Speaker Recognition by Hidden Markov Models and Neural Networks

THESIS

Eric J. Zeek
Captain, USAF

AFIT/GCS/ENG/96D-31

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/96D-31

Speaker Recognition by Hidden Markov Models and Neural Networks

THESIS

Eric J. Zeek
Captain, USAF

AFIT/GCS/ENG/96D-31

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.

AFIT/GCS/ENG/96D-31

Speaker Recognition by Hidden Markov Models and Neural Networks

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Computer Engineering

Eric J. Zeek, B.S.E.E.

Captain, USAF

December, 1996

Acknowledgements

I would like to dedicate this work to my wife Leslie and my son Eli. Leslie's loving support and understanding was vital to sustaining my efforts at AFIT. Eli's arrival, in the middle of the program, was the highlight of my time at AFIT.

Special thanks to Dr. John Colombi whose assistance helped make this thesis a success. He was crucial to my understanding of speaker recognition techniques and hidden Markov models. He was also instrumental in the effort to submit an article to IEEE Transactions on Neural Networks midway through my program.

Finally, I would like to thank the members of my committee, Dr. Martin DeSimio, Dr. Timothy Anderson, and my advisor, Dr. Steven Rogers. Their involvement throughout the thesis process was invaluable. Thank you for all your time and effort.

Eric J. Zeek

Table of Contents

List of Figures

List of Tables

Abstract

As humans, we develop the ability to identify people by their voice at an early age. Getting computers to perform the same task has proven to be an interesting problem. Speaker recognition involves two applications, speaker identification and speaker verification. Both applications are examined in this effort.

Two methods are employed to perform speaker recognition. The first is an enhancement of hidden Markov models. Rather than alter some part of the model itself, a single-layer perceptron is added to perform neural post-processing. The second solution is the novel application of an enhanced Feature Space Trajectory Neural Network to speaker recognition. The Feature Space Trajectory was developed for image processing for temporal recognition and has been demonstrated to outperform the hidden Markov model for some image sequence applications.

Neural post-processing of hidden Markov models is shown to improve performance of both aspects of speaker recognition by increasing the identification rate from 70.23% to 88.44% and reducing the Equal Error Rate from 3.38% to 1.56%. In addition, a new method of cohort selection is implemented based on the structure of the single-layer percceptron.

Feasibility of using Feature Space Trajectory Neural Networks for speaker recognition is demonstrated. Favorable identification results of 65.52% are obtained when using a large training database. The FST configurations tested outperformed a comparable HMM system by 12-24%.

Speaker Recognition by Hidden Markov Models and Neural Networks

I. Introduction

1.1 Background

As humans, we have developed the ability to identify people by merely hearing their voices. We can do this if they are in the same room with us, down the hall, on the telephone, or even talking through a personal address system. What makes this possible? What does our brain use to discriminate one person's voice from another?

It is easy to understand how to differentiate a male speaker from a female speaker because in most cases, the male's voice has a lower pitch. The problem becomes more difficult when trying to discriminate one particular male from a group of all male speakers. Maybe we can use the fact that one speaker has a Southern accent while the others do not. It could also be the case that the speaker pronounces certain words differently than other speakers. We have developed this discriminative ability and use it without giving it much thought. programming a computer or machine to perform the same task has been difficult.

Speech has a temporal component in that when a given word is spoken, sounds must be in a certain order. If these same sounds are produced in permuted order, they will not produce the same word. This temporal information will be the basis of the methods that are used in this thesis for speaker recognition. Similar to the analogy that sounds must be ordered to represent a given word, the manner that a speaker produces that order

is also temporally based. A good illustration is people with a Southern accent. Their pronunciation of certain words may be extended relative to that of New Englanders.

Speaker recognition research is divided into two applications. The first is speaker identification where given a sample of speech, the system finds the closest match in the database and reports it as the result. The second is speaker verification where someone makes a claim about their identity, and the system determines if the claim is valid. This thesis will address both areas.

1.2 Problem Statement

Develop and compare the performance of two temporally based speaker recognition systems; hidden Markov models and Feature Space Trajectory (FST) Neural Networks, each using neural post-processing.

1.3 Scope

The data used in this thesis is from the YOHO database. YOHO speech utterances are from a real-world office environment collected using a high-quality telephone handset. Each utterance consists of a combination lock phrase of the form "twenty-four, sixty-seven, eighty-two" with 8 kHz sampling and 3.8 kHz bandwidth [1].

This database was developed by ITT and is the largest supervised database of its type. YOHO has been configured to allow testing at the 75% confidence level for determination of meeting the 0.1% false rejection and 1.0% false acceptance criteria. This database contains 138 speakers (32 females and 106 males) from which data was collected over 14 sessions for each speaker [1]. For the purposes of this thesis, only the 32 females

will be used. Time constraints drove the need to work on a subset of the database. The entire set of females was chosen because it represents a much more complex problem than performing speaker recognition on a database of 16 males and 16 females.

Speaker recognition performance will be reported in terms of Equal Error Rate (EER). EER is defined as the point where the number of False Acceptances is equal to the number of False Rejections to a system.

1.4 Approach

The first step is the development of a simple, word-based HMM speaker recognition system. The single-layer perceptron (SLP) post-processor is also developed to determine if it provides any enhancement. The second step is the development of a word-based FST speaker recognition system. Once developed, the SLP post-processor is applied to determine the improvement in performance.

1.5 Thesis Organization

Chapter II provides background information on the methods used in this thesis. Chapter III contains a description of the methodology used in the accomplishment of this research. Chapter IV contains the results, and Chapter V contains a discussion of the results and suggestions for future work.

II. Background

2.1 Introduction

This chapter provides the necessary background information to understand the methods used in this thesis for the speaker recognition problem. To start, the YOHO database will be described along with feature generation. Next, hidden Markov models (HMM) will be discussed in detail as well as the Feature Space Trajectory (FST) Neural Network developed by Neiberg and Casasent [2–7]. In addition, the single-layer perceptron will be discussed due to its use as a post-processor following both the HMM and the FST.

2.2 YOHO Database

The YOHO database [1] was developed by ITT and is available from the Linguistic Data Consortium. It was created as a standard to be used in the development of speaker verification systems. The database contains speech from 138 speakers (32 female and 106 male) in the form of combination lock phrases. Each phrase consists of three numbers in the form: "ninety-seven, sixty-three, twenty-four." The vocabulary has been limited such that no value under twenty is permitted, the number eight cannot be used, doublets (twenty-two, thirty-three, etc.) are excluded, and the decade numbers (twenty, thirty, etc.) are not allowed.

The data was collected over a 3-month period in a real-world office environment. A telephone handset was used to collect the data with 8 kHz sampling and 3.8 kHz bandwidth [1]. Each subject took part in 4 enrollment sessions with 24 phrases per session and 10 verification sessions with 4 phrases per session.

2.3 Mel-Frequency Cepstral Coefficients

Mel-Frequency Cepstral Coefficients (MFCC) are used as features [8,9]. Coefficients are obtained from analysis frames 20 msec in length at 10 msec intervals. Twenty-four Mel frequency spectral coefficients (MFSC) are generated by twenty-four triangular filters. The filters are spaced linearly below 1 kHz and logarithmic above. This results in twenty-four MFSC which are reduced to twelve MFCC through application of a Discrete Cosine Transform. Log energy is appended to the twelve MFCCs for a baseline feature set of thirteen dimensions. The transitional coefficients delta and delta-delta are also appended to provided a thirty-nine dimensional feature vector for each frame, as shown below.

$$V_{1-39} = [\ MFCC_{1-12} \quad LogEnergy_{13} \quad \Delta_{14-26} \quad \Delta\Delta_{27-39}\ ] \tag{1}$$

2.4 Hidden Markov Models

The hidden Markov model (HMM) is a probabilistic technique for the modeling of temporal data [10]. HMMs consist of states that can be interconnected in different manners. Two ways that states may be connected include ergodic and left-to-right. The ergodic model is fully interconnected, meaning a transition may occur to any other state. The left-to-right, or Bakis, model has the constraint of starting in the first state, and finishing in the final state without going backwards.

There also exist two subtypes of HMMs, discrete and continuous, that describe the systems they are attempting to model. Discrete HMMs are characterized by a finite observation symbol alphabet and corresponding probability mass function. Continuous HMMs

are characterized by modeling the observations as continuous random variables with associated probability density functions.

2.4.1 Hidden Markov Model Parameters. The number of states and structure of the HMM are important characteristics, but there are additional parameters that are required to define an HMM.

1. N - The number of states of the model.
2. M - The number of observation symbols in each state.
3. π - Initial state distribution. Each entry corresponds to the probability of being in state q_i for the initial observation.
4. A - Transition Matrix (N x N). Each entry (a_{ij}) corresponds to the probability of transitioning from state q_i to state q_j.
5. B - Observation Symbol Distribution Matrix (N x M). Each entry (b_{ik} corresponds to the probability of being in state q_i and observing the symbol k.

Since the parameters N and M can be derived from the matrix dimensions, only the parameters π, A, and B are required to completely define an HMM. This definition is usually in the form of $\lambda = (\pi, A, B)$ [11].

2.5 Hidden Markov Model Building Blocks

There are three basic problems which must be solved to apply HMMs to any task [11]. The first is that given an HMM, λ, and an observation sequence, $\mathbf{O} = \{\mathbf{o_1}, \mathbf{o_2}, \ldots, \mathbf{o_T}\}$, what is the probability that the sequence came from this model, $P(\mathbf{O}|\lambda)$? For example, in a

speaker identification test, each speaker has a model and identification is accomplished by finding the best fitting model, as determined by the highest probability. The second is to find the optimal state sequence, $\mathbf{q}$, given $\mathbf{O}$ & λ. In our speaker identification example, this corresponds to finding the path through a given speaker's model that results in the highest probability. The third problem is to adapt the parameters of a given HMM, $\lambda = (\pi, A, B)$, to maximize $P(\mathbf{O}|\lambda)$. This problem is often referred to as training the HMM. In speaker identification, as with most other HMM applications, this is a crucial aspect. A poorly trained system results in poor results. The following algorithms solve these three problems.

2.5.1 Forward Algorithm. One way to calculate $P(\mathbf{O}|\lambda)$ is the Forward Algorithm. In this procedure, given $\mathbf{O}$ and λ you start with the first observation, $\mathbf{o_1}$, and work "forward" through the data sequence until the end is reached. This produces a probability which is the sum over all possible state sequences evaluated at $\mathbf{o} = \mathbf{o_t}$. The algorithm is shown below [11].

