4-2018

Fusion of Audio and Visual Information for Implementing Improved Speech Recognition System

Vikrant Satish Acharya
Grand Valley State University

Fusion of Audio and Visual Information for Implementing Improved Speech Recognition System

Vikrant Satish Acharya

A Thesis Submitted to the Graduate Faculty of

GRAND VALLEY STATE UNIVERSITY

In

Partial Fulfillment of the Requirements

For the Degree of

Master of Science in Engineering

Electrical Engineering

April 2018

Acknowledgement

The success of any project largely depends on the encouragement and guidance of many people. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.

Firstly, I would like to thank my school, Grand Valley State University and the school of Engineering, for giving the opportunity and encouragement to do this project. It has given me tremendous exposure to the field of Signal and Image Processing and a strong base to apply the knowledge that I have gained through my coursework in a practical environment.

I would like to thank Dr. Nicholas Baine, my thesis advisor, for his support and help in providing valuable inputs. I am also grateful to my committee members, Dr. Samhita Rhodes, Associate Professor, Electrical and Biomedical Engineering, Dr. Robert Bossemeyer, Associate Professor, Computer and Electrical Engineering for their constant support throughout the span of this research. I am thankful to them for being patient with me and for taking out valuable time from their busy schedule.

I would also like to thank the department of Human Research Review Committee for giving me the approval to carry out my project. I would like to show my appreciation to all the Engineering students of Grand Valley State University for helping me in the collection of database. Their help was vital for the completion of the project.

Last but not the least I would like to thank my family and friends for their constant support throughout this project.

Abstract

Speech recognition is a very useful technology because of its potential to develop applications, which are suitable for various needs of users. This research is an attempt to enhance the performance of a speech recognition system by combining the visual features (lip movement) with audio features. The results were calculated using utterances of numerals collected from participants inclusive of both male and female genders. Discrete Cosine Transform (DCT) coefficients were used for computing visual features and Mel Frequency Cepstral Coefficients (MFCC) were used for computing audio features. The classification was then carried out using Support Vector Machine (SVM). The results obtained from the combined/fused system were compared with the recognition rates of two standalone systems (Audio only and visual only).

Table of Contents

List of Tables

List of Figures

1. Introduction:

Use of human biometric modalities such as face, speech, iris, lips, hand gestures, and fingerprints can be utilized to interface with and act as a control signal for a variety of applications. Of those, speech is one of the most natural/common forms of communication. Many interface solutions have been developed, which are based on speech recognition systems, which are implemented using a variety of techniques, ranging from the use of integral microchips [1] to fuzzy logic algorithms used for noisy speech [2]. Even if it is a strong human modality to be used, the systems utilizing it will have challenges in implementation due to poor performance under non-ideal conditions (noisy environment). In such cases, it is beneficial to include and fuse additional modalities such as lip-reading in the system. Two such human modalities can be fused together, improving performance. This processed output can then be used as a command signal to operate a control system.

For example, research has been done to make a wheelchair capable of operating based only on speech recognition. However, in a noisy environment, a situation can occur in which the wheelchair is not able to recognize the correct direction which is spoken by the disabled person. Consequently, this may result in the wheelchair moving in the wrong direction. In such a case, if another modality of lip-reading is fused with the speech recognition, it will be helpful for improving the result of recognizing the correct command for the wheelchair.

Use of such natural modalities is not limited to control of a wheelchair. Other possible applications include security systems or control of computer interfaces (like a mouse), which is used for moving the pointer on screen. Many small operations that a driver of a vehicle must carry out manually, such as changing the climate control and radio settings, can be carried out in an automated way using speech.

This thesis proposal is arranged into six sections as follows: The first section consists of the introduction as explained above; The second section is the background; The third section is of literature review, which discusses many of the popular algorithms, which have been implemented by the researchers so far in the same field of study; The fourth section is the methodology section which explains the proposed audio-visual speech recognition system. Followed by this methodology section, the results of the research and conclusion are discussed in the fifth and sixth sections.

2. Literature Review:

Many speech recognition systems have been developed to make life easier for a human being. Many such systems are not completely robust and can be improved. This research project describes an approach for improving performance of speech recognition systems with the help of fusion of signal processing and image processing algorithms. The purpose of this implementation is to have an improved recognition system developed using the fusion of two human modalities, human speech and human lips.

In the proposed study, three systems will be implemented. First will be a speech recognition system, second will be a lip-reading system, and the third will be an audio-visual speech recognition system. This third system will have the fusion of speech recognition and lip-reading algorithms in it. The main aim of this study is to compare the performances of these three systems and to prove that the feature level fusion-based system performs better than the two stand-alone systems.

Basically, the speech recognition and lip-reading algorithms consist of three main stages of operation. Those three stages are data collection, feature extraction, and feature matching. In the first data collection stage, the speech samples and the face videos will be collected. In the feature extraction stage, these speech samples and videos will be used for extracting important features. A speech signal has a lot of information other than the linguistic message which is required to be suppressed for recognition purpose. This unwanted information includes, characteristics of environment, characteristics of the recording equipment. The task related to emphasizing upon the important linguistic information in a speech signal and suppressing all the other unwanted information is carried out in the stage of feature extraction. The features, which are extracted, are then processed using algorithms selected for these implementations. All such algorithms consist of some specific mathematical calculations, which can be carried out on collected data using the development software MATLAB.

Prior to being used for speech recognition, the process must be performed on a training set, where the samples are known. Features are extracted from each of the samples in the training set and stored in a training database, which are later used for pattern matching and classification.

For speech samples that are unknown, they can have their features extracted and compared to the training database using pattern recognition and are then classified. This classification is desired to match the uttered word or phrase.

To test the system, multiple subjects (persons) are recruited and asked to speak a set of words, which are then classified by the system. The classification from the system is then compared with the true value in each instance. From this a percentage match is calculated to determine the performance of the system.

2.a. Review of algorithms for speech recognition systems:

Speech is one of the primary ways of communication for human beings. Just like that of a fingerprint or iris, speech is also a characteristic that is unique for every individual. The unique information, which can be extracted from every individual, can be used to implement systems that can recognize the speech or voice. Such systems can be very useful for many applications such as ensuring secure access to systems [3], gender recognition, recognition of age, emotion, accent and speaker identity. Due to such a vast span of applications, researchers have always been inspired by this phenomenon of speech recognition. The basic block diagram of a speech recognition system is shown in Figure 1.

