DYNAMIC TIME WARPING DIGIT RECOGNIZER

By

Yu Zeng

A Project Report
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Electrical Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

November 2000

DYNAMIC TIME WARPING DIGIT RECOGNIZER

By

Yu Zeng

Approved:

Joseph Picone
Professor of Electrical and Computer Engineering
(Director of Thesis)

Nicolas Younan
Associate Professor of Electrical and Computer Engineering
(Coordinator)

Nicolas Younan
Associate Professor of Electrical and Computer Engineering
(Committee Member)

Eric Hansen
Assistant Professor of Computer Science
(Committee Member)

James C. Harden
Graduate Coordinator of Computer Engineering in the Department of Electrical and Computer Engineering

G. Marshall Molen
Department Head of the Department of Electrical and Computer Engineering

A. Wayne Bennett
Dean of the College of Engineering

Richard D. Koshel
Dean of the Graduate School

Name: Yu Zeng

Date of Degree: December, 2000

Institution: Mississippi State University

Major Professor: Dr. Joseph Picone

Title of Study: DYNAMIC TIME WARPING DIGIT RECOGNIZERS

Pages in Study: 51

Candidate for Degree of Master of Science

DEDICATION

To my parents, my brother and my husband.

ACKNOWLEDGMENTS

I would like to give my sincere appreciation to my advisor Dr. Joseph Picone, who pointed me to the topic of speech recognition. I would also like to thank Dr. Nicholas Younan and Dr. Eric Hansen for serving on my committee and for providing valuable suggestions and comments. Also, thanks to Mr. Bill Chapman for helping me by proofreading the report. My final thanks go to all other people in ISIP for giving me support and help through the entire project.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Speech is a most natural and efficient way to exchange information for human beings. To make a real “intelligent computer,” it is important that the machine can “hear,” “understand,” and “act upon” spoken information, and also “speak” to complete the information exchange. Therefore, speech recognition is essential for a computer to reach the goal of natural human-computer communication [1].

After more than 40 years of research, many algorithms have been proposed and implemented for automatic speech recognition. The “pattern-matching” method is one of the well-studied approaches. In this method, the system stores one or more prototypes (templates) for each word in the vocabulary, and compares the incoming speech signal with each of them to find the best-matched one as the recognition result. This approach involves two steps, training the template set, and recognition of the test signal via a pattern matching method. Figure 1 shows a block diagram of such a system.

The training process constructs a template for each word in the vocabulary. Speech signals in the training database are first divided into frames of equal length. Then the acoustic front-end converts each frame into a feature vector that captures the properties of the signal in that frame. These feature vectors will be clustered into groups by the pattern clustering block to form a word model. This process is repeated for every word in the vocabulary. The concept is that if enough versions of a pattern to be recognized are included in the training set, the training procedure should be able to adequately characterize the acoustic properties of the pattern [2].

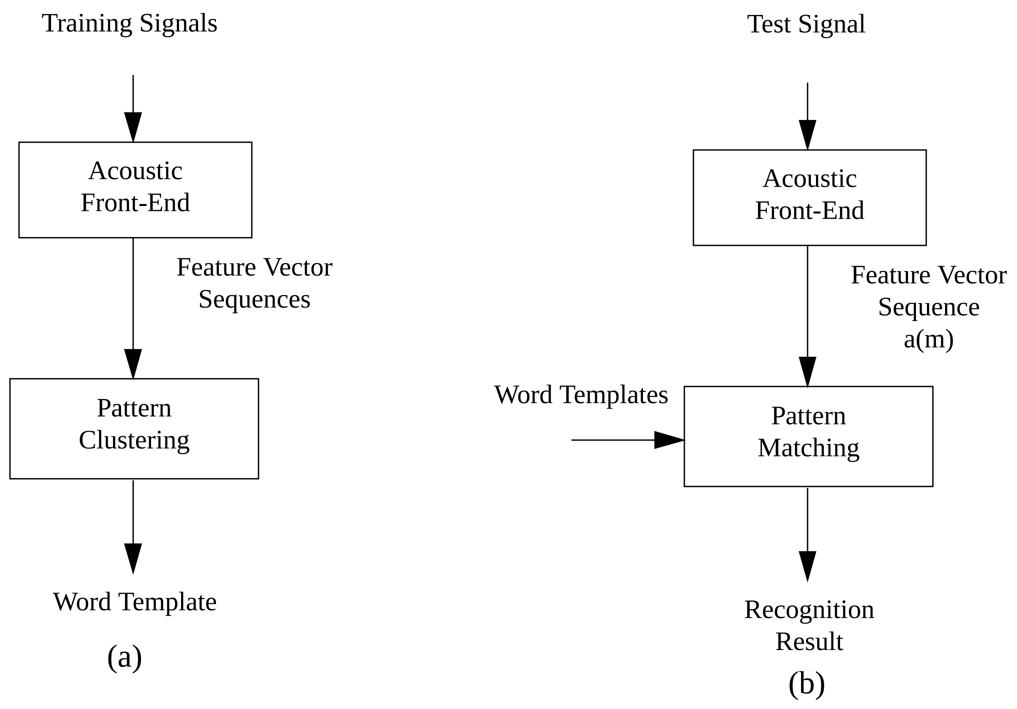


Figure 1. Block diagram of simplified speech recognition system.

The recognition procedure first converts the unknown signal into a sequence of feature vectors using the same acoustic front-end as the training process. This feature vector sequence is then compared to each possible word template learned during training by the pattern matching block. A recognition decision is made based on the goodness of the match, which is quantified by a distance function between two sequences of feature vectors, one representing the word model and the other representing the input signal [2].

This approach raises three questions; the first one is how to derive a set of parameters to represent speech signals in a form which is convenient for subsequent processing. Linear Predictive Coding (LPC) is a dominant representation method for the spectral analysis model of a speech signal. It provides a good model to approximate the vocal tract spectral envelope of speaking. The method of LPC is mathematically precise, simple and straightforward to implement in either software or hardware. Also, it works well in recognition applications [2]. Based on these considerations, we will use an LPC front-end in this project.

The second question is how to generate an efficient and succinct template for each word. Since the test signal will be recognized based on the distance between the signal and the word templates, the goodness of the templates determines the recognition performance. As we discussed before, a template should contain enough information of various renditions of the word it represents in order to reduce the error rate.

The last question is how to find a reasonable definition of dissimilarity between a template and a signal. After all, both the template and the signal are sequences of feature vectors, so this problem can be split into two smaller problems. The first one is to define a distance metric between two feature vectors, or the local distance. The second one is to define a method to combine the distance between each vector pair into the overall distance between two sequences, or the global distance. Itakura distance has been proved to be the best distance measurement for LPC signals. Rather than compute the distance between two vectors directly, it compares the prediction error for a particular speech sample using these two LPC vectors. In this project, we also implemented the Euclidean distance and absolute distance, which are commonly used in engineering computation, to give the user a chance to examine their performance. Dynamic programming (DP) is usually applied to solve the second problem. Its application in speech recognition is known as dynamic time warping (DTW), which stretches or compresses signals nonlinearly to achieve the best matching between them.

The goal of this project is to introduce the concept of speech recognition to student engineers by building a Java-based digit recognition system using the dynamic time warping technique. One of the key difficulties in signal processing courses is the visualization of the theories and concepts. Students need an example system to help them to understand the theory, to develop their intuition by interacting with an actual system, and to reinforce what they learned by practice [3]. Combining what we discussed above, this project resulted in the following two major capabilities:

- DTW match: given a test signal and a template, the applet should be able to find the best matching path and associated matching cost. Also, it should be able to recognize the signal, print out the best matching cost.

- Visualization: to fulfill our educational goals, the applet should be able to synthesize the signal and provide a visualization of the signal, the template, and the actual matching between them. so that a user can visually compare them.

Institute for Signal and Information Processing (ISIP) has developed a set of Java-based demos for courses like Signals and Systems, DSP and Pattern Recognition. An applet for speech recognition will be a good complement to this demo family. The demos are available in the public domain at ISIP's web site: *http://www.isip.msstate.edu/*.

Another important reason that we chose to implement the system as a Java applet is that this way more students can get access to the software. An applet is a Java program that runs inside a Java-enabled web browser. Comparing with other commercial software like Matlab, Visual C++, Visual Basic, etc., Java has a number of advantages:

- Portability. Java is totally platform-independent. The only thing that the user will need is a Java-enabled web browser.

- Java is free! A number of students do not have hundreds of dollars to buy a copy of Matlab or VC.

- Convenient access. The user even does not need to keep a local copy of the applet in their machine. Only type in the address of the applet, and wait for the applet to show up without installing and running it locally.

The report is organized as follows: chapter 2 describes a linear prediction method for generating feature vectors from speech signals. Several methods for computing d_f, the distance between two frames, are described using the LPC representation. Chapter 3 describes the training process that builds the word templates. Chapter 4 introduces the dynamic programming algorithm and it’s application in speech recognition: DTW, i.e., the algorithm to compute the dissimilarity D of two feature vector sequences. Chapter 5 describes the design and implementation details of the applet, and chapter 6 gives a brief summary of this project. The source code and a tutorial for the applet are included at the end of this report.

CHAPTER 2

LINEAR PREDICTIVE CODING

A typical speech signal is an analog signal which varies with time. We use $s(t)$ to denote an analog speech signal function. Part (a) of figure 2 shows a portion of a spoken vowel sound. To process the signal by digital means, it is necessary to sample the continuous-time signal into a discrete-time signal, and then convert the discrete-time continuous-valued signal into a discrete-time, discrete-valued (digital) signal [5]. The digitized signal is shown in part (b) of figure 2, which is denoted as $s(n)$. The properties of a speech signal change relatively slowly with time, so that we can divide the speech signal into a sequence of un-correlated segments, or frames, and process the sequence as if each frame has fixed properties. Under this assumption, we can extract the features of each frame based on the samples inside this frame only. And usually, the feature vector will replace the original signal in further processing, which means the speech signal is converted from a time-varying analog signal into a sequence of feature vectors. Linear predictive coding is one of the most powerful techniques for estimating basic speech properties, such as pitch, formants and spectra [4]. In this chapter, we will discuss the procedure of deriving the linear prediction coefficients from the speech signal, and the distance measurement between two LPC vectors. Because of the short time processing assumption, the discussion in this chapter will be restricted to one frame of speech signal. As shown in part (c) of figure 2, speech samples inside this frame are indexed by n, and range from $s(0)$ to $s(N-1)$.

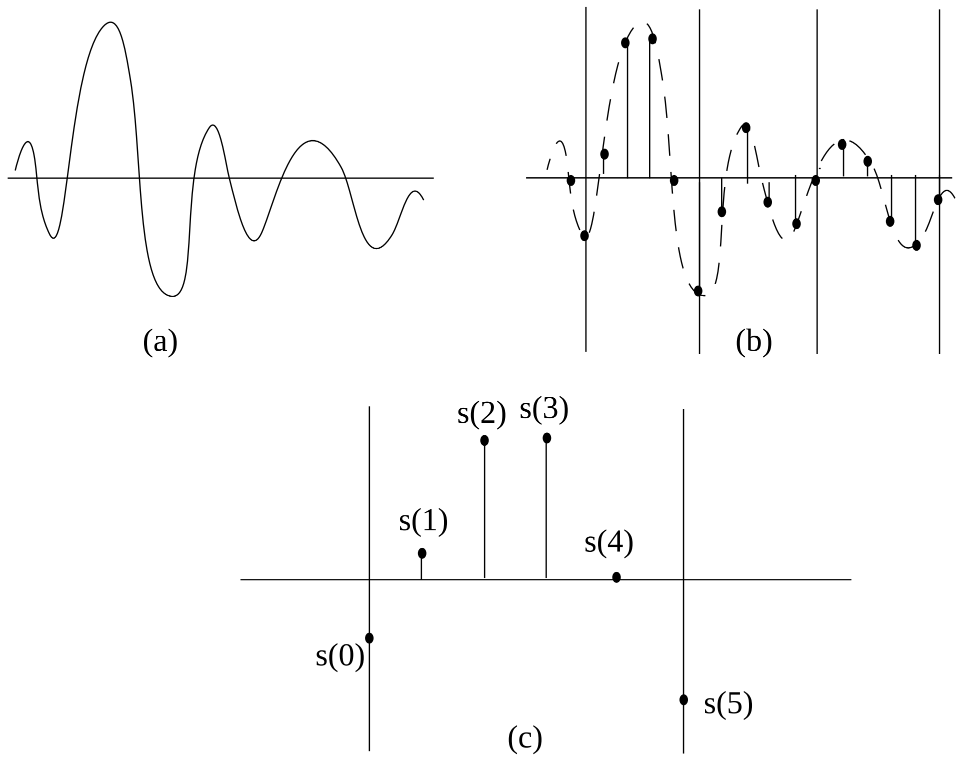


Figure 2. Speech signal: (a) analog signal s(t), (b) digital signal s(n), (c) a frame of s(n).