1. Initialization

$$\alpha_1(i) = \pi_i b_i(\mathbf{o_1}), \quad 1 \leq i \leq N \tag{2}$$

2. Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(\mathbf{o_{t+1}}), \quad \begin{matrix} \mathbf{1 \leq t \leq T-1} \\ \mathbf{1 \leq j \leq N} \end{matrix} \tag{3}$$

3. Termination

$$P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{4}$$

4. NOTE: The forward variable is defined as $\alpha_t(i) = P(\mathbf{o}_1, \mathbf{o}_2, ...\mathbf{o}_t, \mathbf{q}_t = i|\lambda)$ which represents the probability of the observation sequence $\{\mathbf{o}_1, \mathbf{o}_2, ...\mathbf{o}_t\}$ and state i at time t for the given model λ

2.5.2 Backward Algorithm. Instead of beginning with the first observation, we can start with the final observation and work our way "backward". This procedure is primarily used to produce the backward variable for use in training the HMM:

$$\beta_t(i) = P(\mathbf{o}_{t+1}\mathbf{o}_{t+2}...\mathbf{o}_T|\mathbf{q}_t = i, \lambda) \tag{5}$$

The backward variable is defined as the probability of the observation sequence $\{\mathbf{o}_{t+1}, \mathbf{o}_{t+2}...\mathbf{o}_T\}$ given state $\mathbf{q}_t = i$ and a model λ. Calculation of the backward variable follows [11].

1. Initialization

$$\beta_T(i) = 1, \quad 1 \leq i \leq N \tag{6}$$

2. Induction

$$\beta_T(i) = \sum_{j=1}^{N} a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j), \qquad \begin{matrix} t = T-1, T-2, ..., 1, \\ 1 \leq i \leq N \end{matrix} \tag{7}$$

3. Termination

$$P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \pi_i b_i(\mathbf{o}_1) \beta_1(i) \tag{8}$$

2.5.3 Viterbi Algorithm. The Viterbi Algorithm is used to find the most likely state sequence using an algorithm similar to that of the forward procedure. The difference is that instead of keeping track of probabilities for every possible path; it is only concerned

with the best path. An array is used to store the state sequence that corresponds to the optimal probability. The general procedure for the Viterbi Algorithm is found below [11]:

1. Initialization

$$\delta_1(i) = \pi_i b_i(\mathbf{o}_1) \quad 1 \leq i \leq N \tag{9}$$

$$\psi_1(i) = 0 \tag{10}$$

NOTE: $\delta_t(i)$ represents the highest probability for the observations $\{\mathbf{o}_1, \mathbf{o}_2 ... \mathbf{o}_t\}$ along a single path at time t and ends in state i. $\psi_t(i)$ is an array that holds the argument which maximizes δ for each t and $\mathbf{q}_t$.

2. Recursion

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(\mathbf{o}_t) \quad \begin{matrix} 2 \leq t \leq T \\ 1 \leq j \leq N \end{matrix} \tag{11}$$

$$\psi_t(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] \quad \begin{matrix} 2 \leq t \leq T \\ 1 \leq j \leq N \end{matrix} \tag{12}$$

3. Termination

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)] \tag{13}$$

$$\mathbf{q}_T^* = \arg \max_{1 \leq i \leq N} [\delta_T(i)] \tag{14}$$

4. Path (state sequence) backtracking

$$\mathbf{q}_t^* = \psi_{t+1}(\mathbf{q}_{t+1}^*) \tag{15}$$

In most cases, HMMs are implemented using log values to avoid underflow of the host machine. Underflow is caused by the extremely small probabilities that occur when dealing with large models and large amounts of input data.

2.5.3.1 Forced Viterbi Alignment. The Viterbi alignment procedure described above uses all possible state sequences to determine its score. A different approach is to 'force' only certain allowed state sequences. For example, assume an HMM system with word level models using the YOHO database. With the normal Viterbi procedure described in Section 2.5.3, all possible combinations of word models will be examined to determine the best match. In forced Viterbi alignment, only the words corresponding to the utterance will be used to determine the score. This research will use forced Viterbi alignment due to the constraint that the combination lock phrase being uttered is known to the system.

2.5.4 Baum-Welch Re-estimation. The problem of adapting the model parameters to maximize $P(\mathbf{O}|\lambda)$ can be solved in many different ways. One of the most well-known is the Baum-Welch Method. This procedure uses α, β, and γ from the forward and backward algorithms. γ is calculated from α and β using Equation 17, where it represents the probability of being in state i at time t for a given observation sequence, $\mathbf{O}$, and model λ.

The basic steps are to first determine $\xi_t(i,j)$, the probability of being in state i at time t and state j at time $t+1$. Taking the summation of $\xi_t(i,j)$ from $t = 1$ to $t = T$ - 1 yields the expected number of transitions from state i to state j in $\mathbf{O}$. A similar

summation of $\gamma_t(i)$ yields the expected number of transitions from state i in $\mathbf{O}$ [11].

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(\mathbf{o}_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(\mathbf{o}_{t+1})\beta_{t+1}(j)} \tag{16}$$

All the parameters are now in place to perform the re-estimation. The goal is to optimize the model for the given observation sequence. Therefore, π, A, and B must be recalculated. The equations are given below [11]:

$$\bar{\pi}_i = \gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)} \tag{17}$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)} \tag{18}$$

$$\bar{b}_j(k) = \frac{\sum_{\substack{t=1 \\ o_t=v_k}}^{T}\gamma(j)}{\sum_{t=1}^{T}\gamma_t(j)} \tag{19}$$

Application of these steps results in a new HMM, $\bar{\lambda}$. By replacing λ with $\bar{\lambda}$ this procedure can be applied iteratively until there is a minimal difference between λ and $\bar{\lambda}$ in successive iterations.

2.6 Feature Space Trajectory

The Feature Space Trajectory Neural Network was developed by Neiberg and Casasent [2] for application to the multi-class pattern recognition problem. To this point, it has only been used on images [2–7]. This research has focused on extending the techniques that allowed FSTs to work with images so that they could work with speech.

2.6.1 What is a Trajectory? The trajectory and what it represents is the heart of the FST. A trajectory is simply a series of interconnected points in feature space, as in Figure 1. These points are called vertices. Since these points are connected, there is some relationship between them. In the case of speech, these points correspond to the features from a given frame of speech. Therefore, by ordering the points according to their occurrence in a speech utterance, a trajectory naturally encodes the temporal aspect of the speech.

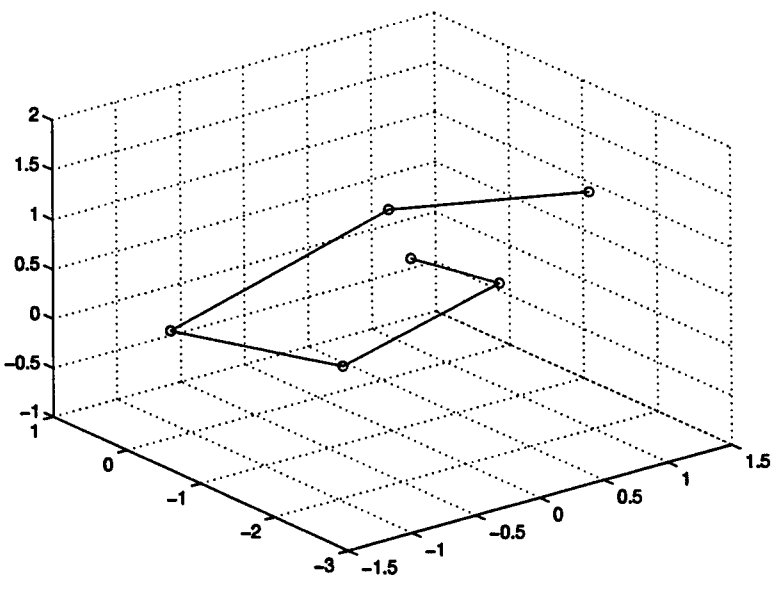


Figure 1. Sample Trajectory in Three Dimensional Space

2.6.2 How do FSTs Work? The first step in constructing an FST system to perform classification is to formulate a database of trajectories for each class of the problem. To perform classification, an unknown trajectory is compared to the database, and the trajectory that is the smallest distance from the unknown is classified as the winner. It is important to note that instead of comparing vertex to vertex, as would happen in a nearest

neighbor classifier, the FST compares an unknown vertex with the closest trajectory from the database [6].

2.6.2.1 Trajectory Creation. Each trajectory is constructed of multiple segments that are defined by a length, l_i, and direction, v_i [6]. In addition, the vector inner products, $c_{i,i+1}$ are required for distance calculations to the test data. These values are calculated based on the feature vectors from the training data, x_i.

$$l_i = ||x_{i+1} - x_i|| \tag{20}$$

$$v_i = \frac{x_{i+1} - x_i}{l_i} \tag{21}$$

$$c_{i,i+1} = x_i \cdot x_{i+1} \tag{22}$$

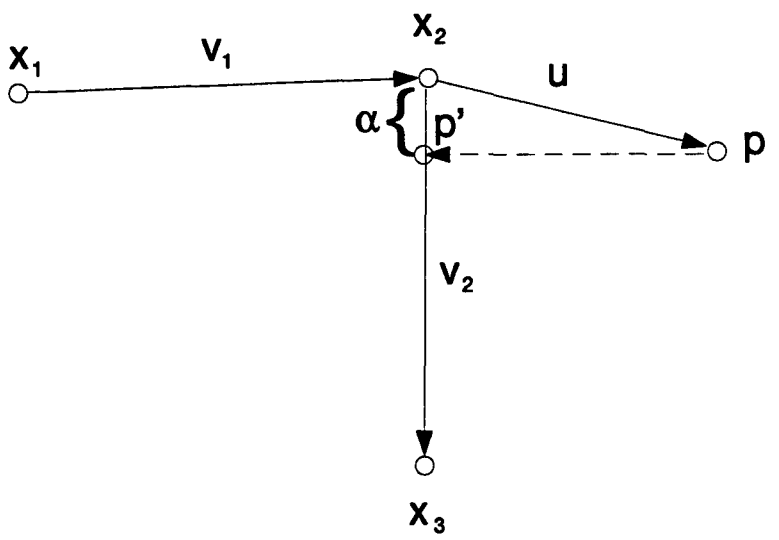


Figure 2. Illustration of Distance Calculation Geometry

2.6.2.2 Distance Calculations. When a test trajectory is tested against the training data, distances are calculated from the unknown vertices, p_i, to the closest segment of the training trajectory. In order to calculate the distance, some intermediate values must be calculated in order to project p_i onto the known trajectory, see Figure 2. The variable α is used to denote the position p' where p projects onto the segment v. In the equation below, u is defined as the distance from x to p.

$$\alpha = u \cdot v \tag{23}$$

There are three possible values for α [6]:

1. α is negative — the point does not fall on the segment. The distance is calculated to the segment start point.

$$d = ||x_i - p|| \tag{24}$$

2. α is positive and less than l — the point is on the segment and distance calculated from p to p'. Let $a = 1 - \frac{\alpha}{l}$; $b = \frac{\alpha}{l}$.

$$d^2 = p \cdot p - 2ap \cdot x_1 - 2bp \cdot x_2 + a^2 c_{1,1} + 2abc_{1,2} + b^2 c_{2,2} \tag{25}$$

3. α is positive and larger than l — the point does not fall on the segment and since the closest point is the endpoint, it is considered with the next segment.