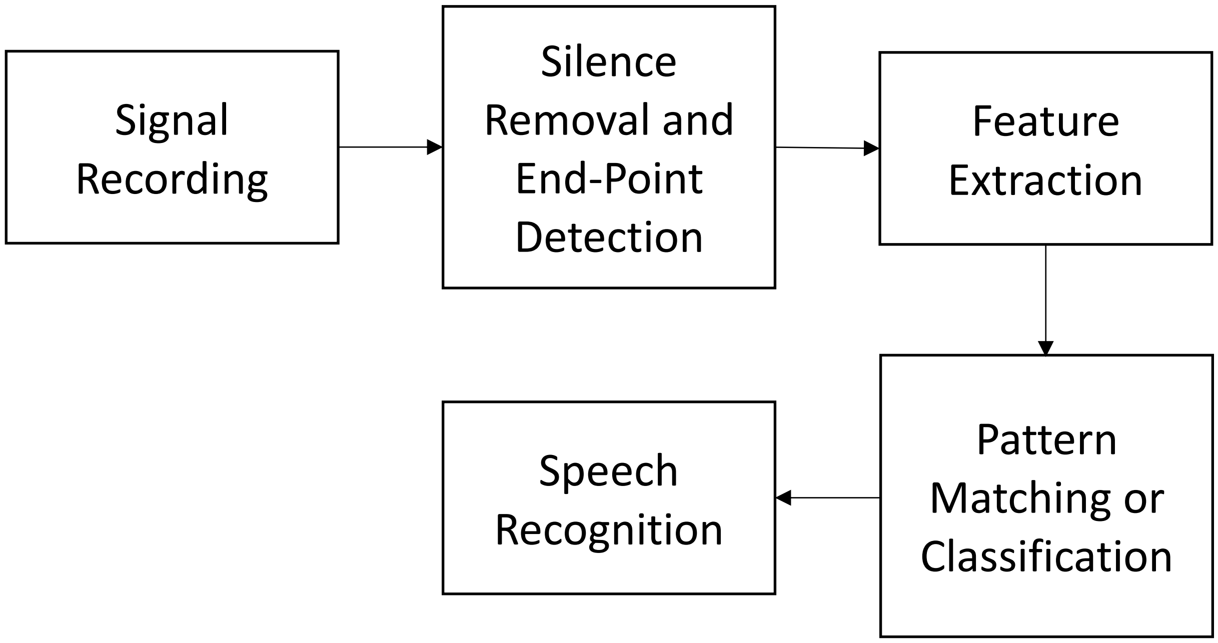

Signal Recording
Silence Removal and End-Point Detection
Feature Extraction
Pattern Matching or Classification
Speech Recognition

Figure 1: Basic block diagram of Speech Recognition System

As shown in the block diagram, the speech signal is recorded in a specific format based on the application. The possible formats are .MP3, .WAV, .AIFF and .AU. The speech specific attributes, which are required for efficient feature extraction, are mainly present in the voiced part of signal [29]. After the signals are recorded, the next step is to remove the silence/unvoiced part of the signal, using silence removal and end-point detection techniques. After that, the necessary features are extracted from the speech signal [3]. These extracted features are then used for pattern matching purposes. Many algorithms have been developed and implemented for silence removal, feature extraction and pattern matching purposes. These algorithms are briefly discussed in the following sections.

2.a.1 Silence Removal and End-Point Detection Algorithms:

For carrying out efficient feature extraction in any speech or speaker recognition application, preprocessing of silence removal and end-point detection is important. Conventionally, a three-state representation is used for classifying the events in speech [29]. These states are, (i)silence, (ii)unvoiced and (iii)voiced. No speech is produced in the silence state. The vocal cords are not vibrating in the unvoiced state whereas they are tensed and periodically vibrating in the voiced state [29]. Silence (background noise) and unvoiced part are distinguished together as silence/unvoiced from voiced part because, low energy content is present in unvoiced part.

The two widely used methods for silence removal are the Zeros Crossing Rate (ZCR) method and the Short Time Energy (STE) method.

***Zeros Crossing Rate (ZCR) Method*:**

The measure of number of times in a given time interval/frame, the amplitude of speech signals passes through a value of zero is called the zero-crossing rate [30]. As mentioned in [30], the definition of zero-crossing rate is:

$$Zn = \sum_{m=-\infty}^{\infty} |sgn[x(m)] - sgn[x(m-1)]| w(n-m)$$

-------(1)

Where, sgn[x(n)] = 1, x(n) ≥ 0

= -1, x(n) < 0

and w(n) = $\frac{1}{2N}$ for, 0 ≤ n ≤ N-1

= 0, Otherwise

Here, *x*(*m*) represents data sequence, *w*(*n*-*m*) represents a limited time window sequence and *N* is the window length. In a speech signal, the energy in the voiced part is concentrated at lower frequencies and the energy in the unvoiced part is found at higher frequencies. Zero crossing rate and energy distribution with frequency are strongly related. If the zero-crossing rate is higher, the signal is unvoiced and if it is lower, then the signal is voiced [30].

***Short Time Energy (STE) Method*:**

In Short-Time Energy algorithm, amplitude of signal is taken into consideration. For unvoiced speech, the amplitude is lower and for the voiced speech, it is higher. These amplitude variations can be represented using energy of speech signal [30]. The short time energy is calculated as:

$$En = \sum_{m=-\infty}^{\infty} [x(m)w(n-m)]^2$$

-------(2)

In STE, it cannot be specified accurately that how much greater is the energy in unvoiced part of a speech signal as it varies in every case [29]. ZCR has one rule, which specifies that, for a clean speech of 10ms, if the ZCR of a portion exceeds 50 then that portion is labeled as unvoiced and if it is about 12, then it is labeled as voiced [29][49].

2.a.2 Feature Extraction Algorithms:

During the process of feature extraction, analysis of the speech signal is carried out. During this process, the required information from the speech signal is identified for producing a meaningful representation of the signal [3]. This feature extraction step mainly includes, measurement of important characteristics of the signal such as energy or frequency response. It also includes parameterization of the signal in which these measurements are augmented with perceptually meaningful derived measurements and then these numbers are conditioned to form the feature vectors [3]. For some applications, it is also required to transform the original type of signal to some other useful form of signal.

***Linear Predictive Coding (LPC) Method*:**

In this method, the speech signal is passed through the speech analysis filter for removing the redundancy in the signal [32]. While doing so, residual error is generated as an output. As compared to original signal, this residual error can be quantized by smaller number of bits [32]. So, this residual error and speech parameters can be transferred instead of transferring the complete signal. A technique in which least mean squared error theory is used for computing a parametric model is called as Linear Prediction (LP). As mentioned in [32], the speech signal is approximated as linear combination of its previous samples. Formants are described by the obtained LPC coefficients and the resonant peak frequencies are called the formant frequencies.

Linear predictive coefficients and peaks in the spectrum of filter are calculated for finding the locations of formants in a speech signal [32].

***Linear Predictive Cepstral Coefficient (LPCC) Method*:**

The main assumption behind this method is that the nature of the sound being produced is based on the shape of the vocal tract [6]. For this purpose, a digital all-pole filter is used for modeling the vocal tract [7]. In this algorithm, one vocal tract transfer function is calculated using a set of Linear Predictive Coefficients. The autocorrelation method used in this algorithm consists of calculating filter gain and the autocorrelation of windowed speech signals [6].

As explained in [6], the vocal tract transfer function is calculated as.

$$V(z) = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}}$$

-------(3)

Where, $V(z)$ is the transfer function, G is the filter gain, a_k is a set of Liner Prediction Coefficients (LPC) and p is the order of all-pole filter. a_k is the same as explained in the previous section.

A matrix of simultaneous equations is calculated in the autocorrelation method involved in this technique.

$$\begin{bmatrix} R[0] & R[1] & \cdots & R[p-1] \\ R[1] & R[2] & \cdots & R[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ R[p-1] & R[p-2] & \cdots & R[0] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} R[1] \\ R[2] \\ \vdots \\ R[p] \end{bmatrix}$$

-------(4)

Where, $R[n]$ is the autocorrelation function of signal. The gain G is calculated as,

$$G = \sqrt{R[0] - \sum_{k=1}^{p} a_k R[k]}$$

-------(5)

Furthermore, in this algorithm a cepstrum of speech sequence is evaluated, which is referred to as cepstral analysis. The Inverse Discrete Fourier Transform (IDFT) of the log magnitude of the DFT of a signal is called as cepstrum. It is calculated using Equation (6).

$$\mathrm{c[n]} = F^{-1}\{log|F\{x[n]\}|\}$$

-------(6)

Where, $x[n]$ is the signal, F is DFT and F^{-1} is IDFT. This Cepstrum is used for estimation of dominant fundamental frequency in clean stationary speech signal.

The Linear Predictive Cepstral Coefficients are evaluated from LPCs using a recursive method. This recursive procedure is calculated as:

$$c[0] = in(G)$$

$$c[n] = a_n + \sum_{k=1}^{n-1} \left(\frac{k}{n}\right) c[k] a_{n-k} \text{ for } 1 \leq \mathrm{n} \leq \mathrm{p}$$

$$c[n] = \sum_{k=1}^{n-1} \left(\frac{n-k}{n}\right) c[n-k] a_k \text{ for n > p}$$

-------(7)

Where, c is the LPCC coefficient and p is the total number of samples in the sequence and a_k are the predictor coefficients.

