International Journal of Engineering Sciences & Emerging Technologies, Dec. 2013.
ISSN: 2231 – 6604 Volume 6, Issue 3, pp: 352-367 ©IJESET

ISOLATED WORD SPEECH RECOGNITION SYSTEM USING DYNAMIC TIME WARPING

Rajesh Makhijani[1], Ravindra Gupta[2]

[1]M. Tech Scholar, IT Dept, Sri Satya Sai Institute of Science & Technology, Sehore (M.P.), India
rajesh.mtit@gmail.com

[2]Assoc. Professor, IT Dept, Sri Satya Sai Institute of Science & Technology, Sehore (M.P.), India
ravindra_p84@rediffmail.com

ABSTRACT

This paper includes a new approach to develop a real time isolated word speech recognition system for human-computer interaction. The system is a speaker dependent system. The main motive behind developing this system is to recognize a list of words in which the speaker says through the microphone. The features used are the mel-frequency cepstral coefficients (MFCC) which gives the good discrimination of the speech signal. The Dynamic Programming algorithm is used in the system measures the similarity between the stored template and the test template for the speech recognition which gives the optimal distance. The recognition accuracy obtained for the system is 90.1%. We made a simple list of four words of numbers (count), i.e., One, Two, Three and Four and stored them under certain command names. After that when the particular word was spoken on microphone, the system recognized the word and displayed the respective command name in which it was stored. This system can be used in many areas after a little modification for the specific function, for e.g., to control a robot using simple commands and also in many other applications.

KEYWORDS: *Isolated Word, Speech Recognition, Dynamic Time Warping, Dynamic Programming, Euclidian Distance.*

I. INTRODUCTION

In speech recognition, the main goal of the feature extraction step is to compute a parsimonious sequence of feature vectors providing a compact representation of the given input signal. The feature extraction is usually performed in three stages. The first stage is called the speech analysis or the acoustic front end. It performs some kind of spectro-temporal analysis of the signal and generates raw features describing the envelope of the power spectrum of short speech intervals. The second stage compiles an extended feature vector composed of static and dynamic features. Finally, the last stage (which is not always present) transforms these extended feature vectors into more compact and robust vectors that are then supplied to the recognizer. Although there is no real consensus as to what the optimal feature sets should look like, one usually would like them to have the following properties: they should allow an automatic system to discriminate between different through similar sounding speech sounds, they should allow for the automatic creation of acoustic models for these sounds without the need for an excessive amount of training data, and they should exhibit statistics which are largely invariant cross speakers and speaking environment.

II. SIMPLE MODEL OF SPEECH PRODUCTION

As we have seen, the production of speech can be separated into two parts: Producing the excitation signal and forming the spectral shape. Thus, we can draw a simplified model of speech production [1]:

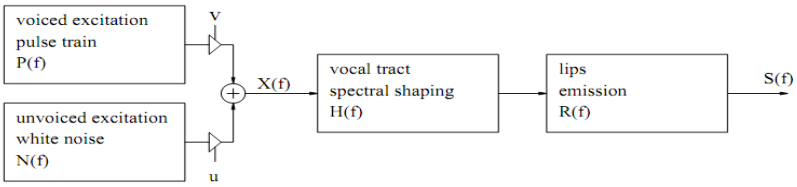

Figure 1. A simple model of speech production

This model works as follows: Voiced excitation is modeled by a pulse generator which generates a pulse train (of triangle–shaped pulses) with its spectrum given by *P(f)*. The unvoiced excitation is modeled by a white noise generator with spectrum *N(f)*. To mix voiced and unvoiced excitation, one can adjust the signal amplitude of the impulse generator *(v)* and the noise generator (u)[7].

2.1. Human speech coefficients

The human ear does not show a linear frequency resolution but builds several groups of frequencies and integrates the spectral energies within a given group. Furthermore, the mid-frequency and bandwidth of these groups are non–linearly distributed. The non–linear warping of the frequency axis can be modeled by the so-called mel-scale. The frequency groups are assumed to be linearly distributed along the mel-scale. The so–called mel–frequency f_{mel} can be computed from the frequency f as follows:

$$f_{mel}(f) = 2595 \cdot log\left(1 + \frac{f}{700\ Hz}\right) \tag{1}$$

The human ear has high frequency resolution in low–frequency parts of the spectrum and low frequency resolution in the high–frequency parts of the spectrum. The coefficients of the power spectrum $|V\ (n)\ |^2$ are now transformed to reflect the frequency resolution of the human ear.

2.2. Cepstrum-wise Transformation

The direct computation of the power spectrum from the speech signal results in a spectrum containing "ripples" caused by the excitation spectrum $X(f)$. Depending on the implementation of the acoustic preprocessing however, special transformations are used to separate the excitation spectrum $X(f)$ from the spectral shaping of the vocal tract $H(f)$. Thus, a smooth spectral shape (without the ripples), which represents $H(f)$ can be estimated from the

$$S(f)\ =\ X(f) \cdot H(f) \cdot R(f) = H(f) \cdot U(f) \tag{2}$$

We can now transform the product of the spectral functions to a sum by taking the logarithm on both sides of the equation:

$$\begin{aligned} \log\big(S(f)\big) &= \log(H(f) \cdot U(f)) \\ &= \log\big(H(f)\big) + \log(U(f)) \end{aligned} \tag{3}$$

This holds also for the absolute values of the power spectrum and also for their squares:

$$\begin{aligned} \log(|S(f)|^2) &= \log(|H(f)|^2 \cdot |U(f)|^2) \\ &= \log(|H(f)|^2) + \log(|U(f)|^2) \end{aligned} \tag{4}$$

In Fig. 1, we see an example of the log power spectrum, which contains unwanted ripples caused by the excitation signal $U(f) = X(f) \cdot R(f)$.

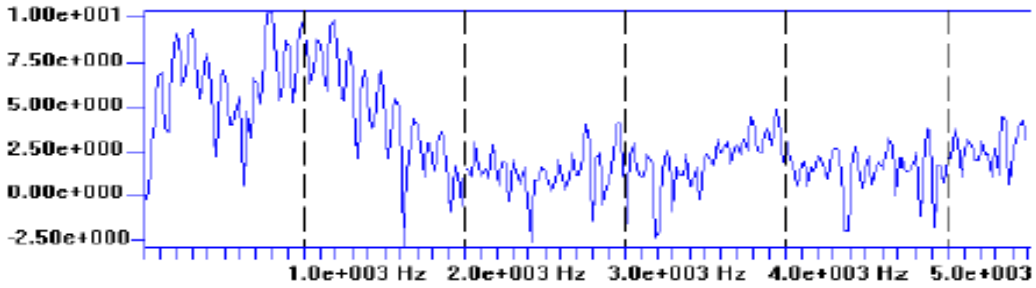

Figure 2. Log power spectrum of the vowel /a: / (f_s = 11 kHz, N = 512). The ripples in the spectrum are caused by X(f)