For each of the vertices p_i, the minimum distances to the training trajectory are summed to get on overall distortion measure between trajectories [2]. The training trajectory which

has the minimal distortion value indicates class membership of the test trajectory. This implementation allows any vertex from the test trajectory to be mapped to any segment on the training trajectory.

2.6.3 Issues with regard to speech. There are two main problems that exist for applying FSTs to speech. In its original form, the FST is very effective when comparing trajectories that contain an equal number of vertices and segments. This makes the summation of minimum distances an acceptable metric. However, if the trajectories differ in size, the sum is no longer a valid distortion measure. The FST in its present form, allows for any point of the unknown trajectory to map to any segment on the training trajectory. That is, the first point may map to the third segment and the second point may map to an earlier segment. Since a speech signal is a temporal process, an order must be established where vertices cannot be mapped to earlier segments. The following sections address these issues.

2.6.3.1 Distortion Metric. The summation of minimum distances is not valid when comparing trajectories of different lengths; however, the mean of the minimum distances corrects this problem. The mean is a valid solution because it normalizes the distortion metric and provides a basis for comparison among trajectories of differing lengths. Consider the mean distance,

$$D = \frac{\sum_{i=1}^{N} d_i}{N} \tag{26}$$

where d_i represents the distance from vertex p_i of the test trajectory to the closest point on the training trajectory. The distance is calculated for all N vertices of the test trajectory.

2.6.3.2 Dynamic Time Warping. Dynamic Time Warping (DTW) has been used to compare speech utterances of different lengths for many years. However, DTW is another technique where vertices are compared to vertices. Therefore the FST must be adapted to provide DTW where an unknown vertex is compared to the closest trajectory. An elegant way of performing DTW 'on-the-fly' is by using Ney's algorithm [12]. This is a one-stage algorithm that has been applied to the problem of connected word recognition. It provides the advantages of word boundary detection and nonlinear time-alignment to enhance recognition performance. The advantage of performing this in the context of the FST algorithm is that errors which may be introduced by missed word boundary detection and improper time alignment are removed [12].

2.6.3.3 Template Generation. It is impossible for a person to say an utterance exactly the same way more than once. This leads to differences in the feature space vertices and therefore differences in trajectories. One approach for template generation may be to construct a database of every utterance for every person in the database. This procedure leads to a training database that is difficult to test against. For example, assume that trajectories will be constructed for each word and that each word will be spoken ten times in training. If we are trying to recognize just two combinations of words, the FST would need to check each of the ten trajectories of the first word with each of the ten trajectories of the second word resulting in 100 comparisons. If you factor in multiple speakers, say 10, the result is 1000 comparisons in order to determine the result. In contrast, if one template per word per speaker could be developed, the number of comparisons drops to 10; two orders of magnitude difference.

The problem is how to effectively reduce multiple utterances of one word into a trajectory that is representative of them all and still allows for discrimination from other speakers. One method for accomplishing this could be to apply Vector Quantization (VQ) to the training data to establish a codebook that represents the trajectory [11]. The codebook is easily generated by standard VQ techniques; however, due to the structure of the FST, the temporal information of the codewords must be maintained. Maintaining temporal information in an FST classifier will be a large focus in this effort and at present is the largest hurdle to overcome in applying FSTs to speech.

2.7 Neural Post-Processing

The optimal Bayes classifier makes use of the maximum *a posteriori* probability. Baum-Welch re-estimation only provides maximum likelihoods while FSTs produce minimum distance based decisions. Since perceptrons can provide outputs which approximate the maximum *a posteriori* probability [13], their use as a post-processor will be investigated. Benson and Bernander investigated this option in their work on speech recognition and achieved favorable results [14]. The single-layer perceptrons will accept the outputs from the HMMs or FSTs as inputs. Thirty-two output nodes will be used with one node for each speaker. In the identification experiment, the output node with the maximum value will be chosen as the identified speaker.

2.8 Summary

This chapter provided background material on the methods that will be used to implement the speaker recognition systems. The HMM was detailed and will be used as

a baseline system for comparison against the newly developed FST system. In addition, enhancement of the methods by adding an SLP post-processor to utilize the maximum *a posteriori* probability will be examined in Chapters III and IV.

III. Approach

3.1 Introduction

This chapter discusses the application of HMMs and FSTs to the speaker recognition problem as well as how these methods can be compared to one another.

3.2 Hidden Markov Model

In order to accomplish speaker recognition using HMMs, all of the basic blocks defined in Chapter II must be used. The Hidden Markov Model Toolkit (HTK) is used to accomplish this task. HTK was developed by Entropic Research Laboratory, Inc and a complete description can be found in [15].

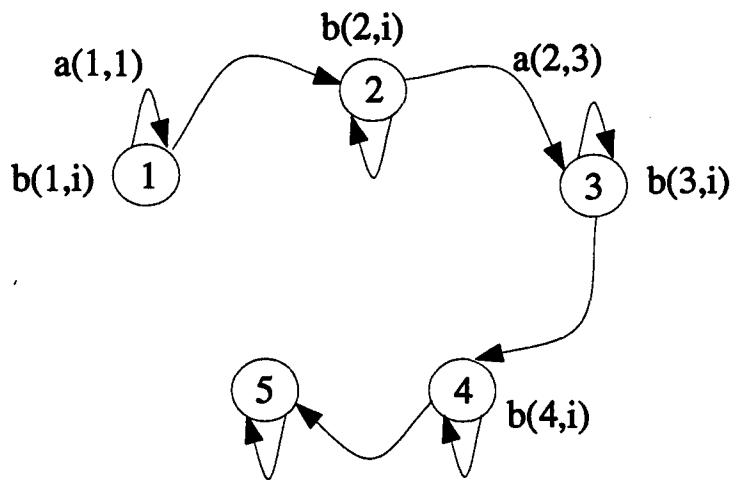


Figure 3. Five State Left-to-Right HMM

The HMM speaker recognition system consists of word models for each speaker. Five-state left-to-right models are created using HTK. All of the training data for the 32 females in YOHO are used to train the models and they were tested against the entire set of test data. Each speaker has 96 combination-lock phrases of training data and 40 combination-lock phrases of test data. A test consisted of running an utterance through each speaker's model and obtaining a log-likelihood score. This score is the basis of classification.

3.3 Speaker Identification

The speaker identification phase is the first step in a speaker verification system. The models that produce the maximum Viterbi log-likelihood value identify the unknown speaker. This corresponds to the basic Bayesian classifier (assuming equal priors) where speaker model i chosen to represent the speaker of a given utterance $\mathcal{U}$ [8].

$$i = \arg\max_{k} \{ \log p(\mathcal{U}|\lambda_k) \}, \ 1 \leq k \leq 32 \tag{27}$$

3.4 Speaker Verification

Speaker verification utilizes the results from the speaker identification phase for selecting reference speakers. Reference speakers, also known as cohorts, are used to normalize the log likelihood ratio of the test utterance using the claimed speaker's model [16–18]. The log-likelihood ratio $\mathcal{L}$ is derived from the Bayes optimal decision rule for classifying true speakers against impostors. Classification is made by comparing $\mathcal{L}$ to a threshold. Define the log-likelihood ratio of an utterance $\mathcal{U}$, with a reference "cohort" set size denoted

by $|\mathcal{C}|$, as the following approximation[19],

$$\mathcal{L}(\mathcal{U}) \equiv \log p(\mathcal{U}|\lambda_{claim}) - \frac{1}{|\mathcal{C}|}\sum_{j\in\mathcal{C}} \log p(\mathcal{U}|\lambda_j) \tag{28}$$

where λ_{claim} is the claimed speaker's model and λ_j is one model from the claimed speaker's cohort set. This has recently been called the *Geometric Mean* normalization [18].

3.4.1 Cohort Selection. The set of cohorts $\mathcal{C}$ will be selected as "close" speakers based on log-likelihood Viterbi scores using all enrollment data. Cohorts are selected for each speaker based on the smallest distortion metric using the three methods listed below.

1. *Difference of Means* [17] or (DOM) sorts by mean difference of log-likelihoods enrollment scores.

$$\begin{aligned} d_{DOM}(\lambda_i, \lambda_j) &\equiv \log \frac{p(\mathcal{U}|\lambda_i)}{p(\mathcal{U}|\lambda_j)} \\ &= \log p(\mathcal{U}|\lambda_i) - \log p(\mathcal{U}|\lambda_j) \end{aligned} \tag{29}$$

2. Reynold's [20] *Symmetric* method sorts on pairwise log-likelihood ratio enrollment information (If speaker i is "close" to speaker j then the reverse must be true).

$$d_{SYM}(\lambda_i, \lambda_j) \equiv \log \frac{p(\mathcal{U}_i|\lambda_i)}{p(\mathcal{U}_i|\lambda_j)} + \log \frac{p(\mathcal{U}_j|\lambda_i)}{p(\mathcal{U}_i|\lambda_j)} \tag{30}$$

3. Second order *Bhattacharyya* measure [21] sorts cohorts using the variance of the enrollment likelihoods, as well.

$$d_B(\lambda_i, \lambda_j) \equiv \frac{(m_i - m_j)^2}{4(\sigma_i^2 + \sigma_j^2)} + \frac{1}{2} \log \left(\frac{\frac{\sigma_i^2 + \sigma_j^2}{2}}{(\sigma_i^2 \sigma_j^2)^{\frac{1}{2}}} \right) \tag{31}$$

where m_i represents the enrollment likelihood mean and σ_i^2 represents the enrollment likelihood variance. Fielding [22] has shown how the use of second-order statistics can be useful for HMM model comparisons.

3.5 HMM with Single-Layer Perceptron Post-Processor

This approach takes advantage of the discriminative power of perceptrons. HMMs are very effective at modeling the temporal characteristics of speech; however, since they rely on maximum likelihood estimation (MLE), they are not necessarily discriminative. This approach employs a SLP as a post-processor to improve classification performance. The SLP accepts as inputs the log likelihoods produced by the HMMs and selects a speaker with the highest probability of having spoken the utterance under test.

In order for the SLP to be effective as a post-processor, the log likelihoods from the HMMs must be normalized. In general, these values are within a specific range without a large degree of separation. For the females from YOHO, the entire set of log likelihoods range from -80 to -61. In terms of probabilities, these values are different by nineteen orders of magnitude, but the log values decrease this distance measure. In order for perceptrons to train effectively, these value must be normalized.