***Perceptual Linear Predictive Cepstral Coefficient (PLPCC) Method*:**

In this algorithm, various concepts of psychophysics of hearing are used. The three concepts of psychophysics which are used in this algorithm are, critical band spectral resolution, equal loudness curve and intensity loudness power law [40]. These concepts are made useful for finding the all-pole model of short-term spectrum of speech [8]. The short-term power spectrum is computed using Fourier Transform, and then it is transformed to a Bark scale, which is a frequency scale on which equal distances correspond with perceptually equal distances [31]. The sensitivity of human hearing at different frequencies is approximated using a function relating the intensity of a sound and its perceived loudness. As mentioned in [31], The auditory spectrum is approximated using the all-pole model of LP and then the LP parameters are transformed to cepstral coefficients. The flow chart of PLPCC [31] is as shown in Figure 2.

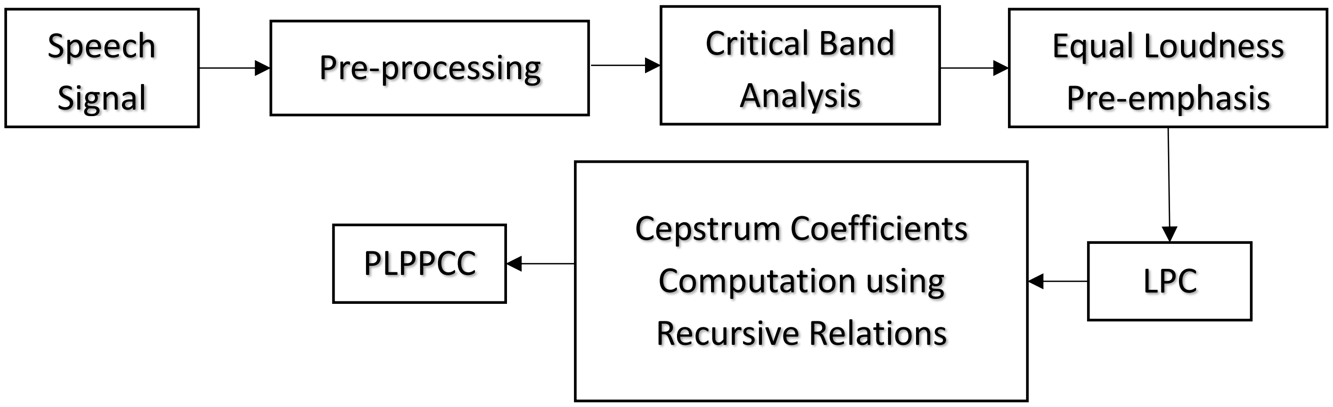

Speech Signal
Pre-processing
Critical Band Analysis
Equal Loudness Pre-emphasis
LPC
Cepstrum Coefficients Computation using Recursive Relations
PLPPCC

Figure 2: Flow chart of PLPCC algorithm

As shown in the flow chart, frame blocking, and windowing is carried out on the speech signal in the pre-processing stage. DFT and its squared magnitude is computed. The power spectrum of signal is then integrated in the overlapping critical band filter responses. The spectrum is then pre-emphasized to simulate the unequal sensitivity of human ear. The inverse

Discrete Fourier Transform (IDFT) is performed and then cestrum coefficients are computed using autoregressive model derived from regression analysis [33].

***Mel Frequency Cepstral Coefficient (MFCC) Method*:**

This algorithm is based on hearing capabilities of human ears. Known variations of the human ear's critical bandwidth with frequency are considered while implementing MFCC algorithm [9]. Steps such as framing, windowing, Discrete Fourier Transform (DFT) and Discrete Cosine Transform (DCT) are included in the implementation of MFCC.

In this method, a set of triangular filters is used for computing a weighted sum of filter spectral components [9]. The magnitude frequency response of every filter is triangular. Sum of the filtered spectral components is calculated for getting each filter output. The MFCC algorithm is discussed further in detail in the methodology section.

***Principal Component Analysis (PCA) Method*:**

The principal component analysis has been used in many applications for feature extraction. The method takes advantage of dimensional reduction. In PCA, the eigenvectors are extracted from the feature vector. Eigenvalues and covariance matrices are calculated for obtaining these eigenvectors. Only those eigenvectors with higher eigenvalues are selected to carry out the dimensional reduction. This is done because the eigenvectors with higher eigenvalues account for higher variability in the data.

In one of the previous algorithms, KL (Karhunen-Loeve) transformation was employed instead of DCT (Discrete Cosine Transform) on the MF (Mel Frequency) output to carry out the feature extraction. This transform used was based on PCA. It was used to reflect the statistics of speech data more precisely than the DCT [4].

In another algorithm proposed, PCA was applied on FFT (Fast Fourier Transform) to calculate the filter bank coefficients [5]. Kernel PCA based algorithm for feature extraction was proposed to overcome the issue of additive noise. In this algorithm, again, PCA was used instead of DCT, in which main speech element was projected onto low order features and noise element was projected onto high order features [5].

2.b. Review of algorithms for Lip reading or Visual Speech Recognition [VSR]:

The concept of lip reading or visual speech recognition [VSR] has attracted many researchers and many algorithms have been proposed by the researchers for implementing automated VSR systems.

Lip reading is important for people who have hearing impairment and can only use the visual signs to understand the speech. In many ways, such automated visual speech recognition is useful for people without disabilities also. Lip reading systems are mainly useful for people without disabilities when the acoustic speech is not understandable [13]. Especially in a noisy environment, this visual component of speech is unconsciously used by everyone for carrying out normal communication [13]. Such systems also have wide range of applications in the fields of human-computer interface, video surveillance, security systems, defense systems and car navigation systems.

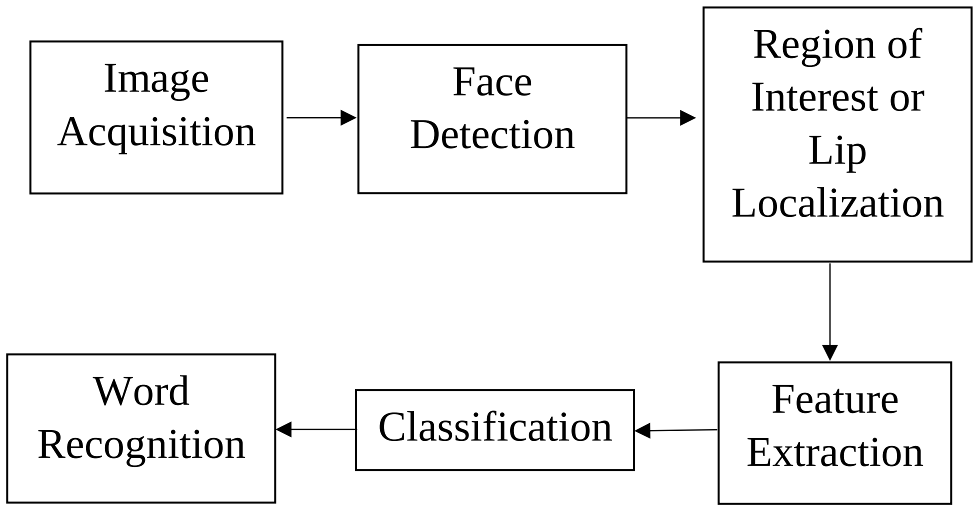


Figure 3: Basic block diagram of Lip Reading System

Figure 3 shows the steps involved in a lip-reading system. The first step of image acquisition involves breaking the video into frames [14]. The number of frames, which are formed from a video, depends upon the algorithms, which are selected for further steps involved in the implementation. Many techniques and algorithms have been proposed and used for all these steps, which are involved in a lip-reading system. The most popular techniques are taken into consideration as a part of the following discussion.