As we recall, it is necessary to compute the speech parameters in short time intervals to reflect the dynamic change of the speech signal. Typically, the spectral parameters of speech are estimated in time intervals of 10 ms. First, we have to sample and digitize the speech signal. Depending on the implementation, a sampling frequency $f_s$between 8 kHz and 16 kHz and usually a 16 bit quantization of the signal amplitude is used. After digitizing the analog speech signal, we get a series of speech samples $s(k \cdot \Delta t)$where $\Delta t = 1/f_s$ or, for easier notation, simply $s(k)$.

$$\hat{s}(d) = FT^{-1}\{\log(|S(f)|^2)\} = FT^{-1}\{\log(|H(f)|^2)\} + FT^{-1}\{\log(|U(f)|^2)\} \quad (5)$$

The inverse Fourier transform brings us back to the time–domain (d is also called the delay or frequency), giving the so–called *cepstrum* (a reversed "spectrum"). The resulting cepstrum is real–valued, since $|U(f)|^2$ and $|H(f)|^2$ are both real-valued and both are even: $|U(f)|^2 = |U(-f)|^2$ and $|H(f)|^2 = |H(-f)|^2$. Applying the inverse DFT to the log power spectrum coefficients $\log|V(n)|^2$ yields:

$$\hat{s}(d) = \frac{1}{N}\sum_{n=0}^{N-1} \log(|V(n)|^2) \cdot e^{\frac{j2\pi dn}{N}}; d = 0, 1, \ldots N-1 \quad (6)$$

Figure 3 shows the result of the inverse DFT applied on the log power spectrum shown in fig. 2.

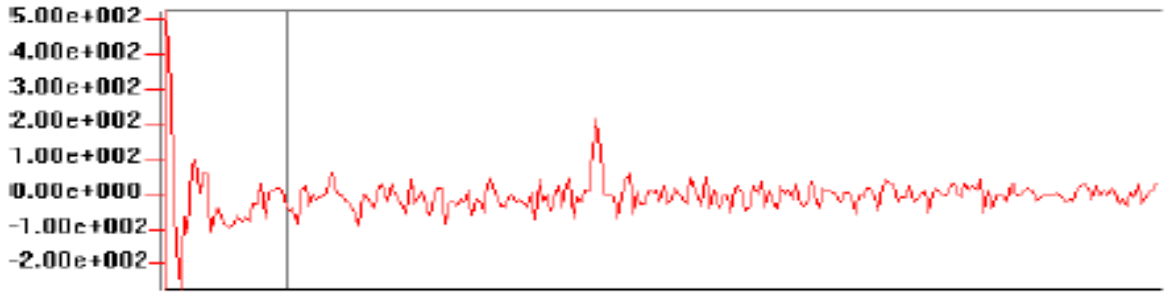

Figure 3. Cepstrum of the vowel /a: / ($f_s = 11\ kHz, N = 512$). The ripples in the spectrum result in a peak in the cepstrum

2.3. Mel Cepstrum

Now that we are familiar with the cepstral transformation and cepstral smoothing, we will compute the mel cepstrum commonly used in speech recognition. As stated above, for speech recognition, the mel spectrum is used to reflect the perception characteristics of the human ear. In analogy to computing the cepstrum, we now take the logarithm of the mel power spectrum (instead of the power spectrum itself) and transform it into the frequency domain to compute the so–called mel cepstrum[2]. Only the first Q (less than 14) coefficients of the mel cepstrum are used in typical speech

recognition systems. The restriction to the first Q coefficients reflects the low–pass filtering process as described above.
Since the mel power spectrum is symmetric due to (5), the Fourier-Transform can be replaced by a simple cosine transform:

$$c(q) = \sum_{k=0}^{K-1} \log\big(G(k)\big) \cdot \cos\left(\frac{\pi q(2k+1)}{2K}\right); q = 0, 1, \dots, Q-1 \qquad (7)$$

While successive coefficients $G(k)$ of the mel power spectrum are correlated, the Mel Frequency Cepstral Coefficients (MFCC) resulting from the cosine transform (7) are de-correlated.

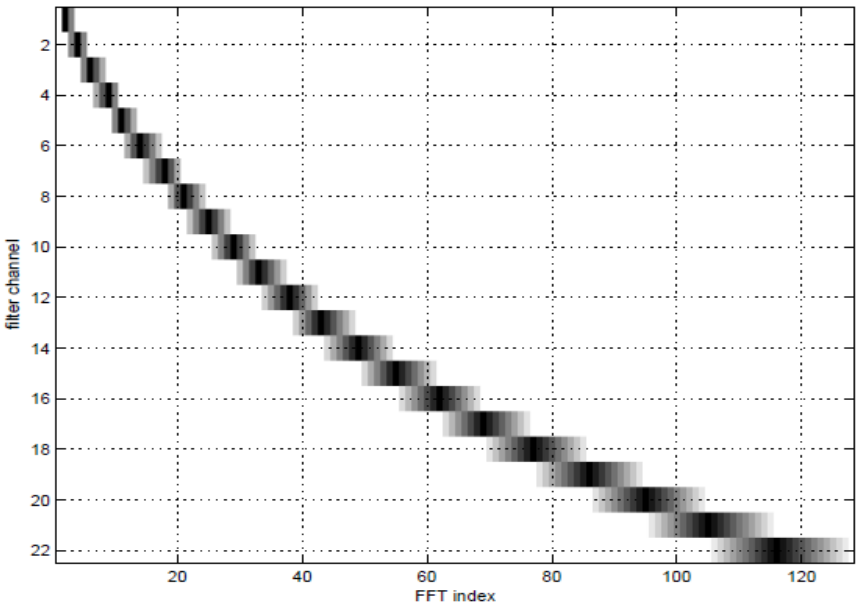

Figure 4. Mel cepstrum of an isolated word

III. FEATURE AND VECTOR SPACE

Until now, we have seen that the speech signal can be characterized by a set of parameters (features), which will be measured in short intervals of time during a preprocessing step. Before we start to look at the speech recognition task, we will first get familiar with the concept of feature vectors and vector space.
If you have a set of numbers representing certain features of an object you want to describe, it is useful for further processing to construct a vector out of these numbers by assigning each measured value to one component of the vector. As an example, think of an air conditioning system which will measure the temperature and relative humidity in your office. If you measure those parameters every second or so and you put the temperature into the first component and the humidity into the second component of a vector, you will get a series of two–dimensional vectors describing how the air in your office changes in time. Since these so–called feature vectors have two components, we can interpret the vectors as points in a two–dimensional vector space[3]. Thus we can draw a two–dimensional map of our measurements as sketched below. Each point in our map represents the temperature and humidity in our office at a given time. As we know, there are certain values of temperature and humidity which we find more comfortable than other values. In the map the comfortable value– pairs are shown as points labeled "+" and the less comfortable ones are shown as "-". You can see that they form regions of convenience and inconvenience, respectively.
Let's assume we would want to know if a value–pair we measured in our office would be judged as comfortable or as uncomfortable by you. One way to find out is to initially run a test series trying out

many value–pairs and labeling each points either "+" or "-" in order to draw a map as the one you saw above.

