Abstract—Currently, speech recognition systems with different levels of complexities are being researched on, with isolated speech recognition being the most basic level of them all. Work presented in this paper is an attempt to build this basic level of speech recognizer. The system presented is a speaker independent isolated speech recognizer with a small vocabulary. The main idea is to give an overall idea of the components of a speech recognition system and the theory behind. The system described has mainly four parts: Edge detection, feature extraction, constant trajectory mapping and neural network recognizer. A simple algorithm is introduced for word boundary detection. Feature extraction is being done using an LPC processor with enhanced features. Apart from that, a Self Organizing Map (SOM) is used to reduce the dimensionality of the LPC output and to get a feature vector of constant size to be fed to the Artificial Neural Network (ANN).

In an analytical point of view, speech consists of convolution in the time domain of two waves generated by formant structure of the vocal tract and excitation of vocal tract called the pitch of the sound [3]. In the case of recognizing the word uttered we need to focus on the shape of Vocal Tract which is unique for the each and every word uttered.

II. PROCESSING OF SPEECH SIGNALS
The greatest common denominator of all speech recognition systems is the signal processing front end, or the feature extractor. It converts the speech waveform to some type of parametric representation which consists of a lower information rate.

A wide range of possibilities exist for parametrically representing the speech signal for the speaker recognition task, such as Linear Prediction Coding (LPC), Mel-Frequency Cepstrum Coefficients (MFCC), and others. The feature extractor presented in this paper is based on the LPC model with enhanced features. LPC provides a good model of the speech signal. This is especially true for the quasi steady state voiced regions of speech in which the all-pole model of LPC provides a good approximation to the vocal tract envelope [1 pg98]. The way in which LPC is applied to the analysis of speech signals leads to a reasonable source-vocal tract representation.

III. ARTIFICIAL NEURAL NETWORKS FOR SPEECH RECOGNITION
An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information [5]. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements.
(neurons) working in unison to solve specific problems. The following Fig. depicts a simple neuron.

![Fig. 1. An Artificial Neuron]

There are N inputs, \( x_1, x_2, \ldots, x_n \), which are summed with weights of the dendrites \( w_1, w_2, \ldots, w_n \), thresholded, and then nonlinarily compressed to give the output \( y \), defined as

\[
y = f \left( \sum_{i=1}^{n} w_i x_i - \phi \right)
\]

\( \Phi \) is an internal threshold or offset, and \( f \) is a nonlinearity either of the types, hard limiter or sigmoid.

**A. Multilayer Perceptron (MLP)**

MLP is the most common topology type of ANN. In an MLP, the units each perform a biased weighted sum of their inputs and pass this activation level through a transfer function to produce their output, and the units are arranged in a layered feedforward topology [8]. The network thus has a simple interpretation as a form of input-output model, with the weights and thresholds being the free parameters of the model. Such networks can model functions of almost arbitrary complexity, with the number of layers, and the number of units in each layer, determining the function complexity. Here learning algorithm is back propagation.

**B. Self Organizing Maps**

The SOM is an unsupervised non-parametric regression process to represent nonlinear, high dimensional data on a low-dimensional illustrative display. The input data points are mapped to SOM units on a usually one or two-dimensional grid [6]. The mapping is learned from the training data samples by a simple stochastic learning process, where the SOM units are adjusted by small steps with respect to the feature vectors that are extracted from the data and presented one after another.

**C. Neural Networks in Speech Recognition**

A lot of research has been carried out to combine ANN computing with conventional algorithms- in particular, Dynamic Time Warping, Hidden Markov Models and Viterbi Search. The ANN contribution to these techniques is principally to serve as an alternative computing structure for carrying out the necessary mathematical operations [2 pg841]. The ANN strategy can also enhance the distance or likelihood computing task by incorporating context or by learning which features are most effective. ANNs can also be added to refine the recognition scores and help improve performance. The underlying techniques, however, remain unchanged.