2.b.1 Face detection and lip localization techniques:

One of the most popular algorithms used for face detection is the Viola Jones algorithm. This algorithm was proposed by Paul Viola and Michael Jones, and it is considered efficient for detecting faces. It was shown that, while operating on 384 by 288 pixel images, this algorithm could detect faces at the rate of 15 frames per second [15]. As explained in [15], this algorithm was divided into three main contributions of object detection framework. The first of them was integral image. Features resembling Haar basis functions were used within this algorithm, and integral image representation was used for evaluating these features rapidly. Haar like features use contrast variance in the adjacent rectangular groups of pixels in an image [26]. This contrast variance is used for determining light and dark areas. Haar-like features are formed by two or three adjacent groups with relative contrast variance [26]. In any image, the total number of Haar-like features is very large. For carrying out fast classification, only the important features are required to be selected. In the Viola-Jones algorithm, this selection of important features was carried out using Adaboost and this was the second contribution of the paper. This Adaboost is a process which was used for finding relevant and irrelevant features. In the third contribution of this algorithm, more complex classifiers were combined in a cascade structure for increasing the speed of the detector.

Another algorithm which is widely used for face detection purpose is the Kanade-Lucas-Tomasi (KLT) algorithm. In this algorithm, initially, feature points are detected and then displacement of these points from one frame to another frame is calculated [16]. The movement of the head is computed using this displacement and optical flow tracker. Figure 4 shows the flow chart of KLT algorithm [16].

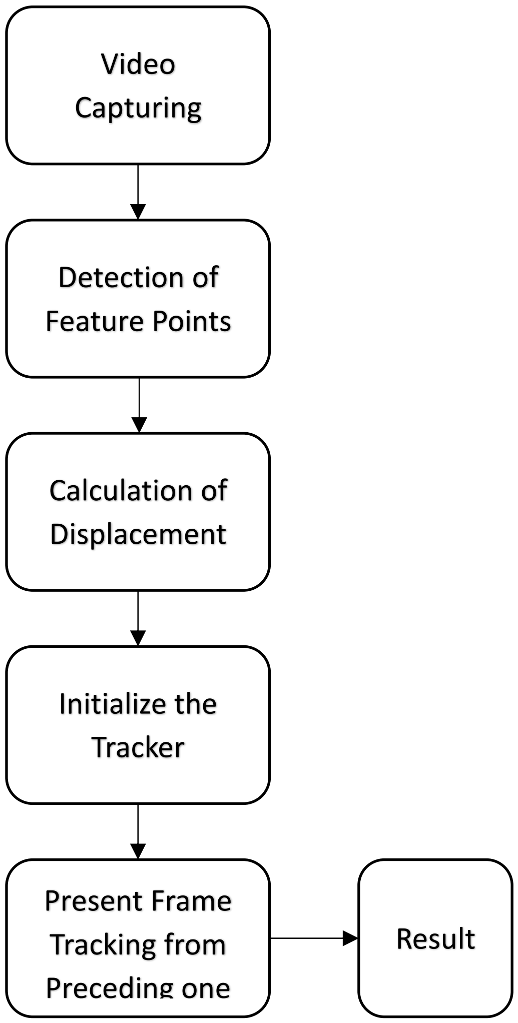

Video Capturing
Detection of Feature Points
Calculation of Displacement
Initialize the Tracker
Present Frame Tracking from Preceding one
Result

Figure 4: Flow chart of KLT algorithm

As explained in [16], two simple steps are used in KLT algorithm for tracking the face. Firstly, traceable feature points from the first frame are found and then calculated displacement is used to track those features in next frame. The traceable feature points used are Harris corners. These Harris corner detectors are popularly used for detecting corners in the field of computer vision. After detecting the corners, optical flow is computed for each translational motion and the corners are detected accordingly in the successive frames by connecting the motion vectors [16].

If (x,y) is considered as one of the corner point, then it is displaced by some variable vector, $(b_1, b_2, \ldots, b_n)$. The coordinates of the new point are then calculated using Equation (9),

$$x^2 = x + b_1$$

$$y^2 = y + b_2$$

-------(9)

The displacement with respect to each coordinate is calculated using the wrap function as follows:

$$W(x;p) = (x + b1; x + b2)$$

-------(10)

Where, p is the displacement parameter.

The methods used for lip localization mainly consist of color space-based techniques. Within these methods, color spaces like RGB (Red-Green-Blue), HSI (Hue-Saturation-Intensity), and YCbCr (Luminance-Component blue- Component red) or L*a*b space are used [14]. As explained in [17], the mouth region of human face contains more red, that is Cr component than blue, that is Cb component. This factor of chrominance color ratio can be used in various ways for lip localization. According to the results in [17], the saturation component of HSI color space with Cr and Cb component gives good results for lip localization.

2.b.2 Feature Extraction Techniques:

Out of all the steps involved in a lip-reading algorithm, feature extraction is the most important and crucial part. In general, feature extraction techniques can be divided into two types [14]. One type is Pixel-Based, and the other type is Lip Contour Based.

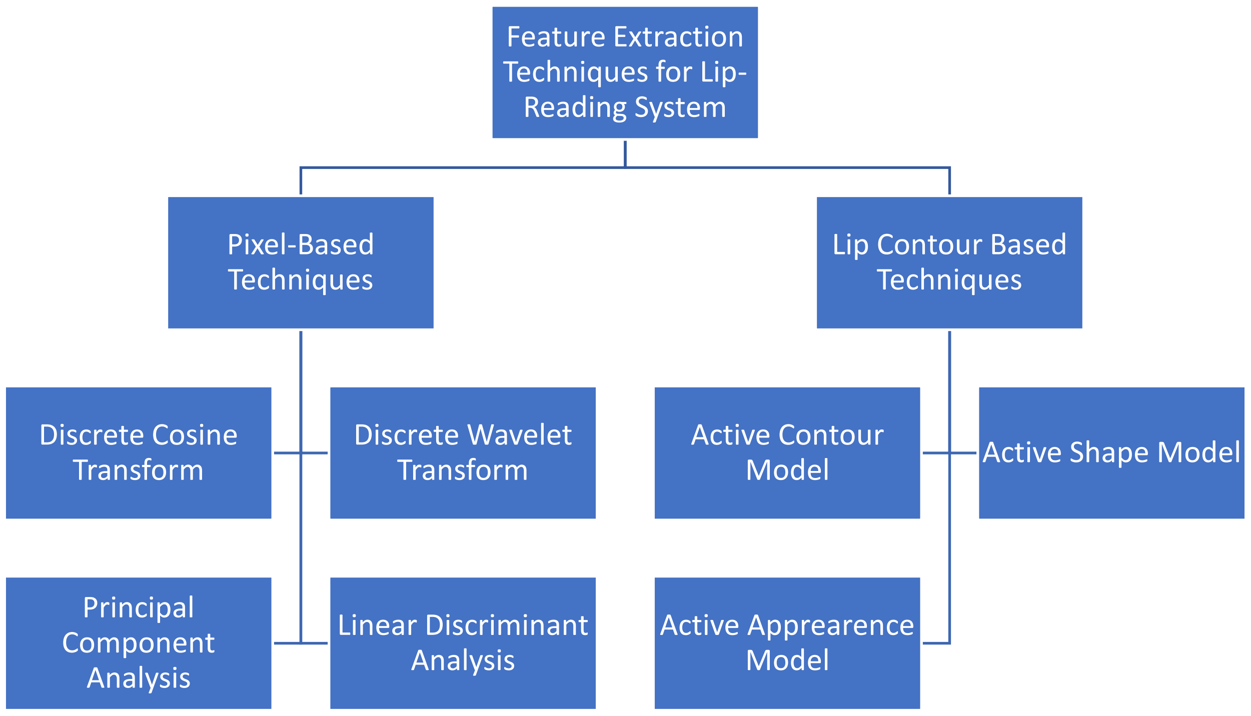

Feature Extraction Techniques for Lip-Reading System
Pixel-Based Techniques
Lip Contour Based Techniques
Discrete Cosine Transform
Discrete Wavelet Transform
Active Contour Model
Active Shape Model
Principal Component Analysis
Linear Discriminant Analysis
Active Apprerence Model