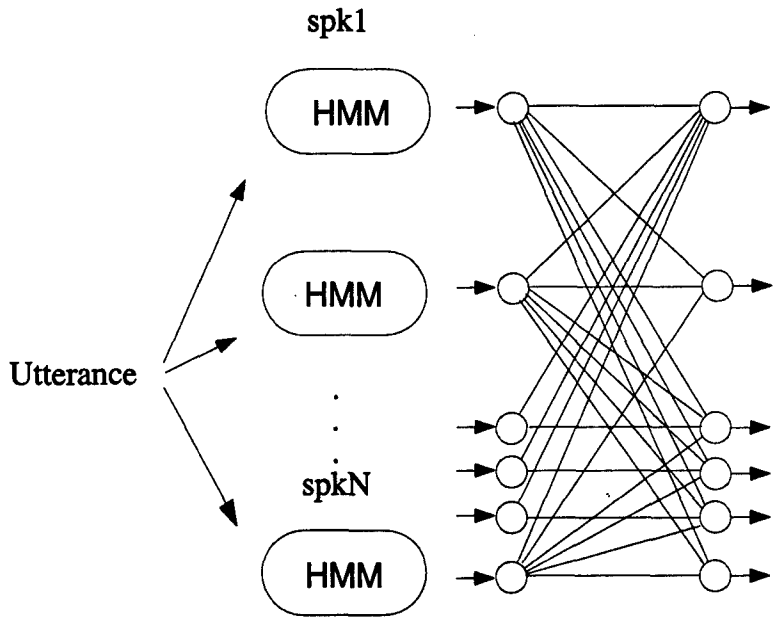


Figure 4. Configuration of Neural Network as Post-Processor

A widely accepted method of normalizing SLP input data is statistical normalization [23]. In this method, each feature, j, (log likelihood from a specific speaker) is normalized such that it has zero mean and unity variance, see Equation 32. This attempts to balance the discriminant nature of each feature with respect to all others.

$$a_{ij}^{s} = \frac{a_{ij} - \mu_j}{\sigma_j} \tag{32}$$

Although statistical normalization is the method used in this thesis, two others were investigated. The first is energy normalization. In this technique, each element, a_{ij} of the feature vector is divided by the vector magnitude such that the resulting vector is of unit magnitude, see Equation 33. Energy normalization attempts to capture discriminant information between elements of a single feature vector. The problem is that the relative

distance relationship with other samples is lost.

$$a_{ij}^{e} = \frac{a_{ij}}{\|a_i\|} \tag{33}$$

The second, compliment coding is a normalization technique which seeks a middle ground between the two techniques mentioned above. The first step is to energy normalize the features to obtain vectors of unit magnitude, as in Equation 33. Next, the compliment of each element is taken and appended to the vector as shown in Equations 34 and 35. This doubles the amount of features, but has the advantage of retaining relative distance information among samples.

$$a_{ij}^{c} = 1 - a_{ij} \tag{34}$$

$$I = [aa^{c}] \tag{35}$$

3.6 Cohort Selection Using Neural Post-Processing

Since the outputs of the SLP can approximate the *a posteriori* probabilities [13] and cohort selection attempts to find the speakers that are "close" to one another, it is possible to use the SLP outputs as criteria for selecting the cohort speakers. The SLP outputs are able to represent higher order relationships better than the raw log likelihoods. Since the HMM relies on determining the maximum likelihood, it can only capture a first order relationship. Therefore, the SLP has the advantage of using more information to make a decision.

In order to select cohorts using the SLP, each utterance is run through the HMMs with the resulting log likelihoods used as inputs to the SLP. Each output of the SLP is

associated with one of the speakers. These outputs of the SLP are ranked from highest to lowest, with the highest representing the speaker with the highest probability of saying the utterance [13], see Table 1. This process is performed on each test utterance. Once all utterances for a particular speaker are complete, the individual rankings are summed to get on overall ranking. This final ranking is used as the basis for cohort selection, see Table 2.

Table 1. Ranking of output node values for one utterance

Output Node	Output Value	Rank
1	0.9789	1
32	0.0679	2
29	0.0398	3
26	0.0305	4
20	0.0200	5
9	0.0094	6
4	0.0089	7
24	0.0079	8
6	0.0074	9
12	0.0065	10
8	0.0044	11
5	0.0044	12
27	0.0032	13
22	0.0030	14
7	0.0026	15
17	0.0025	16
2	0.0021	17
10	0.0018	18
3	0.0016	19
11	0.0015	20
19	0.0013	21
16	0.0007	22
23	0.0006	23
15	0.0005	24
14	0.0005	25
13	0.0004	26
30	0.0004	27
28	0.0002	28
25	0.0000	29
21	0.0000	30
28	0.0000	31
31	0.0000	32

3.7 Feature Space Trajectory

Prior to this effort, the Feature Space Trajectory (FST) Neural Network had not been applied to speech. Therefore, development of such a system must be undertaken in gradual steps. The first step necessary is to determine how speech signals can be transformed into trajectories.

Table 2. Ranking of output node values for all training utterances for speaker 1

Cohorts	Sum of Ranks	Rank
1	96	1
5	762	2
24	961	3
9	965	4
32	987	5
17	1081	6
20	1094	7
22	1174	8
26	1273	9
3	1284	10
23	1309	11
15	1368	12
12	1433	13
2	1504	14
7	1593	15
29	1659	16
16	1696	17
4	1702	18
10	1773	19
8	1784	20
25	1809	21
27	1833	22
6	1845	23
18	1884	24
28	1908	25
19	2061	26
13	2171	27
21	2187	28
31	2262	29
14	2290	30
30	2293	31
11	2467	32

3.7.1 Trajectory Construction. In the original FSTs used in image recognition, series of images are used to characterize an object and create a trajectory [2–7]. Conceptually, the trajectory is created by connecting sequential points in feature space via line segments. Adjacent points represent images that have a temporal order. An analogy can be drawn to speech signals in that features from consecutive frames of sampled speech can be used to create a trajectory.

The next question is at what level of speech should trajectories be constructed. It is possible to create trajectories that represent an entire utterance from YOHO. The problem is that every utterance is different in that one may be "Ninety-three, Fifty-seven, Thirty-two" while another may be "Twenty-four, Forty-six, Eighty-one." Even if the same person spoke both utterances, the trajectories are vastly different due to the difference in words that comprise them. To remedy this problem, word level trajectories have been chosen. By constructing trajectories at the word level, individual words can be concatenated to make every possible utterance in YOHO. If one trajectory is created for each speaker per word, the result is sixteen trajectories for each speaker. In contrast, creating trajectories for every possible utterance would require more than 350,000 trajectories per speaker. In addition to an abundance of storage space required, the training time for each speaker would be a limiting factor.

The next step is determining how to test the trajectories. One solution is to use every instance of each word as a training trajectory. For the word 'ONE' alone this results in 42 trajectories for each speaker in the database. When attempting to test an entire utterance, the number of combinations resulting from concatenating multiple word instances pushes the number of test runs to be large. However, using every instance is useful for development

of the basic techniques for applying FSTs to speech when limiting the testing to only one word from an utterance.

Using this idea for a proof of concept, the initial FST for speech could be developed. However, a second problem is produced. All instances of the word 'ONE' are of different lengths, resulting in different numbers of frames and therefore different length trajectories. The original FST for image recognition is designed to work on a single point or trajectories of equal lengths. In order to overcome this obstacle, a method of comparing utterances of different lengths must be developed.

3.7.2 Incorporating Ney's Algorithm. Ney's algorithm is an efficient way of performing Dynamic Time Warping (DTW) in a one-step process using Dynamic Programming [12]. By using DTW, two signals of different lengths can be compared to one another with one either being stretched or compressed to best match the template. The result from this is a distortion metric which indicates the distance between the two signals.

In Dynamic Time Warping (DTW), a matrix is created containing distances between points of two signals, see Figure 5(a). In the case of speech signals, the points represent feature space locations corresponding to individual frames from an utterance.

The idea is to find the minimum distance path through the matrix while applying certain constraints. The constraints determine the path that can be created. Figure 5(b) shows acceptable transitions and the weights associated with each move.

The idea is to reward transitions that keep the path on the upward diagonal while penalizing those moves that get away from the diagonal. The legal predecessors establish which moves are allowed and the cost of making each one. For example, a move on the

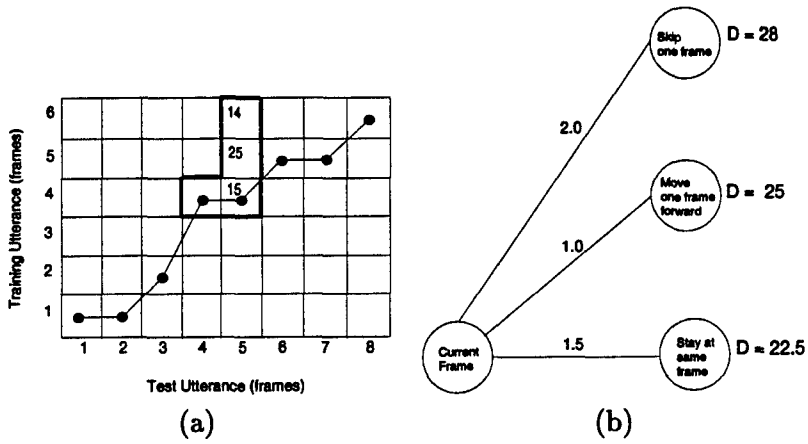


Figure 5. (a) is a representative DTW Distance Matrix and (b) represents the legal moves and their weights

diagonal is good and will not receive a penalty (weight = 1); however, to stay in the same frame or skip a frame ahead will receive a penalty. In Figure 5(b) staying in the same training frame has a weight of 1.5 and skipping a training frame has a weight of 2.0. The weights are important when determining which transition to make. Figure 5(b) shows the detailed distance measurements for movement from frame 4 to frame 5 of the test utterance, the area in the bold box of Figure 5(a). Without the weights, the transition chosen would be to skip one frame due to its distance of 14 which is lower than the values for moving one frame ahead or staying in the same frame. However, when the weights are considered, the transition chosen is to remain in the same frame. Staying in the same frame has a distance of 22.5, moving one frame forward has a distance of 25, and skipping one frame has a distance of 28; thus, the minimum distance of 22.5 forces the transition to remain in the same frame. The values shown in Figure 5 (b) result from multiplying the original distances (Figure 5(a)) by the associated weight.

The original FST finds the nearest point on the training trajectory to a test point. This is done without regard for where the closest point lies on the training trajectory. For speech, since temporal data is involved, it is important to consider whether a point from the end of one trajectory maps to the beginning of another or vice versa. In the same manner, once we have reached a given point in the speech data, we do not want to allow the point to map to an earlier segment of the trajectory. That violates the temporal nature of speech.

Toward this end, Ney's algorithm is incorporated to improve the FST so as to produce a Ney-based FST. In this version of the FST, comparison starts at the beginning of the training trajectory and moves forward just as in the left-to-right HMM. The only moves allowed are to stay on the same segment, move one segment ahead, or move two segments ahead, see Figure 5(b). Staying on the same segment corresponds to a test trajectory needing to be compressed. Moving ahead one segment is the desired result meaning that the test and training trajectories are proceeding at the same rate. Moving two segments ahead means that the test trajectory needs to be expanded to provide a better match to the training trajectory.