2.1. Linear Predictive Modeling for Speech Signal

The basic idea behind the LPC model is that a speech sample $s(n)$ can be approximated as a linear combination of the past p speech samples $s(n-1)$, $s(n-2)$, $\ldots$, $s(n-p)$, such that

$$s(n) \approx -\sum_{k=1}^{p} a_k s(n-k) \tag{2.1}$$

If we define the predicted sample for $s(n)$ as $\hat{s}(n)$, then

$$\hat{s}(n) = -\sum_{k=1}^{p} a_k s(n-k), \tag{2.2}$$

where the terms $-a_k, k = 1, 2, \ldots, p$ define the predictor coefficients of this frame. Taking the z-transform for both sides, we get the system function of the predictor as

$$P(z) = -\sum_{k=1}^{p} a_k z^{-k} \tag{2.4}$$

And the prediction error for sample $s(n)$ is defined as

$$\begin{aligned} e(n) &= s(n) - \hat{s}(n) = s(n) + \sum_{k=1}^{p} a_k s(n-k) \\ &= \sum_{k=0}^{p} a_k s(n-k) \end{aligned}, \tag{2.5}$$

where $a_0 = 1$.

The objective of linear predictive analysis is to determine a set of $-a_k$ which minimize the total squared error, which is defined as

$$\begin{aligned} E &= \sum_{n=n_0}^{n_1} e^2(n) \\ &= \sum_{n=n_0}^{n_1} \left[\sum_{k=0}^{p} a_k s(n-k) \right]^2 \\ &= \sum_{n=n_0}^{n_1} \sum_{i=0}^{p} \sum_{j=0}^{p} a_i s(n-i) s(n-j) a_j \end{aligned}. \tag{2.6}$$

where n_0 and n_1 define the index limits over which error minimization occurs.

If we define

$$\phi(i, j) = \sum_{n = n_0}^{n_1} s(n-i)s(n-j), \tag{2.7}$$

equation (2.6) can be simplified as

$$E = \sum_{i = 0}^{p} \sum_{j = 0}^{p} a_i \phi(i, j) a_j. \tag{2.8}$$

We solve the minimization problem by taking the partial differential of E with respect to $a_k, k = 0, 1, \ldots, p$, and setting to zero. Therefore, from equation (2.8), we get:

$$\frac{\partial}{\partial a_k} E = 0 = 2 \sum_{i = 0}^{p} a_i \phi(i, k) \qquad 0 \le k \le p. \tag{2.9}$$

Since $a_0 = 1$, (2.9) can be extended as

$$\phi(0, k) + \sum_{i = 1}^{p} a_i \phi(i, k) = 0 \qquad 1 \le k \le p. \tag{2.10}$$

In this equation, $\phi(i, j)$, $i = 0, 1, 2, \ldots, p, j = 0, 1, 2, \ldots, p$ are determined by equation (2.7), which only involves samples $s(n)$ from $n = n_0 - p$ to n_1, so that the p unknown predictor coefficients $\{a_k\}$ can be obtained by solving the set of p linear functions [6].

Two specific cases of the selection of n_0 and n_1 have been investigated in detail, they are referred to as the autocorrelation method and the covariance method. We use the former one in this project.

The covariance method is defined by setting $n_0 = p$, and $n_1 = N - 1$, so that the total prediction error is minimized over the interval $[p, N-1]$, and all N samples are involved in the calculation of $\phi(i, j)$. Please refer to [6] for more detail about the covariance method.

The autocorrelation method is defined by setting $n_0 = -\infty$, and $n_1 = \infty$, and defining $s(n) = 0$, for $n < 0$ and $n \geq N$. Under this condition, $\phi(i, j)$ can be simplified as

$$\begin{aligned}\phi(i, j) &= \sum_{n=-\infty}^{\infty} s(n-i)s(n-j) \\ &= \sum_{n=-\infty}^{\infty} s(n)s(n+|i-j|) \\ &= \sum_{n=0}^{N-1-|i-j|} s(n)s(n+|i-j|) \\ &= r(|i-j|)\end{aligned} \quad . \tag{2.11}$$

Replacing $\phi(i, j)$ with $r(|i-j|)$, (2.10) leads to

$$r(k) = \sum_{i=1}^{p} a_i \cdot r(|i-k|), \qquad 1 \leq k \leq p. \tag{2.12}$$

The matrix form of (2.12) is

$$\bar{r} = \bar{R} \cdot \bar{a}, \tag{2.13}$$

where $\bar{r} = \begin{bmatrix} r(1) \\ r(2) \\ \dots \\ r(p) \end{bmatrix}$, $\bar{R} = \begin{bmatrix} r(0) & r(1) & \dots & r(p-1) \\ r(1) & r(2) & \dots & r(p-2) \\ \dots & \dots & \dots & \dots \\ r(p-1) & r(p-2) & \dots & r(0) \end{bmatrix}$, and $\bar{a} = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_p \end{bmatrix}$.

And from (2.13), we can get the solution of the linear predictive coefficients:

$$\bar{a} = \bar{R}^{-1} \cdot \bar{r}, \tag{2.14}$$

Usually, $\vec{a} = [1, a_1, a_2, \ldots, a_p]^T$ is called the LPC vector of a frame of speech signal. In this project, speech signals will be converted into sequences of LPC vectors by the acoustic front-end block of figure 1 before further processing.

2.2. Distance Measurement

From chapter 1, we know that pattern matching recognition is based upon the comparison of two frames. Therefore, a proper distance metric between two LPC vectors is important for this approach to speech recognition.

We use $d_f(\vec{a}, \vec{b})$ to denote the distance between frames of speech signal with LPC parameter sets $\vec{a} = [1, a_1, a_2, \ldots, a_p]^T$, and $\vec{b} = [1, b_1, b_2, \ldots, b_p]^T$. Since d_f is a distance measure, we require that:

1. $d_f(\vec{a}, \vec{b}) \geq 0$,

2. $d_f(\vec{a}, \vec{b}) = d_f(\vec{b}, \vec{a})$,

3. $d_f(\vec{a}, \vec{b}) \leq d_f(\vec{a}, \vec{c}) + d_f(\vec{c}, \vec{b})$.

Any function that meets the above properties is a legitimate metric on the vector space. Therefore, there are many metrics, each having its own advantages and disadvantages. Usually we use particular cases of the Minkowski metric in engineering computations. The Minkowski metric of order s between frame $\vec{a}$ and $\vec{b}$ is

$$d_s(\vec{a}, \vec{b}) \equiv \sqrt[s]{\sum_{k=1}^{p} |a_k - b_k|^s}. \tag{2.15}$$

The ones we will use in this project are

- The l_1 or city block metric,

$$d_1(\vec{a}, \vec{b}) = \sum_{k=1}^{p} |a_k - b_k|. \tag{2.16}$$

- The l_2 or Euclidean metric,

$$d_2(\vec{a}, \vec{b}) \equiv \sqrt[2]{\sum_{k=1}^{p} |a_k - b_k|^2}. \tag{2.17}$$

If the vectors are defined in an orthonormal space, then the Euclidean distance between two vectors conforms exactly to the “natural” distance between them. However, the LPC space is not orthonormal; parameters are highly correlated. So the unweighted Euclidean distance and absolute distance are not appropriate here [1].

The most frequently used distance measurement for LPC vectors was proposed by Itakura in 1975 [8]. To understand this distance measurement, let’s look at this problem from another point of view. In the last section, we know that for a segment of speech signal $s(n)$, if we use LPC vector $\vec{a}$ to model the signal, the prediction error will be

$$E = \sum_{i=0}^{p} \sum_{j=0}^{p} a_i \phi(i, j) a_j. \tag{2.18}$$

In matrix form, this equation is

$$E = \vec{a}^T \cdot \vec{R} \cdot \vec{a}, \tag{2.19}$$

where $\vec{R} = \begin{bmatrix} r(0) & r(1) & \dots & r(p) \\ r(1) & r(2) & \dots & r(p-1) \\ \dots & \dots & \dots & \dots \\ r(p) & r(p-1) & \dots & r(0) \end{bmatrix} = \begin{bmatrix} r(0) & \bar{r}^T \\ \bar{r} & \bar{R} \end{bmatrix}$.

In the paper of [8], Itakura explained the maximum likelihood method to estimate the set of parameters $\vec{a}$ given the signal $s(n)$. Using equation 2.5 as the hypothesize model specified by LPC vector $\vec{a}$, the maximum value of the likelihood function is proportional to the prediction error, which is

$$L^*(S|\vec{a}) = -(N/2)\log(\vec{a}^T \cdot \vec{R} \cdot \vec{a}) + C, \tag{2.20}$$

where S is the signal $\{s(1), s(2), \dots, s(N)\}$, and C is a constance. Let's assume that $\hat{a}$ is the maximum likelihood estimate of $\vec{a}$, and define

$$d(S|\vec{a}) = \log\frac{\vec{a}^T \cdot \vec{R} \cdot \vec{a}}{\hat{a}^T \cdot \vec{R} \cdot \hat{a}}. \tag{2.21}$$

Since $\hat{a}$ is the best estimate of $\vec{a}$, if the model defined by $\vec{a}$ is close to the actual process which generates S, then $\hat{a}$ is close to $\vec{a}$, and $d(S|\vec{a})$ is close to zero, otherwise, $d(S|\vec{a})$ is significantly large. In this sense, $d(S|\vec{a})$ can be regarded as a distance measure between S and the model specified by equation 2.5 [8].

Now let's come back to the problem of measuring the distance between LPC vector $\vec{a}$ and $\vec{b}$. Both of these two vectors refer to a segment of speech signal. If we consider the signal that $\vec{a}$ refers to as S, and use both $\vec{a}$ and $\vec{b}$ to model this signal, then $\vec{a}$ is the one that results in maximum value of likelihood, and function

$$d(S|\vec{a}) = \log\frac{\vec{b}^T \cdot \vec{R} \cdot \vec{b}}{\vec{a}^T \cdot \vec{R} \cdot \vec{a}} \tag{2.22}$$

is the distance between signal S and $\vec{b}$, or between $\vec{a}$ and $\vec{b}$. We re-write the left part as $d(\vec{a}, \vec{b})$, equation 2.22 comes to

$$d(\vec{a}, \vec{b}) = \log\frac{\vec{b}^T \cdot \vec{R} \cdot \vec{b}}{\vec{a}^T \cdot \vec{R} \cdot \vec{a}}, \tag{2.23}$$

and this is called the Itakura distance.

CHAPTER 3

TEMPLATE TRAINING

As described in chapter 1, using the pattern-matching technique to solve the speech recognition problem, the input speech signal is compared with the model of each word by measuring the dissimilarity between them. Chapter 2 has described linear predictive coding and how to measure the distance (dissimilarity) between two LPC vectors. In this chapter, we will discuss the problem of how to obtain the set of acoustic models, in other word, the training problem.

The objective of training is to create a consistent and succinct template for each word which appears in the vocabulary. A trivial idea is to use an utterance for each word spoken by one person as the prototype set. This approach fails because it makes no attempt to estimate the pattern variability [2]. For speaker-independent speech recognition, template training is required in order to achieve high word recognition accuracy for practical tasks. In this paper, the terms acoustic model, prototype, template and reference pattern are interchangeable.