**IV. SYSTEM IMPLEMENTATION**

**A. Speech Recording**

Selected words were recorded from different speakers. The sound recorder utility that comes with the Windows XP was used for this purpose. The sampling rate was selected to be 16kHz, which was sufficient enough to represent the analog speech signal. One sample was encoded using16 bits, the number of channels was 1 (mono), and the audio format is pulse code modulation (PCM).

**B. Edge Detection**

A simple algorithm based on amplitude detection is used here. A word is considered as started if the amplitude crosses over a pre-defined threshold value. Until the amplitude of the speech signal remains over this threshold value, the signal is considered to be in the voiced region. When the signal amplitude stays below the threshold for a predefined time, the end of the signal is detected. This value is selected in such a way that it would not mistakenly cut off the speech signal in an intermediate point.

**C. Feature Extraction**

Linear Predictive Coding (LPC) Model has been used here. The basic idea behind the LPC model is that a given speech sample at time \( n \), \( s(n) \), can be approximated as a linear combination of the past speech samples, such that

\[
S(n) = a_1 s(n-1) + a_2 s(n-2) + \ldots + a_p s(n-p)
\]

Where coefficients \( a_1, a_2, \ldots, a_p \) are assumed to be constant over the speech analysis frame.
1. Pre emphasis
The digitized speech signal \( s(n) \) is put through a first order FIR filter. This would spectrally flatten the signal and make it less susceptible to finite precision effects later in the signal processing.

The transfer function of this fixed first-order system is

\[
H(w) = 1 - ae^{-jw}, \quad 0.9 < a < 1.0 \tag{1}
\]

The most common value for \( a \) is 0.95 and this value was used in our filter. The filter was implemented in MATLAB and the corresponding coefficients were obtained. These were multiplied with the time domain waveform.

2. Frame blocking
In this step, the pre emphasized speech signal is blocked into frames of \( N \) samples, with adjacent frames being separated by \( M \) samples. Here, \( N = 330 \) and \( M = (1/3)N = 110 \).

Each frame begins \( M \) samples after their immediate successor, and overlaps it by \( N - M \) samples. This process continues until all the speech is accounted for within one or more frames. Since \( M \leq N \), the adjacent frames overlap and the resulting LPC spectral estimates would be correlated from frame to frame. Since \( M \) is considerably smaller than \( N \), the LPC spectral estimates from frame to frame would be quite smooth.

3. Windowing
The next step is to window each individual frame so as to minimize the signal discontinuities at the beginning and end of each frame. The technique here is to use a window to taper the signal to zero at the beginning and end of each frame. A typical window used for the autocorrelation method of LPC (and the method that is being most widely used for speech recognition systems) is the hamming window. Our system also has incorporated this type of window.

\[
w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \quad 0 \leq n \leq N-1 (N = 330) \tag{2}
\]

4. Autocorrelation analysis
Each frame of windowed signal is next autocorrelated to give

\[
r(m) = \sum_{n=0}^{N-m} x(n)x(n+m), \quad m = 12 \quad \text{and} \quad N = 330 \tag{3}
\]

Here the highest autocorrelation value is 12, and it is the order of the LPC analysis. The 0th autocorrelation \( r(0) \) is considered as the energy of that particular frame.

5. LPC analysis
The next processing step is the LPC analysis. It converts each frame of \( p+1 \) autocorrelations into an LPC parameter set. The formal method for converting from autocorrelation coefficients to an LPC parameter set has been used here.

This method is known as Durbin’s method and can formally be given as the following algorithm.

\[
l = 0 : \quad E(l) = r(l)
\]

\[
l = 1 : \quad k(l) = \frac{r(l)}{E(l-1)} \quad E(l) = E(l-1)[1 - k(l)^2]
\]

\[
l > 1 : \quad k(l) = \frac{1}{E(l-1)}(r(l) - \sum_{i=1}^{l} a_i r(l-i)) \quad E(l) = E(l-1)[1 - k(l)^2]
\]

\[
a_m = a_m - k(l)a_{m-1}
\]

\[
a_m = k(l)
\]

where \( m \in [1, l] \), \( \forall l \in [1, p] \), and \( a_m = a_m^p \) is the \( n^{th} \)

LPC coefficient. Value of \( p = 12 \).

