapter 4

Warp Factor Detection

In practice, we need to have some methods to estimate the warping factor to correctly match the features and models. For feature warping, three warping factor detection methods that have been proposed are: 1. Maximum Likelihood Strategy [4], 2. Multiple-Pass Strategy [4] and 3. Gaussian Mixture Model (GMM) [12], [13], [14] and [15].

Here we propose three warping factor detection methods for model-based VTLN which are based on those proposed for feature based warping. Our experiments show that all of the methods increase recognition accuracy, but to varying degrees.

4.1 Maximum Likelihood Strategy (ML)

As described in [4], the maximum likelihood strategy estimates the best warping factor from a set of $n$ possible warping factors by warping the incoming utterance by each of the $n$ warping factors and recognizing each of the warped utterance. The warping factor whose recognition result has the highest likelihood chosen as the best, and the corresponding recognition result is output. The overview is shown in figure 4.1 (A).
A similar detection method can be applied to model-based warping, as shown in figure 4.1 (B). The input utterance is decoded with $n$ warped models. The recognition having maximum likelihood is selected as the final recognition result. At the same time, the corresponding warping factor is treated as the correct warping factor.

**4.2 Multiple-Pass Strategy (MP)**

As described in [4], the multiple-pass strategy estimates the warp factor by aligning warped utterances with respect to a hypothesized word string and selects the maximum likelihood of the warped utterances as the estimated warp factor. The
following three-step process, as illustrated in Figure 4.2 (A), is used.
Figure 4.2: MP Strategy for (A) Feature Warping and (B) Model Warping

1) From figure 4.2 (A), the unwarped utterance $X$ and the acoustic model are used to obtain a preliminary transcription of the utterance. The transcription obtained from the unwarped features is denoted as $W$.

2) $\tilde{\alpha}$ is found using equation as follows:

$$\tilde{\alpha} = \arg \max_{\alpha} \Pr(X^\alpha | \lambda, W)$$  \hspace{1cm} (4.1)

The probability is evaluated by probabilistic alignment of each warped set of feature vectors with the transcription $W$. 
3) The warped utterance is decoded with the model $\lambda^\alpha$ to obtain the final recognition result.

A similar approach can be applied to the model-based approach. The main idea is selecting the maximum likelihood of the input utterances with the warped acoustic models as the estimated warp factor given the transcription $W$. The following three-step process, as illustrated in figure 4.2 (B), is used.

1) The input utterance $X$ and the unwarped acoustic model are used to obtain a preliminary transcription of the utterance. The transcription obtained is denoted as $W$.

2) $\bar{\alpha}$ is found using equation as follows:

$$\bar{\alpha} = \arg \max_{\alpha} \Pr(X \mid \lambda^\alpha, W)$$

(4.2)

The probability is evaluated by probabilistic alignment of each warped set of model with the transcription $W$.

2) The input utterance is decoded with the warped model $\lambda^\alpha$ to obtain the final recognition result.

4.3 Gaussian Mixture Model Strategy (GMM)

In the following, we describe a fast method for scale selection in recognition which does not require a preliminary transcription [12], [13], [14] and [15]. The method is based on a Gaussian mixture model that represents the distribution of the feature vectors. As described in [15], the model of generic voiced speech is used to select the warp scale for each speaker. This model is a single state, consisting of a mixture of
256 multivariate Gaussians.

Intuitively, the feature space distributions of untransformed speech from the different classes of speakers vary due to the acoustic differences of speech produced by vocal tracts of different lengths. Therefore, if statistical models of the feature space distribution of each class are available, it may be possible to determine the warping factor by finding out which class distribution is most likely to have generated the given sequence of feature vectors.
Figure 4.3: GMM Strategy for (A) Feature Warping and (B) Model Warping

After a model of generic voiced speech is trained, we use this model to estimate the vocal tract length of speakers based on the acoustic data by selecting the maximum
likelihood of the warped feature set with respect to the given statistical model. During recognition shown in figure 4.3 (A), the probability of the incoming utterance after frequency warping is evaluated against the generic model, and the warping factor is chosen for the warping that yields the highest likelihood over other warping factors. Then, that warped utterance is used for second pass recognition.