3.1. Vector Quantization

Quantization is used to approximate continuous amplitude signals by discrete signals, so as to reduce the data redundancy [10]. 1-dimensional signal quantization is called scalar quantization. If the signal is multi-dimensional, the joint quantization of all dimensions is called as vector quantization (VQ). Figure 3 shows an example of

2-dimensional vector quantization. The space is partitioned into 4 cells, vectors falling in the same cell are highly similar under the distance measurement chosen for the recognizer design. We use the centroid value of that cell as the quantized value of these vectors.

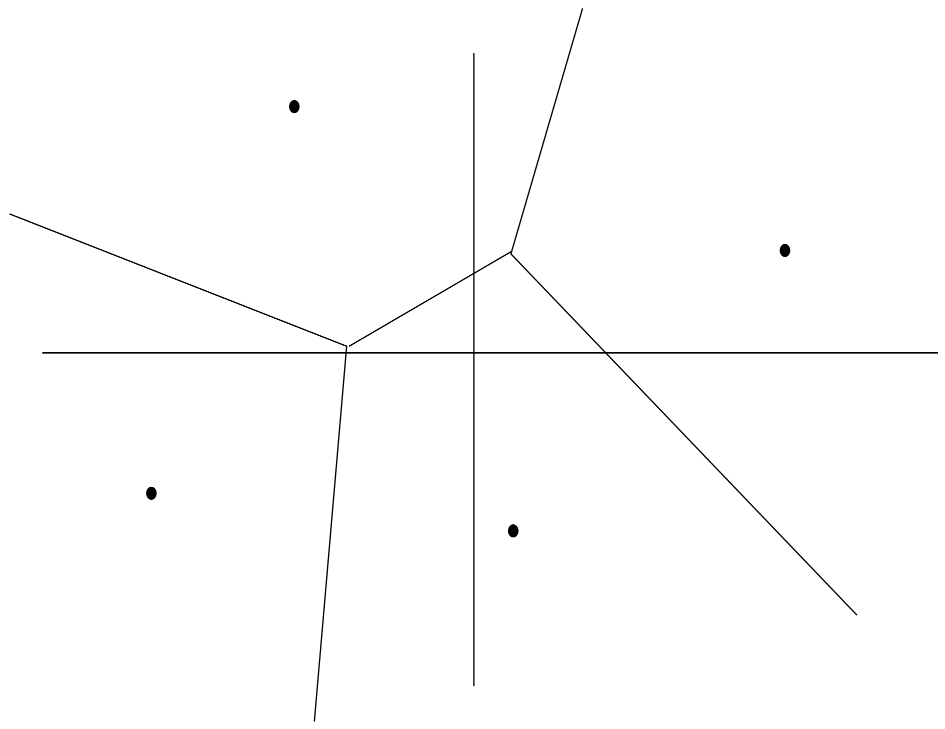


Figure 3. Partitioning of 2-dimensional space into 4 cells.

A vector quantizer is usually trained by a large data set. After the training process, we get a codebook containing the centroid value of each cell. In a speech recognition application, for each word in the vocabulary, we perform this training procedure, and catenate codewords to form the template for this word. The performance of a quantizer is measured by the average distortion. Suppose an original vector is $\vec{x}$, the quantized value is $\vec{z}$, and associated quantization distortion is $d_f(\vec{x}, \vec{z})$. Then the average distortion is

$$D = \frac{1}{N}\sum_{i=1}^{N} d_f(\vec{x}_i, \vec{z}_i), \tag{3.1}$$

where N is number of training vectors.

3.2. Training Database

The major concern of VQ is to find the set of codewords. As mentioned in the last section, this goal is usually achieved by training on a large database. Once the training data set is large enough, we say the result is optimal.

We used the TI digits database as the training database. This database was collected by Texas Instruments via telephone lines, and includes over 100 speakers. The part of training signals that we used in this project is the male-subset. There are 20 speakers, each having two renditions for one word, one in increasing intolerance, and the other in decreasing intolerance. For each word we have 38 to 40 training signals, and the length of these training signals ranges from 60 frames to 120 frames.

3.3. K-means Algorithm

Suppose for one word in the vocabulary, we have N training signals, each with length $L_1, L_2, \ldots, L_N$, where L_i is not necessarily to be equal to L_j when $i \neq j$. We use L to indicate the total number of vectors in all these N signals, that is

$$L = \sum_{i=1}^{N} L_i. \tag{3.2}$$

We also assume that each template has M states. The training goal is to classify these L vectors into M groups. The centroid of each group will be represented using $\vec{z}_i$, where $i = 1, 2, \ldots, M$. Catenating the $\vec{z}_i$ together, we get the template for this word. The K-means algorithm is a simple but powerful classification algorithm for performing unsupervised training. The procedure is:

1. Initialization: Arbitrarily choose M vectors as the initial set of codewords in the codebook.

2. Nearest-Neighbor Search: For each training vector, find the codeword in the current codebook that is closest, and assign this vector to the corresponding class.

3. Centroid update: Compute the centroid of the training vectors in each class and use these new values to form the updated codebook.

4. Iteration: Repeat steps 2 and 3 until the average distortion falls below a pre-determined threshold.

Repeating this procedure for all words in the vocabulary, we get a complete template set.

In this project, the template for each word contains 33 states: 11 states for the word itself, surrounded by a silence model state on each side of the word.

CHAPTER 4

DYNAMIC TIME WARPING ALGORITHM

Many algorithms are currently used for searching a best matching path between two signals. Some of the more successful techniques include hidden Markov models applied at both the phoneme and at the word level. However, dynamic programming (DP) remains the most widely used algorithm for real-time recognition systems. It is considered sufficiently mature to solve sequential decision problems. Silverman and Morgan gave detailed description of the history of the application of DP in speech recognition in [9]. In this chapter, we will look at the general concept of dynamic programming first, then dynamic time warping as an application of DP in speech recognition will be introduced in detail.

4.1. Problem Definition

Because of different accents, speaking styles of individual speakers, etc., waveforms of one word may have a lot of differences. Even spoken by the same speaker, differences still exist because of speaking rate, loudness and stress. Some patterns may have a longer duration and higher amplitude; others may be shorter and weaker. Parts (a) and (b) of figure 4 show two repetitions of the word "speech." Comparing these two signals, phones s, p and ch of signal 1 are shorter, and phone iy is longer, as part (c) of figure 4. Here, we use a dotted line to show signal 1, and a solid line to show signal 2 so that they can be distinguished in the same plot. The problem of comparing these two signals as they are is that their lengths are not the same. From chapter 2, we know that a speech signal can be represented by a sequence of feature vectors. Let's use $\vec{a} = \{a_1, a_2, \ldots, a_I\}$ to indicate signal1, and $\vec{r} = \{r_1, r_2, \ldots, r_J\}$ to indicate signal 2. The lengths of these two signals may not be the same so that I does not necessarily equal

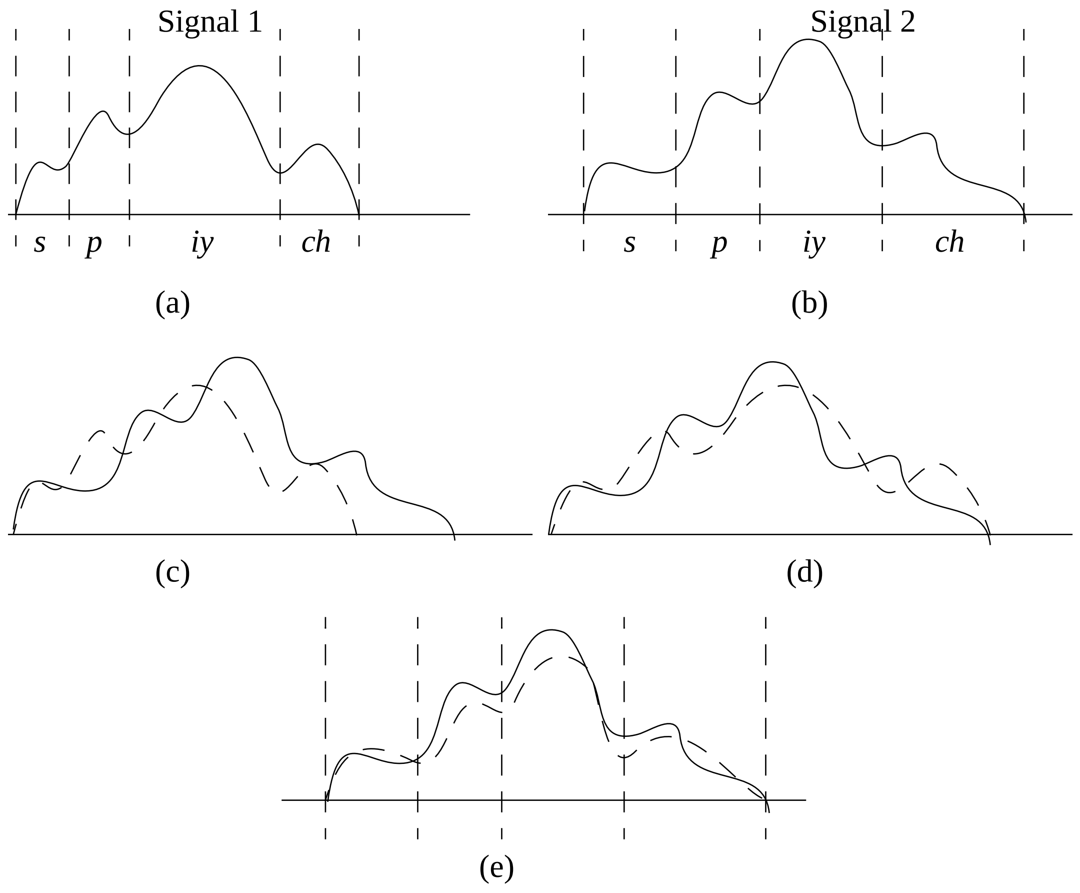


Figure 4. Comparison of different time normalizations for the word "speech."

to J. A simple normalization scheme is to extend the shorter signal linearly to the length of the longer one. Let's assume that $J < I$, and use $D(\vec{r}, \vec{a})$ to denote the distance between signal $\vec{r}$ and $\vec{a}$, which can be computed as

$$D(\vec{r}, \vec{a}) = \sum_{i=1}^{I} d_f(r_{i'}, a_i) \qquad i' = \frac{J}{I} \times i, \tag{4.1}$$

where $d_f(r_{j'}, a_i)$ denotes the distance between frame r_j and a_i.

The result of this linear time normalization alignment is shown as part (d) of figure 4. It assumes that the speaking rate is proportional to the duration of the utterance, and is independent of the sound being spoken. But from the plot of signal $\vec{r}$ and $\vec{a}$, we know that this assumption may not model the real situation of speech utterance. A good normalization scheme should change the duration of one signal according to the characters of the other signal, to achieve a reasonable matching between them, so that the distance $D(\vec{r}, \vec{a})$ can be minimized. Part (e) of figure 4 shows a nonlinear time normalization example. We can see that the parts of s, p and ch of signal 1 are extended, and part iy is compressed [2].

