Speech Recognition Using
Features Extracted from Phase Space Reconstructions

by
Andrew Carl Lindgren, B.S.

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
for the degree of
MASTER OF SCIENCE
Field of Electrical and Computer Engineering

Marquette University
Milwaukee, Wisconsin
May 2003
Preface

A novel method for speech recognition is presented, utilizing nonlinear/chaotic signal processing techniques to extract time-domain based, reconstructed phase space derived features. By exploiting the theoretical results derived in nonlinear dynamics, a distinct signal processing space called a reconstructed phase space can be generated where salient features (the natural distribution and trajectory of the attractor) can be extracted for speech recognition. These nonlinear methodologies differ strongly from the traditional linear signal processing techniques typically employed for speech recognition. To discover the discriminatory strength of these reconstructed phase space derived features, isolated phoneme classification experiments are executed using the TIMIT corpus and are compared to a baseline classifier that uses Mel frequency cepstral coefficient features, which are the typical benchmark. Statistical methods are implemented to model these features, e.g. Gaussian Mixture Models and Hidden Markov Models. The results demonstrate that reconstructed phase space derived features contain substantial discriminatory power, even though the Mel frequency cepstral coefficient features outperformed them on direct comparisons. When the two feature sets are combined, improvement is made over the baseline, suggesting that the features extracted using the nonlinear techniques contain different discriminatory information than the features extracted from linear approaches. These nonlinear methods are particularly interesting, because they attack the speech recognition problem in a radically different manner and are an attractive research opportunity for improved speech recognition accuracy.
Acknowledgments

There have been several people who encouraged, supported, and gave invaluable advice to me during my graduate work. First, I thank my wife, Amy, for her love and support the last two years. I appreciate the encouragement that my parents, in-laws, and friends have given me. I would like to acknowledge my thesis committee: Mike Johnson, Richard Povinelli, and James Heinen. I especially want to thank my advisor, Mike Johnson, for his guidance and solid mentoring. Also, Richard Povinelli, for our vast and tireless discussions on the topic of this research. Finally, I am grateful to the members of the ITR group, the Speech and Signal Processing Laboratory, and the KID Laboratory at Marquette for our awesome, thought provoking debates.
# Table of Contents

LIST OF FIGURES .............................................................................................................. VIII

LIST OF TABLES ................................................................................................................ IX

LIST OF ACRONYMS ........................................................................................................ X

1. INTRODUCTION .............................................................................................................. 1

   1.1 OVERVIEW OF SPEECH RECOGNITION TECHNOLOGY AND RESEARCH ............. 1

       1.1.1 Speech processing for recognition ................................................................. 1

       1.1.2 Historical background of ASR ................................................................. 5

       1.1.3 Current research in ASR ........................................................................... 6

   1.2 DESCRIPTION OF RESEARCH ............................................................................. 6

       1.2.1 Problem statement ...................................................................................... 6

       1.2.2 Speech and nonlinear dynamics ............................................................... 7

       1.2.3 Previous work in speech and nonlinear dynamics ....................................... 9

       1.2.4 Summary .................................................................................................. 10

2. BACKGROUND ............................................................................................................. 13

   2.1 CEPSTRAL ANALYSIS FOR FEATURE EXTRACTION ......................................... 13

       2.1.1 Introduction to cepstral analysis ................................................................. 14

       2.1.2 Mel frequency Cepstral Coefficients (MFCCs) ............................................. 17

       2.1.3 Energy, deltas, and, delta-deltas ............................................................... 18

       2.1.4 Assembling the feature vector ................................................................... 19

   2.2 STATISTICAL MODELS FOR PATTERN RECOGNITION ....................................... 20

       2.2.1 Gaussian Mixture Models ....................................................................... 20
2.2.2 Hidden Markov Models

2.2.3 Practical issues

2.3 SUMMARY

3. NONLINEAR SIGNAL PROCESSING METHODS

3.1 Theoretical background and Takens’ theorem

3.2 Features derived from the RPS

3.2.1 The natural distribution

3.2.2 Trajectory information

3.2.3 Joint feature vector

3.3 Modeling technique and classifier

3.3.1 Modeling the RPS derived features

3.3.2 Modeling of the joint feature vector

3.3.3 Classifier

3.4 SUMMARY

4. EXPERIMENTAL SETUP AND RESULTS

4.1 Software

4.2 Data

4.3 Time lags and embedding dimension

4.4 Number of mixtures

4.5 Baseline systems

4.6 Direct comparisons

4.7 Stream weights

4.8 Joint feature vector comparison
4.8.1 Accuracy results ................................................................. 53
4.8.2 Statistical tests ................................................................. 53
4.9 SUMMARY ..................................................................................... 54

5. DISCUSSION, FUTURE WORK, AND CONCLUSIONS ............... 55

5.1 DISCUSSION ............................................................................... 55
5.2 KNOWN ISSUES AND FUTURE WORK .................................... 56
  5.2.1 Feature extraction ................................................................. 56
  5.2.2 Pattern recognition ............................................................... 58
  5.2.3 Continuous speech recognition ............................................. 59
5.3 CONCLUSIONS ........................................................................... 59

6. BIBLIOGRAPHY AND REFERENCES ........................................... 61

APPENDIX A – CONFUSION MATRICES ........................................... 66

APPENDIX B – CODE EXAMPLES ..................................................... 71
List of Figures

Figure 1: Block diagram of speech production mechanism ........................................... 2
Figure 2: Block diagram of ASR system ................................................................. 4
Figure 3: Example of reconstructed phase space for a speech phoneme ............... 8
Figure 4: Block diagram of speech production model ............................................. 14
Figure 5: Block diagram of the source-filter model ............................................... 15
Figure 6: Cepstral analysis for a speech phoneme .................................................. 16
Figure 7: Hamming window .................................................................................... 17
Figure 8: Block diagram of feature vector computation .......................................... 20
Figure 9: Visualization of a GMM for randomly generated data .......................... 23
Figure 10: Diagram of HMM with GMM state distributions ............................... 26
Figure 11: Two dimensional reconstructed phase spaces of speech phonemes ...... 31
Figure 12: RPS of a typical speech phoneme demonstrating the natural distribution and the trajectory information ................................................................. 35
Figure 13: Relationship between indices for the joint feature vector ....... 37
Figure 14: GMM modeling of the RPS derived features for the phoneme '/aa/' ...... 38
Figure 15: Time lag comparison in RPS ............................................................... 46
Figure 16: False nearest neighbors for 500 random speech phonemes .................. 47
Figure 17: Classification accuracy vs. number of mixtures for RPS derived features .... 49
Figure 18: Accuracy vs. stream weight ............................................................... 52
## List of Tables

Table 1: List of notation used in section 2.1 ................................................................. 13

Table 2: Notation used for section 2.2 ........................................................................ 20

Table 3: Notation for Chapter 3 .................................................................................. 29

Table 4: Notation used in Chapter 4 .......................................................................... 42

Table 5: List of phonemes, folding and statistics ....................................................... 45

Table 6: Direct performance comparison of the feature sets ..................................... 51

Table 7: Comparisons for different stream weights ...................................................... 53

Table 9: Example of a confusion matrix ...................................................................... 66

Table 10: Confusion matrix for $x^{(5,6)}_a$ .................................................................... 67

Table 11: Confusion matrix for $x^{(10,6)}_a$ .................................................................. 67

Table 12: Confusion matrix for $x^{(5,6,6)}_a$ ................................................................. 68

Table 13: Confusion matrix for $x^{(5,6,\Delta)}_a$ ............................................................. 68

Table 14: Confusion matrix for $c_i$ .............................................................................. 69

Table 15: Confusion matrix for $O_i$ ............................................................................ 69

Table 16: Confusion matrix for $y_{u_i, \rho = 0}$ ............................................................... 70

Table 17: Confusion matrix for $y_{u_i, \rho = 0.25}$ .......................................................... 70
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>Artificial Neural Networks</td>
</tr>
<tr>
<td>ASR</td>
<td>Automatic Speech Recognition</td>
</tr>
<tr>
<td>DCT</td>
<td>Discrete Cosine Transform</td>
</tr>
<tr>
<td>DFT</td>
<td>Discrete Fourier Transform</td>
</tr>
<tr>
<td>DTW</td>
<td>Dynamic Time Warping</td>
</tr>
<tr>
<td>EM</td>
<td>Expectation Maximization</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>GMM</td>
<td>Gaussian Mixture Model</td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Markov Model</td>
</tr>
<tr>
<td>HTK</td>
<td>Hidden Markov Modeling Toolkit</td>
</tr>
<tr>
<td>IDFT</td>
<td>Inverse Discrete Fourier Transform</td>
</tr>
<tr>
<td>LPC</td>
<td>Linear Predicative Coding</td>
</tr>
<tr>
<td>MFCC</td>
<td>Mel Frequency Cepstral Coefficients</td>
</tr>
<tr>
<td>PDF</td>
<td>Probability Density Function</td>
</tr>
<tr>
<td>RPS</td>
<td>Reconstructed Phase Space</td>
</tr>
<tr>
<td>TIMIT</td>
<td>Texas Instruments &amp; Massachusetts Institute of Technology speech corpus</td>
</tr>
<tr>
<td>Z</td>
<td>Z Transform</td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Overview of Speech Recognition Technology and Research

1.1.1 Speech processing for recognition

Speech processing is the mathematical analysis and application of electrical signals for information storage and/or retrieval that results from the transduction of acoustic pressure waves gathered from human vocalizations. The teleology of speech processing generally can be subdivided into the broad overlapping categories of speech analysis, coding, enhancement, synthesis and recognition. Speech analysis is the study of the speech production mechanism in order to generate a mathematical model of the physical phenomena. The study of coding endeavors to store speech information for subsequent recovery. Enhancement, in contrast, is the process of improving the intelligibility and quality of noise-corrupted speech signals. The generation of speech from coded instructions is known as speech synthesis. Speech recognition, though, is the process of synthesis in reverse; namely, given a speech signal, produce the code that generated it. Speech recognition is the task that will be addressed in this work.

For the vast majority of applications, the code that speech recognition systems strive to identify is the units of language, usually phonemes (set of unique sound categories for a language) or words, which can be compiled into text [1]. Speech recognition can then be formulated as a decoding problem, where the goal is to map speech signals to their underlying textual representations. The potential applications of automatic speech recognition (ASR) systems include computer speech-to-text dictation, automatic call routing, and machine language translation. ASR is a multi-disciplinary area that draws theoretical knowledge from mathematics, physics, engineering, and social science. Specific topics
include signal processing, information theory, random processes, machine learning / pattern recognition, psychoacoustics, and linguistics.

The development of speech recognition systems begins with the processing of a given speech signal for the purposes of obtaining the discriminatory acoustic information about that utterance. The discriminatory information is represented by computed numerical quantities called features. Current speech processing techniques are based on modeling speech as a linear stochastic process [2]. The underlying speech production model is what is known as a source-filter model, where the vocal tract is excited by a series of pitch pulses (e.g. vowels) or Gaussian white noise (e.g. fricatives), which ultimately results in certain frequency spectra that humans recognize as sounds [2].

![Figure 1: Block diagram of speech production mechanism](image-url)
Features are extracted with the goal of capturing the characteristic frequency spectrum of a particular phoneme. One feature set typically used in ASR are cepstral coefficients (usually Mel frequency cepstral coefficients) [2]. Using the acoustic features, speech recognition algorithms are employed to accomplish the undertaking. The process starts at the phoneme or acoustic level where consecutive, overlapping, short portions (called frames) of speech (~10-30 ms) are assumed to be approximately stationary, and frequency domain-based features are extracted such as Mel frequency cepstral coefficients (MFCCs) [2].

Following feature extraction, machine learning techniques are employed to use the features for recognition. There are many machine learning algorithms that can be utilized for the speech recognition task. A common approach is to use statistical pattern recognition methodologies.