A similar approach can be applied to the model-based approach shown in figure 4.3 (B). Using same GMM (G), we warp the GMM model to estimate the vocal tract length of speakers. To evaluate the warping factor, the probability of the incoming utterance is evaluated against each warped generic model, and the warping factor is chosen for the warping that yields the highest likelihood over other warping factors.

$$\bar{\alpha} = \arg\max_{\alpha} \Pr(X | G^\alpha)$$  \hspace{1cm} (4.3)

where the $G$ is the generic model. Then, the acoustic model will be warped according to that estimated warp factor and will be used for second pass recognition. The process is illustrated in Figure 4.3 (B).

### 4.4 Experimental Results

In this section, the three warping factor detection methods described are applied to our proposed model-based VTLN. Those approaches use the same acoustic adult models described before for recognition. The Gaussian Mixture model is trained with all of the utterances in adult training set of the TI-digits database and consists a mixture of 256 multivariate Gaussians to represent the feature space distribution of each class.

The three approaches with the adult models are tested with the 3750’s adult utterances and the 3750 children’s utterances in testing set of the TI-digits. Then, the models are
warped by the warp factor from 0.9 to 1.3 with the step size 0.025, totally 17 warp factors. The results are shown as follows:

<table>
<thead>
<tr>
<th>Warp Factor Detection used</th>
<th>Performance of Adult utterances</th>
<th>Performance of Children utterances</th>
<th>Word Error Rate Reduction (Adult+Children)</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>98.36%</td>
<td>92.34%</td>
<td>0 %</td>
</tr>
<tr>
<td>ML</td>
<td>98.45%</td>
<td>96.24%</td>
<td>28.21%</td>
</tr>
<tr>
<td>Multiple-Pass</td>
<td>98.44%</td>
<td>96.00%</td>
<td>26.34%</td>
</tr>
<tr>
<td>GMM</td>
<td>98.36%</td>
<td>95.40%</td>
<td>19.99%</td>
</tr>
<tr>
<td>Best (Ideal case)</td>
<td>99.39%</td>
<td>98.14%</td>
<td>69.26%</td>
</tr>
</tbody>
</table>

Table 4.1: Performance of three warp factor detection approaches

The results for the ideal case are obtained by testing each input utterance with the 17 warped models, and selecting warping factor with the highest recognition accuracy as the correct warping factor. This is the ideal case, as the warping factor is calculated by comparison with the transcription of the input utterance, but which is not known in practice. This result is given for reference, as it indicates the upper bound on the performance of the warping factor identification schemes.

Comparing the complexity of 3 methods, GMM is the simplest and the ML is the most computationally complex, as measured by the numbers of forced alignments and recognition passes required. Forced alignment measures the likelihood of the models with respect to the input utterance. However, recognition decodes the whole input utterance into word transcription. Thus, the computational cost of recognition is higher than forced alignment.

GMM is the fastest method for warp factor selection in recognition because it does not decode the whole utterance, but it only tests the likelihood of the models with the input utterance.
Multiple-Pass Strategy is the middle one because it requires running the recognizer to obtain a preliminary transcription of the utterance and the transcription is used to perform the "forced alignment". This forced alignment is more complex than the likelihood evaluation by the Gaussian Mixture Model.

ML is the most complex because it needs to run the recognizer for every warped model to decoded the input utterances and then select the highest likelihood over the phoneme models.

Table 4.2 shows the number of recognition and likelihood testing used for the 3 methods if we use the 17 warp factors in the range from 0.9-1.3 described before. The number of recognition used in ML Strategy is much higher than others, so its computational cost is the highest.

<table>
<thead>
<tr>
<th>Warp Factor Detection used</th>
<th>No. of recognition</th>
<th>No. of likelihood testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>Multiple-Pass</td>
<td>2</td>
<td>17</td>
</tr>
<tr>
<td>GMM</td>
<td>1</td>
<td>17</td>
</tr>
</tbody>
</table>

Table 4.2: No. of operation for the 3 warp factor detection approaches

As the table 4.1 and table 4.2, the recognition accuracy of model warping has a direct relationship to the complexity. ML Strategy has the highest accuracy, but it is the most complex approach; GMM Strategy has the simplest approach, but the accuracy is lower than those complex approaches. Thus, selecting which detection approach is trade off between the recognition accuracy and the computational operations.