Figure 5: Feature extraction Techniques for Lip-Reading Systems

Pixel-Based or Image-Based Techniques:

One of the most popular pixel-based feature extraction method is the use of Discrete Cosine Transform (DCT). DCT is similar to Discrete Fourier Transform (DFT) as both these are used for transforming an image from spatial domain to frequency domain. DCT is mainly used for separating an image into spectral sub bands of different importance [27]. In [18], 2D-DCT was used as one of the methods for the feature extraction purpose. In that attempt, the features were extracted using Equation (11).

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos\frac{\pi(2m+1)}{2M} \cos\frac{\pi(2n+1)}{2N}$$

-------(11)

Considering the image *A* of size *M* by *N*, B_{pq} is the DCT coefficient of image *A* at location (*p*,*q*).

$$\alpha_p = \left\{ \begin{array}{cc} \frac{1}{\sqrt{M}}, & p = 0 \\ \sqrt{\frac{2}{m}}, & 1 \le p \le M-1 \end{array} \right\}$$

$$\alpha_q = \left\{ \begin{array}{cc} \frac{1}{\sqrt{N}}, & q = 0 \\ \sqrt{\frac{2}{n}}, & 1 \le q \le M-1 \end{array} \right\}$$

-------(12)

Another well-known transform used for feature extraction is Discrete Wavelet Transform (DWT). As mentioned in [19], wavelet transform can be used as a multiscale differentiator as it represents singularity of an image at multiple scales. This method is also useful in case the images are captured with multiple orientations like horizontal, vertical, and diagonal. In wavelet decomposition, each stage of filtering splits the image into four parts with the help of low-pass and high-pass filters. Out of these parts, the image with a low-spatial frequency is selected for the next decomposition level. After carrying out three such levels of decomposition, the resultant matrix representing the lowest spatial frequency sub-image is extracted as the feature vector. With the help of this methodology, only those coefficients, which play significant role in lip motion, are selected [19].

Often, these transforms described above are combined with one of the techniques, which is useful for dimensional reduction [19]. Two of the most commonly used methods for dimensional reduction are Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA).

To implement a lip-reading system, PCA was combined with DCT [20]. In this algorithm, 32 x 16 matrix was considered as 1D vector *Xi*, where *i* = {1, 2, 3…. *S*}, and *S* is the number of training samples. Covariance matrix *C* and mean vector *m* were calculated for all these X_i. Eventually eigenvectors and eigenvalues of all *C* were calculated. Only the *K* eigenvectors with the largest eigenvalues were selected to generate the matrix P_{PCA}. This PCA method is used to reduce the mean square error [21]. This mean square error is defined as,

$$E\|x - \hat{x}\|^2 = \sum_{i=1}^{p} E(x_i - \hat{x_i})^2$$

-------(13)

Where, *x* is the random vector such that $x^T = [x_1, x_2, \ldots, x_p]$ and $\hat{x}$ is the projection of *x* into subspace *V*.

LDA method consists of two scatter matrices called within-class scatter matrix *Sw* and between class scatter matrix *Sb*. These matrices are defined as follows:

$$S_b = \frac{1}{N} \sum_{i=1}^{c} liS_{bi}$$

$$S_{bi} = (\mu_i - \bar{x})(\mu_i - \bar{x})^T$$

-------(14)

Where, li is number of data points in class i and $\sum_{i=1}^{c} l_i = N$. μ_i is mean point of ith class, $\bar{x}$ is the mean of all data points and we have *c* known classes, $L_1, L_2, \ldots, L_c$.

$$S_w = \frac{1}{N}\sum_{i=1}^{c} S_{wi}$$

$$S_{wi} = \sum_{j \epsilon L_i} (x_j - \mu_i)(x_j - \mu_i)^T$$

-------(15)

In this algorithm, the data is projected to a lower dimension. Matrices in lower dimension are then defined as:

$$S_b^W = W^T S_b W$$

$$S_w^W = W^T S_w W$$

-------(16)

Next, a transformation matrix *W* is found such that *Sb* is minimized and *Sw* is maximized. This results in the ratio of *Sb* and *Sw* being maximized. This function is calculated as,

$$J(W) = \frac{|W^T S_b W|}{|W^T S_w W|}$$

-------(17)

So, if *J(W)* is maximized, we can get the matrix *W* using,

$$(S_b - \lambda_i S_w) W_i = 0$$

-------(18)

A combination of DCT and LDA algorithms were used in [22] for extracting the features for lip-reading. As mentioned in [22], the transformation matrix *W* and transformed features were calculated to keep maximum distance between different classifications and minimum distance within each classification.

Lip Contour or model-based techniques:

The Active Contour Model (ACM) technique which is also known as Snake was introduced [23]. In this method, an energy minimizing spline was used. This spline was created to concentrate upon features such as lines and edges. Image forces and external constraint forces were used to attract these splines towards the features. As mentioned in [23], image forces are responsible for pushing the snakes towards salient features like lines and edges whereas the external forces push the snakes near a local minimum. It was described within the paper that image energy responsible for pushing the snakes towards salient features is a combination of three energy functionals which are Line Functional, Edge Functional and Terminal Functional.

Two additional well-known model-based methods used for feature extraction are the Active Shape Model (ASM) and the Active Appearance Model (AAM).

In ASM, a statistical shape model calculated using labeled training data is used for the shape constraint [24]. Initially, a mean shape is calculated using aligned images and the PCA technique. Landmark points are then used for calculating inner and outer contours. An approximated shape of lips can then be defined by using Equation (19).

$$x = \bar{x} + P_b$$

-------(19)

In this equation, $\bar{x}$ is the mean shape, P is the matrix of first *t* important eigenvectors and *b* is the vector of *t* weights [24]. In this way, the valid shape of lip images, *x* is obtained.

AAM is an extension of ASM method as it uses combination of model described, using shape variation and the statistical model of grey levels [24]. In AAM, intensity values from normalized training images are sampled into a vector, *g*. Then the appearance model is obtained using Equation (20).

$$g = \bar{g} + P_g b_g$$

-------(20)

In this equation, $\bar{g}$ is the mean, P_g are the main modes of variation, and b_g are the texture parameters. After obtaining model g, the difference between the model and testing images can be calculated by the model estimate onto the target image. The iterative Active Appearance Model algorithm is used for this purpose.

Other than these image-based and model-based techniques, two of the most advanced techniques used for lip reading are Neural Networks and Hidden Markov Model. The Neural Networks technique was first introduced by Beala and Finaly [25] for lip-reading. Later on, techniques like Back Propagation [BP], Time-Delay Neural Network [TDNN], Hidden Markov Model (HMM) were introduced to improve the performance of neural network-based lip-reading systems. Compared to image based and model-based techniques, Neural Networks and Hidden Markov Model techniques are computationally expensive.

2.c Pattern Matching Algorithms for lip-reading and speech recognition systems:

After the feature extraction from audio or visual speech samples is done, pattern matching is required. This block decides to which of the training word class the testing word belongs. For that purpose, the task is to find out the difference between extracted feature vectors. In that case, the distortion between speech samples can be referred as the distance between feature vectors.