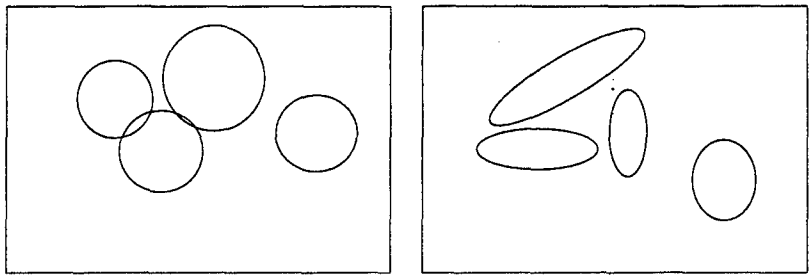
Using the proposed Ney FST, a proof of concept test is constructed where each of the first speaker's test instances of the word 'ONE' would be tested against every speaker's training instances of the word 'ONE.' This shows the ability to use FSTs with speech. However, it is unrealistic to use all training utterances as discussed previously. For the word 'ONE,' each speaker had 42 instances in training data. Therefore, to perform speaker identification requires testing 1344 trajectories. A method for reducing the size of the training set while maintaining performance is the next area of interest

3.7.3 Template Generation. In order to reduce the amount of training templates in the database, it is desirable to find a way of determining one trajectory which is representative of how a given speaker says each word of the YOHO vocabulary. This would result in 16 templates per speaker, one for each word, as opposed to the 576 trajectories resulting from using each instance of the word 'ONE'. Two methods of determining the optimum template are developed.

The first is to pick the instance which minimizes the distance to all other training instances of the word. This process involves using the Ney FST to obtain a distance measure from each training instance to every other training distance. The one that results in the minimum sum of distances to all trajectories is picked as the optimum template.

The second is to Vector Quantize (VQ) all of the training data for a given word from each speaker to produce a codebook. Here, VQ is performed two different ways. One uses a Euclidean distance measure while the other uses a Mahalanobis distance measure. These distances provide different results when creating the codebook, see Figure 6. The Euclidean distance measure assumes the clusters are spherical. Mahalanobis distance, on the other hand, makes no such assumption and uses the mean and variance of the data to determine the shape of the clusters and distances normalized relative to ellipse size and orientation [24].

Codebook generation follows the basic LBG VQ algorithm [11]. However, because the system seeks to take advantage of the temporal nature of the speech signal, a method of recovering the temporal ordering of the codewords is necessary. By applying an utterance to the codebook, it is possible to determine which codeword each frame of speech data mapped



(a) (b)

Figure 6. Illustrations of the cluster shapes as a result of (a) Euclidean distance distortion measure and (b) Mahalanobis distance distortion measure

to. If only one utterance is used, the solution could be obtained quickly by applying the mapping of that utterance to the codebook. However, since multiple utterances exist in the training data, a way of combining each of their mappings into one overall temporal ordering is required.

Two methods are employed to determine the codeword ordering from the mappings of each utterance to the codebook. When the mappings are obtained, it is known which codeword each frame of speech is mapped to. The idea is to determine which codewords are mapped to by frames from the beginning, the middle, and end of the utterance. By analyzing the mappings, it is possible to create the desired order. The first method used was to determine the mean of the frames which map to each codeword. The codeword with the lowest mean is placed first, and the codeword with the highest mean is placed last. In a similar manner, the median of the mapped frame values is also used. This can remove errors in the mean that may surface due to a few unreasonably high or low values. Once again, the codeword with the lowest median is placed first in the ordering and the highest median is placed last.

Therefore, based on Vector Quantization, four different methods of template generation are examined.

1. Euclidean distance with mean ordering.
2. Euclidean distance with median ordering.
3. Mahalanobis distance with mean ordering.
4. Mahalanobis distance with median ordering.

For comparison, the templates from each of the five template generation techniques are tested in the proof of concept experiment.

3.8 FST with SLP Post-Processor

The post-processor used with the FST is the same configuration as that used with the HMM. A single-layer perceptron is used that accepts 32 inputs and provides 32 outputs. The 32 inputs represents the distance scores from the test trajectory to the training trajectory for each speaker. This facilitates comparison to determine the performance enhancement provided.

3.9 Summary

This chapter defined two systems for accomplishing speaker recognition. Multiple cohort selection techniques were introduced including the newly developed SLP technique. This technique takes advantage of the probabilistic properties of perceptron outputs to provide an improved basis for cohort selection. A Ney-based FST was introduced which capitalizes on the temporal structure of trajectories by incorporating the time alignment

techniques of Ney's Dynamic Time Warping Algorithm. In addition, a manner for combining both the HMM and FST with an SLP post-processor was described. The SLP post-processor is used to improve the identification and verification performance of these systems.

IV. Results

4.1 Introduction

This chapter provides results of the speaker recognition techniques described in Chapters II and III. The first technique is the HMM applied to speaker identification. The second technique expands upon the first by using the speaker identification results from the HMM to perform speaker verification. The third technique demonstrates the FST in performing speaker identification. The chapter concludes with a direct comparison of the HMM and FST.

4.2 Hidden Markov Model Speaker Identification

In order to perform speaker identification, word level HMMs are constructed for each speaker. A test phrase is obtained from the YOHO database and the corresponding word models are arranged to set up for forced Viterbi alignment. The resulting scores from each speaker's model are then used to make the identification. The model with the highest Viterbi score is chosen as the identified speaker.

For baseline purposes, speaker level HMMs are also constructed. Similarly, these models consist of 5 states but are ergodic, see Figure 7. The key difference is that there is only 1 model per speaker; but with word level HMMs, there are 16 models per speaker to represent each of the 16 words from the YOHO vocabulary. This results in a large reduction in the number of free parameters. Testing is conducted in the same manner as with word level HMMs.

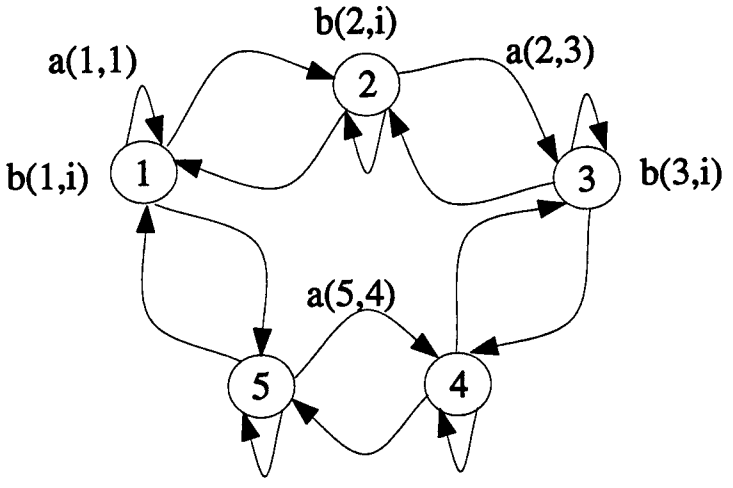


Figure 7. Five State Ergodic HMM - Not all connections shown

Table 3. Closed-set speaker identification Rates(%) for 1, 2, and 4 combination lock phrases using Equation 27.

Method	1	2	4
Speaker Models	70.23	80.16	86.25
Word Models	80.55	91.94	96.88

Speaker level models are very simple representations of the speakers in the database. Using only 5 states, they attempt to model every test utterance in YOHO. This leads to low system performance as shown in the first line of Table 3. Using one combination-lock phrase the speaker model identification rate was 70.23%. Although the rate increased to 80.16% and 86.25% when two and four combination-lock phrases were used, there is still more than 13% error in the system that can be eliminated.

To improve performance, one possible solution is to divide the larger problem into a series of smaller problems. This division is represented by creating HMMs at the word level.

In this manner, there are 5 states used to represent each word. Word level HMMs use a total of 80 states among the required 16 models compared to the 5 states of the speaker models. By providing more states, the performance of the system shows a substantial improvement in Table 3. There is a tradeoff that occurs between desired level of performance and number of free parameters. The word level models perform better but also require more calculations due the increased number of free parameters.

4.2.1 Neural Post-Processor. One way of enhancing the performance of simple HMM systems is to add neural network post-processing to the back-end of the system. In this thesis, single-layer perceptrons (SLP) are used. They are configured to accept 32 inputs and provide 32 outputs. The 32 inputs correspond to the Viterbi score provided by each of the HMMs.

Table 4. Closed-set speaker identification Rates(%) for 1, 2, and 4 combination lock phrases using SLP Post-processor.

Method	1	2	4
Speaker Models	88.44	94.22	96.88
Word Models	93.52	97.97	99.06

The SLP provides improved speaker identification in all cases, see Table 4. These results were obtained using statistical normalization. All normalization techniques discussed in Chapter III were examined, but statistical normalization outperformed them all.

An interesting note is that speaker level models with SLP post-processing perform as well as or better than word level models alone. The simple speaker representation results in lower template storage requirements and in the ability for the system to process the information more rapidly. The addition of the SLP post-processor does add some

overhead; but this is primarily during training. When testing, the results from HMMs are fed directly to the SLP and the calculations are basically instantaneous.

4.3 Hidden Markov Model Speaker Verification

The same HMMs used in the speaker identification experiment are used for speaker verification. The first part of the process is identical to the speaker identification described above. The difference is that during training, the training scores are used to create the "cohort" sets. These cohort sets represent the N speakers that are closest to a given speaker. That is, the system is likely to confuse one of the cohorts as the true speaker. By identifying these cohorts, we are able to remove them from the testing and provide better results as is standard in the speaker recognition community. Three methods of cohort selection are implemented with respect to the HMM outputs:

1. Difference of Means (d_{DOM})
2. Bhattacharyya (d_B)
3. Symmetric (d_{SYM})

The same methods are applied to word level and speaker level HMMs with the results shown in Table 5. Similar to the results from speaker identification, speaker verification performance improves through the use of word models. In all cases, the Equal Error Rate (EER) is reduced by using the word level models. The type of model does not effect the relationship of the results among the cohort selection techniques. For example, in Table 5 Bhattacharyya provided the lowest EER for five cohorts and one combination-lock phrase. This is true for both word and speaker models.