As a conclusion, our results show that warp factor detection methods applied for feature based warping can also be applied to model warping. With our proposed
detection methods, we can select either using feature warping or model warping system because the result shown in last chapter, they are nearly the same. Selecting model warping has the advantage that the front end feature extractor does not need to be changed. In addition, our results in the next chapter show that model warping is computationally less complex.
Chapter 5

Feature vs Model based VTLN in Computational Cost

Chapter 3 showed that the recognition accuracy of feature based VTLN and model based VTLN is similar. In this chapter, we are interested to compare the computational cost of the feature and model based approaches under two assumptions regarding the amount of system memory available. We first list our system configurations used in our comparison. Next, we compare the computational cost of both systems with warp factor detection. First, we assume the system have infinite memory space and compare their overall computational cost. Then, the computational costs of the systems with limited memory space are compared. Results of comparison show that the model warping is more computationally efficient than the feature warping if the system have infinite memory space or the input utterance is long enough.

5.1 System Configuration

To compare both systems, we mention the settings in feature based and model based VTLN systems for the comparison of memory and operations usage: 1. Input Signal and Feature Vector, 2. Configuration of Acoustic Model and 3. Warp Factor Detection.

5.1.1 Input Signal and Feature Vector

The analog speech signal is converted to digital form for analysis. Sampling rate of A/D converter used is 20k. Frame size is 400 samples and the frame shift is 160
samples. These parameters limit the maximum analysis time per frame to be

\[(1/20k)\times 160=8\times 10^{-3}\text{ seconds}\]  \hspace{1cm} (5.1)

This is equivalent to 125 frames/second. The processing for each frame must be finished within \(8\times 10^{-3}\) seconds in order for the whole speech recognition processes to perform in real time.

After the speech is converted to the digital form, feature extraction is performed and the digital speech signal is converted into feature vectors. This means each frame has 160 new samples, padded with the past data to form a frame which has 400 samples, then these 400 samples are extracted to be 13 MFCC feature vectors/frame after feature extraction.

5.1.2 Configuration of Acoustic Model

There are totally 46 phoneme models, where each model contains 3 stages and each state output probability is a single Gaussian mixture. Thus, each state contains 39 means, 39 variances and 6 transition probabilities. Thus, the total memory used to store one set of phoneme models is:

\[\text{Total memory used}=46\times((39+39)\times3+6)=11040\text{ units}\]  \hspace{1cm} (5.2)

5.1.3 Warp Factor Detection

We use the Maximum Likelihood Strategy for all comparison, because it gives the highest recognition accuracy. The comparison of using other 2 strategy will be similar because those configurations of the VTLN systems are quite similar.
5.2 Feature warping vs Model Warping with infinite memory space

In this section, we assume the systems have infinite memory space so that the warped models can be precalculated and stored.

The feature based VTLN performs warping in the feature extractor. The feature based VTLN warps the features at the stage between calculating power spectrum and the binning process.

![Feature Warping Procedure Diagram]

\[ \text{define} \quad \text{"Forward"} = \text{Lifting} \rightarrow \text{DCT} \rightarrow \text{Log} \rightarrow \text{Mel-scaled Filterbanks} \rightarrow \text{Warping} \]

\[ \text{Figure 5.1: Feature Warping Procedures} \]

Figure 5.1 shows the procedures of feature warping. To minimize the computational cost of warping, we assume the data after power spectrum is saved to prevent
repeating operations before this block.

The warping is done by interpolation which has been described in chapter 3. After estimating the new magnitude of each components of power spectrum, the remaining steps are binning, DCT and log, which we defined previously as the “forward” operation. If there are 17 warp factors used, the “forward” operation must be performed 17 times. The total computational cost is one feature extraction, sixteen forward operations for warping and seventeen Viterbi decodings.

The model based VTLN performs warping on the phoneme models. Because the system is assumed to have enough memory space, we can store the pre-warped models into the memory. During Viterbi stage, the pre-calculated models are loaded by table look-up method for recognition. As the memory usage to store 1 set of phoneme model using single mixture are 11,040 units, the total memory usage to store 17 set of warped model is 187,680 units. Table 5.1 concludes the computational cost of both cases.