Now if you have measured a new value–pair and you are to judge if it will be convenient or not to a person, you would have to judge if it lies within those regions which are marked in your map as "+" or if it lies in those marked as "-". This is our first example of a classification task: We have two classes ("comfortable" and "uncomfortable") and a vector in feature space which has to be assigned to one of these classes. — But how do you describe the shape of the regions and how can you decide if a measured vector lies within or without a given region? In the following chapter we will learn how to represent the regions by prototypes and how to measure the distance of a point to a region.

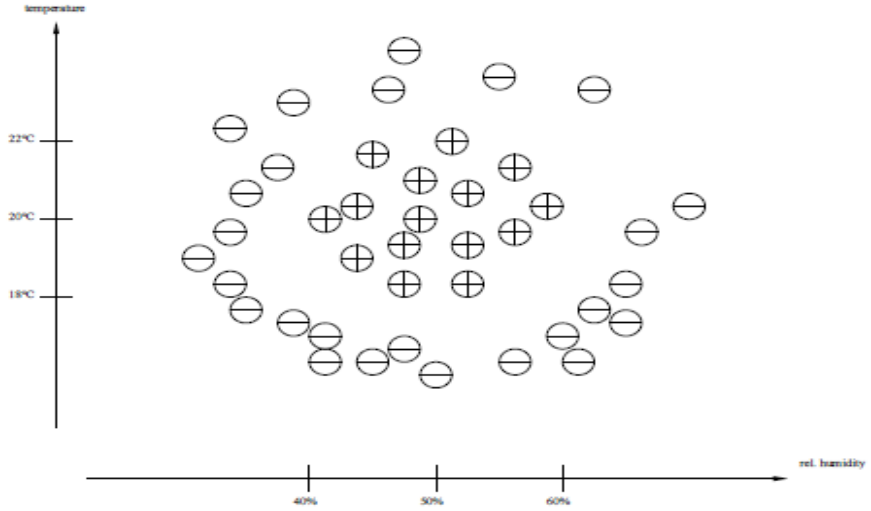

Figure 5. A map of feature vectors

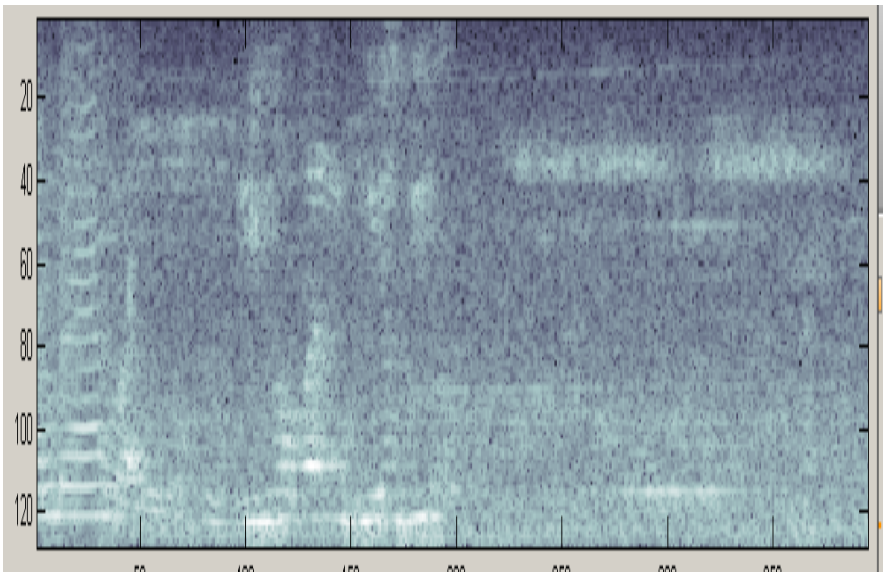

Figure 6. Log power spectrum of word "HELLO"

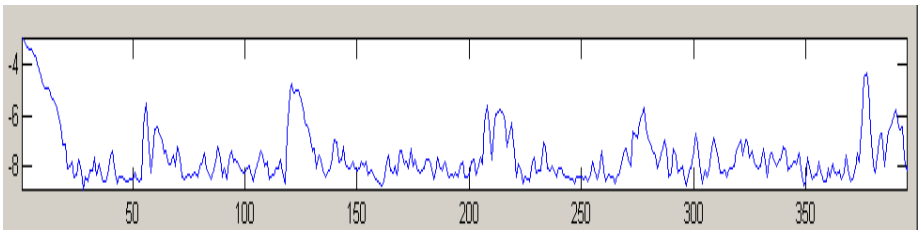

Figure 7. Log energy spectrum of word "HELLO"

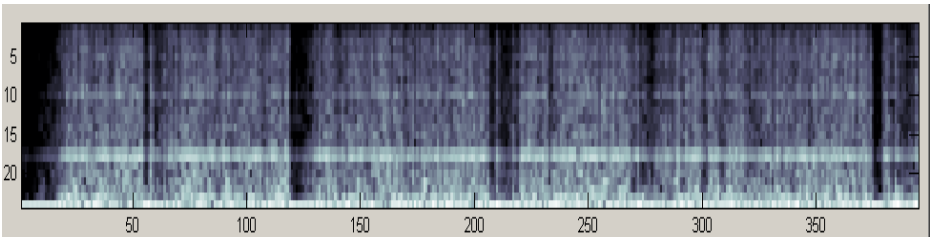

Figure 8. Log mel power spectrum of word "HELLO"

IV. DISTANCE MEASUREMENT

So far, we have found a way to classify an unknown vector by calculation of its class–distances to predefined classes, which in turn are defined by the distances to their individual prototype vectors. Now we will briefly look at some commonly used distance measures. Depending on the application at hand, each of the distance measures has its pros and cons, and we will discuss their most important properties.

4.1. Weighted Euclidean Distance

Both the Euclidean distance and the City Block distance are treating the individual dimensions of the feature space equally, i.e., the distances in each dimension contributes in the same way to the overall distance. But if we remember our example, we see that for real–world applications, the individual dimensions will have different scales also. While in our office the temperature values will have a range of typically between 18 and 22 degrees Celsius, the humidity will have a range from 40 to 60 percent relative humidity. While a small difference in humidity of e.g., 4 percent relative humidity might not even be noticed by a person, a temperature difference of 4 degrees Celsius certainly will. In Figure 9 we see a more abstract example involving two classes and two dimensions. The dimension x_1 has a wider range of values than dimension x_2, so all the measured values (or prototypes) are spread wider along the axis denoted as "x_1" as compared to axis "x_2". Obviously, a Euclidean or City Block distance measure would give the wrong result, classifying the unknown vector as "class A" instead of "class B" which would (probably) be the correct result.