In this paradigm, a probability density function (PDF) of these features is generated from the training examples [2-5]. The PDF is usually a continuous, parametric function that is a sum of weighted Gaussian functions, which is called a Gaussian Mixture Model (GMM). In order to track the rapid changes of the vocal tract as a human utters several sounds (phonemes) consecutively, another model must be employed called a Hidden Markov Model (HMM). A HMM is a finite state machine where each state has an associated GMM which represents one possible configuration of the vocal tract. HMMs also have parameters called transition probabilities that represent the likelihood of transitioning from one vocal tract configuration to another. The parameters of these statistical models are learned or optimized using training data.
After model training is complete, previously unseen test data is presented to the recognizer. The recognizer outputs the most likely language units associated with that test data given the trained models. A more detailed and mathematically rigorous explanation of ASR systems will be given in subsequent chapters.

The pattern recognition algorithms utilized must be robust to the inherent variabilities of speech such as changing speech rates, coarticulation effects, different speakers, and large vocabulary sizes[1, 2]. Other knowledge sources that have discriminatory power, in addition to the acoustic data, can also be incorporated into the recognizer such as language models. The final output of the system can be the single phoneme, a sequence of phonemes, an isolated word, an entire sentence, etc. depending on the specific application and problem domain.

![Figure 2: Block diagram of ASR system](image-url)
1.1.2 Historical background of ASR

The first speech recognition system was built at Bell Labs in the early 1950’s [1]. The task of the system was to recognize digits separated by pauses spoken from a single speaker. The system was built using analog electronics and it performed recognition by detecting the resonant frequency peaks (called formants) of the uttered digit. Despite its crudeness, the system was able to achieve 98% accuracy, and it proved the concept that a machine could recognize human speech [1].

Research continued into the 1960’s and 1970’s fueled by the advent of digital computing technology with focus both on the signal processing and the pattern recognition aspects of developing a speech recognizer. The most significant contributions to speech analysis included to the development of the Fast Fourier Transform (FFT), cepstral analysis, and linear predictive coding (LPC), which replaced the antiquated method of filter banks used in the past. Pattern recognition algorithms such as artificial neural networks (ANN), dynamic time warping (DTW), and Hidden Markov Models (HMM) were successfully applied to speech. These methodologies resulted in several functional and useful systems for a variety of applications [1].

In the 1980’s and early 1990’s, research focused on expanding the capabilities of ASR systems to more complex tasks including speaker independent data sets, larger vocabularies, and noise robust recognition. Progress was made by incorporating speech-tailored signal processing methods for feature extraction such as Mel frequency cepstral coefficients (MFCC). HMM methods were further refined and became the prevailing ASR technology. Also, the research community became more organized for facilitation of data and software sharing so that comparisons among researchers could be made.
Standard speech data was compiled and distributed such as the TIMIT corpus and the Resource Management corpus. Also, speech software toolkits with open source code were made available; the Hidden Markov Modeling Toolkit (HTK) being one example [5]. The period was punctuated by the release of several commercial products; the most famous being personal computer dictation systems including IBM ViaVoice© and Dragon Systems Naturally Speaking© [1].

1.1.3 Current research in ASR

Current research efforts are focused on several different task dependent categories as well as on the different aspects of ASR systems. The task complexities include speaker independent data sets, large vocabulary recognition, spontaneous speech recognition, noise robust recognition, and speaker verification / identification. Additionally, new methodologies have been under investigation, such as rapid speaker adaptation, discriminative training techniques, vocal tract normalization, multi-modal ASR, and improved language modeling. Despite the increased popularity of ASR as a research area, progress over the last 10-15 years has been largely incremental. Most of the ASR literature that describes and implements new methods report small improvements (< 3% increased accuracy) over baseline recognizers on standard corpora.

1.2 Description of Research

1.2.1 Problem statement

The problem addressed in this thesis concerns the investigation of novel acoustic modeling techniques that exploit the theoretical results of nonlinear dynamics, and applies them to the problem of speech recognition. The assessment of ASR systems is driven by results and this concept reduces the problem to the following question: “Does
the use of nonlinear signal processing techniques improve the accuracy of current state-of-the-art (baseline) computer speech-to-text systems?” Although this is the central question addressed in this work, this research also seeks to further the understanding of the nonlinear methods employed, as well as to ascertain the limitations of the current techniques conventionally used in ASR. In summary, therefore, this research will endeavor to boost the results of current methodologies, while concurrently extending the knowledge of these new nonlinear techniques.

1.2.2 Speech and nonlinear dynamics

There are two distinct but related research fields that are addressed in this work: nonlinear dynamics and speech recognition. In the nonlinear dynamics community, a budding interest has emerged in the application of theoretical results to experimental time series data; the most significant being Takens’ Theorem and Sauer and Yorke’s extension of this theorem [6, 7]. Takens’ theorem states that under certain assumptions, the state space (also called phase space or lag space) of a dynamical system can be reconstructed through the use of time-delayed versions of the original signal. This new state space is commonly referred to in the literature as a reconstructed phase space (RPS), and has been proven to be topologically equivalent to the original phase space of the dynamical system, as if all the state variables of that system would have been measured simultaneously [6, 7]. A RPS can be exploited as a powerful signal processing domain, especially when the dynamical system of interest is nonlinear or even chaotic [8, 9]. Conventional linear digital signal processing techniques often utilize the frequency domain as the primary processing space, which is obtained through the Discrete Fourier Transform (DFT) of a time series [10]. For a linear dynamical system, structure appears in the frequency do-
main that takes the form of sharp resonant peaks in the spectrum. However for a nonlin-
ear or chaotic system, structure does not appear in the frequency domain, because the spectrum is usually broadband and resembles noise [8]. In the RPS, though, structure does emerge in the form of complex, dense orbits that form patterns known as attractors. These attractors contain the information about the time evolution of the system, which means that features derived from a RPS can potentially contain more and/or different information than a spectral representation. An example of a RPS for a typical speech pho-
neme is shown below.

![Figure 3: Example of reconstructed phase space for a speech phoneme](image)

The majority of literature that utilizes a RPS for signal processing applications re-
volves around its use for control, prediction, and noise reduction, reporting both positive and negative results [8, 9, 11]. There is only scattered research using RPS features for classification and/or recognition experiments [12-19]. One set of features that can be de-
erived from the RPS relates to a quantity known as the natural measure or natural distribu-
tion [8, 9, 20, 21]. The natural distribution is the fraction of time that the trajectories of an attractor spend in a particular neighborhood of the RPS as the size of the neighborhood
goes to zero and time goes to infinity, or simply the distribution of points in the RPS [22]. For an experimental time series such as speech, the signal is of finite length so the natural distribution can only be estimated via a numerical algorithm or model. Previous work has shown that accurate estimates of the natural distribution can be obtained through both histogram methods and Gaussian Mixture Models [12, 17]. This work has demonstrated that estimations of the natural distribution can be robust, accurate, and useful for recognition/classification, which is the central focus of this thesis.

1.2.3 Previous work in speech and nonlinear dynamics

There have been only a few publications that have examined the possibilities of applying nonlinear signal processing techniques to speech, all of which have been published within the last ten years [12-15, 18, 23-34]. Work by Banbrook et al. [23], Kumar et al. [27], and Narayanan et al. [31] has attempted to use nonlinear dynamical methods to answer the question: “Is speech chaotic?” These papers focused on calculating theoretical quantities such as Lyapunov exponents and Correlation dimension. Their results are largely inconclusive and even contradictory. Other papers by Petry et al. and Pitsikalis et al. have used Lyapunov exponents and Correlation dimension in unison with traditional features (cepstral coefficients) and have shown minor improvements over baseline speech recognition systems. Central to both sets of these papers is the importance of Lyapunov exponents and Correlation dimension, because they are invariant metrics that are the same regardless of initial conditions in both the original and reconstructed phase space [8]. Despite their significance, there are several issues that exist in the measuring of these quantities on real experimental data. The most important issue is that these measurements are very sensitive to noise [35]. Secondarily, the automatic computation of these quanti-
ties through a numerical algorithm is not well established and this can lead to drastically differing results. The overall performance of these quantities as salient features remains an open research question.

In addition to these speech analysis and recognition applications, nonlinear methods have also been applied to speech enhancement. Papers by Hegger et al. [24, 25] demonstrated the successful application of what is known as local nonlinear noise reduction to sustained vowel utterances. In our previous work [13], we extended this algorithm into a complete speech enhancement algorithm. The results showed mild success and demonstrated some of the potential limitations of the algorithm despite the optimistic claims made in the papers by Hegger et al.

With the exception of our work [12], the use of the natural distribution as a feature set derived from the RPS is yet to be exploited for speech recognition, which is the primary focus of this thesis.

1.2.4 Summary

As described, nonlinear signal processing techniques have several potential advantages over traditional linear signal processing methodologies. They are capable of recovering the nonlinear dynamics of the signal of interest possibly preserving more and/or different information. Furthermore, they are not constrained by strong linearity assumptions. Despite these facts, the use of nonlinear signal processing techniques also have disadvantages as well, which is why they have not been widely used in the past. Primarily, they are not as well understood as conventional linear methods. Additionally, the RPS can be difficult to interpret, because little is understood in the way of attaching physical meaning to particular attractor characteristics that appear in the RPS. Salient features for
classification/recognition have yet to be firmly established, and this work is clearly in the early stages, which is what motivates this research pursuit. Moreover, researchers have just begun to study nonlinear signal processing techniques for a variety of engineering tasks with mixed success. The use of nonlinear methodologies especially as it applies to speech recognition is truly in its infancy with very little work published, most of which has been in the last three years. Therefore, this thesis constitutes some of the very first work performed on the subject.

In order to ascertain the feasibility of the nonlinear techniques, discover the discriminatory power of the RPS derived features, and expand the knowledge of these nonlinear signal processing methods, work was conducted on the task of isolated phoneme classification. The motivation for restricting the scope of the research to isolated phoneme classification was two-fold. The first reason is that isolated phoneme classification allows one to focus in on the performance of the features, because the classification is performed using the acoustic data alone, and in turn, makes comparisons among different feature sets as “fair” as possible. Secondly, the task is less complex than continuous speech recognition, where issues of phoneme duration modeling create additional difficulties. The topic of phoneme duration will be discussed in full detail later.

The statistical model employed in this work was a one state Hidden Markov Model with a Gaussian Mixture Model state distribution. Again, this model was chosen to make comparisons “reasonable” and to keep the focus on the acoustic data.

Experiments were performed on both baseline MFCC feature sets, RPS derived feature sets, and joint MFCC and RPS feature sets. The choice of these feature sets allows
investigation into the discriminatory and representational power of each set as well as the sets used in unison.