<table>
<thead>
<tr>
<th>Warping</th>
<th>No. of Feature Extraction</th>
<th>No. of “Forward” Operation for warping</th>
<th>No. of Viterbi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature</td>
<td>1</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>Model</td>
<td>1</td>
<td>0</td>
<td>17</td>
</tr>
</tbody>
</table>

Table 5.1: Overall Computational Cost for Feature and Model Warping

From the table 5.1 shown, we see the different between feature and model warping is feature warping need 16 extra forward operations for warping. It is a large cost for the feature warping because every input frame performs 16 forward operations more than the model warping. Thus, model warping is better than the feature warping as its lower computational cost with the help of the pre-calculated warped models.
5.3 Feature warping vs Model Warping with limited memory space

In this section, we assume the systems do not have enough memory to store all of the warped models, so that we need to warp the models on line. Thus, we focus on the computational cost of warping the phoneme models. First, approximate the computational cost of feature warping and model warping. We count each addition or multiplication as one floating point operation (flop) using the operation equations. Then, we use Matlab to simulate the result.

The system setting of feature warping is described in section 5.2. It stores the power spectrum data and then performs the “forward” warping processes in the feature extraction. In the forward process, the binning coefficients are pre-calculated to save the computational cost. We assume 512 FFT, 26 filterbanks and 39 features are used. The number of operations used is approximated by their equations. Table 5.2 shows the approximated number of floating point operations needed in one frame feature warping.

<table>
<thead>
<tr>
<th>Operation</th>
<th>No. of Addition</th>
<th>No. of Multiplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) 256 Warping</td>
<td>8*256</td>
<td>6*256</td>
</tr>
<tr>
<td>(2) Binning</td>
<td>256*2</td>
<td>256*2</td>
</tr>
<tr>
<td>(3) Log</td>
<td>1*26</td>
<td>8*26</td>
</tr>
<tr>
<td>(4) DCT</td>
<td>13*26</td>
<td>13*26</td>
</tr>
<tr>
<td>(5) Lifting</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>(6) Derivatives</td>
<td>13*5</td>
<td>13*5</td>
</tr>
<tr>
<td>(7) Accelerations</td>
<td>13*5</td>
<td>13*5</td>
</tr>
<tr>
<td>Total:</td>
<td>3054</td>
<td>2737</td>
</tr>
</tbody>
</table>

Table 5.2: Approximated no. of flops for 1 frame feature warping

One feature warping contains 256 warping for each power spectrum coefficient using equation 3.1. Binning converts the 256 warped coefficients to 26 filterbank
coefficients. Each filterbank overlaps another, so we count 2 times more operations in both cases. We assume using Fast Log2 with N=128 described in section 2.2.2.5, so it needs 1 addition and 8 multiplications. DCT compresses 26 coefficients to 13 MFCC, so we assume each DCT has 13*26 additions and multiplications. Then the 13 derivatives MFCC and 13 accelerations MFCC are calculated from 13 MFCC by equation (5.3) where \( c_i \) is MFCC and derivatives MFCC at particular time frame respectively.

\[
\Delta c_i(i) = \sum_{k=-K}^{K} k c_{i+k}(i)
\]  

(5.3)

K=2 is used in our case, this means one derivatives or accelerations MFCC needs five coefficients in each calculation. Thus, overall computational cost for one feature warping is 3054+2737 = 5791 flops.

The model warping transforms the 39 means of acoustic model including 13 MFCC, 13 derivatives MFCC and 13 accelerations MFCC. The transformation is called “backward-forward” operation described in Chapter 3. We warp the original acoustic model to form the warped models. 13 MFCC take forward operation of model warping containing invert liftering, IDCT and exp and each backward operation contains log, DCT and liftering which is shown in figure 3.2. Table 5.3 shows the approximated number of flops needed in 13 MFCC warping.