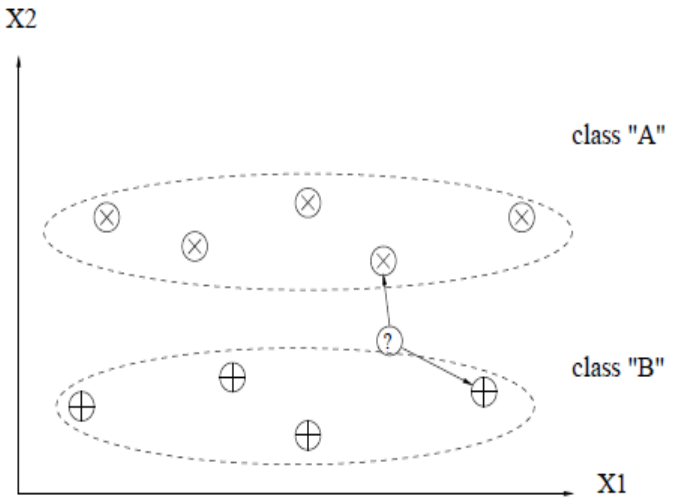

Figure 9. Two dimensions with different scales

To cope with this problem, the different scales of the dimensions of our feature vectors have to be compensated when computing the distance. This can be done by multiplying each contributing term

with a scaling factor specific for the respective dimension. This leads us to the so–called "Weighted Euclidean Distance".

$$d^2_{Weighted\_Euclid}(\vec{x},\vec{p}) = \sum_{i=0}^{DIM-1} \lambda_i \cdot (x_i - p_i)^2 \tag{8}$$

As before, this can be rewritten as:

$$d^2_{Weighted\_Euclid}(\vec{x},\vec{p}) = (\vec{x}-\vec{p})' \cdot \tilde{\Lambda} \cdot (\vec{x}-\vec{p}) \tag{9}$$

Where $\tilde{\Lambda}$ denotes the diagonal matrix of all the scaling factors:

$$\tilde{\Lambda} = \text{diag}\begin{bmatrix} \lambda_0 & & & & \\ & \cdots & & & \\ & & \lambda_i & & \\ & & & \cdots & \\ & & & & \lambda_{DIM-1} \end{bmatrix}$$

The scaling factors are usually chosen to compensate the variances of the measured features :

$$\lambda_i = \frac{1}{{\sigma_i}^2}$$

The variance of dimension i is computed from a training set of N vectors $\{\vec{x}_0, \vec{x}_1, \ldots, \vec{x}_{N-1}\}$. Let $x_{i,n}$ denote the i-th element of vector $\vec{x}_n$, then the variances ${\sigma_i}^2$ can be estimated from the training set as follows:

$${\sigma_i}^2 = \frac{1}{N-1}\sum_{n=0}^{N-1}\left(x_{i,n} - m_i\right)^2 \tag{10}$$

Where m_i is the mean value of the training set for dimension i:

$$m_i = \frac{1}{N}\sum_{n=0}^{N-1} x_{i,n}$$

4.2. Mahalanobis Distance

So far, we can deal with different scales of our features using the weighted Euclidean distance measure. This works very well if there is no correlation between the individual features as it would be the case if the features we selected for our vector space were statistically independent from each other. What if they are not? Figure 10 shows an example in which the features x_1 and x_2 are correlated.
Obviously, for both classes A and B, a high value of x_1 correlates with a high value for x_2 (with respect to the mean vector (center) of the class), which is indicated by the orientation of the two ellipses. In this case, we would want the distance measure to regard both the correlation and scale properties of the features. Here a simple scale transformation as in (8) will not be sufficient. Instead, the correlations between the individual components of the feature vector will have to be regarded when computing the distance between two vectors. This leads us to a new distance measure, the so–called Mahalanobis Distance:

$$d_{Mahalanobis}\left(\vec{x}, \overline{\vec{p}}, \tilde{C}\right) = (\vec{x}-\vec{p})' \cdot \tilde{C}^{-1} \cdot (\vec{x}-\vec{p}) \tag{11}$$

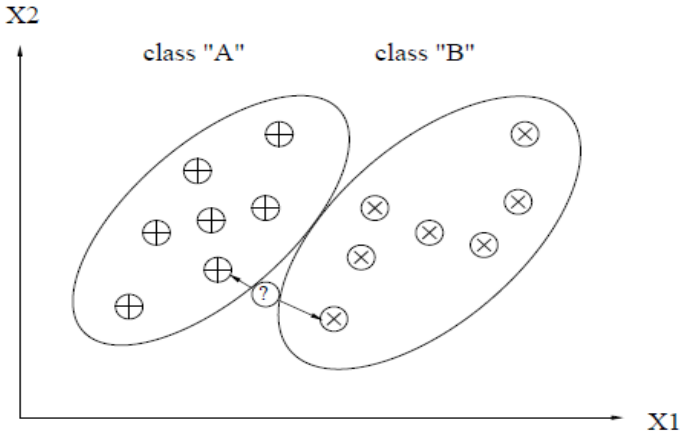

Figure 10. Correlated Features

Where $\tilde{C}^{-1}$ denotes the inverse of the covariance matrix $\tilde{C}$:

$$\tilde{C} = \begin{bmatrix} \sigma_{0,0} & \cdots & \sigma_{0,j} & \cdots & \sigma_{0,DIM-1} \\ \vdots & & \vdots & & \vdots \\ \sigma_{i,0} & \cdots & \sigma_{i,j} & \cdots & \sigma_{i,DIM-1} \\ \vdots & & \vdots & & \vdots \\ \sigma_{DIM-1,0} & \cdots & \sigma_{DIM-1,j} & \cdots & \sigma_{DIM-1,DIM-1} \end{bmatrix}$$

The individual components $\sigma_{i,j}$ represent the co-variances between the components i and j. The covariance matrix can be estimated from a training set of N vectors $\{\vec{x}_0, \vec{x}_1, \ldots, \vec{x}_{N-1}\}$ as follows:

$$\vec{C} = \frac{1}{N-1} \sum_{n=0}^{N-1} (\vec{x}_n - \vec{x}_m) \cdot (\vec{x}_n - \vec{x}_m)' \tag{12}$$

Where $\vec{m}$ is the mean vector of the training set:

$$\vec{m} = \frac{1}{N} \sum_{n=0}^{N-1} \vec{x}_n \tag{13}$$

Looking at the single element of the matrix C, say, $\sigma_{i,j}$, this can also be written as:

$$\sigma_{i,j} = \frac{1}{N-1} \sum_{n=0}^{N-1} (x_{i,n} - m_i) \cdot (x_{j,n} - m_j) \tag{14}$$

From (10) and (14) it is obvious that $\sigma_{i,i}$, the values of the main diagonal of $\tilde{C}$, are identical to the variances ${\sigma_i}^2$ of the dimension i. Two properties of $\tilde{C}$ can be seen immediately from (14): First, $\tilde{C}$ is symmetric and second, the values $\sigma_{i,i}$ of its main diagonal are either positive or zero. A zero variance would mean that for this specific dimension of the vector we chose a feature which does not change its value at all).