Table 5. Equal Error Rates for Speaker Verification using HMM

Method	$\|\mathcal{C}\|$	Speaker Models (Word Models)		
		1	2	4
d_{DOM}	1	19.92 (16.09)	17.50 (11.09)	15.94 (6.25)
	2	15.55 (11.95)	12.97 (8.12)	10.62 (5.31)
	3	12.73 (10.46)	10.16 (6.25)	8.12 (4.69)
	4	11.88 (9.90)	8.89 (5.91)	7.83 (4.06)
	5	11.42 (9.77)	8.90 (5.77)	7.19 (3.47)
d_B	1	21.56 (16.48)	19.53 (10.97)	17.80 (6.25)
	2	15.16 (12.97)	12.50 (8.47)	11.22 (5.00)
	3	14.37 (11.10)	11.43 (7.19)	9.63 (4.69)
	4	11.88 (10.23)	8.93 (5.78)	8.44 (3.75)
	5	11.17 (9.52)	8.27 (5.65)	6.92 (3.38)
d_{SYM}	1	22.34 (15.32)	20.64 (11.41)	18.73 (7.45)
	2	17.19 (12.81)	14.81 (8.26)	13.12 (5.37)
	3	15.30 (11.80)	12.66 (7.22)	10.62 (4.31)
	4	12.59 (10.63)	9.84 (6.56)	8.75 (4.38)
	5	11.95 (9.77)	9.34 (6.54)	7.23 (4.38)

4.3.1 Cohort Selection Using Perceptron Outputs. The neural post-processor provides another means of determining the cohort speakers. Under certain conditions, outputs of SLPs approximate *a posteriori* probabilities and therefore provide an enhanced basis for classification [13]. The three methods of cohort selection applied to the stand alone HMM systems are again employed. In addition, a new method of cohort selection based on the SLP outputs is used. The new method ranks the values of the output layer to provide a cohort list for each utterance. It then combines all test utterance rankings for each speaker to form an overall cohort set.

The results in Table 6 show for small cohort set size, speaker verification performance is degraded by the SLP outputs. However, as the cohort set increases in size, the performance improves past that of the systems without the SLP for the Difference of Means and Symmetric methods. The Bhattacharyya method experiences a reduction in performance. The new SLP cohort selection method provides the best EER in 63.3% of the cases exam-

Table 6. SLP Post-Processor Speaker Verification Equal Error Rates

Method	$\|\mathcal{C}\|$	Speaker Models (Word Models)		
		1	2	4
d_{DOM}	1	20.41 (18.20)	17.37 (13.15)	15.24 (9.12)
	2	15.78 (13.68)	13.27 (9.04)	10.63 (5.31)
	3	13.13 (10.23)	10.32 (5.94)	9.06 (3.39)
	4	10.70 (9.05)	8.13 (5.16)	6.52 (3.12)
	5	9.84 (7.34)	7.19 (4.22)	5.94 (1.88)
d_B	1	27.56 (19.92)	25.61 (15.16)	23.48 (11.28)
	2	20.15 (14.61)	17.54 (9.86)	15.37 (5.70)
	3	17.02 (10.72)	14.72 (6.41)	11.94 (3.07)
	4	14.46 (9.53)	11.91 (5.78)	10.60 (2.57)
	5	11.56 (7.89)	9.67 (4.53)	8.12 (1.59)
d_{SYM}	1	21.87 (19.84)	18.92 (15.30)	17.43 (10.62)
	2	17.73 (14.77)	15.19 (9.53)	12.81 (5.31)
	3	13.91 (10.94)	11.09 (6.70)	9.96 (4.06)
	4	11.58 (9.06)	8.94 (5.03)	7.52 (2.19)
	5	9.84 (7.97)	7.46 (4.36)	6.56 (1.56)
d_{SLP}	1	19.46 (18.58)	16.10 (13.91)	14.39 (9.98)
	2	15.14 (13.58)	12.50 (8.63)	10.25 (4.98)
	3	12.67 (10.00)	10.03 (5.47)	9.05 (3.15)
	4	11.39 (8.66)	8.75 (5.00)	7.20 (3.10)
	5	9.84 (7.50)	7.31 (4.19)	5.62 (1.88)

ined. This is significant not only in obtaining an improvement in EER, but also in the fact that cohort selection is made much simpler. Instead of relying on calculations of means and/or variances, it is simple a series of rankings which can be computed quickly.

4.4 HMM/FST Comparative Test

In order to compare HMMs with FSTs, an HMM speaker identification system is developed that uses the 5 state, left-to-right, word models. This type of HMM structure is closely related to the trajectories used in the FST because both force movement from the beginning to the end. In the HMM, you must start in the first state and can either stay in the same state or transition to the next. For the FST, the similar case is true in that when comparing trajectories, the mapping may stay at the same segment, move one ahead,

or move two segments ahead. The HMM was limited to only moving one state because with only five states, a two state skip represents a large movement through the HMM structure. To facilitate comparison with the FST systems developed, test data consists only of utterances of the word 'ONE' from each of thirty-two female speakers. The HMM system was able to obtain a speaker identification rate of 9.90%. This rate is extremely low, but it performs three times better than chance (1/32 = 3.13%). This rate will be compared against the FST speaker identification systems.

4.5 Feature Space Trajectory Speaker Identification

The first test uses all instances of the word 'ONE' from the training data as individual trajectories. Each speaker has forty-two such instances. Speaker identification is performed by comparing each instance of the word 'ONE' from the test data against each of the forty-two training trajectories from all thirty-two speakers. This results in 1344 comparisons for each test utterance. This method performs speaker identification at the rate of 65.52% correct.

The second test uses a single utterance from each speaker's training data as the template. This instance is chosen by calculating the cumulative Ney-based FST distance to all other training instances and identifying the minimum. This method performs best of all single template techniques, see Table 7.

The other FST speaker identification tests use all of the training data from a speaker to establish a sixteen word codebook. This codebook attempts to represent all training data in the form of one sixteen vertex trajectory. The advantage to this technique is that the number of comparison per test utterance is reduced from 1344 to 32. A reduction of

more than one order of magnitude. The four methods described in Chapter IV are used. The results are shown in Table 7

Table 7. Speaker identification accuracy for various single template generation techniques

Method	Stand Alone(%)	SLP Post-Processor
Ney-FST Min Distance Traj	33.71	53.90
Mahalanobis/Median	25.71	54.48
Mahalanobis/Mean	24.95	49.71
Euclidean/Mean	25.14	37.33
Euclidean/Median	21.12	35.81

4.5.1 FST Speaker Identification using Neural Post-Processing. A SLP perceptron is implemented as a post-processor to enhance the performance of the FST speaker identification system. The SLP accepts the 32 distance measures as inputs and provides 32 outputs. The output node with the maximum value is chosen as the node which represents the identified speaker. The results are shown in the last column of Table 7.

These results are much improved over the stand alone FST system and once again demonstrate the enhancement provided by neural post-processing. In all cases, the perceptron post-processor was able to improve the identification rate. The Mahalanobis based distance templates provided the largest improvement with their scores doubling. The template created by determining the training trajectory that minimizes the distance to all other training trajectories showed the next best improvement of over 20%. The Euclidean based templates showed minimal improvement. These results show that the Mahalanobis based templates provide greater discriminative ability. Such templates are able to produce distance measures which the SLP can exploit more effectively than the Euclidean based

templates. In other words, as far as the SLP is concerned, the Mahalanobis based distance metrics are better features than the distance metrics produced by the other templates.

4.6 Conclusions

This chapter has provided the results of the various tests performed during this effort. A new Ney-based FST speaker identification was developed. The performance is poor when compared with other speaker identification systems; however, this system had severe constraints. The FST system is only using single word test utterances. A comparable HMM system using only single word test data was developed and performed much worse than the FST. This result is important because although performance is not at the desired level, it is outperforming an established speaker identification method with only small amounts of data.

V. Conclusions and Recommendations

5.1 Introduction

This chapter provides the conclusions and recommendations based on the results detailed in the previous chapter. Discussion of the HMM with SLP post-processing is given first. Next, the results of applying the FST to speaker recognition is accomplished. The third section analyzes the comparative test between HMMs and FSTs. Finally, recommendations for follow-on research is provided.

5.2 Hidden Markov Models with SLP Post-Processing

SLP post-processing provides a definite enhancement of speaker recognition performance for the HMM based systems developed. These results were consistent for both ergodic and left-to-right HMM systems. The speaker based model accuracy went from 86.25% to 96.88% for speaker identification, and equal error rates dropped from 6.92% to 5.62%. The word based model accuracy went from 96.88% to 99.06% identification, and equal error rates dropped from 3.38% to 1.56%. These results show the utility of using SLP post-processing of simple HMM systems. It permits simpler HMMs with less free parameters to obtain performance similar to that of more complex HMM systems.

5.3 Feature Space Trajectories

This research has produced the first speaker recognition system using FSTs. Thus, the feasibility of applying FSTs to speaker recognition has been proven. They can be implemented in a stand alone manner or incorporated with neural post-processing. The

key issue regarding the use of FSTs in speech was found to be template generation. The goal will be to keep the level of performance, but reduce the size of the training database. The ultimate solution would be to have one template per word for every speaker. New systems were developed toward this goal and showed favorable results because they were able to double the performance of a similar HMM based system. These were the first such FST systems ever developed and have laid valuable groundwork for further research in this area.

5.4 Hidden Markov Models vs Feature Space Trajectories

In order to make a direct comparison between HMMs and FSTs and their application to speaker recognition, a test was devised based on the limited performance of the FST. The HMM system used 5 state left-to-right word models and was tested using all test instances of the word 'ONE.' The HMM speaker identification was 9.90%. All of the FST speaker identification methods more than doubled this result with the best method obtaining a rate of 65.52%. This shows the ability of the FST to outperform the HMM on limited amounts of test data and justifies the need for further research.

5.5 Recommendations

Application of FSTs to speaker recognition had not been accomplished prior to this effort. This thesis provides a solid foundation to show that FSTs offer a promising area for research. The primary concern for follow-on research should be template generation. The test conducted that used each individual training utterance provided outstanding results

that exceed HMM performance. This suggests that an effective method of reducing the number of templates could also outperform a similar HMM.

This research focused on a feature set consisting of Mel-Frequency Cepstral Coefficients (MFCC). These features provide good performance in HMM based system; but, a detailed investigation into the proper feature set for FSTs may prove valuable.