<table>
<thead>
<tr>
<th>Operation</th>
<th>No. of Addition</th>
<th>No. of Multiplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>(8) inv. Lifting</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>(9) IDCT</td>
<td>13*26</td>
<td>13*26</td>
</tr>
<tr>
<td>(10) exp</td>
<td>10*26</td>
<td>10<em>2</em>26</td>
</tr>
<tr>
<td>(11) 26 Warping</td>
<td>8*26</td>
<td>6*26</td>
</tr>
<tr>
<td>(3) Log</td>
<td>1*26</td>
<td>8*26</td>
</tr>
<tr>
<td>(4) DCT</td>
<td>13*26</td>
<td>13*26</td>
</tr>
<tr>
<td>(5) Liftering</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>Total:</td>
<td>1170</td>
<td>1586</td>
</tr>
</tbody>
</table>

Table 5.3: Approximated no. of flops for 13 MFCC model warping
There are 26 coefficients taken exponential function and it is approximated by using
Taylor series with n=10 in our case.

\[ e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} \]  \hspace{1cm} (5.4)

Besides, the approximated number of addition and multiplication of warping 13
derivatives MFCC and 13 accelerations MFCC is counted according to the equations
3.4-3.7. Table 5.4 and Table 5.5 show the approximated warping operations of
derivatives MFCC and that of accelerations MFCC respectively.

<table>
<thead>
<tr>
<th>Operation</th>
<th>No. of Addition</th>
<th>No. of Multiplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>(8) inv. Lifting</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>(9) IDCT</td>
<td>13*26</td>
<td>13*26</td>
</tr>
<tr>
<td>(11) 26 Warping</td>
<td>8*26</td>
<td>6*26</td>
</tr>
<tr>
<td>(4) DCT</td>
<td>13*26</td>
<td>13*26</td>
</tr>
<tr>
<td>(5) Lifting</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>(12) Others</td>
<td>0</td>
<td>13*2</td>
</tr>
<tr>
<td>Total:</td>
<td>884</td>
<td>884</td>
</tr>
</tbody>
</table>

Table 5.4: Approximated no. of flops for 13 derivatives MFCC model warping

<table>
<thead>
<tr>
<th>Operation</th>
<th>No. of Addition</th>
<th>No. of Multiplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>(8) inv. Lifting</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>(9) IDCT</td>
<td>13*26</td>
<td>13*26</td>
</tr>
<tr>
<td>(11) 26 Warping</td>
<td>8*26</td>
<td>6*26</td>
</tr>
<tr>
<td>(4) DCT</td>
<td>13*26</td>
<td>13*26</td>
</tr>
<tr>
<td>(5) Lifting</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>(13) Others</td>
<td>13*2</td>
<td>3<em>13</em>2</td>
</tr>
<tr>
<td>Total:</td>
<td>910</td>
<td>936</td>
</tr>
</tbody>
</table>

Table 5.5: Approximated no. of flops for 13 accelerations MFCC model warping

The “Others” in both tables are approximated depending on the number of addition
and multiplication in the equations. Thus, the total operations taken by one model
warping is 1170+1586+884+884+910+936 = 6370 flops. Thus, the ratio of the
computation cost of model warping to that of feature warping is 6370/5791 = 1.10.

To prove this theoretical ratio, we use Matlab to count the flops using same
configuration. We run the feature warping and model warping 10,000 times with the
random warp factors selected range from 0.9 to 1.3. The results show that the feature
warping contains 91,244,788 flops and the model warping contains 260,109,088 flops.
Thus, the ratio of the computation cost of model warping to that of feature warping is
260,109,088/91,244,788 = 2.85.

The ratio of the simulation result is larger than that of the theoretical result. The
different is mainly due to Matlab including the calculation of coefficient indexing, but
the theoretical approximation ignoring it. Thus, we use the simulation result for the
comparison of computational cost.

The computational cost of feature warping is not fixed, as it depends on the length of
input utterance. If we define the sampling rate be \(a\) and assume a 160 sample frame
shift, the number of frame processed per second for feature warping is:

\[
\frac{\text{sampling rate}}{160} \text{ frames/second} = \frac{a}{160} \text{ frames/second} \tag{5.5}
\]

We also define the number of phoneme models to be \(p\) with \(s\) states and \(m\) mixtures
and the number of warp factors used to be \(w\). To approximate the length of input
utterances to be computed is equivalent to the computational cost of model warping,
the calculation is:

\[
\frac{\text{Total cost of model warping}}{\text{Total cost of feature warping/frame}} = \frac{2.85 psmw}{(a/160)w} \text{ seconds} \tag{5.6}
\]

In our case, the sampling rate is 20k and there are 46 phoneme models with 3 states
and 1 mixture. Thus,

\[
\frac{2.85 \times 46 \times 3 \times 1 \times w}{(20000/160)w} = 3.146 \text{ seconds} \tag{5.7}
\]

This means the computational cost of model warping is less than the feature warping
if the length of input utterance is longer than 3.146 seconds.
Chapter 6

Conclusion

In this thesis, we have presented the implementation of speech recognition using Digital Signal Processor (DSP). We have described the architecture of DSP and some algorithms which can help the speech recognition able to be run by DSP.