The remainder of the thesis is outlined as follows. Chapter 2 summarizes conventional methods used in speech recognition, namely cepstral analysis and the statistical models typically used in speech: Hidden Markov Models and Gaussian Mixture Models. Chapter 3 develops the nonlinear methods employed in this research along with the feature extraction schemes utilized. Chapter 4 describes the experimental setup and the results of this work. Finally, Chapter 5 details the discussion of the results, draws conclusions, and proposes several different possible future research avenues in this area.
2. Background

2.1 Cepstral analysis for feature extraction

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>time index</td>
</tr>
<tr>
<td>$s[n]$</td>
<td>signal in the time domain</td>
</tr>
<tr>
<td>$e[n]$</td>
<td>excitation signal in the time domain</td>
</tr>
<tr>
<td>$h[n]$</td>
<td>linear time invariant filter in time domain</td>
</tr>
<tr>
<td>$s_f[n]$</td>
<td>frame of a signal in time domain</td>
</tr>
<tr>
<td>$\omega$</td>
<td>frequency index</td>
</tr>
<tr>
<td>$S(\omega)$</td>
<td>signal in the frequency domain</td>
</tr>
<tr>
<td>$E(\omega)$</td>
<td>excitation signal in the frequency domain</td>
</tr>
<tr>
<td>$H(\omega)$</td>
<td>linear time invariant filter in frequency domain</td>
</tr>
<tr>
<td>$S_f(\omega)$</td>
<td>frame of a signal in the frequency domain</td>
</tr>
<tr>
<td>$q$</td>
<td>quefrecny index</td>
</tr>
<tr>
<td>$C(q)$</td>
<td>cepstral coefficients in the quefrecny domain</td>
</tr>
<tr>
<td>$F_{mel}$</td>
<td>Mel frequency scale</td>
</tr>
<tr>
<td>$F_{Hz}$</td>
<td>linear frequency scale</td>
</tr>
<tr>
<td>$t$</td>
<td>index of frames</td>
</tr>
<tr>
<td>$c$</td>
<td>MFCC of a frame</td>
</tr>
<tr>
<td>$E$</td>
<td>energy measure</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>delta vector</td>
</tr>
<tr>
<td>$\Delta\Delta$</td>
<td>delta-delta vector</td>
</tr>
<tr>
<td>$\mathbf{O}$</td>
<td>composite feature vector</td>
</tr>
</tbody>
</table>

Table 1: List of notation used in section 2.1
2.1.1 Introduction to cepstral analysis

The aim of cepstral analysis methods is to extract the vocal tract characteristics from the excitation source, because the vocal tract characteristics are what contain the information about the phoneme utterance [2]. Cepstral analysis is a form of homomorphic signal processing, where nonlinear operations are used to give the equations the properties of linearity [2].

One typical model used to represent the entire speech production mechanism is given in figure below.

![Block diagram of speech production model](image)

Figure 4: Block diagram of speech production model

Although this model is accurate, the analysis can be made simpler by replacing the glottal, vocal tract, and lip radiation filters, by a single vocal tract filter given in the next figure. This model is obtained by collapsing all these separate filters into a vocal tract filter by the convolution operation.
An analytical model of this block diagram can be formulated in the following way. According to the conventional source-filter model representation, a speech signal is composed of an excitation source convolved with the vocal tract filter.

\[ s[n] = h[n] \ast e[n] \Rightarrow S(\omega) = H(\omega)E(\omega) \]  

(2.1.1)

By taking the logarithm of the magnitude of both sides, the equation is converted from a multiplication to an addition.

\[ \log|S(\omega)| = \log|H(\omega)E(\omega)| = \log|H(\omega)| + \log|E(\omega)| \]  

(2.1.2)

Then, by taking the inverse discrete Fourier Transform (IDFT) of \( \log|S(\omega)| \), the cepstrum is obtained in what is known as the quefrency domain [2].

\[ C(q) = \text{IDFT} \left\{ \log|S(\omega)| \right\} = \text{IDFT} \left\{ \log|H(\omega)| \right\} + \text{IDFT} \left\{ \log|E(\omega)| \right\} \]  

(2.1.3)

The cepstrum then allows the excitation signal to be completely isolated from the vocal tract characteristics, because the multiplication in the frequency domain has been converted to an addition in the quefrency domain.

A visual demonstration of the computation of the cepstrum for a frame of speech data is shown in the following figure. Notice that the ripple in log spectral magnitude, which is due to the excitation source, is isolated from the spectral envelope of the vocal tract.
Cepstral coefficients, $C(q)$, can be used as features for speech recognition for several reasons. First, they represent the spectral envelope, which is the vocal tract. The vocal tract characteristics are understood to contain information about the phoneme that is produced. Second, cepstral coefficients have the property that they are uncorrelated with one another, which simplifies subsequent analysis [1, 2]. Third, their computation can be done in a reasonable amount of time. The last and most important reason is that cepstral coefficients have empirically demonstrated excellent performance as features for speech recognition for many years [2].
2.1.2 Mel frequency Cepstral Coefficients (MFCCs)

The most popular form of cepstral coefficients are known as Mel frequency cepstral coefficients (MFCCs). MFCCs are computed in a similar way as the methods described in section 2.1.1, but have been slightly modified. The procedure is as follows:

a) The signal is filtered using a pre-emphasis filter to accentuate the high-frequency content. This is also performed to compensate for the spectral tilt that is present in most speech phoneme spectra.

\[ s'_f[n] = h[n] * s_f[n]^{\text{Mel}} \Rightarrow S'_f(z) = H(z) S_f(z) \]  
\[ H(z) = 1 - 0.97 z^{-1} \]  

b) The signal is multiplied by overlapping windows and divided into frames usually using ~20-30 ms windows with ~10-15ms of overlap between windows.

\[ s'_f[n] = s'_f[n] \cdot w[n] \]  

A hamming window is typically used for \( w[n] \).

\[ w[n] = \begin{cases} 
0.54 - 0.46 \cos \left( \frac{2\pi n}{N} \right), & 0 \leq n \leq N \\
0, & \text{otherwise}
\end{cases} \]

Figure 7: Hamming window
c) The discrete Fourier Transform (DFT) is taken of every frame \( s'_n \) followed by the logarithm.

\[
S''(\omega) = \log|DFT\{s'_n[n]\}|
\]

\[[2.1.8]\]

d) Twenty-four triangular-shaped filter banks with approximately logarithmic spacing (called Mel banks) are applied to \( S''(\omega) \). The energy in each of these bands is integrated to give 24 coefficients in the spectrum. The Mel scale is a warping of the frequency axis that models the human auditory system and has been shown to empirically improve recognition accuracy. A curve fit is usually used to approximate the empirically derived scale given below.

\[
F_{mel} = 2595\log_{10}\left(1 + \frac{F_{fe}}{700}\right)
\]

\[[2.1.9]\]

e) The Discrete Cosine Transform (DCT) is taken to give the cepstral coefficients.

\[
\tilde{c} = DCT\{S''(\omega)\}
\]

\[[2.1.10]\]

f) The first 12 coefficients (excluding the 0\(^{th}\) coefficient) are the features that are known as MFCCs \( c_i \).

2.1.3 Energy, deltas, and, delta-deltas

In addition to MFCC’s, typically other elements are appended to the feature vector. One prominent measure is the energy. The energy can be an important metric for distinguishing among different phonemes [5]. Although the 0\(^{th}\) cepstral coefficient can be used
as an energy measure, it is usually preferred to use the energy in the framed time series as given below.

\[ E = \log \sum_{n=1}^{N} s_f^2[n] \quad (2.1.11) \]

The temporal change of cepstral coefficients over a sequence of frames is also a valuable feature. This temporal change or derivative has boosted the accuracy of ASR systems for many applications [3, 12]. One way to estimate the derivative of the MFCCs is to use a simple linear regression formula [5].

\[ \Delta_i = \frac{\sum_{\theta=1}^{\theta} \theta (c_{i+\theta} - c_{i-\theta})}{2 \sum_{\theta=1}^{\theta} \theta^2} \quad (2.1.12) \]

The second derivatives are estimated simply by taking the linear regression formula given in equation (2.1.12) and applying to \( \Delta_i \) [5].

\[ \Delta\Delta_i = \frac{\sum_{\theta=1}^{\theta} \theta (\Delta_{i+\theta} - \Delta_{i-\theta})}{2 \sum_{\theta=1}^{\theta} \theta^2} \quad (2.1.13) \]

These derivative estimates are referred to as deltas and delta-deltas respectively. The derivative estimates of the energy are computed using the same formulas as for the MFCCs.

### 2.1.4 Assembling the feature vector

After all the analysis is performed on every frame, the computed elements can be compiled into a feature vector (given below) and then as input to an ASR system.

\[ \mathbf{O}_i = \begin{bmatrix} c_i & E_i & \Delta_i^c & \Delta_i^\theta & \Delta\Delta_i^c & \Delta\Delta_i^\theta \end{bmatrix} \quad (2.1.14) \]

The feature vector is composed as follows: i.) 12 MFCCs ii.) energy iii.) deltas of the 12 MFCCs iv.) delta of the energy v.) delta-deltas of the 12 MFCCs vi.) delta-delta of the
energy. The total number of elements in the feature vector is then 39. A illustration of the computation of the feature in block diagram form is displayed next.

![Block diagram of feature vector computation](image.jpg)

**Figure 8: Block diagram of feature vector computation**

### 2.2 Statistical models for pattern recognition

#### 2.2.1 Gaussian Mixture Models

| $p(\cdot)$ | probability density function |
| $x$ | an arbitrary feature vector |
| $m$ | mixture index |
| $w$ | mixture weight |
| $\mu$ | mean vector |
| $\Sigma$ | covariance matrix |
| $k$ | index of training vectors |
| $\lambda$ | set of parameters of a statistical model |
| $i$ | iterative index |
| $\omega$ | class |
| $c$ | class index |
| $q$ | index of testing vectors |

**Table 2: Notation used for section 2.2**
The first step in building a statistical pattern recognition system is choosing an appropriate model to capture the distribution of the underlying features. Gaussian Mixture Models (GMM) are an excellent choice, because they have several desirable properties. Their primary advantage is that they are flexible and robust, because they can accurately model a multitude of different distributions including multi-modal data. In fact, they can represent any arbitrary probability density function (PDF) in the limit as the number of mixtures go to infinity \((M \rightarrow \infty)\) [36]. The equations for a GMM are given below.

\[
p(x) = \sum_{m=1}^{M} w_m p(x|m)
\]

\[
p(x|m) = (2\pi)^{\frac{d}{2}} |\Sigma_m|^{\frac{1}{2}} e^{-\frac{1}{2}(x - \mu_m)' \Sigma_m^{-1} (x - \mu_m)} = N(x; \mu_m, \Sigma_m)
\]

Frequently, there is no prior knowledge of the distribution of the data and consequently, a GMM will be a suitable model for the task.

After the model has been chosen, the parameters of that PDF have to be learned from training data. The typical data-driven method used to estimate these model parameters is to discover the parameters that maximize the data likelihood, known as the Maximum Likelihood method. Assuming that the data points (feature vectors) are statistically independent from one another, the objective or criterion function can be formulated known as the likelihood function \(L\) or the log likelihood function \(\ell\).

\[
\ell = \prod_{k=1}^{K} p(x_k) \Rightarrow L = \sum_{k=1}^{K} \log p(x_k)
\]

This objective function is optimized through the well-known Expectation Maximization (EM) algorithm [36-38]. EM Maximization is usually formulated as a “missing data” problem. The missing information in the case of a GMM is what mixture component a
particular data point \((x_i)\) was sampled from. The Maximum Likelihood estimator is given by

\[
\lambda^{(i+1)} = \arg \max_{\lambda} \left\{ \log p(y|\lambda) \right\},
\]

where \(y\) is the complete data and \(x\) is the incomplete data. The function inside of the argmax can be marginalized, differentiated, and set equal to zero to yield the following update equations for the parameters of a GMM given by

\[
\begin{align*}
W_m^{(i+1)} &= \frac{1}{K} \sum_{i=1}^{K} \frac{w_m^{(i)} p(x_i|m)}{p(x_i)}, \\
\mu_m^{(i+1)} &= \frac{\sum_{i=1}^{K} w_m^{(i)} p(x_i|m) x_i}{\sum_{i=1}^{K} w_m^{(i)} p(x_i|m)}, \\
\Sigma_m^{(i+1)} &= \frac{\sum_{i=1}^{K} w_m^{(i)} p(x_i|m) (x_i - \mu_m^{(i+1)}) (x_i - \mu_m^{(i+1)})^T}{\sum_{i=1}^{K} w_m^{(i)} p(x_i|m)}. 
\end{align*}
\]

EM is an iterative algorithm that is gradient-based and is guaranteed to find only a local maximum [36, 38]. Because it is gradient-based, EM is somewhat sensitive to initial conditions. A common method for initialization is mixture splitting. Mixture splitting begins with a one mixture GMM that is optimized from the training data. The one mixture GMM is then split into a two mixture GMM by dividing the largest weight by 2 and perturbing the mean vectors. Optimization occurs again until convergence is obtained and the procedure is repeated until the number of desired mixtures is reached.
In addition to EM’s sensitivity to initial conditions, there is no well-established method for setting the number of mixtures in a GMM. The number of mixtures in the GMM is related to the complexity of the model and data; the more mixtures in the GMM the more complex the model, and consequently, the more complex the data distribution. Care must be taken in setting the number of mixtures; too few mixtures can lead to inadequate representations of the data, ultimately resulting in poor classification accuracy. Too many mixtures leads to over-fitting and data memorization, which results in high training accuracy, but poor testing performance [37]. This is due to the fact that there is not enough training data available to properly estimate all the parameters of the model. In general, the number of mixtures should be set appropriately between these two extremes, so that the models have superior performance and generalize well to unseen test data.