4.2.1. Clusters With Individual and Shared Covariance Matrices

In our Fig. 10, the two classes A and B each consisted of one cluster of vectors having its own center (mean vector that is) and its own covariance matrix. But in general, each class of a classification task may consist of several clusters which belong to that class, if the "shape" of the class in our feature

space is more complex. As an example, think of the shape of the "uncomfortable" class of Fig. 5. In these cases, each of those clusters will have its own mean vector (the "prototype vector" representing the cluster) and its own covariance matrix. However, in some applications, it is sufficient (or necessary, due to a lack of training data) to estimate one covariance matrix out of all training vectors of one class, regardless to which cluster of the class the vectors are assigned. The resulting covariance matrix is then class specific but is shared among all clusters of the class while the mean vectors are estimated individually for the clusters. To go even further, it is also possible to estimate a single "global" covariance matrix which is shared among all classes (and their clusters). This covariance matrix is computed out of the complete training set of all vectors regardless of the cluster and class assignments.

4.2.2. Using the Mahalanobis Distance for the Classification Task

With the Mahalanobis distance, we have found a distance measure capable of dealing with different scales and correlations between the dimensions of our feature space. It plays a very significant role when dealing with the Gaussian probability density function.
To use the Mahalanobis distance in our nearest neighbor classification task, we would not only have to find the appropriate prototype vectors for the classes (that is, finding the mean vectors of the clusters), but for each cluster we also have to compute the covariance matrix from the set of vectors which belong to the cluster. We will later learn how the "k–means algorithm" helps us not only in finding prototypes from a given training set, but also in finding the vectors that belong to the clusters represented by these prototypes. Once the prototype vectors and the corresponding (either shared or individual) covariance matrices are computed, we can insert the Mahalanobis distance according to (11) into the equation for the class distance (12) and get:

$$d_{\omega_v}(\vec{x}) = \min_k \left\{ d_{Mahalanobis}\left(\vec{x}, \vec{p}_{k,\omega_v}, \tilde{C}_{k,\omega_v}\right) \right\}; k = 0, 1, \dots, \left(K_{\omega_v} - 1\right) \quad (15)$$

Then we can perform the Nearest Neighbor classification

V. DYNAMIC TIME WARPING

In the last section, we were dealing with the task of classifying single vectors to a given set of classes which were represented by prototype vectors computed from a set of training vectors. Several distance measures were presented, some of them using additional sets of parameters (e.g., the covariance matrices) which also had to be computed from training vectors. How does this relate to speech recognition? As we saw in section 2, our speech signal is represented by a series of feature vectors which are computed every 10ms. A whole word will comprise dozens of those vectors, and we know that the number of vectors (the duration) of a word will depend on how fast a person is speaking. Therefore, our classification task is different from what we have learned before. In speech recognition, we have to classify not only single vectors, but sequences of vectors. Let's assume we would want to recognize a few command words or digits. For an utterance of a word w which is T_X vectors long, we will get a sequence of vectors $\tilde{X} = \{\vec{x}_0, \vec{x}_1, \dots, \vec{x}_{X-1}\}$ from the acoustic preprocessing stage. What we need here is a way to compute a "distance" between this unknown sequence of vectors $\tilde{X}$ and known sequences of vectors $\widetilde{W}_k = \left\{\vec{w}_{k0,}\vec{w}_{k1,} \dots, \vec{w}_{kT_{W_k}}\right\}$ which are prototypes for the words we want to recognize. Let our vocabulary (here: the set of classes Ω) contain V different words $w_0, w_1, \dots w_{V-1}$. In analogy to the Nearest Neighbor classification task, we will allow a word w_v (here: class$w_v \in \Omega$) to be represented by a set of prototypes $\widetilde{W}_{k,\omega_v}, k = 0, 1, \dots, \left(K_{\omega_v} - 1\right)$ to reflect all the variations possible due to different pronunciation or even different speakers.

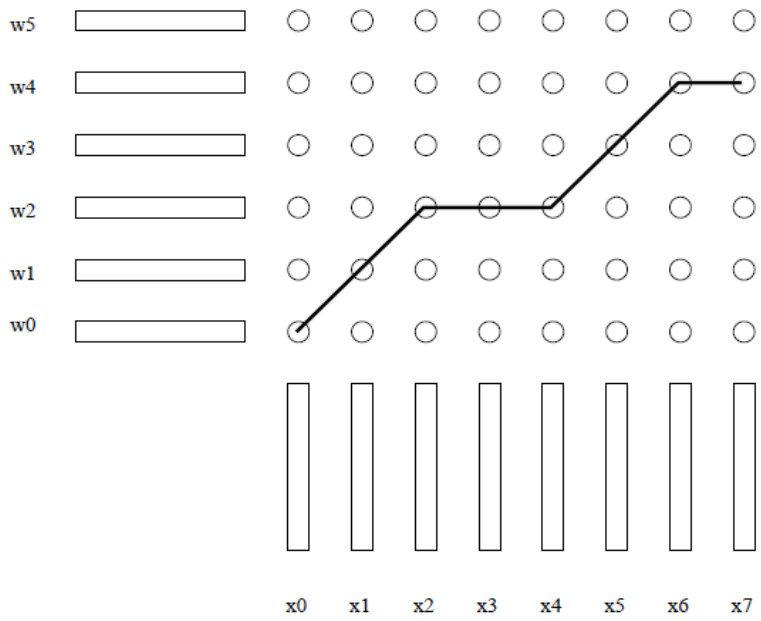

Figure 11. Possible assignment between vector pairs of X and W

5.1. The Dynamic Programming Algorithm

Definition of the Classification Task

If we have a suitable distance measure $D(\tilde{X}, \widetilde{W}_{k,\omega_v})$ between $\tilde{X}$ and a prototype $\widetilde{W}_{k,\omega_v}$, we can define the class distance $D_{\omega_v}(\tilde{X})$ as follows:

$$D_{\omega_v}(\tilde{X}) = \min_k \{D(\tilde{X}, \widetilde{W}_{k,\omega_v})\}\ ; k = 0, 1, \dots, (K_{\omega_v} - 1) \tag{16}$$

Where $\omega_v$the class of utterances is whose orthographic representation is given by the word w_v.