The speech recognition system must respond to users with a wide range of ages. We have proposed a model based VTLN approach to addressing this issue. First, we compared feature and model-based approaches to VTLN. Our results indicate that despite the coarser frequency resolution used in the transformation, the model-based approach achieves performance which is essentially identical with that obtained using a feature-based approach. Our results also indicate that the performance of the model-based approach can be improved by reducing the variability of input utterances by applying VTLN to the training data. Best performance of the feature based and model based approach occurred at the same warping factor.

We also compared our proposed approach to one proposed previously. Our results indicate that both performances are similar at the selected warp factor 1.2. This shows that our proposed method is competitive with the previous method.

After compared the two model warping approaches, we investigated three proposed warp factor detection methods, based on methods used for the feature based approach. The results show that the more complex system is used, better performance is archived.
Finally, we compared the computational cost of the model warping with the feature warping using two systems, one which has unlimited memory and another one which has limited memory. With unlimited memory space, the system can perform the model warping faster as the warped models can be precalculated and stored. With limited memory space, the computation cost of model warping is equivalent to about 3.1 seconds for performing feature warping in our case. If the input utterance is longer than that value, the model warping is less computationally intensive. In addition, the model based VTLN has the advantage that the front-end feature extraction need not be modified.
APPENDIX A

Program Code for DSP Implementation of Speech Recognizer

Our system contains two parts: PC program and DSP speech recognizer. The PC program provides a Graphical User Interface for inputting active keyword being recognized by DSP. The steps are following:

1. Input the keyword.
2. The keyword is then put into Word List.
3. The phoneme sequence of word list is transformed by lookup the database.
4. The button “DSP Process” is pressed, then the compiled DSP program code, the phoneme sequence (Ph_seq.txt) and models (Hmm.txt) are downloaded to DSP for speech recognition.

![Image: Graphical User Interface of PC Program]

Figure 1: Graphical User Interface of PC Program
After receiving the phoneme sequence and the program code, DSP concatenates the keyword models and performs the speech recognition. The following section gives the architectural view of the program design for the speech recognizer. The system block diagram is shown below:

![Diagram](image)

*Figure II: Program Flow of DSP Implementation*

Shown in figure 2, the system is divided into two parts: Feature Extraction and Viterbi. The Feature Extraction is performed in real time. The MFCC feature vectors of each frame are stored into memory. Then, the batch conversion of $\Delta MFCC$, $\Delta^2 MFCC$, and Viterbi is run after the real time feature extraction.

The program codes are built according to each block diagram:

Aic.asm - contains the program code to control the A/D converter.

Cck.asm - main program defines all the memory location and the feature extraction
code.

Ph_seq.txt - phoneme sequence generated by PC program

Hmm.txt - phoneme models

Hamm.asm - contains the coefficients for pre-emphasis and hamming windows

Fft_sin.asm - contains the coefficients for FFT operation

Lowt.asm - contains the coefficients for binning in filterbank

F_bank.asm - contains the coefficients for DCT

Delta.asm - contains the program to generate $\Delta MFCC$

Caldelta.asm - contains the program code to generate $\Delta^2 MFCC$

Viterbi.asm - contains the Viterbi program code

Mean.asm - contains the mean of the phone models

Cov.asm - contains the covariances of the phone models

Transp.asm - contains the transition probabilities of the phone models
B_garb.asm - contains mean, covariances and transition probabilities of filler model

Word_seq.asm - contains the word sequences for dynamic keyword models generation from phones models

Log.asm - contains the program code to perform fast log

Invf_exp.asm - contains the program code to perform exp(-x)

Stack.asm - defined the memory address of stack operation
Bibliography