Appendix A. Feature Space Trajectory Code

A.1 FSTNN Code

The FSTNN was implemented using code developed in Matlab. The training code was developed by Gary Brandstrom during his thesis and was not altered [6]. Brandstrom's test code was used as a starting point for the incorporation of Ney's algorithm and was used with minor modifications.

```
%-------------------------------------------------------
%  Filename: fst_tstNEY.m
%  Developed by minimal changes to code developed
%  by Gary Brandstrom (fst_tst.m).  Changes developed
%  by Eric Zeek and Neal Bruegger
%
%-------------------------------------------------------
%  c =  Vector Inner Products
%
%  v =  matrix where each row represents a segment between
%   points on a trajectory
%
%  len= length of segment between points on a trajectory
%
%  T =  a matrix where each row represents a feature vector
% from a known image sequence.
%
%  P =  a matrix where each row represents a feature vector
% from a unknown image sequence.
%
%---------------------  variables out  -----------------
%
%  D =  vector representing distance from each point of P to
% the trajectory
%
%  Pp = nearest point on trajectory where each point P projects
%
%  S =  vector representing sequence of line segment projections
%-------------------------------------------------------

function [S,D,Pp,d] = fst_tstNEY (c,v,len,T,P)

[s,dim]=size(T); % s = number of vertices in training trajectory,
% dim = length of feature vector
ss = size(P,1);          % ss = number fo vertices in test trajectory
Pptemp=zeros(s,dim);     % initialize
```

```
ind = 1;

for n=1:ss,              % loop once for each test point
  u=T(1,:)-P(n,:);
  d(n,1)=sqrt(u*u');
  Pptemp(1,:)=T(1,:); % Pptemp is P prime

% now find distance of nth test point to each segment
    ef=T*P';             % e is (i,n) and f is (i+1,n)
  for i=1:s-1;           % for each segment
    u=P(n,:)-T(i,:);
    alpha=u*v(i,:)';

    if alpha<=0,         % closest point is start of segment
      temp=P(n,:)-T(i,:);
      d(n,i+1)=sqrt(temp*temp');
      Pptemp(i+1,:)=T(i,:);
    elseif alpha>len(i), % closest point is next segment
      temp=P(n,:)-T(i+1,:);
      d(n,i+1)=sqrt(temp*temp');
      Pptemp(i+1,:)=T(i+1,:);
    else,                % closest point is on segment
      a=1-alpha/len(i);
      b=alpha/len(i);
      d(n,i+1)=P(n,:)*P(n,:)' - 2*a*ef(i,n) - 2*b*ef(i+1,n)
               + 2*a*b*c(i,i+1) + a*a*c(i,i) + b*b*c(i+1,i+1);
      d(n,i+1)=sqrt(d(n,i+1));
      Pptemp(i+1,:)=a*T(i,:)+b*T(i+1,:);
    end
  end

% Addition of weights to movement in distance matrix
if ind < s-1,
  d(n,ind+2) = 2*d(n,ind+2);
  d(n,ind) = 1.5*d(n,ind);
  [D(n),ind2]=min(d(n,ind:ind+2));
else
   d(n,ind) = 1.5*d(n,ind);
  [D(n),ind2]=min(d(n,ind:s));
end;

ind = ind + ind2 - 1;
Pp(n,:)=Pptemp(ind,:);  % point on traj corresp to min dist
S(n)=ind; % segment where Pp projects
end
```

A.2 FST Testing Code

```
%-------------------------------------------------------
%  Filename: zfstnnNEY.m
%  Tests all test data against every training trajectory
%  for all speakers.  Results are placed in confmatNEY.mat
%
%-------------------------------------------------------

% Set up array of words in the vocabulary
format = '%s';
fid = fopen('/home/bach1/ezeek/YOHO/Scripts/yohowords.list','r');
for i = 1:16,
  eval (['word' int2str(i) ' = fscanf(fid,format,1);'])
end;

% Initialize necessary data structures
load /home/bach1/ezeek/YOHO/SID/females.list
results = zeros(4,32);
Dist = zeros(100,32);
count = 0;

for j = 1:32,      % which speakers to use in verification test
  for k = 1:1,     % which words to verify against
    eval(['cd /home/bach1/ezeek/YOHO/verify/' int2str(females(j)) '/word'])
    eval(['word = word' int2str(k) ';'])
    eval(['load ' word '.num;'])
    eval(['frames = ' word ';'])
    eval(['load ' word '.dat;'])
    eval(['raw = ' word ';'])
    nSeq = size(frames,1);
    cum = cumsum(frames);
    for l=1:nSeq, % number of word instances
      for p = 1:4,
        data(p,:) = raw(l+((p-1)*nSeq),:);
      end;
      if l > 1,
        eframe = (nSeq*4+1)+cum(l-1)-(4*(l-1));
      else
        eframe = (nSeq*4+1);
      end;
      sframe = eframe + frames(l) - 4;
      for q = 5:frames(l),
        sframe = sframe - 1;
        data(q,:) = raw(sframe,:);
      end;
```

```
      P = data(1:frames(l),:);

% Now test this utterance against all training utterances
      count = count + 1;
      for m = 1:32,
% Only test word of test utterance
          eval(['cd /home/bach1/ezeek/YOHO/enroll/'
 int2str(females(m)) '/word'])
          eval(['load ' word '.num;'])
          eval(['tframes = ' word ';'])
          ntSeq = size(tframes,1);
          cd /home/bach1/ezeek/YOHO/FST/Training
          best = Inf;
          for o = 1:ntSeq,
            eval(['load ' int2str(females(m)) word int2str(o) ])
            [S,D,Pp,d] = fst_tstNEY(c,v,len,T,P);
            temp = mean(D);
            if temp < best
              Dist(count,m) = temp;
              best = temp;
    end;
          end;
      end;
    end;
    raw = [];
  end;
end;
save confmatNEY results Dist count;


%-------------------------------------------------------
%  Filename: zfstnnNEYbest.m
%  Tests all test data against each speaker's
%  ''best'' template.  Results placed in BESTall.mat
%
%-------------------------------------------------------


format = '%s';
fid = fopen('/home/bach1/ezeek/YOHO/Scripts/yohowords.list','r');
for i = 1:16,
  eval (['word' int2str(i) ' = fscanf(fid,format,1);'])
end;

load /home/bach1/ezeek/YOHO/SID/females.list
Dist = zeros(1000,32);
```

```
count = 0;

for j = 1:32,     % which speakers to use in verification test
  for k = 1:1,   % which words to verify against
    eval(['cd /home/bach1/ezeek/YOHO/verify/' int2str(females(j)) '/word'])
    eval(['word = word' int2str(k) ';'])
    eval(['load ' word '.num;'])
    eval(['frames = ' word ';'])
    eval(['load ' word '.dat;'])
    eval(['raw = ' word ';'])
    nSeq = size(frames,1);
    cum = cumsum(frames);
    for l=1:nSeq, % number of word instances
      for p = 1:4,
        data(p,:) = raw(l+((p-1)*nSeq),:);
      end;
      if l > 1,
        eframe = (nSeq*4+1)+cum(l-1)-(4*(l-1));
      else
        eframe = (nSeq*4+1);
      end;
      sframe = eframe + frames(l) - 4;
      for q = 5:frames(l),
        sframe = sframe - 1;
        data(q,:) = raw(sframe,:);
      end;
      P = data(1:frames(l),:);

% Now test this utterance against all training utterances
      count = count + 1;
      cd /home/bach1/ezeek/YOHO/FST/Training
      for m = 1:32,
% Only test word of test utterance
            eval(['load ' int2str(females(m)) word ])
            [S,D,Pp,d] = fst_tstNEY(c,v,len,T,P);
            Dist(count,m) = mean(D);
      end;
    end;
    raw = [];
  end;
end;
save BESTall Dist count;


%-------------------------------------------------------
%  Filename: zfstnnNEYmah1.m
```

```
%  Tests all test data against each speaker's
%  Mahalanobis distance median ordered template.
%  Results placed in mahMEDall.mat
%
%-------------------------------------------------------
% Get test utterances

format = '%s';
fid = fopen('/home/bach1/ezeek/YOHO/Scripts/yohowords.list','r');
for i = 1:16,
  eval (['word' int2str(i) ' = fscanf(fid,format,1);'])
end;

load /home/bach1/ezeek/YOHO/SID/females.list
Dist = zeros(1000,32);
count = 0;

for j = 1:32,     % which speakers to use in verification test
  for k = 1:1,   % which words to verify against
    eval(['cd /home/bach1/ezeek/YOHO/verify/' int2str(females(j)) '/word'])
    eval(['word = word' int2str(k) ';'])
    eval(['load ' word '.num;'])
    eval(['frames = ' word ';'])
    eval(['load ' word '.dat;'])
    eval(['raw = ' word ';'])
    nSeq = size(frames,1);
    cum = cumsum(frames);
    for l=1:nSeq, % number of word instances
      for p = 1:4,
        data(p,:) = raw(l+((p-1)*nSeq),:);
      end;
      if l > 1,
        eframe = (nSeq*4+1)+cum(l-1)-(4*(l-1));
      else
        eframe = (nSeq*4+1);
      end;
      sframe = eframe + frames(l) - 4;
      for q = 5:frames(l),
        sframe = sframe - 1;
        data(q,:) = raw(sframe,:);
      end;
      P = data(1:frames(l),:);

% Now test this utterance against all training utterances
      count = count + 1;
```

```
      cd /home/bach1/ezeek/YOHO/FST/Training/MAHmed16
      for m = 1:32,
% Only test word of test utterance
            eval(['load ' int2str(females(m)) word '])
            [S,D,Pp,d] = fst_tstNEY(c,v,len,T,P);
            Dist(count,m) = mean(D);
      end;
    end;
    raw = [];
  end;
end;
save mahMEDall Dist count;


%-------------------------------------------------------
%  Filename: zfstnnNEYmah2.m
%  Tests all test data against each speaker's
%  Mahalanobis distance mean ordered template.
%  Results placed in mahMEANall.mat
%
%-------------------------------------------------------
% Get test utterances

format = '%s';
fid = fopen('/home/bach1/ezeek/YOHO/Scripts/yohowords.list','r');
for i = 1:16,
  eval (['word' int2str(i) ' = fscanf(fid,format,1);'])
end;

load /home/bach1/ezeek/YOHO/SID/females.list
Dist = zeros(1000,32);
count = 0;

for j = 1:32,     % which speakers to use in verification test
  for k = 1:1,   % which words to verify against
    eval(['cd /home/bach1/ezeek/YOHO/verify/' int2str(females(j)) '/word'])
    eval(['word = word' int2str(k) ';'])
    eval(['load ' word '.num;'])
    eval(['frames = ' word ';'])
    eval(['load ' word '.dat;'])
    eval(['raw = ' word ';'])
    nSeq = size(frames,1);
    cum = cumsum(frames);
    for l=1:nSeq, % number of word instances
      for p = 1:4,
        data(p,:) = raw(l+((p-1)*nSeq),:);
```

```
      end;
      if l > 1,
        eframe = (nSeq*4+1)+cum(l-1)-(4*(l-1));
      else
        eframe = (nSeq*4+1);
      end;
      sframe = eframe + frames(l) - 4;
      for q = 5:frames(l),
        sframe = sframe - 1;
        data(q,:) = raw(sframe,:);
      end;
      P = data(1:frames(l),:);

% Now test this utterance against all training utterances
      count = count + 1;
      cd /home/bach1/ezeek/YOHO/FST/Training/MAHmean16
      for m = 1:32,
% Only test word of test utterance
            eval(['load ' int2str(females(m)) word ])
            [S,D,Pp,d] = fst_tstNEY(c,v,len,T,P);
            Dist(count,m) = mean(D);
      end;
    end;
    raw = [];
  end;
end;
save mahMEANall Dist count;


%-------------------------------------------------------
%  Filename: zfstnnNEYeuc1.m
%  Tests all test data against each speaker's
%  template developed using Euclidean distance with
%  median ordering.   Results placed in eucMEDall.mat
%
%-------------------------------------------------------



format = '%s';
fid = fopen('/home/bach1/ezeek/YOHO/Scripts/yohowords.list','r');
for i = 1:16,
  eval (['word' int2str(i) ' = fscanf(fid,format,1);'])
end;

load /home/bach1/ezeek/YOHO/SID/females.list
```

```
Dist = zeros(1000,32);
count = 0;

for j = 1:32,     % which speakers to use in verification test
  for k = 1:1,   % which words to verify against
    eval(['cd /home/bach1/ezeek/YOHO/verify/' int2str(females(j)) '/word'])
    eval(['word = word' int2str(k) ';'])
    eval(['load ' word '.num;'])
    eval(['frames = ' word ';'])
    eval(['load ' word '.dat;'])
    eval(['raw = ' word ';'])
    nSeq = size(frames,1);
    cum = cumsum(frames);
    for l=1:nSeq, % number of word instances
      for p = 1:4,
        data(p,:) = raw(l+((p-1)*nSeq),:);
      end;
      if l > 1,
        eframe = (nSeq*4+1)+cum(l-1)-(4*(l-1));
      else
        eframe = (nSeq*4+1);
      end;
      sframe = eframe + frames(l) - 4;
      for q = 5:frames(l),
        sframe = sframe - 1;
        data(q,:) = raw(sframe,:);
      end;
      P = data(1:frames(l),:);

% Now test this utterance against all training utterances
      count = count + 1;
      cd /home/bach1/ezeek/YOHO/FST/Training/VQmed16
      for m = 1:32,
% Only test word of test utterance
            eval(['load ' int2str(females(m)) word ])
            [S,D,Pp,d] = fst_tstNEY(c,v,len,T,P);
            Dist(count,m) = mean(D);
      end;
    end;
    raw = [];
  end;
end;
save eucMEDall Dist count;

%-------------------------------------------------------
%  Filename: zfstnnNEYeuc2.m
```

```
%  Tests all test data against each speaker's
%  Euclidean distance mean ordering template.
%  Results placed in eucMEANall.mat
%
%-------------------------------------------------------

% Get test utterances

format = '%s';
fid = fopen('/home/bach1/ezeek/YOHO/Scripts/yohowords.list','r');
for i = 1:16,
  eval (['word' int2str(i) ' = fscanf(fid,format,1);'])
end;

load /home/bach1/ezeek/YOHO/SID/females.list
Dist = zeros(1000,32);
count = 0;

for j = 1:32,      % which speakers to use in verification test
  for k = 1:1,   % which words to verify against
    eval(['cd /home/bach1/ezeek/YOHO/verify/' int2str(females(j)) '/word'])
    eval(['word = word' int2str(k) ';'])
    eval(['load ' word '.num;'])
    eval(['frames = ' word ';'])
    eval(['load ' word '.dat;'])
    eval(['raw = ' word ';'])
    nSeq = size(frames,1);
    cum = cumsum(frames);
    for l=1:nSeq, % number of word instances
      for p = 1:4,
        data(p,:) = raw(l+((p-1)*nSeq),:);
      end;
      if l > 1,
        eframe = (nSeq*4+1)+cum(l-1)-(4*(l-1));
      else
        eframe = (nSeq*4+1);
      end;
      sframe = eframe + frames(l) - 4;
      for q = 5:frames(l),
        sframe = sframe - 1;
        data(q,:) = raw(sframe,:);
      end;
      P = data(1:frames(l),:);

% Now test this utterance against all training utterances
```

```
      count = count + 1;
      cd /home/bach1/ezeek/YOHO/FST/Training/VQmean16
      for m = 1:32,
% Only test word of test utterance
            eval(['load ' int2str(females(m)) word ])
            [S,D,Pp,d] = fst_tstNEY(c,v,len,T,P);
            Dist(count,m) = mean(D);
      end;
    end;
    raw = [];
  end;
end;
save eucMEANall Dist count;
```