4.2. Dynamic Programming

Dynamic programming is commonly used to solve sequential decision problems. The nonlinear matching described in the last section is usually performed in a grid plane as shown in figure 5. As defined in section 4.1, the signals to be matched are $\vec{a}$ and $\vec{r}$. In the grid plane, signal $\vec{a}$ is aligned along the x-axis, and signal $\vec{r}$ is aligned along the y-axis. Each intersection in this plane is defined as a node, where node (i, j) means matching frame i of signal $\vec{a}$ with frame j of signal $\vec{r}$. The node $(0, 0)$ is called the original node, which is a dummy node that all paths start from. The cost associated with this matching is defined as the frame distance $d_f(i, j)$ between feature vectors $a(i)$ and $r(j)$, which was discussed in chapter 2. The cost at node $(0, 0)$ is defined to be zero, which is

$$d_f(0, 0) = 0. \tag{4.2}$$

A path is defined as the catenation of node pairs $(i_{k-1}, j_{k-1}) \rightarrow (i_k, j_k)$, which means extending from node (i_{k-1}, j_{k-1}) to node (i_k, j_k). Here we use i_k to indicate the index of signal $\vec{a}$ at time k, and j_k is the index of signal $\vec{r}$ at time k. A path which starts

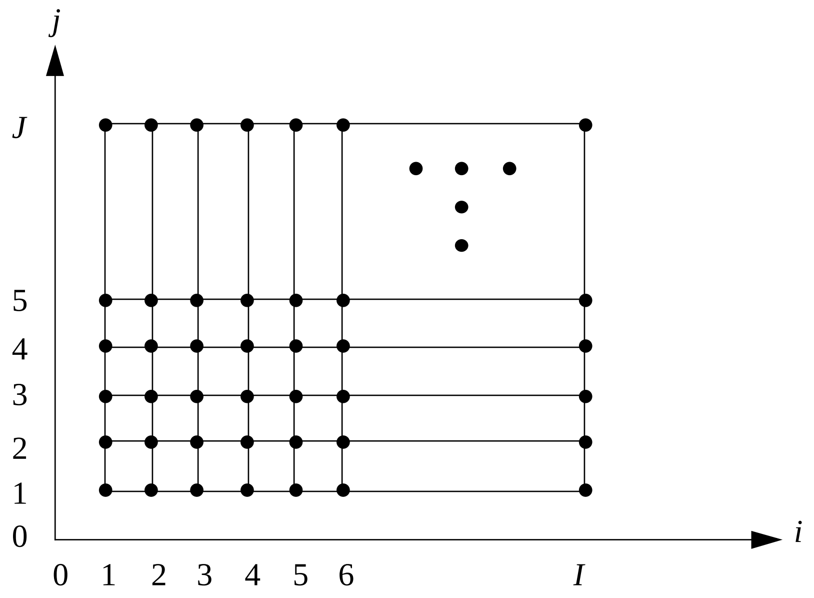


Figure 5. The grid plane to illustrate dynamic programming.

from node $(0, 0)$ and ends at node (i_k, j_k) has an overall cost, which means the accumulated cost from the starting point of this path until it meets node (i_k, j_k). We use $D(i_k, j_k)$ to denote this type of cost, and it is defined as

$$D(i_k, j_k) = D(i_{k-1}, j_{k-1}) + d_f(i_k, j_k)\,. \tag{4.2}$$

Since node $(0, 0)$ is the dummy starting node for all paths, we define

$$D(0, 0) = 0\,. \tag{4.3}$$

Tracing the right part of equation (4.2) back to $k = 1$, we get

$$D(i_k, j_k) = \sum_{m=0}^{k} d_f(i_m, j_m)\,. \tag{4.4}$$

The problem can be written as finding a sequence of nodes (i_k, j_k) which minimizes the accumulated cost for a complete path ending at node (I, J):

$$\begin{aligned} D^*(i_k, j_k) &= min[D(i_{k-1}, j_{k-1})] + d_f(i_k, j_k) \\ &= min\left[\sum_{m=0}^{k} d_f(i_m, j_m)\right] \end{aligned}\,. \tag{4.5}$$

In this equation, $(i_k, j_k) = (I, J)$.

4.3. Dynamic Time Warping

As discussed in chapter 1, the pattern matching method in speech recognition is used to measure the distance between the test signal and all of the templates, and then pick the template with smallest distance. The word that this template refers to is the recognition result. Therefore, the recognition problem can be simplified as finding the distance between a signal and a template. From section 4.1, we know that the challenge of this problem is the diversity of the speaking behavior of humans. The differences imported due to speaking rate, style or accents etc., should not affect the recognition result. The discussion of the last section indicated that dynamic programming can be used to find a nonlinear matching path between the test signal and the template, so that the distance between them is minimized. This application of dynamic programming is called dynamic time warping.

Dynamic time warping uses the same grid plane of figure 5 to define the search space. Here, signal $\vec{a}$ is the test signal, and $\vec{r}$ is the template signal. We will search for the best path along the index of the test signal. As the specific application of dynamic programming in speech recognition, DTW applies particular constraints to the problem according to *a priori* information known about the speech signal, and these constraints define how abruptly the time-scale of one signal can be changed relative to the other signal. The paths of the search space which are considered unreasonable are pruned [1]. The commonly-used constraints will be discussed in the following subsections.

1. Endpoint Constraints

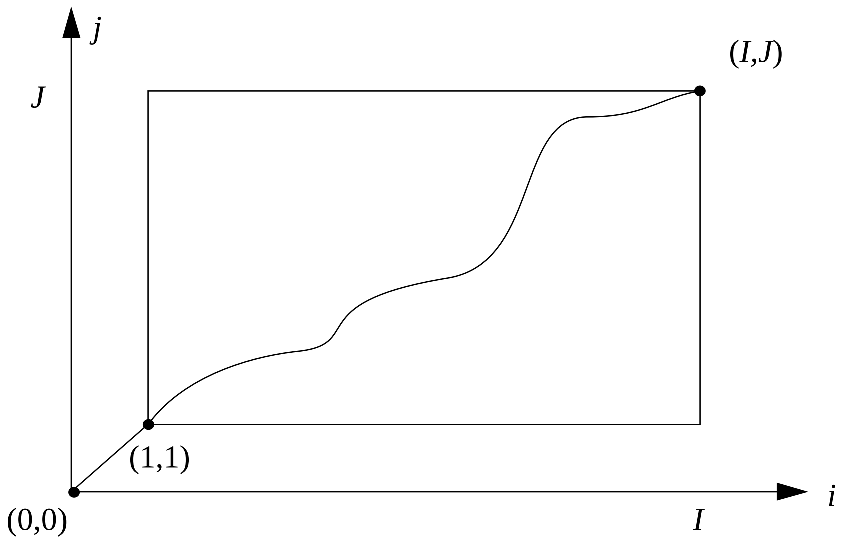


Figure 6. The ending-point constraints.

Starting point constraint: the first frame of the test signal must be matched with the first frame of the template signal:

$$(i_1, j_1) = (1, 1). \tag{4.6}$$

Ending point constraint: the last frame of the test signal must be matched with the last frame of the reference signal

$$(i_k, j_k) = (I, J). \tag{4.6}$$

Under these constraints, a typical matching path would look like the curve in figure 6.

2. Continuity and monotonicity constraints:

The continuity constraint states that there is no “breaking” point in a path, which means

$$i_k - i_{k-1} = 1. \tag{4.7}$$

In other words, the matching must be made along the axis of the test signal one frame after another.

The monotonicity constraint requires vectors of a word to be clustered in monotonically increasing order, which means

$$j_k - j_{k-1} \geq 0. \tag{4.8}$$

Figure 7 is an example of non-monotonic matching. The signal of “pest” is listed along x-axis, and “pets” is listed along y-axis. The phoneme sets of both the words “pets” and “pest” are $\{p, eh, s, t\}$. Without monotonicity constraints, these two signals can be matched along the solid line, which gives the incorrect information that these two signals are similar.

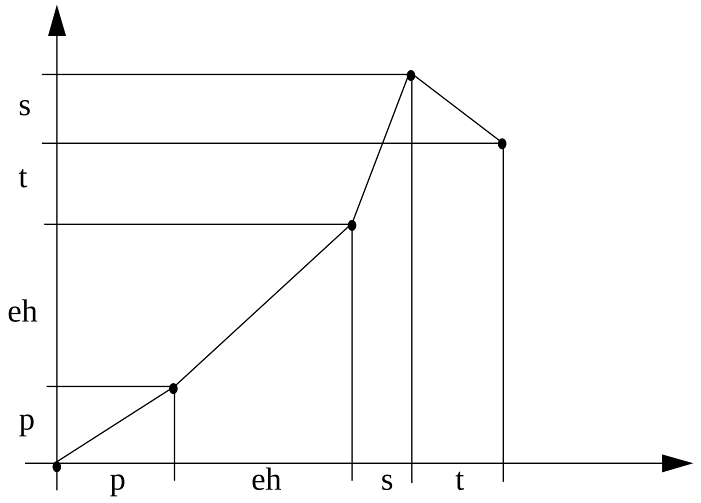


Figure 7. Non-monotonic matching between “pest” and “pets”.

3. Global Path Constraint

This constraint is used to restrict the extent of compression or expansion of speech signals over long ranges of time. The variation of the speaking rate of human beings is considered to be limited in a reasonable range, which means that we can prune the unreasonable search space, and limit the search to the “legal” region. The commonly-used global path constraints is shown in figure 8.

Here the search region is limited by four straight lines whose slopes are s and $1/s$, and pass through node $(1, 1)$ and (I, J). The shadow area is the legal search region.

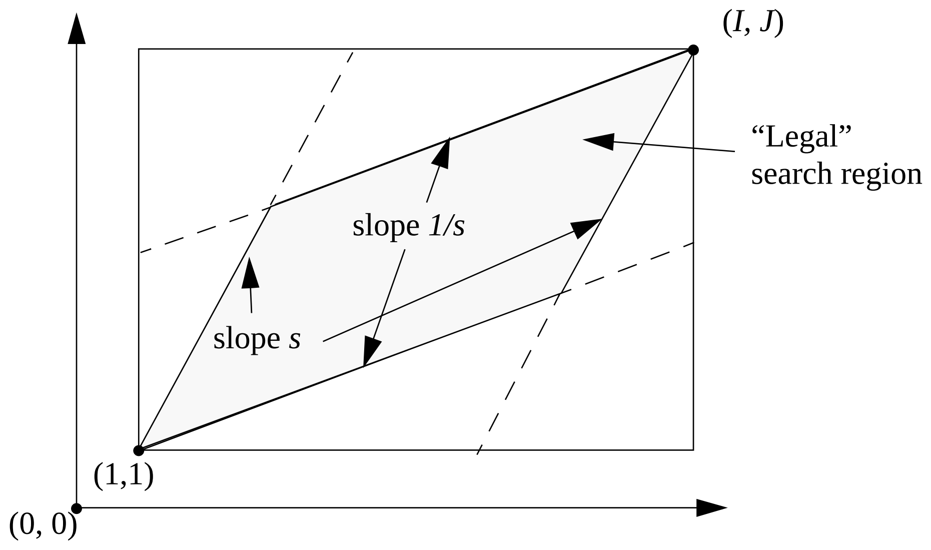


Figure 8. The global path constraints.

4.4. DTW Algorithmic Details

In this section, we will develop the DTW algorithm which will be used in this project. First of all, let us define the variables and search space which will be used in the algorithm development.

This algorithm should take the LPC vector sequence of the template and test signal as input, and the output should be the matching path and the accumulated cost associated with the path. We need two arrays to store the template and test signal. Also we need to store the path and cost. Let's use $template[j][k]$ to denote the k^{th} component of the j^{th} frame of the template, where $0 \le k \le p$, and p is the order of the LP coding. The range of i is from 1 to J. Also $test[i][k]$ is the k^{th} component of the i^{th} frame of the test signal, and j ranges from 1 to I. We use figure 9 as the search plane. In the figure, each column reflects a frame of test signal, and each row reflects a frame of template. An intersection is a possible node in the path, where node (i, j) means frame i of test signal is matched with frame j of the template, and the accumulated cost for the path going through node (i, j) will be stored in $cost[i][j]$. Every node in column $i-1$ is a possible predecessor of node (i, j). We store the choice of predecessor for each node, so that the path can be obtained by tracing from the ending point to the starting point. The value of $predecessor[i][j]$ is

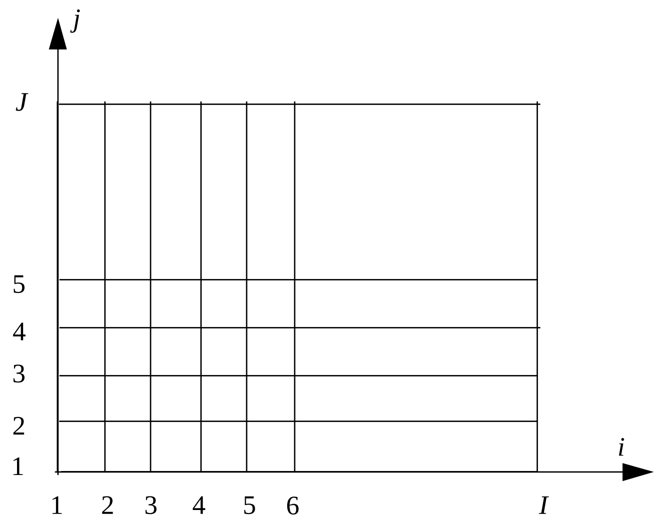


Figure 9. The search space of DTW.

the index of the template of the chosen predecessor of node (i, j).