A visualization of a GMM for random data is shown below.

Figure 9: Visualization of a GMM for randomly generated data
The mean vector is represented as the center of each ellipse, while the covariance matrices are represented as the ellipses. The figure reveals the ability of a GMM to find the clusters in the data and demonstrates the model’s capability to represent complex data distributions.

Parameter estimation is carried out for each GMM, $p(x|\omega_c)$, associated with every class, $\omega_c$, using the corresponding labeled training data. Following parameter estimation, a decision rule must be formulated in order to classify the previously unseen test data, i.e. associate the correct test data with its estimated class. The typical decision rule utilized in statistical pattern recognition is known as Bayes’ decision rule [37]. Bayes’ Theorem is given by

$$p(\omega_c | x) = \frac{p(x|\omega_c)P(\omega_c)}{P(x)}.$$  \hspace{1cm} (2.2.8)

Bayes’ Theorem is able to connect the probability of a certain class model given the data, $p(\omega_c | x)$, with the data likelihood under that class model, $p(x|\omega_c)$, weighted by the prior probability, $P(\omega_c)$, and the probability of the data, $P(x)$. The probability of data is a constant for every class in the set and one can assume that the prior probabilities for each class are equal. Making this assumption leads to the following maximum likelihood decision rule,

$$\omega^* = \arg \max_{c=1,2,3,...C} \sum_{q=1}^{o} \log \left[ p(x_q | \omega_c) \right] \Rightarrow \omega^* = \arg \max_{c=1,2,3,...C} \sum_{q=1}^{o} \log \left[ p(x_q | \omega_c) \right]$$ \hspace{1cm} (2.2.9)

A unique property of the Bayes’ decision rule is that if the PDFs of the classes are truly known and the prior probabilities are equal, this decision rule is the optimal classifier; meaning that no other classifier can outperform it [37]. This fact makes the use of
2.2.2 Hidden Markov Models

Hidden Markov Models (HMM) are statistical models used to track the temporal changes of non-stationary times series or in case of ASR, sequences of various vocal tract configurations. One could interpret the position of the vocal tract as represented by a particular GMM. The HMM then links these GMMs together into a specific time sequence. HMMs are particularly important in continuous speech recognition where the goal is to classify a sequence of phonemes as they proceed in time. Although this is not the explicit task here, the general concept of an HMM is presented to connect this work with that used in continuous speech. HMMs have their origin in theory of Markov Chains and Markov Processes [39]. HMMs, similar to GMMs, are formulated as “missing data” models [5, 37, 40]. In the case of a HMM, the “missing data” is knowing what state a particular observation originated from.

There are three main algorithms that are typically discussed in the literature when dealing with HMMs [40]. All of the algorithms rely heavily on dynamic programming concepts. The first is how to compute the probability of a particular observation sequence given an HMM model, which is solved by the forward-backward algorithm. The next is how to estimate the parameters of a HMM model given a sequence of observations. A version of the EM algorithm called the Baum-Welch algorithm is used to perform this parameter optimization. The final is how to find the most likely sequence of states that a given an observation sequence traversed. An algorithm called Viterbi is executed to find that sequence.
Using these algorithms, the Maximum Likelihood estimators for the parameters can then be derived as recursive update equations for the state transition probabilities as well as the parameters of the associated GMMs. The update equations for the GMM parameters are very similar to the equations described in section 2.2.1 except they contain additional weighting factors for the state occupancy likelihoods. After parameter learning is finalized, the final output of the recognizer can be performed using the Viterbi algorithm. A rigorous derivation and explanation of HMMs can be found in [2, 5, 40].

![Diagram of HMM with GMM state distributions](image)

**Figure 10**: Diagram of HMM with GMM state distributions

### 2.2.3 Practical issues

The statistical pattern recognition methods presented here perform “soft-decisions”. Soft-decision algorithms reserve all classification decisions until the very end of the process. Other “hard-decision” machine learning algorithms make decisions about classi-
fication throughout the entire process. In general, soft-decision methods are superior to
schemes that rely on hard decisions [36, 41, 42].

For the remainder of this work, a one state HMM will be used with a GMM state dis-
distribution for the task of isolated phoneme classification. A one state HMM is nearly
equivalent to a GMM except for the fact that the single self-transition probability adds a
weight to the update equations of the parameters of its associated GMM state distribu-
tion; though for all practical purposes, they are equivalent.

The covariance matrices for all experimentation were constrained to be diagonal. Di-
agonal matrices were used to ensure that they could be inverted as is necessary for
evaluation of equation (2.2.2). If the covariance matrices were left to be arbitrary or full,
a situation could arise where after EM optimization, the matrix might become ill condi-
tioned or singular. In such cases, there are no well-established methods or “work
arounds” to resolve this issue, and further progress halted. Therefore, in order to avoid
completely the possibility of this problem occurring, the off-diagonal elements were set
to zero and only the diagonal elements were optimized. This constraint does limit the
flexibility of the GMM to some extent, but such a sacrifice is prudent to circumvent this
potential computational pitfall that crops up quite frequently in practice.

Another key pragmatic issue is the methods used during the training process. As ex-
plained beforehand, the typical method used to increment the number of mixtures in a
GMM model is to use binary splitting to reach the desired number of mixtures, while si-
multaneously executing EM after every increment until convergence. The question is
this: when do you know that the GMM converged or what is the practical stopping crite-
rian? There are two typical stopping criteria: the parameter change from one iteration to
the next is smaller then some adjustable and arbitrary tolerance threshold, or until a certain arbitrary number of iterations is performed. In this work, an arbitrary number of iterations of EM were performed, after which, the models were assumed to have converged. Changing the number of iterations performed can and will effect the final testing accuracy in unpredictable ways, if the parameters are far from convergence. It is important therefore to set the number of iterations reasonably large so as to ensure some convergence, but small enough to be computationally feasible. In practice, though, it is difficult to know where the models are in the convergence process. Consequently, one strong caveat that precipitates from this issue is that results can differ slightly depending on the specific training strategy employed.

2.3 Summary

This chapter has presented a condensed review of the fundamental elements used in ASR systems. A typical ASR system will use MFCC features as the “front end” with an HMM that has GMM state distributions as the statistical models. The Baum-Welch algorithm is employed for parameter optimization, and the final classification is performed using the Viterbi algorithm and Bayes’ decision rule. If the task is continuous word recognition, additional language models can be utilized such as bigram or trigram models [2, 5].
### 3. Nonlinear Signal Processing Methods

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>time index</td>
</tr>
<tr>
<td>( x[n] ) or ( x_n )</td>
<td>time series</td>
</tr>
<tr>
<td>( x_n )</td>
<td>row vector in RPS</td>
</tr>
<tr>
<td>( d )</td>
<td>dimension of embedding</td>
</tr>
<tr>
<td>( \tau )</td>
<td>time lag</td>
</tr>
<tr>
<td>( X )</td>
<td>trajectory matrix</td>
</tr>
<tr>
<td>( \mu_x )</td>
<td>mean vector (centroid)</td>
</tr>
<tr>
<td>( \sigma_r )</td>
<td>standard deviation of the radius</td>
</tr>
<tr>
<td>( \eta )</td>
<td>number of points in overlap</td>
</tr>
<tr>
<td>( \rho )</td>
<td>stream weight</td>
</tr>
<tr>
<td>( t )</td>
<td>index for cepstral features</td>
</tr>
<tr>
<td>( i )</td>
<td>iterative index</td>
</tr>
<tr>
<td>( s )</td>
<td>stream index</td>
</tr>
<tr>
<td>( b(\cdot) )</td>
<td>stream model of GMMs</td>
</tr>
<tr>
<td>( V^d )</td>
<td>volume of a neighborhood region in the RPS</td>
</tr>
</tbody>
</table>

Table 3: Notation for Chapter 3

#### 3.1 Theoretical background and Takens’ Theorem

The study of the nonlinear signal processing techniques employed in this research begins with phase space reconstruction, also called phase space embedding. Phase space reconstruction has its origin in the field of mathematics known as topology, and the details of the theorems in their historical context can be found in [6, 7, 43, 44]. The princi-
ple thrust of these theorems is that the original phase space of the dynamical system can be reconstructed by observing a single state variable of the system of interest. Furthermore, reconstructed phase spaces (RPS) are assured, with probability one, to be topologically equivalent to the original phase space of the system if the embedding dimension is large enough; meaning that the dimension of the RPS must be greater than twice the dimension of the original phase space to ensure full reconstruction. This property also guarantees that certain invariants of the system will also be identical.

It is important at this point to clearly define the nomenclature used when referring to phase spaces. The term phase space usually refers to an analytical state space model where the state variables and possibly their derivatives are plotted against one another [45]. A reconstructed phase space, on the contrary, is formed from experimental measurements of the state variables themselves, because the analytical state space model is complex and/or not available. There are many potential methods for forming a RPS from a measured time series. The most common way is through the use of time-delayed versions of the original signal. Another method would use linear combinations of the time-delayed versions of the original signal. The final method would be to experimentally measure multiple state variables simultaneously. Additionally, any mixture of these, be it time-delay, linear combination, or multiple state variables, also can constitute a RPS.

A RPS can be produced from a measured state variable, \( x[n] \) or \( x_n \), \( n = 1, 2, 3 \ldots N \), via the method of delays by creating vectors in \( \mathbb{R}^d \) given by

\[
\mathbf{x}_n = \begin{bmatrix}
    x_n & x_{n-\tau} & x_{n-2\tau} & \cdots & x_{n-(d-1)\tau}
\end{bmatrix}, \quad n = 1 + (d - 1)\tau, 2 + (d - 1)\tau, 3 + (d - 1)\tau \ldots N
\]

(3.1.1)

The row vector, \( \mathbf{x}_n \), defines the position of a single point in the RPS. These row vectors are completely causal so that all the delays are referenced time samples that occurred in
the past and not the future, although it is also common to make the row vectors non-causal by referencing time samples that are future values. The row vectors then can be compiled into a matrix (called a trajectory matrix) to completely define the dynamics of the system and create a RPS.

\[
X = \begin{bmatrix}
  x_{1+(d-1)\tau} & x_{1+(d-2)\tau} & x_{1+(d-3)\tau} & \cdots & x_1 \\
  x_{2+(d-1)\tau} & x_{2+(d-2)\tau} & x_{2+(d-3)\tau} & \cdots & x_2 \\
  x_{3+(d-1)\tau} & x_{3+(d-2)\tau} & x_{3+(d-3)\tau} & \cdots & x_3 \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  x_N & x_{N-(d-1)\tau} & x_{N-(d-2)\tau} & \cdots & x_{N-(d-1)\tau} 
\end{bmatrix} \quad (3.1.2)
\]

By plotting the row vectors of the trajectory matrix, a visual representation of the system dynamics becomes evident as shown below.