Bibliography

1. Joseph P. Campbell, Jr., "Testing with the YOHO CD-ROM voice verification corpus", in *Proc. of the 1995 ICASSP*, 1995, pp. 541–545.
2. Leonard Neiberg and David P. Casasent, "Feature space trajectory (FST) classifier neural network", in *SPIE*, 1994, vol. 2353, pp. 276–292.
3. Leonard Neiberg and David P. Casasent, "Feature space trajectory neural net classifier", in *SPIE*, 1995, vol. 2492, pp. 361–372.
4. Leonard Neiberg et al., "Feature space trajectory neural net classifer: 8-class distortion invariant tests", in *SPIE*, 1995, vol. 2588, pp. 540–555.
5. Leonard Neiberg and David P. Casasent, "Feature space trajectory neural net classifier: confidences and thresholds for clutter and low contrast objects", in *SPIE*, 1996, vol. 2760, pp. 435–446.
6. Gary Brandstrom et al., "Space object identification using spatio-temporal pattern recognition", in *SPIE*, 1996, vol. 2760, pp. 475–486.
7. David P. Casasent and Leonard M. Neiberg, "Classifier and shift-invariant automatic target recognition neural networks", *Neural Networks*, vol. 8, no. 7/8, pp. 1117–1129, 1995.
8. Captain John M. Colombi, *Generalized Hidden Filter Markov Models Applied to Speaker Recognition*, PhD thesis, Air Force Institute of Technology, March 1996.
9. Thomas W. Parsons, *Voice and Speech Processing*, McGraw-Hill Book Company, 1987.
10. Alan B. Poritz, "Hidden Markov models: A guided tour", in *Proc. of the 1988 ICASSP*. IEEE, 1988, vol. 1, pp. 7–13.
11. Lawrence Rabiner and Biing-Hwang Juang, *Fundamentals of Speech Recognition*, Prentice Hall, 1993.
12. Hermann Ney, "The use of a one-stage dynamic programming algorithm for connected word recognition", *IEEE Trans. on Speech and Audio Processing*, vol. ASSP-32, no. 2, pp. 263–271, April 1984.
13. Dennis W. Ruck, Steven K. Rogers, et al., "The multilayer perceptron as an approximation to a Bayes optimal discriminant function", *IEEE Trans on Neural Networks*, vol. 1, no. 4, Dec. 1990.
14. Ronald Benson and Ojvind Bernander, "Enhanced Algorithms for Cockpit Voice Recognition Systems", Tech. Rep., Tanner Research, Inc, 1995, SBIR Contract F33615-95-C3602.
15. Entropic Research Laboratory, *HTK: Hidden Markov Model Toolkit*, 1993.
16. A. Higgins, L. Bahler, and J. Porter, "Speaker verification using randomized phrase prompting", *Digital Signal Processing*, vol. 1, pp. 89–106, 1991.
17. Sadaoki Furui, "An overview of speaker recognition technology", in *Proc. of the 1994 ECSA Workshop on Speaker Recog. Ident. and Ver.*, 1994.

18. C.S. Liu, H.C. Wang, and C.H. Lee, "Speaker verification using normalized log-likelihood score", *IEEE Trans. on Speech and Audio Processing*, vol. 4, no. 1, pp. 56–60, January 1996.

19. Tomoko Matsui and Sadaoki Furui, "Similarity normalization method for speaker verification based on a posteriori probability", in *Proc. of the 1994 ECSA Workshop on Speaker Recog. Ident. and Ver.*, 1994.

20. Douglas A. Reynolds, "Speaker identification and verification using Gaussian mixture speaker models", *Speech Communication*, vol. 17, no. 1-2, pp. 91–108, 1995.

21. Kenneth H. Fielding, *Spatio-temporal Pattern Recognition using Hidden Markov Models*, PhD Dissertation, Air Force Institute of Technology, June 1994, AFIT/DS/ENG/94J-02.

22. Kenneth H. Fielding et al., "An application of embedology to spatio-temporal pattern recognition", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 32, no. 2, pp. 768–774, April 1996.

23. Steven K. Rogers and Matthew Kabrisky, *An Introduction to Biological and Artificial Neural Networks for Pattern Recognition*, SPIE, 1991.

24. Jianchang Mao and Anil K. Jain, "A self-organizing network for hyperellipsoidal clustering (HEC)", *IEEE Trans on Neural Networks*, vol. 7, no. 1, pp. 16–29, January 1996.

Vita

Captain Eric Joseph Zeek [redacted]o. He entered the United States Air Force Academy in June 1987. He was awarded a Bachelor of Science in Electrical Engineering degree as well as a commission upon graduation in May 1991. His first assignment was as an Intelligence Collection Requirements Manager in the National Air Intelligence Center (NAIC) at Wright-Patterson Air Force Base, Ohio. In April 1993, he moved to the Foreign Materiel Exploitation Branch of NAIC to become the Chief Engineer. In this capacity, Captain Zeek was responsible for all in-house foreign materiel exploitation projects. This included scheduling, budget, and personnel allocations. In May 1995, Captain Zeek entered the Computer Systems program at the Air Force Institute of Technology.

[redacted]

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1996	Master's Thesis

4. TITLE AND SUBTITLE
SPEAKER RECOGNITION BY HIDDEN MARKOV MODELS AND NEURAL NETWORKS

5. FUNDING NUMBERS

6. AUTHOR(S)
Eric J. Zeek

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCS/ENG/96D-31

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Dr. Timothy Anderson
AL/CFBA
Wright-Patterson AFB, OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

As humans, we develop the ability to identify people by their voice at an early age. Getting computers to perform the same task has proven to be an interesting problem. Speaker recognition involves two applications, speaker identification and speaker verification. Both applications are examined in this effort. Two methods are employed to perform speaker recognition. The first is an enhancement of hidden Markov models. Rather than alter some part of the model itself, a single-layer perceptron is added to perform neural post-processing. The second solution is the novel application of an enhanced Feature Space Trajectory Neural Network to speaker recognition. The Feature Space Trajectory was developed for image processing for temporal recognition and has been demonstrated to outperform the hidden Markov model for some image sequence applications. Neural post-processing of hidden Markov models is shown to improve performance of both aspects of speaker recognition by increasing the identification rate from 70.23% to 88.44% and reducing the Equal Error Rate from 3.38% to 1.56%. In addition, a new method of cohort selection is implemented based on the structure of the single-layer percceptron. Feasibility of using Feature Space Trajectory Neural Networks for speaker recognition is demonstrated. Favorable identification results of 65.52% are obtained when using a large training database. The FST configurations tested outperformed a comparable HMM system by 12-24%.

14. SUBJECT TERMS
hidden Markov model, feature space trajectory, neural network, speaker recognition, speaker identification, speaker verification, dynamic time warping, Ney's algorithm

15. NUMBER OF PAGES
70

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- **DOD** - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- **DOE** - See authorities.
- **NASA** - See Handbook NHB 2200.2.
- **NTIS** - Leave blank.

Block 12b. Distribution Code.

- **DOD** - Leave blank.
- **DOE** - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
- **NASA** - Leave blank.
- **NTIS** - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.