Now let's develop the algorithm. According to the endpoint constraints, all paths start from node $(1, 1)$; this means $predecessor[2][j] = 1$, for $1 \le j \le J$.

To satisfy the global path constraints, the nodes to be examined are restricted to a certain range. From figure 8, we know that the upper boundary of the value of the j^{th} index is defined by the value of the i^{th} index:

$$maxJ = min\begin{pmatrix} s \times i - s + 1 \\ \frac{1}{s} \times i - \left(\frac{1}{s} \times I - J\right) \end{pmatrix}, \tag{4.9}$$

and the lower boundary is:

$$minJ = max\begin{pmatrix} s \times i - (s \times I - J) \\ \frac{1}{s} \times i - \frac{1}{s} + 1 \end{pmatrix}, \tag{4.10}$$

where s is the slope which defines the constraints.

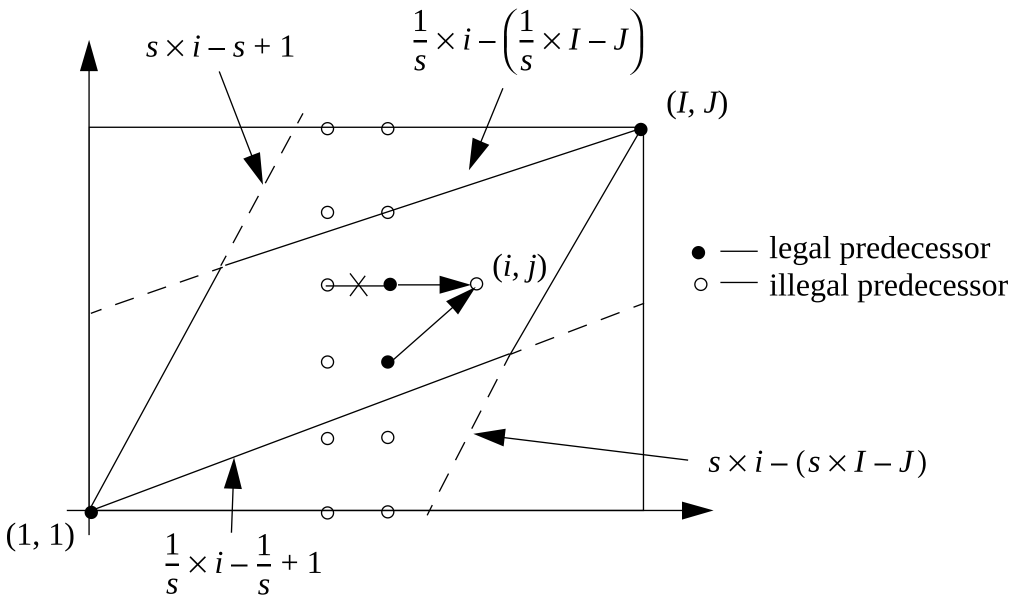

$s \times i - s + 1$
$\frac{1}{s} \times i - \left(\frac{1}{s} \times I - J\right)$
(I, J)
(i, j)
● — legal predecessor
○ — illegal predecessor
$s \times i - (s \times I - J)$
$(1, 1)$
$\frac{1}{s} \times i - \frac{1}{s} + 1$

Figure 10. Legal predecessors for node (i, j).

And for node (i, j), a slope of value s means that if $predecessor[i_{k-s}][j_{k-s}] = j$, then node (i_{k-1}, j) is not a legal predecessor. Combining the requirement of monotonicity constraints, the choice of predecessors should also satisfy equation 4.8. Figure 10 shows the possible predecessors for node (i, j) for $s = 2$.

As we discussed in previous sections, at each node (i, j), the DTW algorithm computes the accumulated cost of the paths coming from all possible predecessors. Pick out the one with minimum cost, and store the index of the template frame of that predecessor in $predecessor[i][j]$, and store the accumulated cost in $cost[i][j]$. After the computation for all frames of the test signal, the algorithm traces back the path from node (I, J) to node $(1, 1)$. Starting from $i = I$ and $j = J$, the value of $predecessor[i][j]$ is the index of the template frame for column $i - 1$. This means node $(I - 1, predecessor[I][J])$ is the predecessor node of node (I, J), and it will be stored in array $path[i]$. The process repeats until node $(1, 1)$; then array $path[i]$ will be reversed to form the ordered sequence of indices for the template frames. The procedure of this forward-DTW and backtracing algorithm is shown in figure 11.

```
1. Initialization:  $cost[1][1] = d_f(1, 1)$,
                    $predecessor[2][j] = 1 \qquad 0 \le j \le J$.
2. Search for best path:
   for $i = 2, \ldots, I$,
       $maxJ = min\left(\begin{matrix} s \times i - s + 1 \\ \frac{1}{s} \times i - \left(\frac{1}{s} \times I - J\right) \end{matrix}\right)$,
       $minJ = max\left(\begin{matrix} s \times i - (s \times I - J) \\ \frac{1}{s} \times i - \frac{1}{s} + 1 \end{matrix}\right)$.
       for $j = minJ, \ldots, maxJ$
           for $k = j - s, \ldots, j$
               if $predecessor[i - s][j] = j$
                   continue
               else
                   compute $tempcost[i][k] = cost[i-1][k] + d_f[i][j]$
           next $k$
           $predecessor[i][j] = \underset{k}{\arg min}\{tempcost([i][k])\}$.
           $cost[i][j] = cost[i-1][predecessor[i][j]] + d_f[i][j]$ .
       next $j$
   next $i$
3. Back trace:
   $path[1] = J$,
   for $i = I, I-1, \ldots, 2$,
           $path[i] = predecessor[i][j]$
   next $i$
   reverse $path[i]$
```

Figure 11. The forward-DTW and backtracing algorithm.

CHAPTER 5

SOFTWARE DESIGN AND IMPLEMENTATION

As mentioned in Chapter 1, the goal of this project is to build an educational demo. It requires that the users should be able to choose different signals, distance measurements and constraints. More importantly, the system should be able to give the user a clear idea what the signal “looks like,” what the grid path looks like, and the comparison of the frames that are matched together. Also, the user would want to listen to the test signal, since we are dealing with speech signals. To fulfill these requirements, this applet is designed to contain two major parts. The first part is the core computation part, which is the search engine for the best matching path. The other part is the user interface. We use the spectrogram, energy plot and waveform to display the selected signals. There are six panels in the applet:

- Menu Panel -- designed to control the whole applet, from which user can select different signals, templates and compute modes;
- Test Signal Panel -- used to show the spectrogram, energy plot and waveform of a selected test signal;
- Template Panel -- used to show the spectrogram and energy plot of a selected template;
- Gridpath Panel -- used to show the matching path for a selected signal-template pair;
- Warped Output Panel -- used to show the spectrogram of a warped template after DTW matching;
- Process Description Panel -- used to describe the procedure that the applet is working on, and to show the matching score and recognized result.

The applet layout is shown in Figure 12.

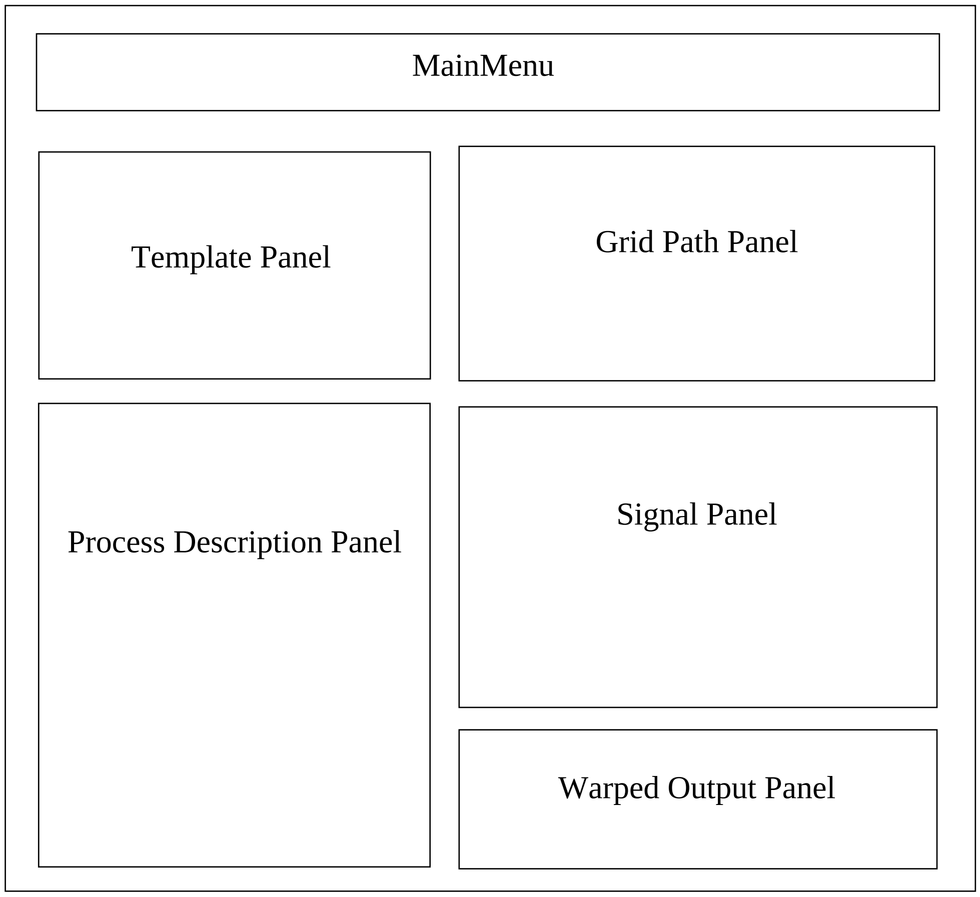


Figure 12. Applet layout.

There are 13 classes involved in this recognition system:

1. MainMenu

This class is the main control of the whole applet. From this class, the user can select different templates and test signals from a pull-down menu. The "Models" menu provides 11 templates that are used in this recognition system, and the "Signal" menu provides one test signal for each word. The user can also use "From File" to input the URL of the ideal signal, and "Select Signal" to choose from a database of 1210 choices. Users can choose different constraints and distance measurements from related menu. In the

“Edit” menu, the “Scaling” allows the user to specify the display region they are interested in, and “Reset” will set the display region to the original value.

2. lpcData

This class is used to represent a linear predictive coded speech signal. It reads in the signal from an ASCII file, and stores it as a sequence of LPC vectors in a 2-dimensional array.

3. match

This class is the computational core of the applet. After selecting the signals, the distance measurement, and the slope for global path constraints, it searches the least cost matching path using the dynamic time warping algorithm. After this forward-searching procedure, the backTrace() method will trace the path from the matrix which stores the index of the predecessor for each node to form the path array. This result will be sent to the PathPanel class to draw the path in a grid pane.

4. PathDrawingPanel

This class is used to draw the best path after searching. It accepts a path which is an 1-dimension integer array and draws the path by connecting adjacent nodes together.

5. PathPanel

This class applies a title frame to the PathDrawingPanel.

6. SpectrumDrawingPanel

This class is used to draw the spectrogram for a selected signal or template, and for the warped template. It accepts a signal represented as a sequence of linear prediction coefficients, computes the energy for each frequency internal of each frame, maps the energy value to a color, and draws a rectangle for it. Connecting the color cells together, a spectrogram for the signal is drawn. There are six methods involved in the spectrogram drawing process: attachData() is used in accepting the signal to be drawn; attachPath() is used to accept a warping path so that the normalized template can be obtained by concatenating the frames from the template according to the path; computeSpectrum() is used to compute the energy spectrum for the attached signal; drawVerticalSpectrum() is used to draw a spectrogram whose time index is set along the y-axis. This method is used to draw the spectrogram of selected templates. drawHorizontalSpectrum() is used to draw a spectrogram whose time index is set along the x-axis, and it's used to draw the spectrogram for a selected test signal. drawWarped() is used to draw the spectrogram for the normalized template according to the warping path.