![Figure 11: Two dimensional reconstructed phase spaces of speech phonemes](image)

Upon qualitative examination of the figure above, it is obvious that different phonemes demonstrate differing structure in the RPS. The RPS of vowels seem to exhibit circular...
orbits which is indicative of periodicity likely originating from a voicing source excitation, while the fricatives are a bubble of orbits indicative of noise probably due to the uncorrelated white noise excitation.

Although Takens’ theorem gives the sufficient condition for the size of the dimension of the embedding, the theorem does not specify a time lag. Furthermore, the precise dimension of the original phase space is unknown, which means that it is also unknown in what dimension to embed the signal to make the RPS. Methods to determine appropriate time lags and embedding dimension will be discussed in depth later.

When performing linear signal processing, the analytical focus is on the frequency domain characteristics such as the resonant peaks of the system [8]. The RPS-based nonlinear methods, though, concentrate on geometric structures that appear in the RPS. The orbits or trajectories that the row vectors of the trajectory matrix form produce these geometric patterns. These orbits can be loosely called attractors. Attractors are defined as a bounded subset of the phase space in which the orbits or trajectories reside as

\[ \text{time} \to \infty [22]. \]

A particular pattern in a RPS can only be an ‘attractor’ in the strict sense if it can be shown mathematically via a proof. A proof can in general only be created for systems where the dynamics are known and not for experimental time series data. Regardless of this formalism, it is common to refer to the geometric patterns that appear in the RPS generated from experimental time series data as attractors.

There are three types of invariant measures that exist for nonlinear systems: metric (Lyapunov exponents and dimensions), natural measure or distribution, and topological. The focus of this work is on the natural distribution and the use of it as features for speech recognition.
3.2 Features derived from the RPS

3.2.1 The natural distribution

The primary feature set developed is the natural distribution or natural measure. As
aforementioned the natural distribution is the fraction of time that the trajectories of the
attractor reside in a particular neighborhood of the RPS as $\text{time} \to \infty$ and as $V^d \to 0$ or
simply the distribution of row vectors, $x_n$, in the RPS.

For implementation then, the feature vector is given by

$$x_{n}^{(d,\tau)} = \frac{x_n - \mu}{\sigma}, \quad n = 1 + (d - 1)\tau, 2 + (d - 1)\tau, 3 + (d - 1)\tau \ldots N,$$

(3.2.1)

where $\mu$ (centroid) and $\sigma$ (standard deviation of the radius) are defined by

$$\mu = \frac{1}{N - (d - 1)\tau} \sum_{n=1+(d-1)\tau}^{N} x_n,$$

(3.2.2)

$$\sigma = \frac{1}{N - (d - 1)\tau} \sum_{n=1+(d-1)\tau}^{N} \left\| x_n - \mu \right\|^2.$$

(3.2.3)

The $\mu$ in equation (3.2.1) serves to zero-mean the distribution in the RPS. Usually
speech signals are close, if not exactly zero-meaned anyway, but this subtraction ensures
it. The $\sigma$ serves to “normalize” the distribution in the RPS. The reasons for doing this
have their origin in the fact that the natural distribution is subject to translation and scal-
ing effects of the original amplitude of the signal. The normalization is performed via
$\sigma$ to realign natural distributions that have similar shape, but differing amplitude ranges.

Upon examination of equation (3.2.1), it is obvious that this normalization procedure is
similar to a z-score normalization [39], but this is done using a multidimensional mo-
ment, the standard deviation of the radius \( (\sigma_r) \), instead of the one-dimensional classic standard deviation \( (\sigma_x) \).

The motivation for using this natural distribution feature set is well founded. Upon inspection of equation (3.2.1), it is clear that the natural distribution features endeavor to capture the time evolution of the attractor in the RPS as the distinguishing characteristic of speech phonemes. This feature set affirms that the natural distribution and its attractor structure (or part of it anyway), remains consistent for utterances of the same phoneme, while differing in an appreciable way among utterances of different phonemes. It is reasonable to assert this, because it makes sense to consider the fact that the system dynamics of the speech production mechanism as captured through the natural distribution would represent a particular phoneme utterance, and that some part of the dynamics would approximately remain constant for a particular utterance of the same phoneme. It is precisely this conjecture that this research seeks to examine.

### 3.2.2 Trajectory information

In addition to the natural distribution of the RPS, trajectory information can be incorporated into the feature vector to capture the motion or temporal change as time increases.

As evident from the figure below, the trajectory information could have substantial discriminatory power in cases where two utterances may have similar natural distributions in the sense that the row vectors visit similar neighborhoods of the RPS, but take radically different paths as the attractors evolve.
In order to capture this trajectory information, two different augmented feature vectors can be assembled. One method uses first difference information to compute the time change of two consecutive row vectors in the RPS.

\[
\mathbf{x}_{n}^{(d,r,\&\&d)} = \left[ \mathbf{x}_{n}^{(d,r)} | \mathbf{x}_{n}^{(d,r)} - \mathbf{x}_{n-1}^{(d,r)} \right]
\]  

(3.2.4)

This method is easy to compute and relatively straightforward, but is susceptible to noise amplification due to the vector-by-vector subtraction. Another method would be to compute the deltas as described in 2.1.3. Because this method performs a linear regression, it tends to smooth out the effects of noise.

\[
\mathbf{x}_{n}^{(d,r,\&\&\&\&\&d)} = \left[ \mathbf{x}_{n}^{(d,r)} | \Delta_n \right]
\]  

(3.2.5)

\[
\Delta_n = \frac{\sum_{\delta=1}^{\delta^3} \theta \left( \mathbf{x}_{n+\delta}^{(d,r)} - \mathbf{x}_{n+\delta}^{(d,r)} \right)}{2\sum_{\delta=1}^{\delta^3} \theta^2}
\]  

(3.2.6)

Regardless of the method used, the augmented feature vector still constitutes an RPS as described in section 3.1, and will possibly increase the discriminatory power of a classifier built using these features.
3.2.3 Joint feature vector

The RPS derived features can also be used in unison with the MFCC feature set to create a joint or composite feature vector. The motivation for such a feature vector would be two-fold. The first reason is that MFCC feature set has been successful for speech recognition in the past, and utilizing them with the RPS derived feature set will increase classification accuracy, if the information content between the two is not identical. The second reason is that it will be interesting to see how the incorporation of two different sets of features extracted from radically dissimilar processing spaces and methodologies will fuse together to help ascertain the precise information content and discriminatory power of the RPS derived feature set when compared to the MFCC feature set.

There are two central issues that arise when assembling this joint feature vector: probability scaling and feature vector time speed mismatch. The first issue will be handled using different probability streams and will be discussed further in the next section.

The feature vector time speed mismatch issue arises due to the fact the RPS derived feature vectors change at a rapid rate (one feature vector per time sample). The MFCC feature set, however, requires an analysis window in order to perform the necessary signal processing steps. The typical overlap size \( \eta \) of the analysis windows is usually around 10ms in length, or 160 time samples for 16 KHz sampling rate. This implies that for every MFCC feature vector, there are approximately 160 RPS derived feature vectors. To address this problem, the MFCC features were replicated for 160 time samples and then time aligned to the RPS derived features. Replication and alignment gives the following joint feature vector using the delta method for the trajectory information given below by
\[ y_n = \left[ x_n^{(d, r, \& A)} \ | \ O_i \right] , \] (3.2.7)

where \( x_n^{(d, r, \& A)} \) is defined in equation (3.2.5) and \( O_i \) is defined in equation (2.1.14). The indices are given by

\[
n = 1 + (d - 1)\tau, 2 + (d - 1)\tau, 3 + (d - 1)\tau \ldots N ,
\]

\[
t^{(i+1)} = \begin{cases} 
1, & \text{if } n = 1 + (d - 1)\tau \\
t^{(i)} + 1, & \text{if } n - (d - 1)\tau \geq (t^{(i)} + 1)\eta .
\end{cases} \] (3.2.8)

The total number of elements in \( y_n \) would be 49; the first 10 elements are the RPS derived features, while the next 39 would be MFCCs, energy, deltas, and delta-deltas. The recursive equation for \( m \) defines the replication of the MFCC feature vectors every \( \eta \) time samples to ensure time alignment. In cases where the number of time samples were not a multiple of \( \eta \), zero padding was performed so that the analysis window was the correct length. The zero padding was only used in the computation of the MFCC features and not performed for the RPS derived features.

![Figure 13: Relationship between indices for the joint feature vector](image_url)

Now that all the RPS derived feature vectors have been formulated the next step is to build models and design a classifier.
3.3 Modeling technique and classifier

3.3.1 Modeling the RPS derived features

The primary modeling technique utilized for the RPS derived features were statistical models. The model was a one state HMM with a GMM state PDF as described in sections 2.2 and 2.3. As aforesaid, this model choice is flexible, robust, and particularly well suited to the RPS derived features, because the goal, in the end, is to estimate the natural distribution of the RPS as represented by the feature vectors. Furthermore, the GMM parameters, when viewed as a clustering algorithm, gravitate towards the attractor, attempting to adhere to its shape. From this perspective, it is straightforward to interpret the function of the GMM, and how it is working to represent the attractor structure. An example of the GMM modeling technique for the RPS derived features is shown below.

![Figure 14: GMM modeling of the RPS derived features for the phoneme '/aa/'](image)

The attractor is the dotted line, while the solid lines are one standard deviation of each mixture in the model, and the crosshairs are the mean vectors of each mixture. This plot
demonstrates the ability of the GMM to model the complex natural distribution of the RPS as well as its ability to capture the characteristic attractor structure.

As stated earlier, the chief issue that arises when using a GMM is choosing the correct number of mixtures, which is directly related to the complexity of the model. In general, the mixture number must be large enough to capture all the salient patterns present in the training data. The number of mixtures needed to attain a high-quality estimate of the RPS derived features far exceeds the usual number used for MFCC features (typically ~8-16 mixtures). The reason for the large number of mixtures is that attractor patterns can be quite complex, because just one attractor includes a substantial amount of data (~300-3000 row vectors). Obviously, data insufficiency issues that cause over-fitting are not relevant here; by way of comparison, there is roughly 160 times more data for the RPS derived features than for the MFCC features. The precise method to determine the correct number of mixtures for the RPS derived features will be covered in detail in the next chapter.

3.3.2 Modeling of the joint feature vector

The key question when modeling the joint feature vector \((\mathbf{y}_n)\) is how to develop a modeling technique that properly unites the two different knowledge sources; namely the RPS derived features and the MFCC features. A naïve approach would be to simply build a joint distribution of the entire composite feature vector \((\mathbf{y}_n)\) such as a 49 dimensional GMM. Although this method seems logical, there is a major problem of feature set weighting that is involved. Implicit to this approach is the fact that each of the feature sets has equally weighted importance for classification. This implicit assumption of
equality of features is almost certainly incorrect, because the RPS derived features and
the MFCC features were extracted from the data in radically different ways.

A possible solution to this weighting problem would be to introduce another method for
knowledge source combination that allows the flexibility of differing weights.

One such model entails two different GMM models each built over the different fea-
ture sets, which are then reweighted and combined. This method uses streams, and the
composite stream model is given below

\[
b(y_s) = \prod_{s=1}^{S} \left( \sum_{n=1}^{N} w_{n,s} N(y_{n,s}; \mu_{n,s}, \Sigma_{n,s}) \right)^{s}.
\]  

(3.3.1)

In this case, \( S = 2 \) (\( s = 1 \) is the RPS derived features, while \( s = 2 \) is the MFCC features),
and therefore, equation (3.3.1) can be further simplified by taking the logarithm,

\[
\log |b(y_s)| = (1 - \rho) \log \left( \sum_{n=1}^{M} w_{n,1} N(y_{n,1}; \mu_{n,1}, \Sigma_{n,1}) \right) + \rho \log \left( \sum_{n=1}^{M} w_{n,2} N(y_{n,2}; \mu_{n,2}, \Sigma_{n,2}) \right).
\]

(3.3.2)

The interpretation of equation (3.3.2) is rather straightforward, where \( \rho \) is simply a
weighting factor (called a stream weight) of the log probability of each GMM for the two
sets of features. The stream weight, \( \rho \), is constrained to be \( 0 \leq \rho \leq 1 \) to ensure proper
normalization. The parameters of each GMM in the stream model can be learned via EM
using modified update equations. A more detailed discussion of the stream models can be
found in [5].