The most common method used for finding out the differences is Euclidean distance. This Euclidean distance is referred as the local distance between the feature vectors. This method of pattern matching has remained useful for many speech recognition algorithms, but it has certain drawbacks. If the length of feature vectors is not the same, then this method of pattern matching is not suitable for getting accurate recognition results. In that case, many other methods of pattern matching can be used.

Dynamic Time Warping (DTW):

In Dynamic Time Warping, the entire processing is executed in small steps [10]. Local distance measurement is carried out for all of these small steps. This approach allows for the different durations of all the utterances of same word. Additionally, even if the words are of the same length, the different parts of the word are spoken with different emphasis. Therefore, the rates with which speech signals for different words are created are different. For overcoming these problems, certain time alignment is performed using the DTW algorithm [10]. DTW is a good algorithm for finding out the lowest distance path without a lot of computation.

Support Vector Machine (SVM):

Many of the speech recogntion applications implemented using SVM are developed in combination with Hidden Markov Models (HMM) algorithm. In HMM, certain stochastic models are generated and the probability of unknown utterances getting generated by each model is compared [11]. In SVM algorithm, the distance between samples and the classification border is maximised. Unseen patterns are generalised by maximizing this distance. This distance is referred to as margin. Compared to Neural Network classifiers, SVMs are better performers because they don’t have problems like convergence and stability.

One algorithm was implemented using pure SVM in which the Token Passing Model algorithm was utilized [12]. This algorithm is an extension of the Viterbi algorithm. In this approach, one probability matrix was built with one row per class and one column per frame. Then, the Token Passing Model algorithm was used for obtaining the chain of recognised words from this matrix.

When the performance of this algorithm was compared with the HMM algorithm, it was observed that SVMs are better performers than HMMs. It was eventually concluded in the approach that, for a small database, SVMs are able to improve recognition accuracy of HMMs [12]. Similar results can be obtained with a huge database as well.

Other techniques such as Gaussian Mixture Model and Vector Quantization are also used for pattern matching purpose.

2.d Applications of Speech Recognition Systems:

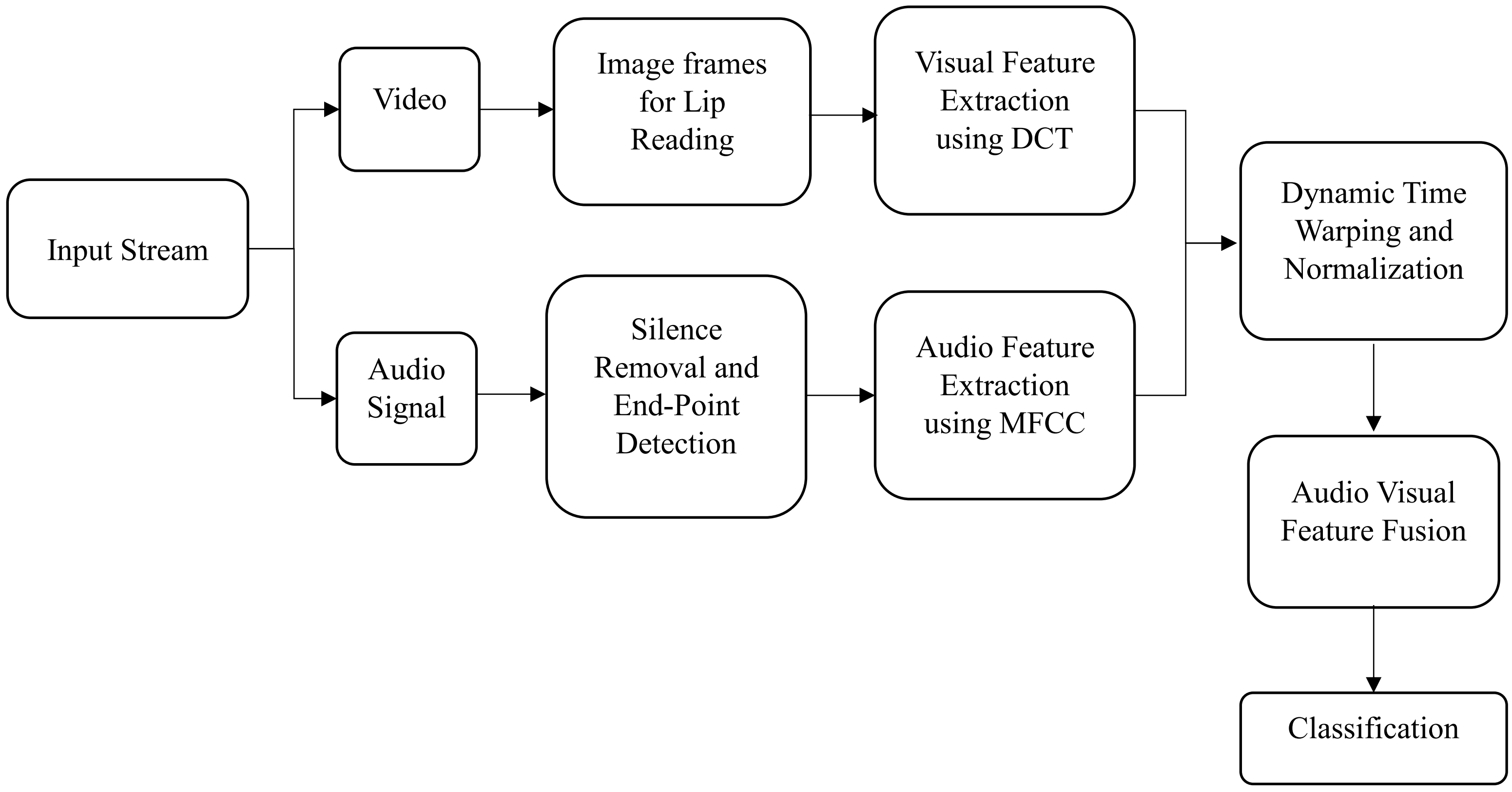
Presently, there is a vast number of applications of speech recognition systems. One of the main applications of it is present in the field of security systems. These systems are mainly used for automatic speaker identification or speaker verification.

Now these applications are divided into different categories such as isolated word recognition systems, keyword recognition systems, and multiple word recognition systems. All these applications are mainly used in the field of communication. There are present systems such as automated operator services in which all the functions of an operator are handled automatically by the speech recognition systems. These functions include billing and general inquiries. Such a system was introduced by AT&T. Additionally, certain voice dialing systems have been developed by AT&T and Bell Atlantic. In such systems, it is possible to complete the call without pushing the buttons on the telephone. Other than telecommunication services like AT&T, such automated speech recognition based customer service applications are also present in banking systems and airports.

Some real world applications such as live subtitling on television, off-line notetaking systems, speech to text conversion and dictation tools for various professions like the medical field are also present. These systems are also used for applications like speech enhancement. The applications related to noise reduction, processing of degraded speech, bandwidth reduction and interference reduction come under this category. Other similar applications such as waveform coding, distortion compensation and multiplexing are also present. Other command and control applications include avionics, battle management, interface to computer systems, and database management [3].

3. Methodology:

Figure 6 shows a block diagram of implemented Audio-Visual Speech Recognition System.



Input Stream
Video
Image frames for Lip Reading
Visual Feature Extraction using DCT
Audio Signal
Silence Removal and End-Point Detection
Audio Feature Extraction using MFCC
Dynamic Time Warping and Normalization
Audio Visual Feature Fusion
Classification

Figure 6: Block Diagram of proposed Audio-Visual Speech Recognition System

As shown in the block diagram, audio and video parts of each utterance from the database were separated before carrying out the feature extraction. The software called ‘Any Video Converter’ was used for separating these audio and video parts. After separating the audio and video components, the video stream was used for visual feature extraction, and the audio stream was used for audio feature extraction.