Then we can write our classification task as:

$$\tilde{X} \in \omega_v \Leftrightarrow v = \arg \min_v \{D_{\omega_v}(\tilde{X})\} \tag{17}$$

This definition implies that the vocabulary of our speech recognizer is limited to the set of classes $\Omega = \{\omega_0, \omega_1, \dots, \omega_{(V-1)}\}$. In other words, we can only recognize V words, and our classifier will match any speech utterance to one of those V words in the vocabulary, even if the utterance was intended to mean completely different. The only criterium the classifier applies for its choice is the acoustic similarity between the utterance $\tilde{X}$ and the known prototypes $\widetilde{W}_{k,\omega_v}$ as defined by the distance measure $D(\tilde{X}, \widetilde{W}_{k,\omega_v})$.

Distance between Two Sequences of Vectors

As we saw before, classification of a spoken utterance would be easy if we had a good distance measure $D(\tilde{X}, \widetilde{W})$ at hand (in the following, we will skip the additional indices for ease of notation). What should the properties of this distance measure be? The distance measure we need must:

- Measure the distance between two sequences of vectors of different length ($T_X$and T_W, that is)
- While computing the distance, find an optimal assignment between the individual feature vectors of $\tilde{X}$ and $\widetilde{W}$

- Compute a total distance out of the sum of distances between individual pairs of feature vectors of $\tilde{X}$ and $\widetilde{W}$.

Comparing Sequences with Different Lengths

The main problem is to find the optimal assignment between the individual vectors of $\tilde{X}$and $\widetilde{W}$. In Fig. 12, we can see two sequences $\tilde{X}$ and $\widetilde{W}$ which consist of six and eight vectors, respectively. The sequence $\widetilde{W}$ was rotated by 90 degrees, so the time index for this sequence runs from the bottom of the sequence to its top. The two sequences span a grid of possible assignments between the vectors. Each path through this grid (as the path shown in the figure) represents one possible assignment of the vector pairs. For example, the first vector of $\tilde{X}$ is assigned the first vector of $\widetilde{W}$, the second vector of $\tilde{X}$ is assigned to the second vector of $\widetilde{W}$, and so on.

Fig. 12 shows as an example the following path P given by the sequence of time index pairs of the vector sequences (or the grid point indices, respectively):

$$P = \left\{g_1, g_2, \dots, g_{L_p}\right\} = \{(0,0), (1,1), (2,2), (3,2), (4,2), (5,3), (6,4), (7,4)\} \qquad (18)$$

The length L_P of path P is determined by the maximum of the number of vectors contained in $\tilde{X}$ and $\widetilde{W}$. The assignment between the time indices of $\widetilde{W}$ and $\tilde{X}$ as given by P can be interpreted as "*time warping*" between the time axes of $\widetilde{W}$and $\tilde{X}$. In our example, the vectors $\vec{x}_2$, $\vec{x}_3$ and $\vec{x}_4$were all assigned to $\vec{w}_2$, thus warping the duration of $\vec{w}_2$ so that it lasts three time indices instead of one. By this kind of time warping, the different lengths of the vector sequences can be compensated.

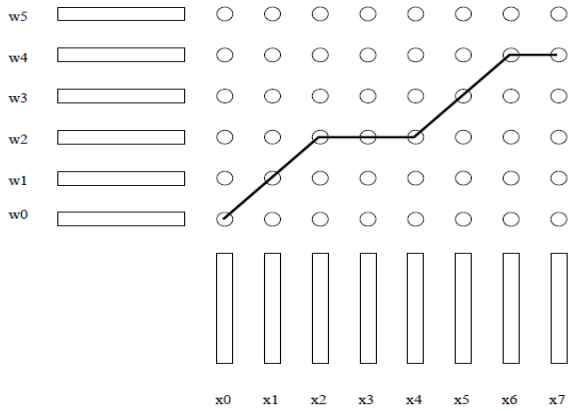

Figure 12. Possible assignment between the vector pairs of $\tilde{X}$ and $\widetilde{W}$

For the given path P, the distance measure between the vector sequences can now be computed as the sum of the distances between the individual vectors. Let l denote the sequence index of the grid points. Let $d(g_l)$ denote the vector distance $d\ (\vec{x}_2, \vec{w}_j)$ for the time indices i and j defined by the grid point $g_l = (i, j)$ (the distance $d\ (\vec{x}_2, \vec{w}_j)$ could be the Euclidean distance from section 4.1). Then the overall distance can be computed as:

$$D\left(\tilde{X}, \widetilde{W}, P\right) = \sum_{l=1}^{L_P} d\ (g_l) \qquad (19)$$

Finding the Optimal Path

As we stated above, once we have the path, computing the distance D is a simple task. How do we find it? The criterion of optimality we want to use in searching the optimal path P_{opt} should be to minimize $D\left(\tilde{X}, \widetilde{W}, P\right)$:

$$P_{opt} = \arg\min_{P}\{D(\tilde{X}, \widetilde{W}, P)\} \quad (20)$$

Fortunately, it is not necessary to compute all possible paths P and corresponding distances $D(\tilde{X}, \widetilde{W}, P)$to find the optimum.

Out of the huge number of theoretically possible paths, only a fraction is reasonable for our purposes. We know that both sequences of vectors represent feature vectors measured in short time intervals. Therefore, we might want to restrict the time warping to reasonable boundaries: The first vectors of $\tilde{X}$ and $\widetilde{W}$ should be assigned to each other as well as their last vectors. For the time indices in between, we want to avoid any giant leap backward or forward in time, but want to restrict the time warping just to the "reuse" of the preceding vector(s) to locally warp the duration of a short segment of speech signal. With these restrictions, we can draw a diagram of possible "local" path alternatives for one grid point and its possible predecessors (of course, many other local path diagrams are possible):

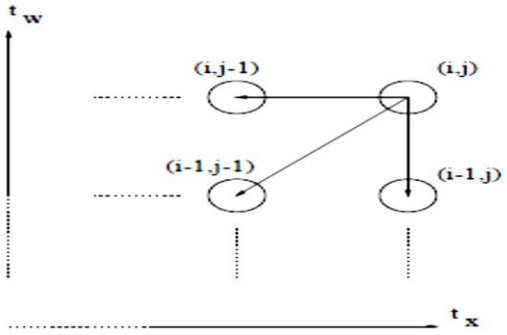

Figure 13. Local path alternatives for a grid point

Note that Fig. 13 does not show the possible extensions of the path from a given point but the possible *predecessor paths* for a given grid point. We will soon get more familiar with this way of thinking. As we can see, a grid point (i, j) can have the following predecessors:

- $(\mathrm{i}-1, \mathrm{j}0)$: keep the time index j of $\widetilde{\mathrm{X}}$ while the time index of $\widetilde{\mathrm{W}}$ is incremented.
- $(\mathrm{i}-1, \mathrm{j}-1)$: both time indices of $\widetilde{\mathrm{X}}$ and $\widetilde{\mathrm{W}}$ are incremented.
- $(\mathrm{i}, \mathrm{j}-1)$: keep of the time index i of $\widetilde{\mathrm{W}}$ while the time index of $\widetilde{\mathrm{X}}$ is incremented.