6. LPC parameter Conversion to Cepstral Coefficients.
This is a very important parameter set derived directly from the LPC parameter set. The cepstral coefficients are the coefficients of the Fourier transform representation of the log magnitude spectrum. They are more robust and reliable for speech recognition than the LPC coefficients.

The following recursion has been used to calculate the LPC cepstral coefficients.

\[
c_m^p = a_m + \sum_{k=1}^{m-1} k c_{m-k} a_{m-k} \quad m \in [1, p]
\]

\[
m = 12 \quad \text{(number of Cepstral coefficients per a frame)}
\]

7. Parameter Weighting
Because of the sensitivity of the low-order cepstral coefficients to overall spectral slope and the sensitivity of the high-order cepstral coefficients to noise, it has become a standard technique to weight the cepstral coefficients by a tapered window so as to minimize these sensitivities. The new parameter \( c_m^w \) can be obtained by,

\[
c_m^w = w(m)c_m
\]

where

\[
w(m) = 1 + Q/2 \sin (\pi m/Q)
\]

Here \( Q \approx (3/2) \times p \) (\( p = 12 \), for the system)

D. Constant trajectory mapping
The ultimate result of the feature extractor is a set of 12 arrays corresponding to the calculated 12 cepstral coefficients. The sizes of these arrays are similar, and the size depends on the length of the wave file. So, when considering an utterance, the length of the arrays usually
differs for different people, because the time duration of the utterance differs from person to person.

This is a challenge since these varying length outputs can’t be fed into the ANN. The input to the ANN should be constant.

The most common approach for this is to use the obtained trajectory to generate a sequence of labels, normally by means of a vector quantization (VQ) scheme [4]. Although it has been successfully used in practical speech recognition systems, it is still throwing away information. It does not keep information about the topological relationships between the output labels. One of the objectives of the present work is to convert the word recognition problem into a trajectory recognition problem, with the dimensionality of the trajectories reduced. In this manner, the trajectory classification is highly simplified. The basic idea behind the approach employed in this work is to use the output of a SOM trained with the output of the LPC Cepstrum block to obtain reduced state space trajectories that preserve some of the behavior of the original trajectory. The problem is now reduced to find the correct number of neurons for constituting the SOM and their geometrical arrangement.

Using the Self Organizing Map, each and every LPC trajectory with the variable length is mapped to a constant trajectory of 6 clusters, while preserving the input space.

1. Initialization
The random values were chosen for initial weight vector $w_j(0)$. These values were normalized around unity. The fact to be mentioned here is that, for all the iterations, the weights were initialized with the same set of vectors, since the SOM was not powerful enough to classify the data sets properly, when its initial weights have a large variation. Let $w_j$ be the synaptic weight vector of neuron $j$.

\[ w_j = [w_{j1}, w_{j2}, \ldots, w_{jm}]^T \quad j = 1, 2, \ldots, l \]  (7)

2. Sampling
The vector $x$ was chosen from the input space with certain probability and it represents the activation pattern applied to the lattice. Then this vector was normalized around unity. This made the input vector closer to the initial weight vector values, making the convergence faster. Let $x$ be the $m$-dimensional input vector then,

\[ x = [x_1, x_2, \ldots, x_m]^T \]  (8)

3. Similarity matching
The best similarity matching neuron $i(x)$, winner was found at the step $n$ as follows:

\[ i(x) = \arg \min_j \| x(n) - w_j \| , \quad j = 1, 2, \ldots, l \]  (9)

4. Updating
Updating of the weights has been done using the following formulae:

\[ w_j(n + 1) = w_j(n) + \eta(n)h_{i,j(x)}(n)[x(n) - w_j(n)] \]  (10)

$\eta(n)$ -learning rate

\[ h_{i,j(x)}(n) \] -neighborhood function around the winning neuron $i(x)$ at the $n^{th}$ iteration.