7. EnergyDrawingPanel

This class is used to draw the energy plot for a selected test signal and template. Like the SpectrumDrawingPanel class, this class accepts a signal represented as a sequence of linear prediction coefficients. It computes the energy for each frame, and draws the energy plot as a function of time. There are four methods involved in the drawing process: attachData() is used in accepting the signal, computeEnergy() is used to get the energy amplitude for each frame, drawVertical() is used to draw the energy plot along the y-axis, and drawHorizontal() is used to draw the energy plot along the x-axis.

8. WaveformDrawingPanel

This class is used to draw the waveform for a selected test signal which has an associated audio file. It first reads in the signal from a ".au" file, and stores each sample as a 16-bit integer, then draws the signal by connecting adjacent samples.

9. SignalPanel

This class is used to display a selected test signal which contains a SpectrumDrawingPanel, a EnergyDrawingPanel and a WaveformDrawingPanel.

10. TemplatePanel

This class is used to display a selected template which contains a SpectrumDrawingPanel and a EnergyDrawingPanel.

11. WarpOutputPanel

This class is used to display the normalized template which contains a SpectrumDrawingPanel.

12. ProcessBox

This class is used to type out the matching cost for the selected test signal and template, or to tell the user the recognition result. It also prints out the matching condition, including the distance metric and selected slope.

13. SignalChooser

This class is used to provide a window for the user to choose test signal from a database.

CHAPTER 6

SUMMARY

In this project, we defined several fundamental problems in automatic speech recognition. We choose linear prediction as a representation for the spectrum of a speech signal, and generated a template set using the k-means clustering algorithm. Next, we used dynamic time warping algorithm to search for the least cost matching path between a test signal and a template. The system was implemented as a Java applet. The applet allows the user to select different test signals, choose models, distance measurements and constraints. The user can configure the applet to compute the matching score between a selected test signal and template, or to recognize an unknown signal. The applet also allows the user to compare performance between different distance measurement and constraints. We have tested the applet using the TI digits database. The test data set contains 1210 signals, which does not have overlapping with the training set that we used to generate the template set. The word error rate is as low as 9.7%, comparing with only 431 training signals, this error rate is very low. Besides, the applet provides spectrogram, energy plot and waveform drawing functions to the signals, and graphical result display, which helps the user to understand what is going on. The applet was implemented on JDK 1.2. We used Java Swing to implement the GUI, which is 100% pure Java. The look and feel of the applet is now platform independent.

During the development of this project, we also gained a comprehensive understanding of the theory involved in a DTW speech recognition system. Besides the dynamic time warping algorithm, we also examined linear prediction modeling and vector quantization. Furthermore, we visualized these theories using this tool. This project is a good beginning to the development of tools for a fundamental of speech recognition course. It serves a nice complement to other speech recognition-specific tools available on the ISIP web site.

REFERENCES

[1] J.R. Deller, J.G. Proakis, J.H.L. Hansen, *Discrete Time Processing of Speech Signals*, MacMillian Publishing Co., New York, New York, USA, 1993.

[2] L. Rabiner and B.H. Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1993.

[3] J. Shaffer, J. Hamaker and J. Picone, “The Visualization of Signal Processing Concepts,” *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1853-1856, Seattle, Washington, USA, May 1998.

[4] L.R. Rabiner and R.W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1978.

[5] J.G. Proakis and D.G. Manolakis, *Digital Signal Processing Principles, Algorithms, and Applications.* Prentice-Hall, Upper Saddle River, New Jersey 1996.

[6] J.D. Markel and A.H. Gray, Jr., *Linear Prediction of Speech*, Springer-Verlag, Berlin, Heidelberg, 1976.

[7] E.L. Bocchieri and G.R. Doddington. “Frame Specific Statistical Features for Speaker-Independent Speech Recognition,” *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol. 34, no. 755-764, August 1986.

[8] F. Itakura, “Minimum Prediction Residual Principle Applied to Speech Recognition,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 1, pp. 67-72, February 1975.

[9] H.F. Silverman and D.P. Morgan, “The Application of Dynamic Programming to Connected Speech Recognition,” *IEEE ASSP Magazine*, pp. 7-25, July 1990.

[10] X.D. Huang, Y.Ariki, M.A. Jack, *Hidden Markov Models for Speech Recognition*, Edinburgh University Press, 1990.

APPENDIX

Source Code

Please refer to http://www.isip.msstate.edu/projects/speech/education/demos/util/dynamic_time_warping/src/v2.0/ for detail.

APPENDIX

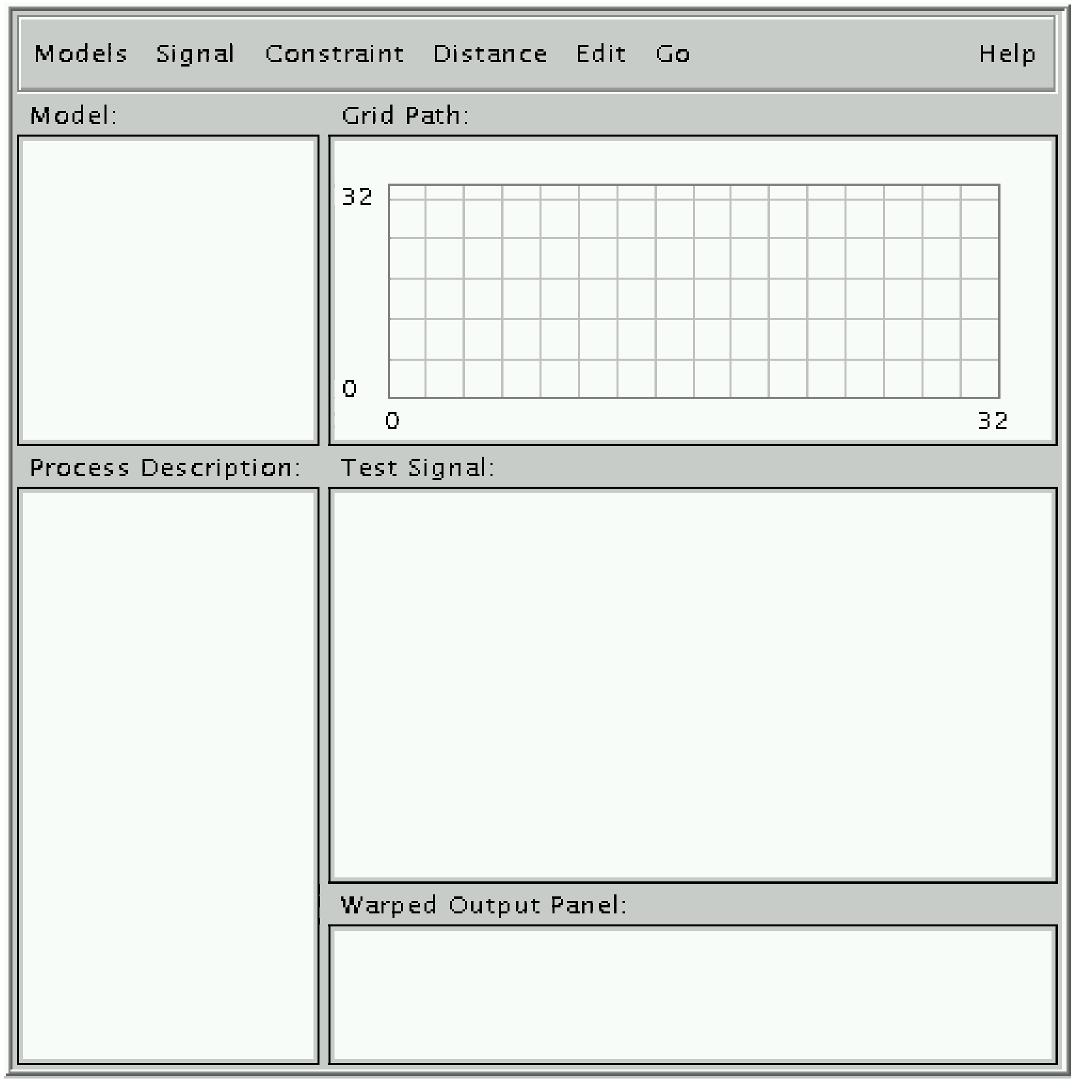
Tutorial

This document is meant as a tutorial for the first-time user. In this tutorial we walk the user through the different features of the applet with hands-on examples. In doing this, we try to reveal all of the features of the applet. We release this version of the software in hopes that it will be beneficial to its users.

STARTING UP

The first thing the user needs to do is to start the applet from our web site http://www.isip.msstate.edu/projects/speech/education/demos/dynamic_time_warping/. You could also download the source code from http://www.isip.msstate.edu/projects/speech/education/demos/util/dynamic_time_warping/v2.0/ and compile it yourself. At this point in the tutorial we will assume that you can access the applet and that you have started it.

Once the applet has loaded, you will see the following screen:



This screen has been partitioned into a Menu area, a Grid Path panel, a Test Signal panel, a Model panel, a Warped Output panel and a Process Description panel. All the elements have been enclosed in separate rectangular border for clarity.

The menu items are used to choose a model, a test signal to recognize, a path constraint or a distance measurement; also, you can scale the display range for signals.

The Model panel displays spectrogram and energy plots for a selected acoustic model.

The Test Signal panel draws the spectrogram, energy plot and waveform for a selected test signal.

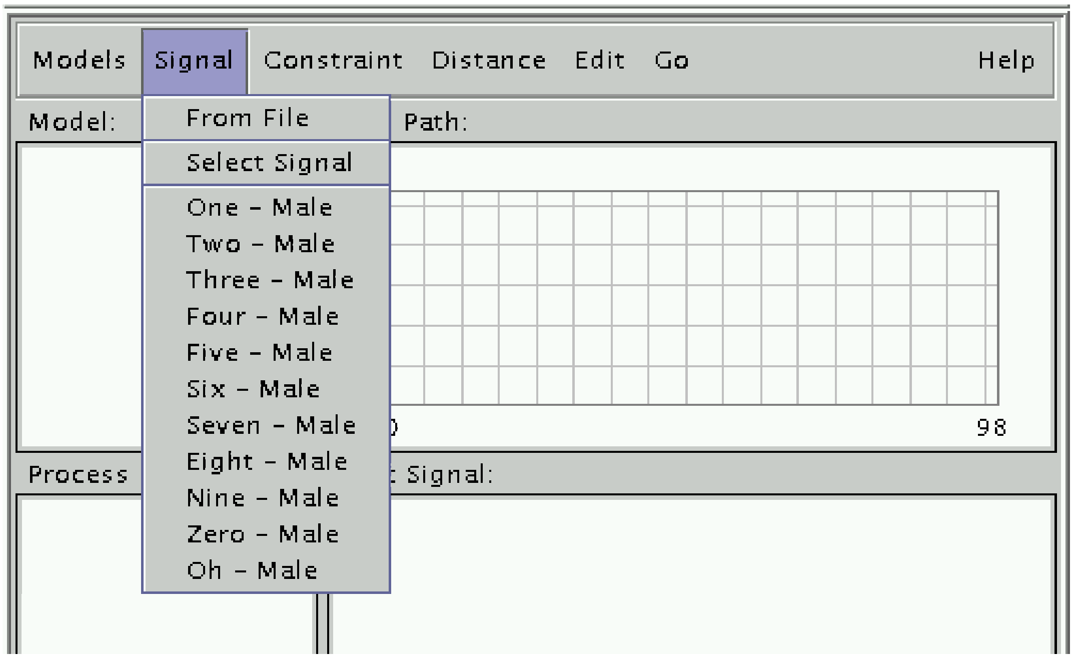
The Warped Output panel draws spectrogram for warped model.

The Grid Path panel shows the matching path between a selected model and test signal.

The Process Description panel tells the user what the applet is working on, the matching cost and recognition result.