All possible paths *P* which we will consider as possible candidates for being the optimal path P_{opt} can be constructed as a concatenation of the local path alternatives as described above. To reach a given grid point (i, j) from $(i-1, j-1)$, the diagonal transition involves only the single vector distance at grid point (i, j) as opposed to using the vertical or horizontal transition, where also the distances for the grid points $(i-1, j)$ or $(i, j-1)$ would have to be added. To compensate this effect, the local distance $d(\vec{w}_i, \vec{x}_j)$ is added twice when using the diagonal transition.

Bellman's Principle

Now that we have defined the local path alternatives, we will use Bellman's Principle to search the optimal path P_{opt}. Applied to our problem, Bellman's Principle states the following:

If P_{opt} is the optimal path through the matrix of grid points beginning at $(\mathbf{0}, \mathbf{0})$ and ending at (T_{W-1}, T_{X-1}), and the grid point (i, j) is part of path P_{opt}, then the partial path from $(\mathbf{0}, \mathbf{0})$ to (i, j) is also part of P_{opt}.

From that, we can construct a way of iteratively finding our optimal path P_{opt}: According to the local path alternatives diagram we chose, there are only three possible predecessor paths leading to a grid point (i, j): The partial paths from $(0, 0)$ to the grid points $(i-1, j), (i-1, j-1)\ and\ (i, j-1)$).

Let's assume we would know the optimal paths (and therefore the accumulated distance $\delta(.)$ along that paths) leading from (0, 0) to these grid points. All these path hypotheses are possible predecessor paths for the optimal path leading from $(0, 0)$to (i, j).

Then we can find the (globally) optimal path from $(0, 0)$ to grid point (i, j) by selecting exactly the one path hypothesis among our alternatives which minimizes the accumulated distance $\delta(i, j)$ of the resulting path from $(0, 0)$to (i, j).

The optimization we have to perform is as follows:

$$\delta(i,j) = \min \begin{cases} \delta(i, j-1) + d(\vec{w}_i, \vec{x}_j) \\ \delta(i-1, j-1) + 2 \cdot d(\vec{w}_i, \vec{x}_j) \\ \delta(i-1, j) + d(\vec{w}_i, \vec{x}_j) \end{cases} \quad (21)$$

By this optimization, it is ensured that we reach the grid point (i, j) via the optimal path beginning from $(0, 0)$ and that therefore the accumulated distance $\delta(i, j)$ is the minimum among all possible paths from $(0, 0)$to (i, j).

Since the decision for the best predecessor path hypothesis reduces the number of paths leading to grid point (i, j) to exactly one, it is also said that the possible path hypotheses are recombined during the optimization step.

Applying this rule, we have found a way to iteratively compute the optimal path from point $(0, 0)$ to point (T_{W-1}, T_{X-1}), starting with point $(0, 0)$:

- ***Initialization:*** For the grid point $(0, 0)$, the computation of the optimal path is simple: It contains only grid point $(0, 0)$ and the accumulated distance along that path is simply $\mathrm{d}(\vec{\mathrm{w}}_0, \vec{\mathrm{x}}_0)$.
- ***Iteration:*** Now we have to expand the computation of the partial paths to all grid points until we reach $(\mathrm{T}_{\mathrm{W}-1}, \mathrm{T}_{\mathrm{X}-1})$: According to the local path alternatives, we can now only compute two points from grid point $(0, 0)$: The upper point $(1, 0)$, and the right point $(0, 1)$. Optimization of (3.46) is easy in these cases, since there is only one valid predecessor: The start point (0.0). So we add $\delta(0, 0)$ to the vector distances defined by the grid points $(1, 0)$ and $(0, 1)$ to compute $\delta(1, 0)$and $\delta(0, 1)$. Now we look at the points which can be computed from the three points we just finished. For each of these points (i, j), we search the optimal predecessor point out of the set of possible predecessors (Of course, for the leftmost column $(\mathrm{i}, 0)$ and the bottom row $(0, \mathrm{j})$ the recombination of path hypotheses is always trivial). That way we walk trough the matrix from bottom-left to top-right.
- ***Termination:*** $\mathrm{D}(\widetilde{\mathrm{W}}, \widetilde{\mathrm{X}}) = \delta(\mathrm{T}_{\mathrm{W}-1}, \mathrm{T}_{\mathrm{X}-1})$ is the distance between $\widetilde{\mathrm{W}}$ and $\widetilde{\mathrm{X}}$.

The optimal path is known only after the termination of the algorithm, when we have made the last recombination for the three possible path hypotheses leading to the top– right grid point (T_{W-1}, T_{X-1}). Once this decision is made, the optimal path can be found by reversely following all the local decisions down to the origin $(0, 0)$. This procedure is called *backtracking*.

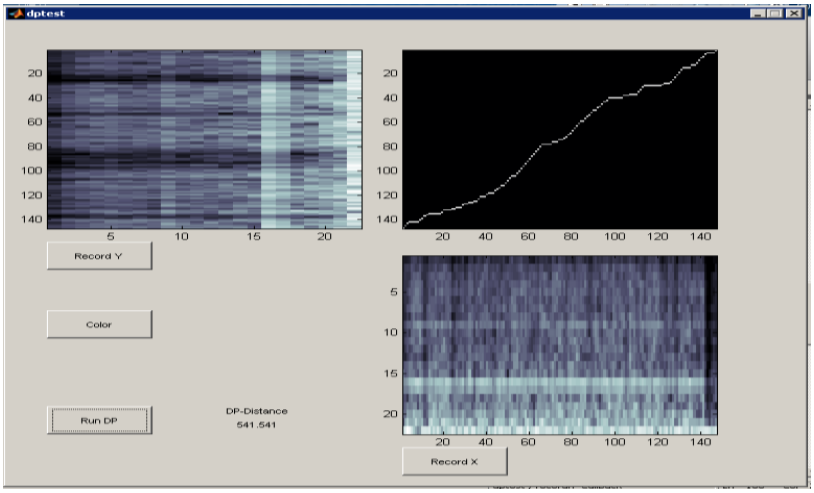

Figure 14. Constructed graphical user interface for calculating distance between two words

Now all we have to do is to run the DTW algorithm for each time index j and along all columns of all prototype sequences. At the last time slot $(T_W - 1)$ we perform the optimization step for the virtual grid point, i.e., the predecessor grid point to the virtual grid point is chosen to be the prototype word having the smallest accumulated distance. Note that the search space we have to consider is spanned by the length of the unknown vector sequence on one hand and the sum of the length of all prototype sequences of all classes on the other hand. Figure 14 shows the constructed graphical user interface for calculating distance between two words. The backtracking procedure can of course be restricted to keeping track of the final optimization step when the best predecessor for the virtual grid point is chosen. The classification task is then performed by assigning the unknown vector sequence to the very class to which the prototype belongs to whose word end grid point was chosen.