The variation of the topological neighborhood $h_{i,j(x)}(n)$ is,

\[ h_{i,j(x)}(n) = \exp \left( -\frac{d_{jj}^2}{2\sigma^2(n)} \right) \]  (11)

here, $d_{jj}^2$ is the lateral distance between the excited neuron $j$ and winning neuron $i$.

\[ d_{jj}^2 = \| r_j - r_i \|^2 \]  (12)

where $r_j$ is the position of neuron $j$ and $r_i$ is the position of the neuron $i$.

\[ \sigma(n) = \sigma_0 \exp \left( -\frac{n}{\tau_1} \right) , n = 0, 1, 2, \ldots \]  (13)

The changing of the learning rate is as follows:

\[ \eta(n) = \eta_0 \exp \left( -\frac{n}{\tau_2} \right) , n = 0, 1, 2, \ldots \]  (14)

E. ANN Recognizer

Input Nodes -72
Hidden Nodes -72
Output Nodes -10

Our Neural network is a three layers (input, hidden, and output), fully connected Neural Network with back-propagation algorithm. We initialized the Network weights with random number between –1 and 1 when we first create the Network. After that each node values (hidden and output) are calculated and each nodes output in the Network is approximated with a sigmoid function (the activation function- $1 \slash (1 + \exp (-num))$). Node values are calculated by multiplying each input which are coming to that node by the connection weight and adding all those results to the multiplied value of BIAS (normally set to 1) and bias weight, so that it can approximate nonlinear and differentiable functions. After getting the output from the
Network, a delta between that output and a desired output is computed, and it is back propagated to update the Network’s weights. Learning here is adjusting the Network’s weights (input weights, hidden weights, and bias weights) until the Network’s error is small enough thus increasing the Network’s accuracy in approximating the function.

Algorithms Used

For every hidden node calculate Hidden Node Value
\[ h_j = \text{bias} \cdot b_j + \sum (x_i \cdot w_i) \]  
(15)
\[ h_j = \text{sigmoid} (h_j) \]  
(16)

For every Output node calculate every output node value
\[ o_k = \text{bias} \cdot b_k + \sum (h_j \cdot w_j) \]  
(17)
\[ o_k = \text{sigmoid} (o_k) \]  
(18)

Error Propagation

Delta for each output node
\[ o \_d k = o_k \cdot (1 - o_k) \cdot (d_k - o_k) \]  
(19)
Od - output delta
D – desired output value
O - output node value

Delta for each hidden node
\[ h \_d j = h_j \cdot (1 - h_j) \cdot \sum (o \_d k \cdot w_k) \]  
(20)
Hd - hidden delta
W – hidden-output weights

Update weight

Update input Weight
\[ w_i = B P \_\text{LEARNING} \cdot h_j \cdot x_i \]  
(21)
W - input-hidden weights

Update bias for hidden nodes
\[ b_j = B P \_\text{LEARNING} \cdot B I A S \cdot h_j \]  
(22)
B - hidden bias weights

Update hidden Weight
\[ w_j = B P \_\text{LEARNING} \cdot o \_d k \cdot h_j \]  
(23)
W - hidden weight

Update bias for output nodes
\[ b_k = B P \_\text{LEARNING} \cdot B I A S \cdot o \_d k \]  
(24)
B – output node bias

Testing function

For every hidden node
\[ h_j = \text{bias} \cdot b_j + \sum (x_i \cdot w_i) \]  
(25)
\[ h_j = \text{sigmoid} (h_j) \]
B-hidden node bias weight

V. TEST RESULTS

Though the complete process of isolated word recognizer could not be proven for accuracy, most of the modules, when taken separately, were functioning properly. For the testing of the system, we recorded about 2000 utterances. It comprised of speech signals obtained for 10 words from 40 individuals. Each word was recorded 5 times from each individual.

A test set was also prepared by recording the same set of words from a set of individuals. Part of this set was the speakers used in recording the training samples. Other part was a new set of people.

A. Constant Trajectory Mapping

When testing the system, one of the most crucial parts was the extensive tests carried out on the SOM.

Fig. 4 shows the constant trajectory mapping output of a certain speech signal.

When we compare Fig. 4 with Fig. 5, which shows how the signal looks like before it is fed to the SOM, we can see that although the cepstral coefficient representation has been
remained roughly the same after the constant trajectory mapping, this mapping is not very much accurate. To clarify this point, below we present a representation of one cepstral coefficient along with its corresponding constant trajectory mapping.