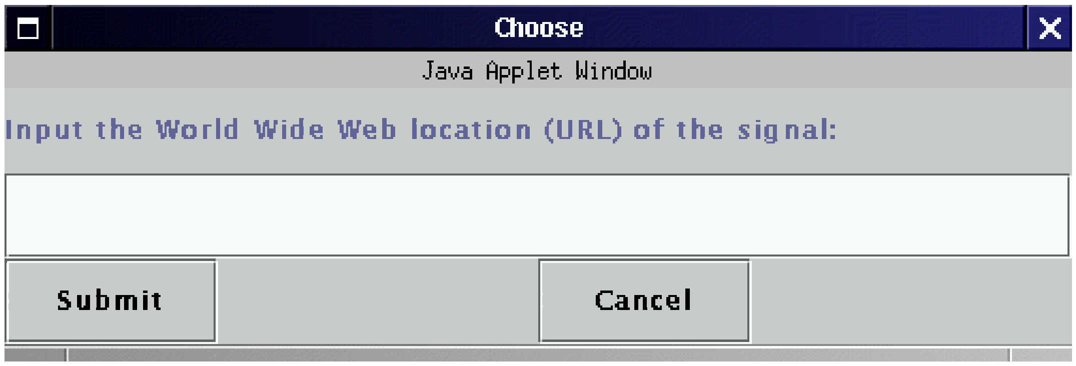
CHOOSING THE SIGNAL AND MODEL

The first step in using this tool is to decide which test signal you want to recognize. Click on **Signal**, and you will see that there are three ways to specify the test signal:



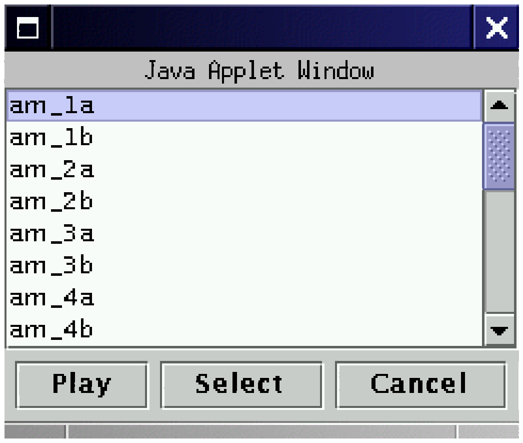
From top to bottom, we have *From File, Select Signal* and a small list specifying "One -- Male", "Two -- Male", etc.

1. The *From File* choice will bring you this input window:



If you have a signal in the proper format, you can type the URL of the file in the text field. By clicking the "Submit" button, the system will read in this file, and use it as the test signal. The "Cancel" button will cause you go back to the main screen.

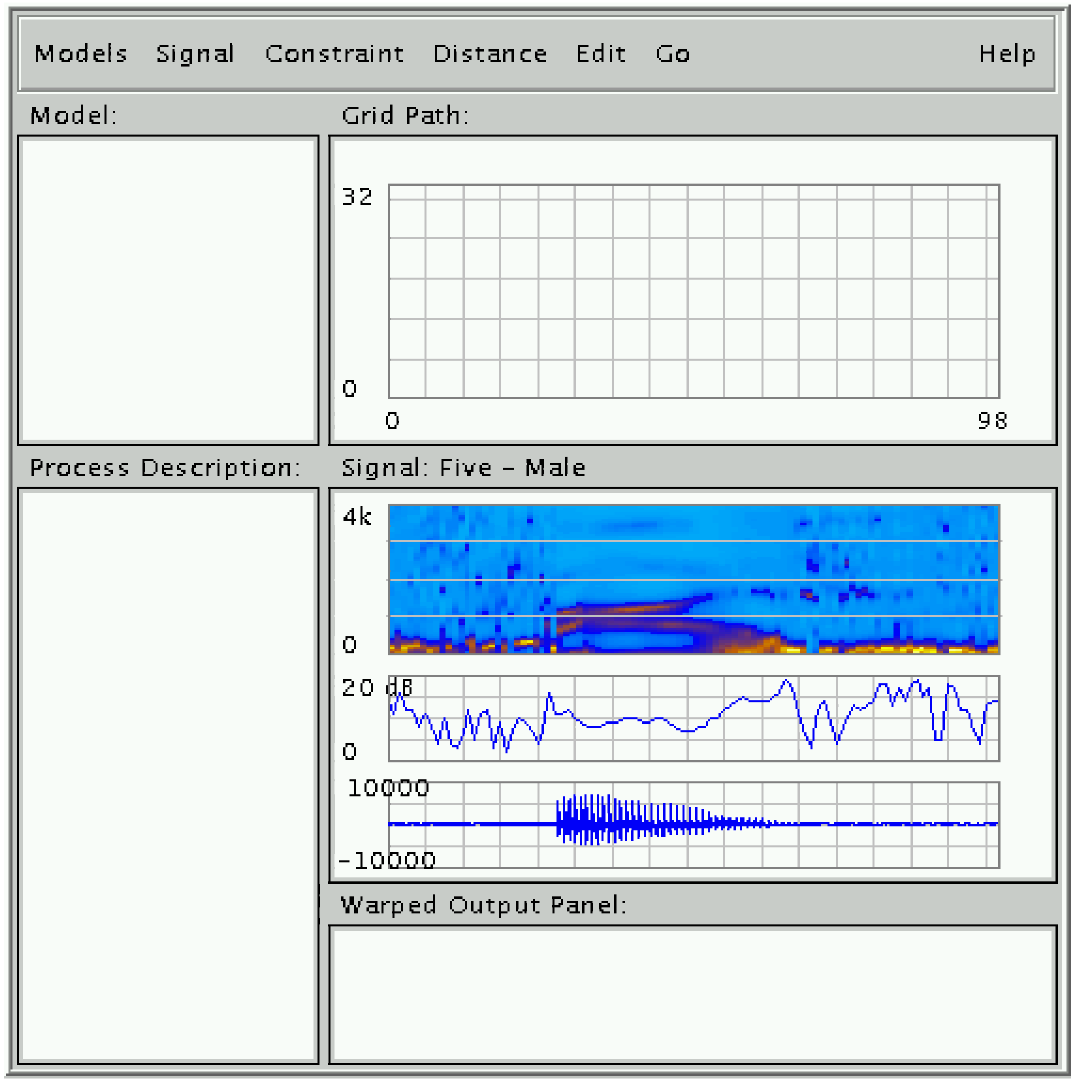
2. The *Selected Signal* choice will bring you this window:



You can choose a signal from the list, and click “Select” to confirm your choice. The “Play” button gives you a chance to listen to the chosen signal before making a decision. Also, the “Cancel” button will let you return to the main screen.

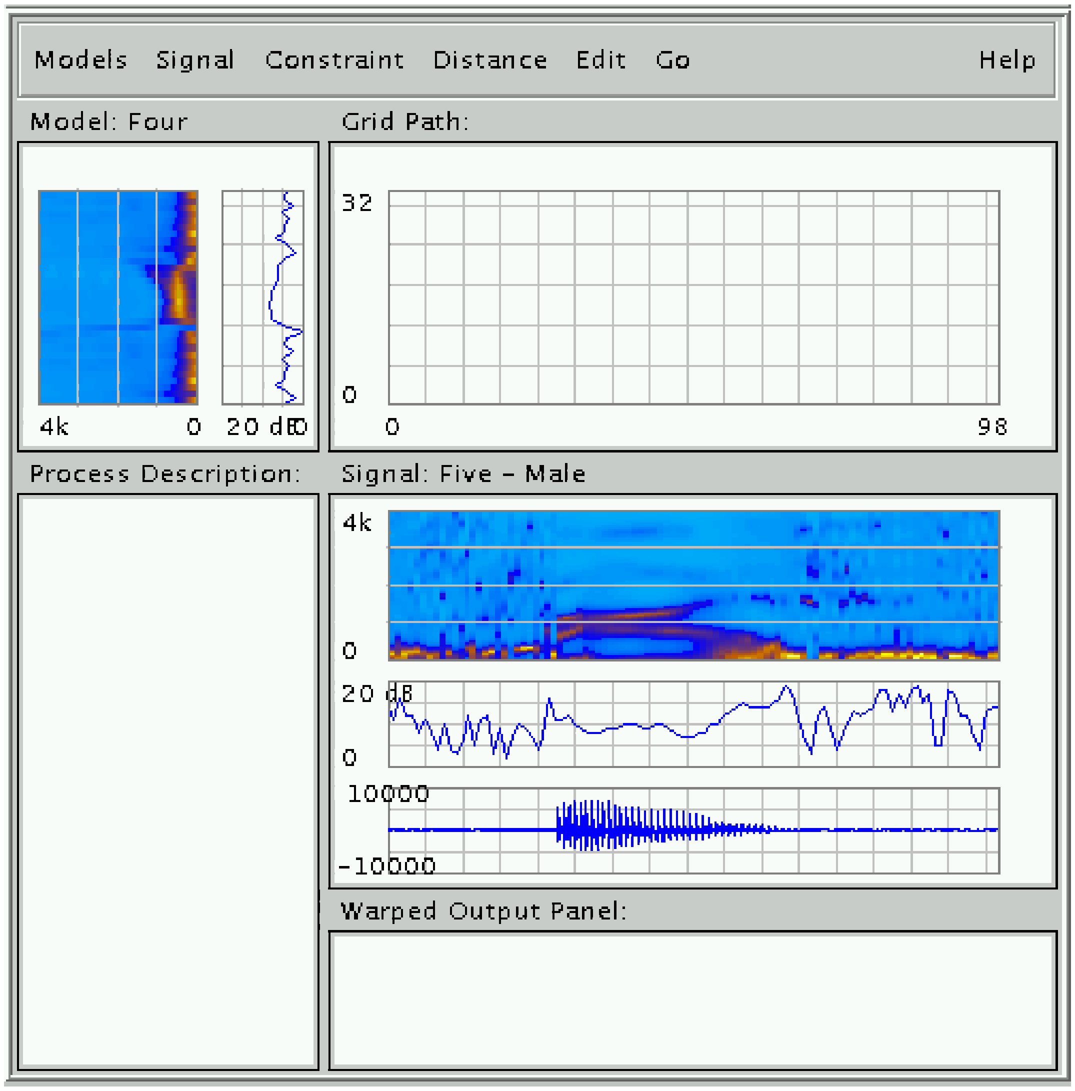
3. The limited signal list. This is the simplest way to specify your choice. The system has already integrated one test signal for each word in the vocabulary, and lists the name of each under the choice of *Selected Signal*. Clicking one of them brings it to the front stage.

In this tutorial we will use signal 5, so select “Five -- Male” now. If you see the following screen, then you did the right thing:



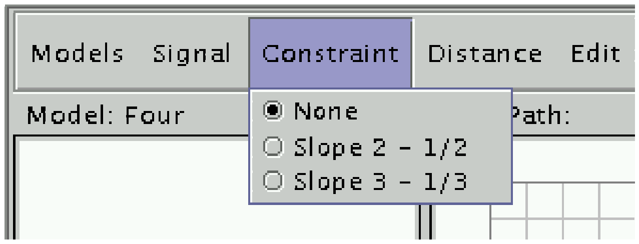
The next thing you may want to do is to specify a model. This system was built to recognize 10 digits plus the word “Oh” as an alternative for “zero.” The menu choice **Models** gives you a complete list of the templates that are used in this system. Clicking on one of them allows you specify the template that you want to use.

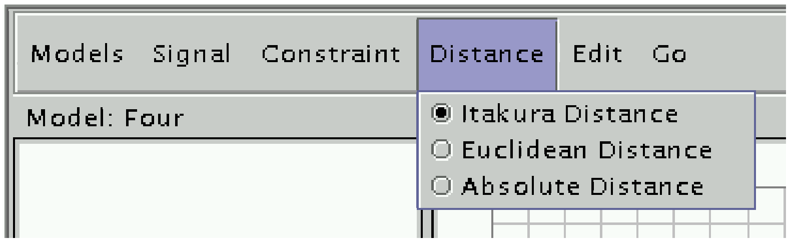
In this tutorial we will use model 4, so select “Four” from **Models** now, and you should see this:



SPECIFY PARAMETERS

Selecting constraint and distance measurements can be done by clicking the **Constraint** and **Distance** menus.