Although update equations for the GMM parameters in a stream model can be formu-
lated, the choice of an appropriate stream weight, \( \rho \), remains an open question [5]. In
general, there is no well-established method to estimate \( \rho \), because it is difficult to solve
for the re-estimation equation using Maximum Likelihood. One reasonable method that
can be utilized to ascertain \( \rho \) would be examination of empirical classification accuracy
as $\rho$ varies. Because this approach is straightforward, it was adopted here, and the precise methodologies used will be discussed in the next chapter.

### 3.3.3 Classifier

The classifier used in conjunction with these models was a Bayes’ Maximum Likelihood classifier as described in section 2.2.1. A GMM or stream model is built and learned as described in sections 3.3.1 and 3.3.2 for every class in the set using available training data. After parameter optimization, the unseen test data is classified according to equation (2.2.9).

### 3.4 Summary

Chapter 3 has provided the theoretical framework and methodologies for the novel nonlinear techniques described in this work. The central premise is that the nonlinear methods can recover the full dynamics of the generating system through the RPS. Salient features can be extracted from the RPS that have substantive discriminability for speech phonemes. GMMs offer accurate and elegant modeling of these features. Subsequent classification can be carried out using these GMMs in an unambiguous manner to complete the entire ASR system architecture.
4. Experimental Setup and Results

<table>
<thead>
<tr>
<th>$\mathbf{x}_n^{(5,6)}$</th>
<th>RPS derived features capturing natural distribution ($d = 5, \tau = 6$, Total = 5 elements)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbf{x}_n^{(10,6)}$</td>
<td>RPS derived features capturing natural distribution ($d = 10, \tau = 6$, Total = 10 elements)</td>
</tr>
<tr>
<td>$\mathbf{x}_n^{(5,6,&amp;\tau)}$</td>
<td>RPS derived features capturing natural distribution with first difference trajectory information appended ($d = 5, \tau = 6$ &amp; first difference, Total = 10 elements)</td>
</tr>
<tr>
<td>$\mathbf{x}_n^{(5,6,&amp;\tau)}$</td>
<td>RPS derived feature capturing natural distribution with delta trajectory information appended ($d = 5, \tau = 6$ &amp; $\Delta_n$, Total = 10 elements)</td>
</tr>
<tr>
<td>$\mathbf{c}_n$</td>
<td>12 MFCC features (Total = 12 elements)</td>
</tr>
<tr>
<td>$\mathbf{O}_n$</td>
<td>12 MFCCs, energy, delta 12 MFCCs, delta energy, delta-delta 12 MFCCs, delta-delta energy (Total = 39 elements)</td>
</tr>
<tr>
<td>$\mathbf{y}_n$</td>
<td>Joint feature vector: RPS derived feature capturing natural distribution with delta trajectory information appended &amp; 12 MFCCs, energy, delta 12 MFCCs, delta energy, delta-delta 12 MFCCs, delta-delta energy (Total = 49 elements)</td>
</tr>
</tbody>
</table>

Table 4: Notation used in Chapter 4

4.1 Software

Most of the experiments carried out in this work were performed using HTK [5].

HTK is a software package that has been used in the ASR research community for almost 15 years. The software has become, to some extent, “standard” in the community for most types of research in ASR. The software is free and available at [http://htk.eng.cam.ac.uk/](http://htk.eng.cam.ac.uk/). Also, some of the software for front-end processing as well as file parsing was performed using MATLAB [46], PERL, and DOS batch files. Some ex-
ample scripts and codes used in this work are available in the appendix so that results can be duplicated.

4.2 Data

The data set used for the experiments was taken from the TIMIT database [47]. TIMIT is a well-known corpus in the speech community that has been used for numerous studies over the last 15 years. TIMIT is a medium-size, speaker independent database containing dialect regions all across the United States. It consists of 630 speakers that have read 10 prompted sentences. TIMIT is also a unique corpus, because it contains expertly labeled, phonetic transcriptions/segmentations performed by linguists that most other corpora do not possess. This makes TIMIT ideal for performing isolated phoneme classification experiments, although it can be used for continuous recognition as well. All of the data in the corpus was digitized using 16 bits and sampled at frequency of a 16 kHz (62.5 µs per time sample).

Using the expertly labeled, time-stamped phoneme boundaries present in TIMIT, all the phoneme exemplars from the “SI” and “SX” marked sentences were extracted according to these segmental boundaries. The “SI” and “SX” sentences were chosen, while the “SA” sentences were omitted, because the “SI” and “SX” sentences are phonetically balanced and different among speakers, whereas all the speakers spoke the “SA” sentences. It is a common practice in the literature to use only the “SI” and “SX” sentences when performing recognition/classification experiments over TIMIT [48]. Additionally, the training data was taken as the predefined training partition, and likewise, the testing data was taken as the predefined testing partition.
TIMIT is a speaker-independent corpus, which means that the speakers in the training set are not the same as the speakers in the testing set. This aspect of the task challenges the ability of ASR systems to generalize to other speakers.

The original phonetic transcriptions contain a total of 64 phonetic labels. For most practical purposes, this phoneme set is too large to perform recognition over, so common methods have been developed to reduce this set. The methodology adopted in this work for phoneme set folding can be found in [48]. The set of 64 was reduced to a set of 48 for training, and then further reduced for testing to 39. The following table summarizes the folding procedure.
<table>
<thead>
<tr>
<th>Phoneme</th>
<th>Example</th>
<th>Folded for Training</th>
<th># of Training exemplars</th>
<th># of Testing exemplars</th>
</tr>
</thead>
<tbody>
<tr>
<td>/aa/</td>
<td>cct</td>
<td>2256</td>
<td>847</td>
<td></td>
</tr>
<tr>
<td>/ae/</td>
<td>bat</td>
<td>2292</td>
<td>772</td>
<td></td>
</tr>
<tr>
<td>/ah/</td>
<td>big</td>
<td>2266</td>
<td>860</td>
<td></td>
</tr>
<tr>
<td>/ao/</td>
<td>about</td>
<td>1865</td>
<td>761</td>
<td></td>
</tr>
<tr>
<td>/aw/</td>
<td>bough</td>
<td>728</td>
<td>216</td>
<td></td>
</tr>
<tr>
<td>/ax/</td>
<td>thg</td>
<td>3892</td>
<td>1435</td>
<td></td>
</tr>
<tr>
<td>/ay/</td>
<td>bit</td>
<td>1934</td>
<td>686</td>
<td></td>
</tr>
<tr>
<td>/b/</td>
<td>bob</td>
<td>2181</td>
<td>886</td>
<td></td>
</tr>
<tr>
<td>/ch/</td>
<td>church</td>
<td>820</td>
<td>259</td>
<td></td>
</tr>
<tr>
<td>/cl/</td>
<td>unvoiced closure</td>
<td>/pcl/, /kcl/, /qcl/</td>
<td>12518</td>
<td>4300</td>
</tr>
<tr>
<td>/d/</td>
<td>dad</td>
<td>2432</td>
<td>841</td>
<td></td>
</tr>
<tr>
<td>/dh/</td>
<td>they</td>
<td>2376</td>
<td>896</td>
<td></td>
</tr>
<tr>
<td>/dx/</td>
<td>butter</td>
<td>1864</td>
<td>634</td>
<td></td>
</tr>
<tr>
<td>/eh/</td>
<td>big</td>
<td>3277</td>
<td>1247</td>
<td></td>
</tr>
<tr>
<td>/el/</td>
<td>bottle</td>
<td>951</td>
<td>343</td>
<td></td>
</tr>
<tr>
<td>/en/</td>
<td>button</td>
<td>630</td>
<td>216</td>
<td></td>
</tr>
<tr>
<td>/epi/</td>
<td>openthetic closure</td>
<td></td>
<td>908</td>
<td>332</td>
</tr>
<tr>
<td>/er/</td>
<td>bird</td>
<td>/axr/</td>
<td>4138</td>
<td>1692</td>
</tr>
<tr>
<td>/ey/</td>
<td>bat</td>
<td>2271</td>
<td>802</td>
<td></td>
</tr>
<tr>
<td>/f/</td>
<td>bef</td>
<td>2215</td>
<td>911</td>
<td></td>
</tr>
<tr>
<td>/g/</td>
<td>gag</td>
<td>1191</td>
<td>452</td>
<td></td>
</tr>
<tr>
<td>/gh/</td>
<td>hay</td>
<td>/hv/</td>
<td>1660</td>
<td>561</td>
</tr>
<tr>
<td>/h/</td>
<td>hit</td>
<td>4248</td>
<td>1438</td>
<td></td>
</tr>
<tr>
<td>/ix/</td>
<td>roses</td>
<td>7370</td>
<td>2502</td>
<td></td>
</tr>
<tr>
<td>/iy/</td>
<td>beat</td>
<td>4626</td>
<td>1810</td>
<td></td>
</tr>
<tr>
<td>/jh/</td>
<td>judge</td>
<td>1013</td>
<td>295</td>
<td></td>
</tr>
<tr>
<td>/k/</td>
<td>kick</td>
<td>3794</td>
<td>1204</td>
<td></td>
</tr>
<tr>
<td>/l/</td>
<td>fed</td>
<td>4425</td>
<td>1858</td>
<td></td>
</tr>
<tr>
<td>/m/</td>
<td>mom</td>
<td>/em/</td>
<td>3566</td>
<td>1407</td>
</tr>
<tr>
<td>/nx/</td>
<td>none</td>
<td>/nx/</td>
<td>6896</td>
<td>2434</td>
</tr>
<tr>
<td>/ng/</td>
<td>sing</td>
<td>/eng/</td>
<td>1220</td>
<td>378</td>
</tr>
<tr>
<td>/ow/</td>
<td>boat</td>
<td>1653</td>
<td>601</td>
<td></td>
</tr>
<tr>
<td>/oy/</td>
<td>box</td>
<td>304</td>
<td>127</td>
<td></td>
</tr>
<tr>
<td>/p/</td>
<td>pop</td>
<td>2588</td>
<td>957</td>
<td></td>
</tr>
<tr>
<td>/q/</td>
<td>glottal stop</td>
<td>deleted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/r/</td>
<td>red</td>
<td>4681</td>
<td>1859</td>
<td></td>
</tr>
<tr>
<td>/s/</td>
<td>sis</td>
<td>6176</td>
<td>2172</td>
<td></td>
</tr>
<tr>
<td>/sh/</td>
<td>shoe</td>
<td>1317</td>
<td>460</td>
<td></td>
</tr>
<tr>
<td>/sil/</td>
<td>silence</td>
<td>/h#/, /#h/, /pau/</td>
<td>891</td>
<td>366</td>
</tr>
<tr>
<td>/t/</td>
<td>tot</td>
<td>3948</td>
<td>1367</td>
<td></td>
</tr>
<tr>
<td>/th/</td>
<td>thief</td>
<td>745</td>
<td>259</td>
<td></td>
</tr>
<tr>
<td>/uh/</td>
<td>book</td>
<td>500</td>
<td>215</td>
<td></td>
</tr>
<tr>
<td>/uw/</td>
<td>boot</td>
<td>/ux/</td>
<td>1952</td>
<td>572</td>
</tr>
<tr>
<td>/v/</td>
<td>very</td>
<td>1994</td>
<td>710</td>
<td></td>
</tr>
<tr>
<td>/vcl/</td>
<td>voiced closure</td>
<td>/bcl/, /dcl/, /gcl/</td>
<td>7219</td>
<td>2553</td>
</tr>
<tr>
<td>/w/</td>
<td>wet</td>
<td>2216</td>
<td>903</td>
<td></td>
</tr>
<tr>
<td>/y/</td>
<td>yet</td>
<td>995</td>
<td>376</td>
<td></td>
</tr>
<tr>
<td>/z/</td>
<td>go</td>
<td>3682</td>
<td>1236</td>
<td></td>
</tr>
<tr>
<td>/zh/</td>
<td>measure</td>
<td>149</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>48</td>
<td>132833</td>
<td>48072</td>
</tr>
</tbody>
</table>

Table 5: List of phonemes, folding and statistics
During recognition, errors or confusions among the following phonemes were ignored: \{/sil/, /cl/, /vcl/, /epi/\}, \{/el/, /l/\}, \{/en/, /n/\}, \{/sh/, /zh/\}, \{/ao/, /aa/\}, \{/ih/, /ix/\}, and \{/ah/, /ax/\}. Performing these foldings gives a total of 39 classes.