![Fig. 5. Cepstral Coefficient 1](image)

![Fig. 6. Reduced Trajectory of Cepstral Coefficient 2](image)

According to the above figure, it can be seen that the SOM has not been able to map the cepstral coefficients in to a reduced trajectory properly.

This variation can be due to several reasons. First and foremost, it should be mentioned that finding SOM algorithms with optimum parameter values depends on trial-and-error approach. Since there is no hard-and-fast rule to find out those parameters, in order to find an optimum solution for the SOM, it is important that we test the SOM with different combinations of number of clusters, neighborhood radius value, initial parameter values, etc, which is something that we were not being able to carry out completely during our project duration.

**B. ANN Recognizer**

Testing with the ANN was carried out in different ways.

First, the ANN was trained using iterative training scheme, where the signal was fed into the recognizer one by one. Under this training scheme, two alternatives were used for the ANN convergence algorithm. In the first approach, after giving an input to the ANN, the error was calculated, if this error was less than a pre-defined value, the training was carried on, otherwise the training was stopped and the whole process was started all over again.

The variation in the learning error is shown in Fig. 7.

![Fig. 7. Iterative Training-maximum error condition](image)

In the second approach, the training was continued until the training error goes beyond a pre-defined minimum value.

The variation in the learning error is shown in Fig. 8.

![Fig. 8. Iterative Training-minimum error condition](image)

According to figures 7 and 8, it can be seen that the iterative training approach has not worked out. (Here, the back propagation learning coefficient=0.5, learning rate=0.5)

**Batch training**

Since the iterative training scheme didn’t work out, we tried out the batch training approach, where all the training samples were fed into the recognizer simultaneously. In both cases, the training was continued until the training error goes beyond a pre-defined minimum value.

The input was sent into the ANN with the batch size 10. The variation in the learning error is shown in Fig. 9.

![Fig. 9. Batch Training-training error](image)

(Here, the back propagation learning coefficient=0.5, learning rate=0.5)
VI. FUTURE DIRECTIVE

This system can be easily extended to an isolated word recognizer with medium sized vocabulary. A problem that arises here is the size of the ANN output increases proportional to the number of words. As a remedy for this, we can introduce the ANN output as a digital sequence number. This way, we can use the ANN to recognize a fair amount of words with the use of a relatively small number of output nodes. For an example, 12 output nodes can be used to recognize $2^{12}$ output words.

As another development, the ANN recognizer could be used to identify the basic sounds in the language, called phonemes. This way, despite of the size of the vocabulary, the number of the output nodes will remain the same. But to do this, we have to introduce sophisticated edge detection in order to break down the incoming signal into its constituent phonemes. The recognized phonemes could then be formulated into words using a language model.

VII. CONCLUSION

In this paper, we tried to introduce a simple speaker independent speech recognition system. The main idea behind the work was to experiment with the emerging concepts such as Artificial Neural Networks and Self Organizing Maps, in order to find out their support for the field of speech recognition.

The process of isolated speech recognition has been implemented completely, though there were problems with the functionality of the SOM and the ANN. Though the correct functionality of the SOM and the ANN could not be obtained, this could be done if we were able to hit on the correct coefficients for the algorithms.

Another reason for the low accuracy of the system is the unwanted noise. In the capturing process on the sound card and because of the environment it introduces lots of noise to the process. The problem faced here is while trying to reduce one type of noise, it introduces another type of noise. The system has been developed in a modularized manner so that it could be easily extended to recognize connected or continuous speech.

ACKNOWLEDGMENT

The authors would sincerely like to thank Dr. Thrishantha Nanayakkara, Senior Lecturer, Department of Mechanical Engineering, University of Moratuwa, Dr. Chatura de Silva, Senior Lecturer, Department of Computer Science and Engineering, University of Moratuwa, for their supervision on this work, and Mr. Shantha Fernando, Senior Lecturer, Department of Computer Science and Engineering, University of Moratuwa, for his excellent coordination of the project.

REFERENCES