Of course, this is just a different (and quite complicated) definition of how we can perform the DTW classification task we already defined in (3.42). Therefore, only a verbal description was given and we did not bother with a formal description. However, by the reformulation of the DTW classification we learned a few things:

- The DTW algorithm can be used for real–time computation of the distances
- The classification task has been integrated into the search for the optimal path
- Instead of the accumulated distance, now the optimal path itself is important for the classification task

VI. RESULTS

To verify the approach of dynamic programming, several graphical user interface programs were created in MATLAB. Several snapshots of them were shown earlier, the final program consists of a graphical user interface in which the user will input four words and the program will then detect the word spoken by the user and display it on the screen. Snapshot of these are shown as below for reference:-
In the phase of execution four words were given as input and were stored in the system in the GUI. Those words were One, Two Three and Four respectively, These words were stored in separate commands, Then when the words are pronounced again, using the “Record” functionality, the system recognizes the word uttered and shows the command Number in which it was stored after matching it.

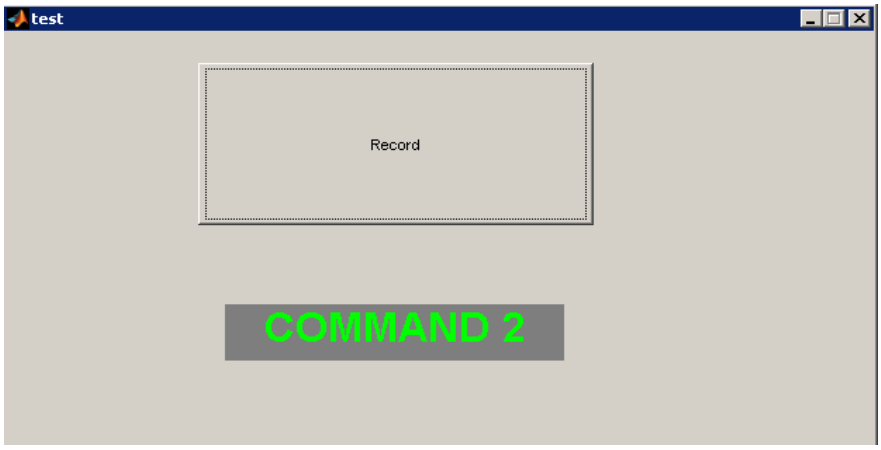

Figure 15. Output of program showing the utterance of word "TWO"

Figure 15 above shows the output generated on the GUI when the word "TWO" was uttered in the form of command number since it was stored at "COMMAND 2" earlier.

Figure 16. Output of program showing the utterance of word "THREE"

As stated earlier, Figure 16 here also shows the output generated on the GUI when the word "THREE" was uttered in the form of command number since it was stored at "COMMAND 3" earlier.

VII. CONCLUSION

Speech is the primary, and the most convenient means ofcommunication between people. Whether due to technological curiosity to build machines that mimic humans or desire to automate work with machines, research in speech and speaker recognition, as a first step toward natural human-machine communication, has attracted much enthusiasm over the past five decades. We have also encountered a number of practical limitations which hinder a widespread deployment of application and services. In most speech recognition tasks, human subjects produce one to two orders of magnitude less errors than machines. There is now increasing interest in finding ways to bridge such a performance gap. What we know about human speech processing is very limited. Although these areas of investigations are important the significant advances will come from studies in acoustic-phonetics, speech perception, linguistics, and psychoacoustics. Future systems need to have an efficient way of representing, storing, and retrieving knowledge required for natural conversation. This paper attempts

to provide a comprehensive survey of research on speech recognition and to provide someyear wise progress to this date. Although significant progress has been made in the last two decades, there is still work to bedone, and we believe that a robust speech recognition system should be effective under full variation in: environmental conditions, speaker variability s etc. Speech Recognition is a challenging and interesting problem in and of itself. We have attempted in this paper to provide a comprehensive cursory, look and review of how much speech recognition technology progressed in the last 60 years. Speech recognition is one of the most integrating areas of machine intelligence, since; humans do a daily activity of speech recognition. Speech recognition has attracted scientists as an important discipline and has created a technological impact on society and is expected to flourish further in this area of human machine interaction. We hope this paper brings about understanding and inspiration amongst the research communities of ASR.

REFERENCES

[1] Palden Lama and Mounika Namburu, "Speech Recognition with Dynamic Time Warping using MATLAB", *Project Report, CS 525, Spring 2010.*

[2] S.A.R Al-Haddad, S.A. Samad, A. Hussain, K.A. Ishak and A.O.A. Noor, "Robust Speech Recognition Using Fusion Techniques and Adaptive Filtering" *American Journal of Applied Sciences 6 (2): 290-295, 2009.*

[3] H. F. Olson and H. Belar, Phonetic Typewriter, *J. Acoust. Soc. Am., 28(6): 1072-1081, 1956*.

[4] Fadhilah Rosdi, Raja N Ainon, " Isolated Malay Speech Recognition Using Hidden Markov Models", *Proceedings of International Conference on Computer and Communication Engineering, Kuala Lumpur, Malaysia, 2008, May 13-15.*

[5] M.A. Anusuya, S.K. Katti, "Speech Recognition by Machine: A Review", *International Journal of Computer Science and Information Security (IJCSIS), Vol. 6, No. 3, 2009.*

[6] Siva Prasad Nandyala, Dr. T. Kishore Kumar, "Real Time Isolated Word Speech Recognition System for Human Computer Interaction", *International Journal of Computer Applications (IJCA), Vol. 12, No. 2, 2010.*

[7] B. Plannerer, An introduction to Speech Recognition, March 28, 2005.

[8] H. Ney. The use of a one-stage dynamic programming algorithm for connected word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing, ASSP-32(2):263–271, April 1984*

[9] S. Young. "Statistical modelling in continuous speech recognition",*Proc. Int. Conference on Uncertainity in Artificial Intelligence, Seattle, WA, August 2001*.

[10] M. Muller, "Dynamic Time Warping", *Information Retrieval for Music and Motion, Springer, 2007.*

[11] T. Bin Amin, I. Mahmood, "Speech Recognition using Dynamic Time Warping", *Conf. Proc. Advances in Space Technologies, 2008.*

[12] Lawrence Rabiner, Biing-Hwang Juang, Fundamentals of Speech Recognition, *Pearson Education, New Delhi, 2006.*