4.3 Time lags and embedding dimension

As described in section 3.1, Takens’ theorem gives the sufficient condition for the size of the embedding, but does not specify a time lag. In practice, the determination of the time lag can be difficult, because there is no theoretically well-founded method to ascertain it, where an obvious criterion function can be formulated. Despite this caveat, two heuristics have been developed in the literature for establishing a time lag: the first zero of the autocorrelation function and the first minimum of the auto-mutual information curve [8]. These heuristics are premised on the principle that it is desirable to have as little information redundancy between the lagged versions of the time series as possible. If the time lag is too small, the attractor structure will be compressed and the dynamics will not be apparent. If the time lag is too large, the attractor will spread out immensely, which also obscures the structure as evident from the figure below.

![Figure 15: Time lag comparison in RPS](image)

Utilizing qualitative inspection of many phoneme attractors and by looking at the first zero of the autocorrelation and first minimum of the auto-mutual information curves for
these phonemes, a time lag of 6 (τ = 6) was determined to be appropriate for all subsequent analysis [49]. This choice also is consistent with the results in [50]. A time lag of 6 is equal to 375 μs delay for data sampled at 16 kHz.

As explained previously, the proper embedding dimension must also be chosen to ensure accurate recovery of the dynamics, since the original dimension of the phase space of speech phonemes is unknown. Again, there is no theoretically proven scheme for discovering it, but a common heuristic known as false nearest neighbors is typically used as explained in [8]. The false nearest neighbors algorithm calculates the percentage of false crossings of the attractor as a function of the embedding dimension. False crossings indicate that the attractor is not completely unfolded. The algorithm works by increasing the dimension of the embedding until the percentage of false crossings drop below some threshold.

Five hundred random phonemes were selected from the training partition of the corpus. The false nearest neighbors algorithm given in [51] was executed on them using a threshold of 0.1 % and a time lag of 6 (τ = 6) resulting in the histogram shown below.

Figure 16: False nearest neighbors for 500 random speech phonemes
The mode of this distribution is at an embedding dimension of $5 \ (d = 5)$, and this was the embedding dimension chosen. In addition, this choice is consistent with the results in [23]. Incidentally, the $x^{(5,6,\&;/d)}_n$ and $x^{(5,6,\&;\Delta)}_n$ actually constitute a 10 dimensional RPS, because the trajectory information is simply linear combinations of time delayed versions of time series. For comparison, a $(d = 10, \tau = 6)$ feature vector, $x^{10,6}_n$ was used as well.

The choice of time lag and embedding dimension are not independent. In practicality, certain time lags will permit the attractor to unfold at a lower dimension than others. This allows for a smaller embedding, which is advantageous for subsequent analysis. Furthermore, a RPS may be embedded at a dimension less than the sufficient condition of greater than $2d$, if the dynamics of interest reside on a manifold of a lower dimension. Although proper reconstruction of the dynamics is in general a relevant issue, the goal of this work is to find a time lag and embedding dimension that yield favorable classification accuracy in the final analysis.

### 4.4 Number of mixtures

In order to determine how many mixtures are necessary to model the RPS derived features, isolated phoneme classification experiments were performed. The feature vector, $x^{(5,6)}_n$, described in section 3.2.1, was modeled using a one state HMM with a GMM state distribution. Training was conducted using the training set for a particular number of GMM mixtures, and then testing was performed over the testing set described in section 4.2. A classification experiment was carried out after each mixture increment and parameter retraining. The method employed to increment the number of mixtures was the
binary split scheme explained in section 2.2.1. The empirical classification accuracy can then be plotted as a function of the number of mixtures as displayed below.

![Image: Classification accuracy vs. number of mixtures for RPS derived features](image)

**Figure 17: Classification accuracy vs. number of mixtures for RPS derived features**

Upon inspection of the graph, it is evident that the classification accuracy asymptotes at approximately 128 mixtures, where the ‘elbow’ of the plot is located. If the number of mixtures were further increased past 256 mixtures, test accuracy would at some point drop due to overfitting. The number of mixtures was therefore set at 128 for all following experimentation. A 128 mixture GMM appears to retain a balance between the complexities of the distribution of the RPS derived features and the overfitting issue elucidated in section 2.2.1.

### 4.5 Baseline systems

Whenever a novel experimental system, especially in ASR, is designed, built, and implemented, it must be compared to a baseline system. Ideally, the baseline system
should be the state-of-the-art system available at that time and the architecture of that system should be well known in the research community at large. In this case, the baseline system that is used for comparison was very similar to the one described in sections 1.1 and 2.3. This baseline system is familiar and frequently considered as the standard in most of the literature [1-4], although it is not necessarily “state-of-the-art” in the sense that it incorporates all known methods ever shown to boost accuracy.

A summary of the parameters used for MFCC feature computation is as follows: a pre-emphasis coefficient of 0.97, a Hamming window with a size of 25 ms (400 time samples), frame-speed/overlap size of 10 ms (160 time samples), and 24 triangular filter-banks. It should be noted that these parameter choices are common. The precise HTK configuration file can be found in the appendix. The model used was a one state HMM with a GMM state distribution. The mixture number was set to 16 in all experiments using the MFCC features, which is quite typical. Mixture incrementing for these baseline systems was executed by increasing the number of mixtures in the model by one. Following this, incremental retraining was performed until 16 mixtures were reached.

### 4.6 Direct comparisons

To test the performance of the classifier systems that use the RPS derived features and the MFCC features, experiments were carried out using the respective feature vectors alone. Performance is evaluated by simply examining the number of correct classifications each system achieved using the various features with their appropriate classifier systems lay out beforehand. The summary of the performance is displayed below. Confusion matrix information on these experiments is given in appendix.
As apparent from the table, the RPS derived feature sets attained approximately ~75\% the accuracy of the baseline, which translates into ~9182–7532 more correct classifications for the baseline. Adding the extra 5 dimensions to $x_n^{(5,6)}$, improved the accuracy by ~2.5\%. The appended RPS derived features that contained additional trajectory information elements boost the $x_n^{(5,6)}$ feature vector accuracy by more than ~7\%.

### 4.7 Stream weights

In order to determine the correct stream weights to model the joint feature vector, $y_n$, the testing accuracy was measured as a function of the stream weight, $\rho$, in equation (3.3.2). The mixtures in the stream model 1 ($s=1$), which contained the RPS derived features, were incremented using the binary split method, while the other stream model...
(s = 2), which contained the MFCC feature set was incremented one mixture at a time simultaneously with stream model 1. The stream model was initially trained up to the desired number of mixtures, which was $M_1 = 128, M_2 = 16$ using $\rho = 0.5$. For all the other values of $\rho$ that were tested, $\rho$ was varied and then retraining took place using the parameters from the $\rho = 0.5$ model as the seed. Doing the training in this manner eliminates the need to retrain each model from a flat start, since the mixture clusters are approximately in an acceptable location anyway [5]. As apparent from equation (3.3.2), when $\rho = 0$, the model is essentially equivalent to the classifier system that uses $x^{(5,6,\&A)}_s$, and when $\rho = 1$, it is approximately equal to the MFCC feature set, $O$, except the MFCCs were replicated according to equation (3.2.8). This makes the $\rho = 1$ stream model in effect on par as the baseline.

The plot of the testing accuracy versus $\rho$ is shown below.

![Figure 18: Accuracy vs. stream weight](image-url)
As labeled in the figure, the peak of the plot is at $\rho = 0.25$.

### 4.8 Joint feature vector comparison

#### 4.8.1 Accuracy results

The table below summarizes the comparisons made for values of $\rho$, the stream weight.

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>Test set accuracy (48,072 total test exemplars)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak value of joint feature vector</td>
<td>0.25</td>
</tr>
<tr>
<td>Baseline</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 7: Comparisons for different stream weights**

The higher accuracy was 57.85 % when $\rho = 0.25$. This ultimately results in a difference of 1350 more training exemplars correctly classified by the joint feature beyond those of the baseline.

#### 4.8.2 Statistical tests

Two statistical tests were executed over the joint feature vector results on the $\rho = 0.25$ and $\rho = 1$ classifiers. The first statistical test poses the question: Is the true error of $\rho = 0.25$ classifier smaller than the error for the $\rho = 1$ classifier under the assumptions that the testing exemplars are independent and drawn from the same distribution as the training exemplars? Using the confidence interval methods described in [41, 42], the error is different to a significance level $\gg 0.99$.

The other statistical test poses the question: Which classifier will achieve better accuracy on a new set of testing exemplars drawn from the same task and problem domain
(e.g. performance on more hypothetical data collected identically in the manner in which TIMIT was acquired, under the same conditions, equipment, etc.)? Using this test outlined in [42], the \( \rho = 0.25 \) classifier will perform better than the \( \rho = 1 \) classifier to a significance level \( \gg 0.99 \).

4.9 Summary

This chapter presented the experimental framework and results for the isolated phoneme classification experiments that were run using both RPS derived features and baseline MFCC features. The software and data (HTK and TIMIT) utilized are well known in the community. The time lag, embedding dimension, mixture weights, and stream weights were determined using empirical methods. Direct comparisons and joint feature vector comparisons were made using overall accuracy, confusion matrices, and other statistical tests. The next chapter will interpret these results, draw conclusions, and propose possible future work in this area.
5. Discussion, Future Work, and Conclusions

5.1 Discussion

The first set of results to analyze is the direct comparisons between the RPS derived features and the MFCC features. The MFCC feature sets obviously outperformed the RPS derived feature sets, achieving approximately between 13-23 % increased accuracy.

Despite this fact, there are several fascinating aspects of the results that warrant attention. First, these results affirm the discriminatory power of the RPS derived features. The natural distribution contains some information about which phoneme was uttered. Additionally, the statistical models (GMM/HMM) are able to capture at least a portion of the information theoretically present in the RPS (see section 3.1). Second, the results demonstrate that the RPS derived features can generalize to a speaker independent task. This fact is particularly interesting considering the fact that the excitation source and phase information were retained for the RPS derived feature calculation, since the method is time domain based. MFCC feature computation, naturally, discards the phase information and removes the excitation source via liftering, which is filtering in the quefrency domain. Lastly, the inclusion of the trajectory information boosted the accuracy over the natural distribution alone. The trajectory information inclusion confirms that the temporal change of the feature vectors also can be employed for discrimination. The ten-dimensional natural distribution feature vector demonstrates that proper choice of time delay coordinates can aid in recognition, since the ten-dimensional natural distribution feature vector only achieved 2.5% improvement, whereas the trajectory information boosted accuracy by 7 %.
The results obtained from the joint feature vector provide insight into the information content of it. The $\rho = 0.25$ classifier attained a 2.81% increase in accuracy over the $\rho = 1$ system (baseline). This result suggests that the RPS derived features contain discriminatory information not present in the MFCC features, otherwise the result would have been the same or lower. The statistical tests further affirm these results by providing knowledge about the bounds on the error and the expected performance of the classifiers on more testing data.