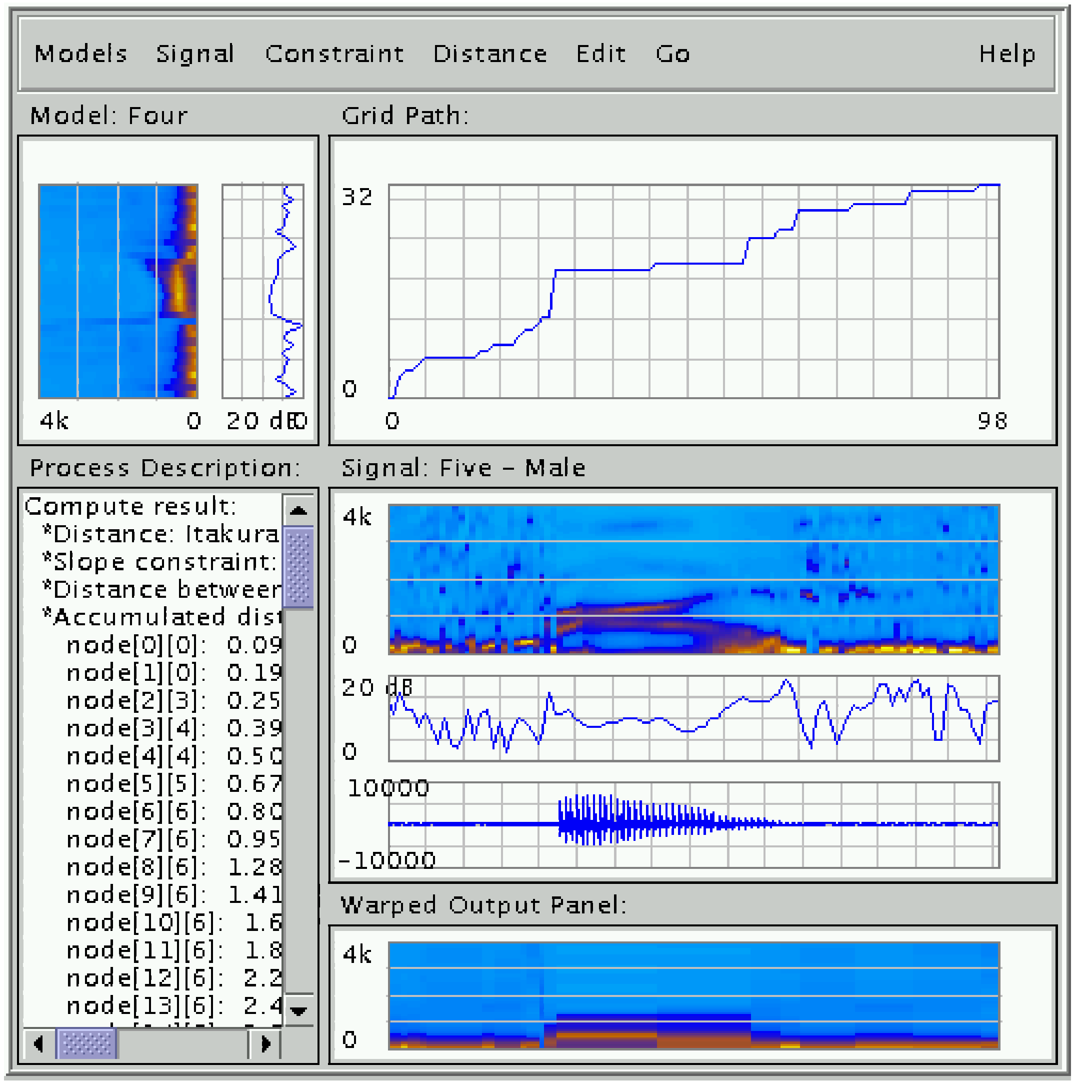
The applet provides three alternatives for global path constraints: *None, Slope 2 - 1/2 and Slope 3 - 1/3*. It also provides three distance metrics: *Itakura Distance, Euclidean Distance and Absolute Distance*. Let's use the default values in this tutorial, which are *None* for **Constraints** and *Itakura Distance* for **Distance**.

At this time, you have already selected a test signal and model, and have specified the constraint and distance metrics, so you are on the way to seeing what dynamic-time-warping is.

INVOKING THE TOOL

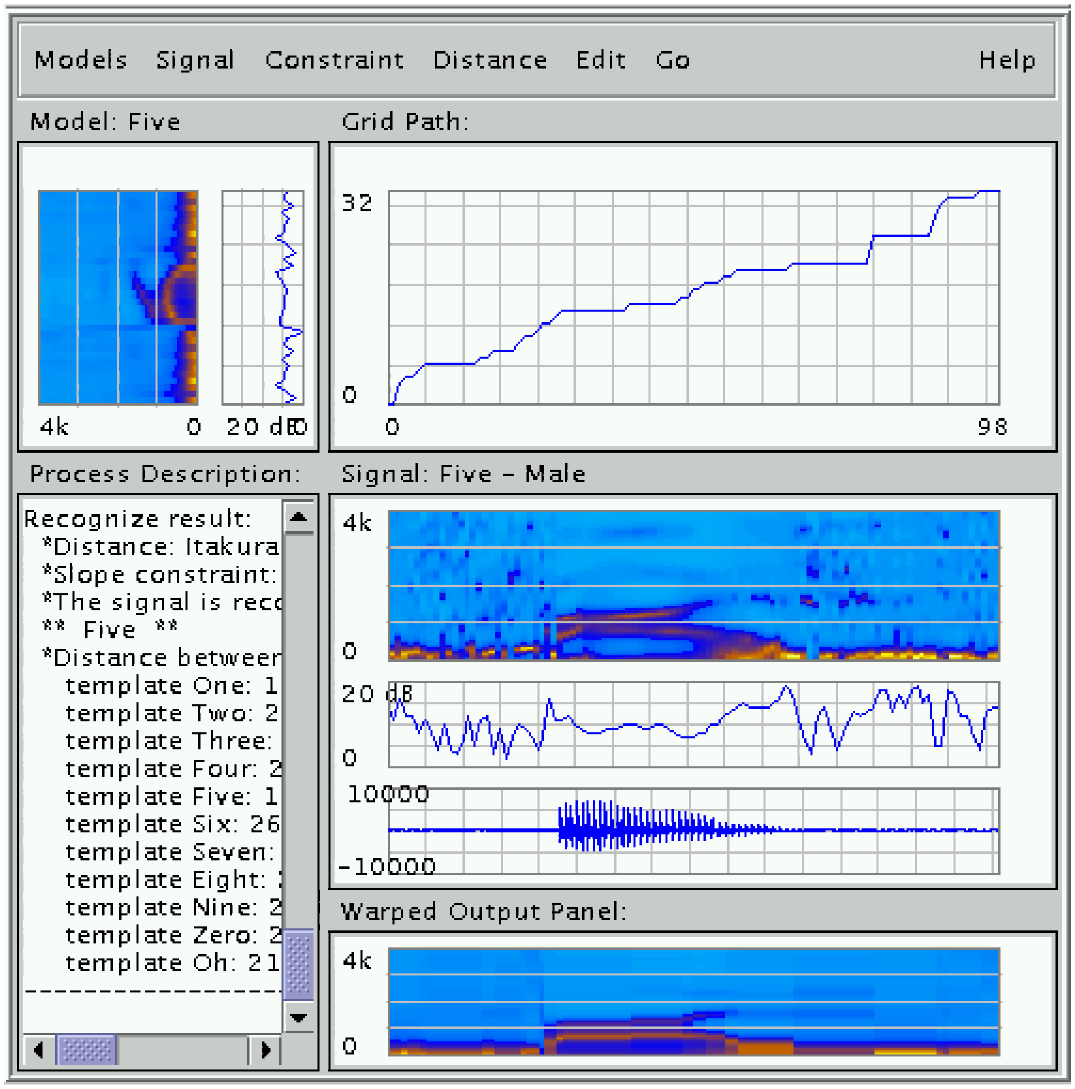
This system provides two process modes:

- The "Compute" mode will give you the matching between the selected test signal and the model. By choosing *Compute* from the **Go** menu, you should get the following on your screen:



```
Compute result:
  *Distance: Itakura
  *Slope constraint:
  *Distance between
  *Accumulated dist
    node[0][0]:  0.09
    node[1][0]:  0.19
    node[2][3]:  0.25
    node[3][4]:  0.39
    node[4][4]:  0.50
    node[5][5]:  0.67
    node[6][6]:  0.80
    node[7][6]:  0.95
    node[8][6]:  1.28
    node[9][6]:  1.41
    node[10][6]:  1.6
    node[11][6]:  1.8
    node[12][6]:  2.2
    node[13][6]:  2.4
```

- The “Recognize” mode will compare the matching score between the test signal and all of the templates, and choose the template which gives the least matching cost as the recognition result. Choose *Recognize* from the **Go** menu, and you will see this:

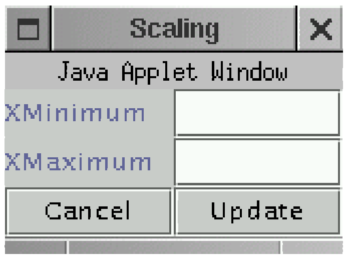


```
Recognize result:
 *Distance: Itakura
 *Slope constraint:
 *The signal is reco
 **  Five  **
 *Distance between
   template One: 1
   template Two: 2
   template Three:
   template Four: 2
   template Five: 1
   template Six: 26
   template Seven:
   template Eight:
   template Nine: 2
   template Zero: 2
   template Oh: 21
-------------------
```

Another choice that the **Go** menu provides is *Play.* This choice will play the selected test signal, if it is one provided with the applet.

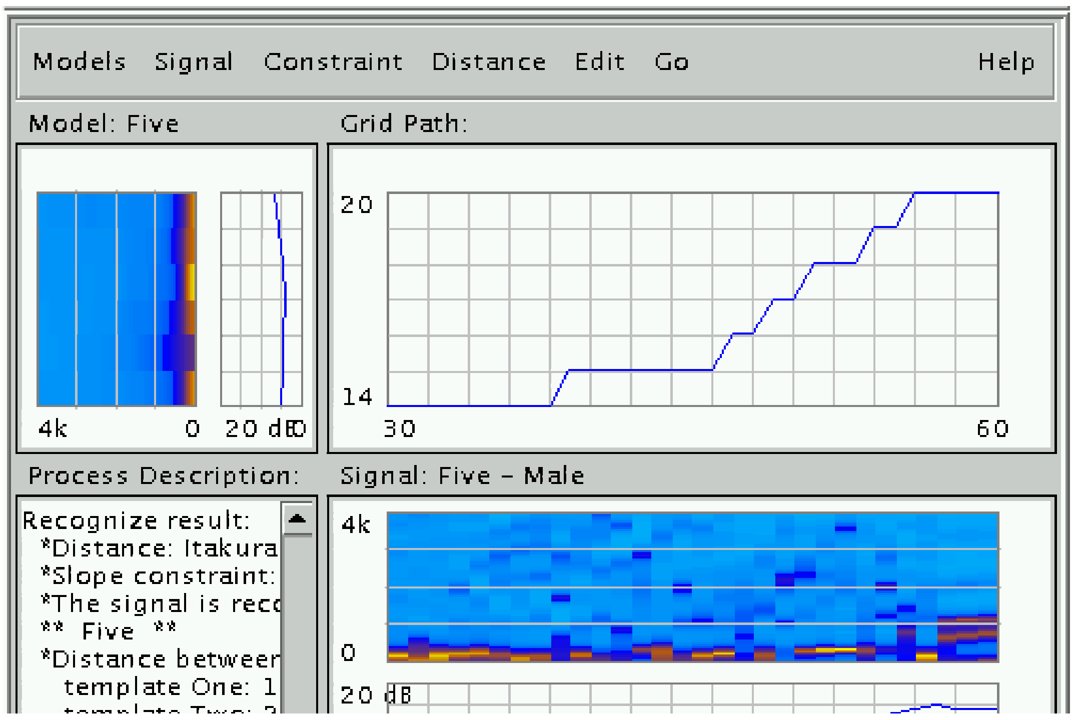
FEATURES OF EDIT MENU

After the matching process, you will see a nice grid path through the template-signal plane. If you want to have a closer look at some part of the path, or the spectrogram and energy plots of the signal, click on the **Edit** menu, and choose *Scaling*, and the following window will open on your screen:



After specifying desired minimum and maximum values of the x-axis, click "Update," and the display range of all display panels will change according to the value that was input. Clicking "Cancel" will return to the main screen.

If we set "XMinimum" as 30, and "XMaximum" as 60, the main screen will be updated like this:



Recognize result:
*Distance: Itakura
*Slope constraint:
*The signal is reco
** Five **
*Distance between
template One: 1