Another interesting characteristic of these results is the shape of the curve. The curve implies that as soon as RPS derived features are incorporated, the accuracy increases over the baseline. This fact, along with where the peak accuracy occurs, shows that additional information, beyond that contained in the MFCC features, is actually present, but that it needs to be combined in an intelligent way.

5.2 Known issues and future work

5.2.1 Feature extraction

There are two main but related issues that arise when computing the RPS derived features: amplitude scaling and the absence of an energy measure. Due to the nature of the technique, the features are based in the time domain, and therefore, the dynamic range of the time series affects the range of the data in the RPS. The dynamic range or amplitude of the signal is known to be irrelevant to the classification process, because the amplitude is affected by experimental nuances such as the distance the speaker is from the microphone during data collection. One advantage of the MFCC features is that they are totally invariant to this issue, because the energy is normalized out on a frame-by-frame basis. In the case of the RPS derived features though, the problem is partially solved by using the
normalization procedure described in section 3.2, but future work could include discovering a robust method to make this approach independent of amplitude scaling effects. This issue is of particular concern when performing continuous recognition, because there is no way to normalize the data on a phoneme-by-phoneme basis, since the time boundary information would not be present. This concern seriously hampers effective implementation for a continuous speech task.

The other related issue is that of computing an energy measure. Again, the MFCC features incorporate an energy measure by computing it for each frame, given in equation (2.1.10). For the proposed RPS derived features though, there is no analysis window, and therefore, no way to compute a meaningful energy measure that can be incorporated directly into the feature vector. One method would be to compute the energy over an entire phoneme, and then replicate it for each feature vector. But, once again, for continuous recognition, phoneme boundaries are unknown, and computing such a measure may be difficult. To reiterate then, both of these issues must be addressed in order for the RPS derived features to have long-term success for speech recognition applications.

Another notion that warrants further study is whether a higher dimensional RPS would improve classification accuracy. All of the experiments presented here utilized a five and ten dimensional RPS, whereas a larger dimensional RPS could further expand out the characteristic attractor structure. The premise is that a larger dimensional RPS may produce bigger differences between phoneme attractors that would have been perhaps overlapping in a smaller dimensional RPS. Further experimentation could be performed on time lags as well especially on how they relate to speaker variability.
All of the methods presented here can be applied to any arbitrary time series that has originated from a dynamical system. Investigation into the implementation of speech models, similar to the source-filter model employed in the linear regime, but applied to the RPS may produce valuable results. Also, features could be extracted from a frame that uses a RPS as processing space such as higher-order statistical moments. Such a feature set could be integrated into the existing frame based ASR systems with relative ease.

5.2.2 Pattern recognition

In addition to modifications that could be made in the feature extraction process, specific pattern recognition improvements could be employed to further expand the methods. One such method would be the implementation of a fully connected HMM. By substituting the simple one state HMM model (128 mixture GMM state distribution), with a 128 state HMM (single mixture GMM state distribution), the trajectory information could be captured through the convenient statistical framework of the transition probabilities. The transition probability parameters would represent the probability of attractor moving from one neighborhood of the RPS to another. The initial experimentation into this approach found that the method can be effective, but the computational cost is very high, because the model is complex. The computation time of such an approach would have to be reduced to make the method feasible. Also on the topic of trajectory, higher-order trajectory information such as delta-deltas could boost accuracy similar to the result of the first delta presented here.

The method of data fusion or classifier combination implemented was a stream model. Although this method is well founded, it is rather naïve and does not necessarily
do the best job of unifying the knowledge extracted from radically different processing spaces. An improved method of fusing the data would be desirable.

5.2.3 Continuous speech recognition

In addition to the isolated phoneme classification experiments presented and described here, the task of continuous speech recognition using the RPS derived features was examined. Above and beyond the issues of amplitude scaling and energy measure calculation, there is another difficulty that occurs when attempting to employ the RPS derived features for a continuous speech recognition task. The issue is state duration. Implicit to the approach of the MFCC features is the use of an analysis window, which automatically enforces a duration during continuous speech recognition; the fastest transition that the recognizer can make is ~10 ms, since that is the speed at which the feature vector changes in time. A ~10 ms speed generally coincides with the amount of time that any particular speech phoneme is stationary and relates to the rate at which phonemes change in time. This implicit duration then makes the classic left-to-right HMM with constant self-transition probabilities functional. In case of the RPS derived features, though, the feature vector changes on a time sample-by-time sample basis, and therefore this implicit duration that is present in the MFCC features no longer exists. Some resolution to the issue of state duration must be instituted in order for the RPS derived features to have ultimate success in the future for continuous speech recognition applications.

5.3 Conclusions

This thesis has presented a novel technique for speech recognition using features extracted from phase space reconstructions. The methods have a sound theoretical justification provided by the nonlinear dynamics literature. The specific approach transfers the
analytical focus from the frequency domain to the time domain, which presents a radically unique way of viewing the speech recognition problem and offers an opportunity to capture the nonlinear dynamical information present in the speech production mechanism. The features derived from the RPS are created using the natural distribution and trajectory information of phoneme attractors. By using statistical models (GMM/HMM), the salient information contained in the RPS derived features can be captured for subsequent classification.

The experiments that were run affirm the discriminatory power of the RPS derived features. The direct comparisons demonstrated the potential of these features for speech recognition applications. The joint feature vector results imply that the information content between the RPS derived features and the MFCC feature sets are not identical, because the accuracy was boosted over the baseline.

As a direct outcome of this work, several possible future research avenues were discovered in conjunction with issues that are inherent to the method. Questions of amplitude scaling, energy measures, better modeling techniques, and state duration present a number of interesting areas of continued investigation.

Overall, this work deviates strongly from mainstream research in speech recognition. This research has extended the fundamental understanding of the speech recognition problem and simultaneously expanded the knowledge of the nonlinear techniques presented for classification applications. In conclusion, reconstructed phase space analysis is an attractive research avenue for increasing speech recognition accuracy as demonstrated through the results, and future work will determine its overall feasibility and long-term success for both isolated and continuous speech recognition applications.
6. Bibliography and References


Appendix A – Confusion Matrices

In addition to inspection of the raw accuracy numbers (e.g. # of correct classifications/# of attempted classifications), other, more detailed results can be compiled, which give supplementary information on the performance of the classifier system. On such representation that reveals the specific classification errors or class confusions is a confusion matrix.

A confusion matrix is formed by creating rows and columns for each class in the set. The rows represent the expert labels or true classes of the testing exemplars, while the columns correspond to the classifier system’s output or it’s hypothesized class label. The confusions are then tabulated and inserted into the correct position in the matrix. For example, looking at the table below, the phoneme ‘/p/’ was classified correctly twelve times, and classified as ‘/r/’ twice, ‘/s/’ once, and ‘/sh/’ never. It is clear therefore, that the main diagonal of the matrix, displayed in table form, are the correct classifications, while the off-diagonal elements are the errors. Consequently, the confusion matrix uncovers correlated errors and further illustrates how a classifier is performing on each class individually. The confusion matrices present here were generated automatically using HTK.

<table>
<thead>
<tr>
<th></th>
<th>/p/</th>
<th>/r/</th>
<th>/s/</th>
<th>/sh/</th>
</tr>
</thead>
<tbody>
<tr>
<td>/p/</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>/r/</td>
<td>4</td>
<td>24</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>/s/</td>
<td>0</td>
<td>1</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>/sh/</td>
<td>0</td>
<td>3</td>
<td>7</td>
<td>21</td>
</tr>
</tbody>
</table>

Table 8: Example of a confusion matrix

Note: The notation for the various feature vectors labeled in following confusion matrices is given in Chapter 4.
Table 9: Confusion matrix for $X_n^{(5,6)}$

<table>
<thead>
<tr>
<th>Class</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>25</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>4</td>
<td>9</td>
<td>11</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>f</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>g</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>h</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>i</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>j</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Overall Results

<table>
<thead>
<tr>
<th>Class</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>25</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>4</td>
<td>9</td>
<td>11</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>f</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>g</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>h</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>i</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>j</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 10: Confusion matrix for $X_n^{(10,6)}$

<table>
<thead>
<tr>
<th>Class</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>25</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>4</td>
<td>9</td>
<td>11</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>f</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>g</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>h</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>i</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>j</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Term</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 11: Confusion matrix for $X^y_{5, 6, 5/d}$

<table>
<thead>
<tr>
<th>Term</th>
<th>3</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 12: Confusion matrix for $X^y_{5, 6, 5/A}$

<table>
<thead>
<tr>
<th>Term</th>
<th>3</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 15: Confusion matrix for $y_2$, $\rho = 0$

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>214</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>188</td>
<td>16</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>188</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>188</td>
<td>16</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>188</td>
</tr>
</tbody>
</table>

Table 16: Confusion matrix for $y_2$, $\rho = 0.25$

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>214</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>188</td>
<td>16</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>0</td>
<td>188</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>188</td>
<td>16</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>188</td>
</tr>
</tbody>
</table>
Appendix B – Code Examples

This appendix provides some of the important computer files and code used in this work. Most of the experiments were carried out using HTK and MATLAB. The first script was used to generate the MFCC feature set using HTK. This script can be used for the purposing of duplicating the baseline.

```
SOURCEKIND     = WAVEFORM
SOURCEFORMAT   = HTK
SOURCERATE     = 625
ZMEANSOURCE    = FALSE
TARGETKIND     = MFCC_E_D_A
TARGETFORMAT   = HTK
TARGETRATE     = 100000
SAVEWITHCRC    = TRUE
WINDOWSIZE     = 250000.0
USEHAMMING     = TRUE
PREEMCOEF      = 0.97
USEPOWER       = FALSE
NUMCHANS       = 24
LOFREQ         = -1.0
HIFREQ         = -1.0
LPCORDER       = 12
CEPLIFTER      = 22
NUMCEPS        = 12
RAWENERGY      = TRUE
ENORMALISE     = TRUE
ESCALE         = 1.0
```
The following code is written in MATLAB and can be used to generate the RPS derived features. The highest level call would be to “normalize(●)” after embedding the time series.

```matlab
function y = normalize(x);
global dim;           % global variable for the dimension of the RPS
centerOfMass = cm(x); % centroid or mean
for i = 1:dim
    x(:,i) = x(:,i)-centerOfMass(i); % subtract off the mean vector
end
RadialMoment = rg(x); % calculate the standard deviation of the radius
y = x./RadialMoment;  % divide off the standard deviation of the radius
return;

function phaseSpace = embed(timeSeries,lags)
N = length(timeSeries); % Determine total number of points in original time series
lags = [0 lags];        % Put the zero delay for the first element
Q = length(lags);       % Determine total number of dimensions
maxLag = max(lags);     % Determine the maximum lag
pointsInPhaseSpace = N - maxLag; % number of points in reconstructed phase space
% Create the reconstructed phase space
for i = 1:Q,
    lag = maxLag - lags(Q-i+1); % lags are subtracted from time index
    phaseSpace(i,(1:pointsInPhaseSpace)) = ...
    timeSeries(1+lag:pointsInPhaseSpace+lag);
end
return;

function y = rg(x)
y = sqrt(sum(sum((x.^2).'))/length(x(:,1))); return;

function y = cm(x)
global dim
n = length(x(:,1));
y = zeros(1,dim);
for i = 1:dim
    y(i) = sum(x(:,i));
end
y = y./n; return;
```