3.a. Audio Feature Extraction:

Before carrying out the feature extraction from the speech samples, a silence removal and end-point detection step was implemented for better efficiency.

3.a.1 Silence removal and end-point detection:

The method used in proposed algorithm mainly consisted of Probability Density Function (PDF) of background noise and a Linear Pattern Classifier. The voiced part of sample was classified from the silence/unvoiced part using this classifier. As mentioned in [29], the results obtained from this algorithm are better than Zero Crossing Rate (ZCR) and Short Time Energy (STE) methods. As explained in [29], this method consists of 5 steps as follows:

In the first step, the mean and standard deviation of first 1600 samples of the audio were calculated with a sampling rate of 16000 samples per second, using Equations (21) and (22).

$$\mu = \frac{1}{1600} \sum_{i=1}^{1600} x(i)$$

-------(21)

$$\sigma = \sqrt{\frac{1}{1600} \sum_{i=1}^{1600} (x(i) - \mu)^2}$$

-------(22)

Where, μ is the mean and σ is the standard deviation. These two values are used to characterize the background noise.

In the second step, Mahalanobis distance was calculated for every sample using Equation (23).

$$Mahalanobis\ distance\ = \frac{|x\ -\ \mu|}{\sigma}$$

-------(23)

The Gaussian Probability density is described using a bell-shaped curve and it is calculated using mean μ and variance σ^2 [29]. This is usually written as, $\mathrm{p(x)} \sim \mathrm{N}(\mu\ \sigma^2)$ and it is read as, x distributed normally with mean μ and variance σ^2 [29]. In this curve, the peak occurs at $x =\ \mu$ and width is proportional to standard deviation, σ. The probabilities follow the Equation (24):

$$\Pr[|\mathrm{x}\ -\ \mu|]\ \leq\ \sigma\ \ =\ 0.68$$

$$\Pr[|\mathrm{x}\ -\ \mu|]\ \leq\ 2\sigma\ \ =\ 0.95$$

$$\Pr[|\mathrm{x} - \mu|] \leq 3\sigma = 0.997$$

-------(24)

As shown in Equation (24), 99.7% of the Gaussian distribution is in the range of

$|\mu|\ \leq\ 3$. Hence, the Mahalanobis distance was then checked whether it was greater than 3. If it was greater than 3, the sample was treated as voiced sample, otherwise it was treated as silence/unvoiced. This was done because the threshold of 3 rejects the samples up to 99.7% according to Equation (24) and hence it accepts only the voiced samples [29].

The audio was divided into 10ms non-overlapping windows. In each window, the Mahalanobis distance was calculated for each sample.

In the third step, all the voiced samples (as determined by Mahalanobis distance) were marked as 1 and all the unvoiced samples were marked as 0. In step four, each window would then be classified as voiced or unvoiced based on a majority of samples within that window. If the majority of samples (>50%) were classified as voiced (Mahalanobis distance), then the window is classified as voiced; otherwise, it was classified as unvoiced.

In the fifth step, parts of the signal that include voiced speech were collected together for further processing with unvoiced sections removed.

3.a.2 Feature Extraction:

Based on information in literature, Mel Frequency Cepstral Coefficient (MFCC) was used in the proposed system. This technique was chosen to take advantage of the frequency bands, which are positioned logarithmically on the Mel scale in MFCC. This approximates the human auditory system's response more closely than other techniques. Also, based on the comparisons in made in [3], it was observed that, MFCC has better performance as compared to other methods used for feature extraction. Figure 7 shows the block diagram for MFCC.

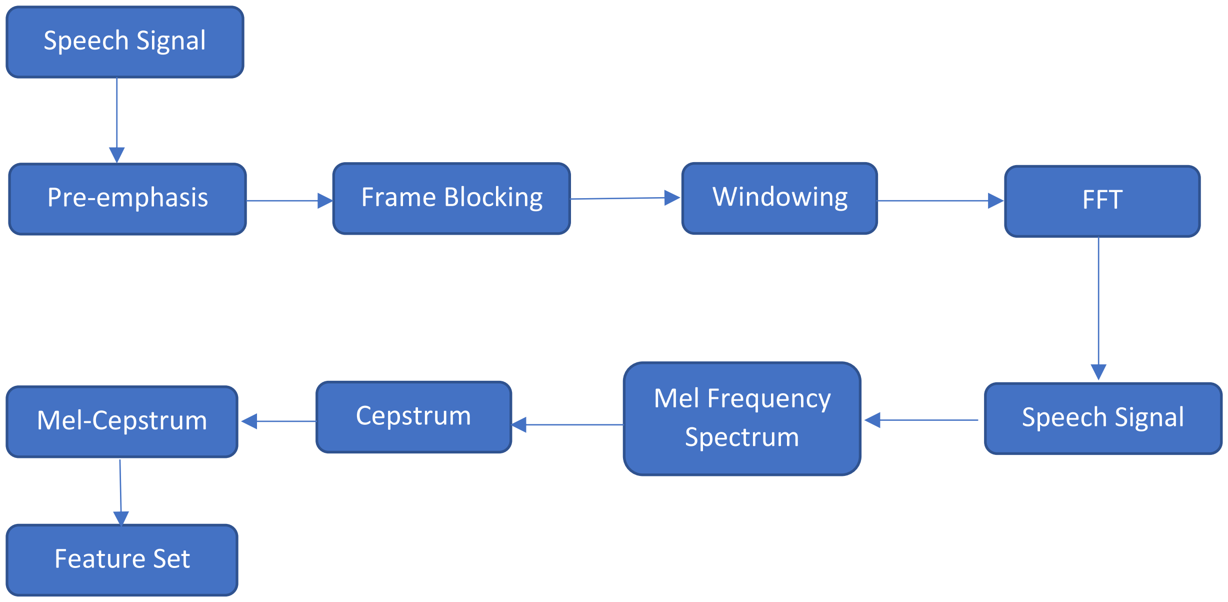


Figure 7: Audio Feature Extraction using MFCC Technique

***Pre-emphasis*:**

In the first stage of pre-emphasis, the amount of energy in the high frequencies is boosted. Considering the spectrum for voiced segments like vowels, more energy is present at lower frequencies than the higher frequencies. This energy drop across frequencies is due to the nature of glottal pulse. If this high energy frequency is boosted, then these higher formants become more available for feature extraction, which in turn improves the performance of the system.

A first order high-pass filter was used for carrying out this pre-emphasis. For the input signal $x[n]$, the filter equation is given as:

$$y[n] = x[n] - ax[n-1]$$

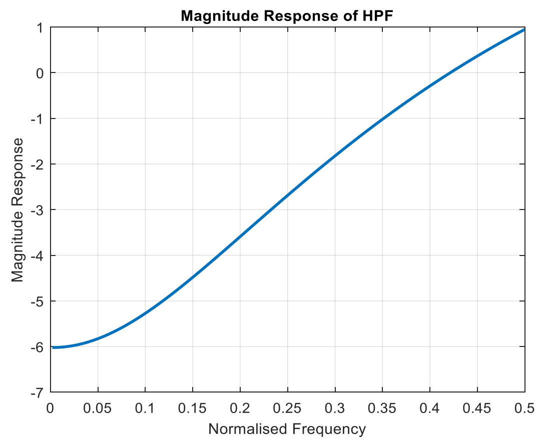
-------(25)

Where, $0.9 \leq a \leq 1.0$. Its transfer function is given as:

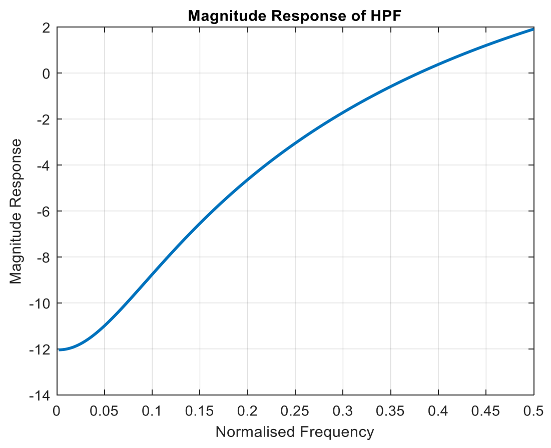
$$H(z) = 1 - az^{-1}$$

-------(26)

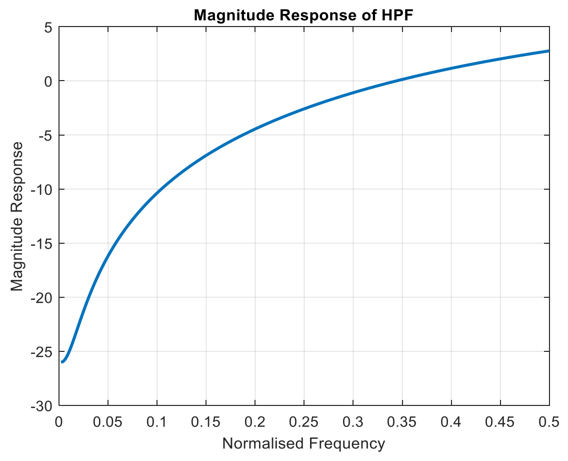
Figure 8 shows the magnitude response of this high pass filter for different values of a. As shown in the figure, for $a = 0.95$, the gain is 6db/octave. A filter with this value, as shown in Figure 8c, was used for the final implementation.



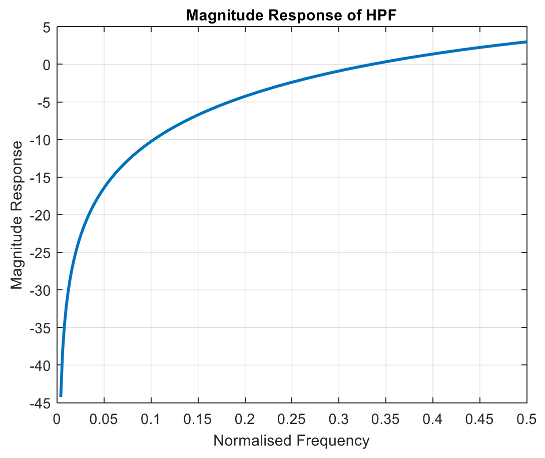
(a)



(b)



(c)



(d)

Figure 8: Magnitude response of pre-emphasis filter for different values of a
(a) a = 0.5 (b) a = 0.75 (c) a = 0.95 (d) a = 1

***Windowing*:**

Since the spectrum of every speech signal changes quickly, the spectral features are not extracted from the entire utterance. Speech is a non-stationary signal; hence, its statistical properties are not constant across time. Because of this, speech is windowed to extract the spectral features. For a small window, it can be assumed that the signal is stationary.

The three important parameters of windowing are, wideness of the window in milliseconds, the offset between successive windows, and the shape of the window. Every window of the speech signal is called a frame and is paired with images. Frame size is the number of milliseconds in the frame and frame shift is the number of milliseconds between left edges of successive windows. In the proposed system, frame size of 25ms and frame shift of 15ms was used. The signal was extracted using Equation (27).

$$y[n] = w[n]s[n]$$

-------(27)

Where, $s[n]$ is the value of signal at time sample n and $w[n]$ is the value of window at time sample n.

Hamming window was used during the implementation. This window avoids the discontinuities in the signal as it shrinks the values of signal towards zero at window boundaries. Equation (28) gives the value of Hamming window.

$$w[n] = 0.54 - 0.46\,cos(2\pi n/(L-1), \qquad if\ \ 0 \leq n \leq L-1$$

$$w[n] = 0, \qquad\qquad otherwise$$

-------(28)

Where, L is the window length.

Figure 9 shows the curves of generalized hamming window for different values of α.

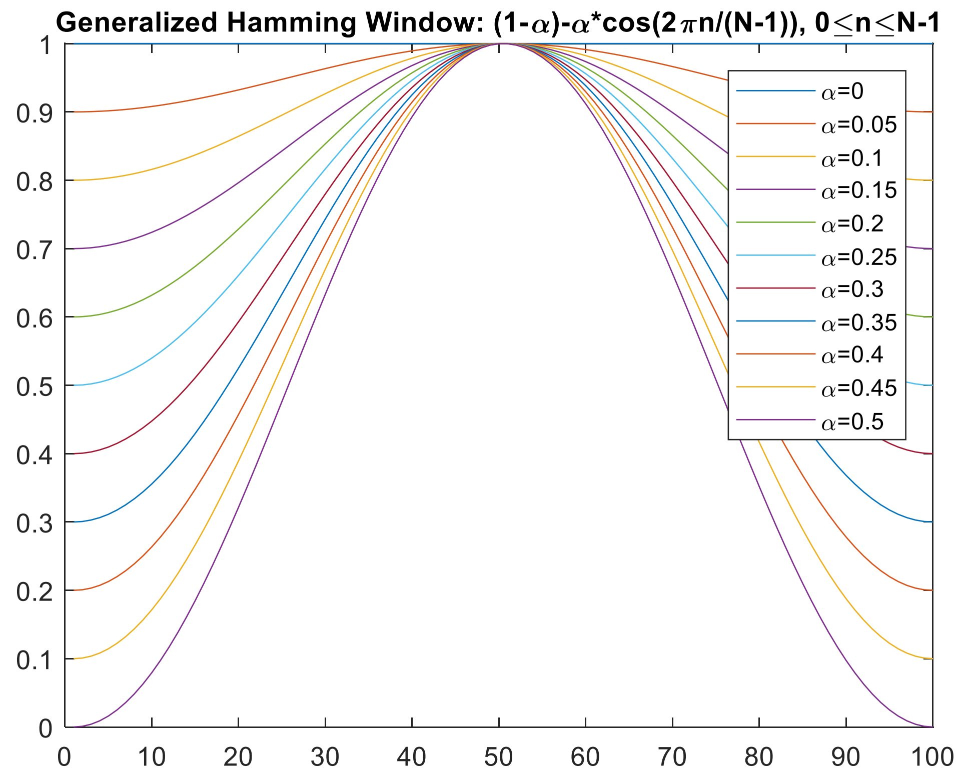


Figure 9: Hamming window curves

As explained by Equation 28, the value of α was chosen to be 0.46.

***Discrete Fourier Transform*:**

Discrete Fourier Transform (DFT) was used for extracting the spectral information from the windowed signal. The windowed signal $x[n]$ was input to the DFT. The output of DFT represented the magnitude and phase of the frequency component in the original signal. Fast Fourier Transform (FFT) is the commonly used algorithm for computing DFT. For this system, a 512-point FFT was used with a frequency resolution of 31.25 Hz/bin.

***Mel Filter Bank and log*:**

FFT yields information about the amount of energy at each frequency band. Human hearing is less sensitive at higher frequencies (above 1000 Hz) [28]. This property of human hearing was used for feature extraction. The frequency output of DFT was warped onto the mel scale in this MFCC feature extraction. Mel is the unit of pitch. It is defined that any pair of sounds which are perceptually equidistant in speech are separated by an equal number of mels. The mel frequency is calculated from the raw acoustic frequency using Equation (29).

$$mel(f) = 1127\, ln(1 + (f/700))$$

-------(29)

A bank of filters for collecting the energy from each frequency band was created during MFCC computation. Total 26 filters were used to calculate the filter bank. Figure 10 shows this mel bank of triangular filters.

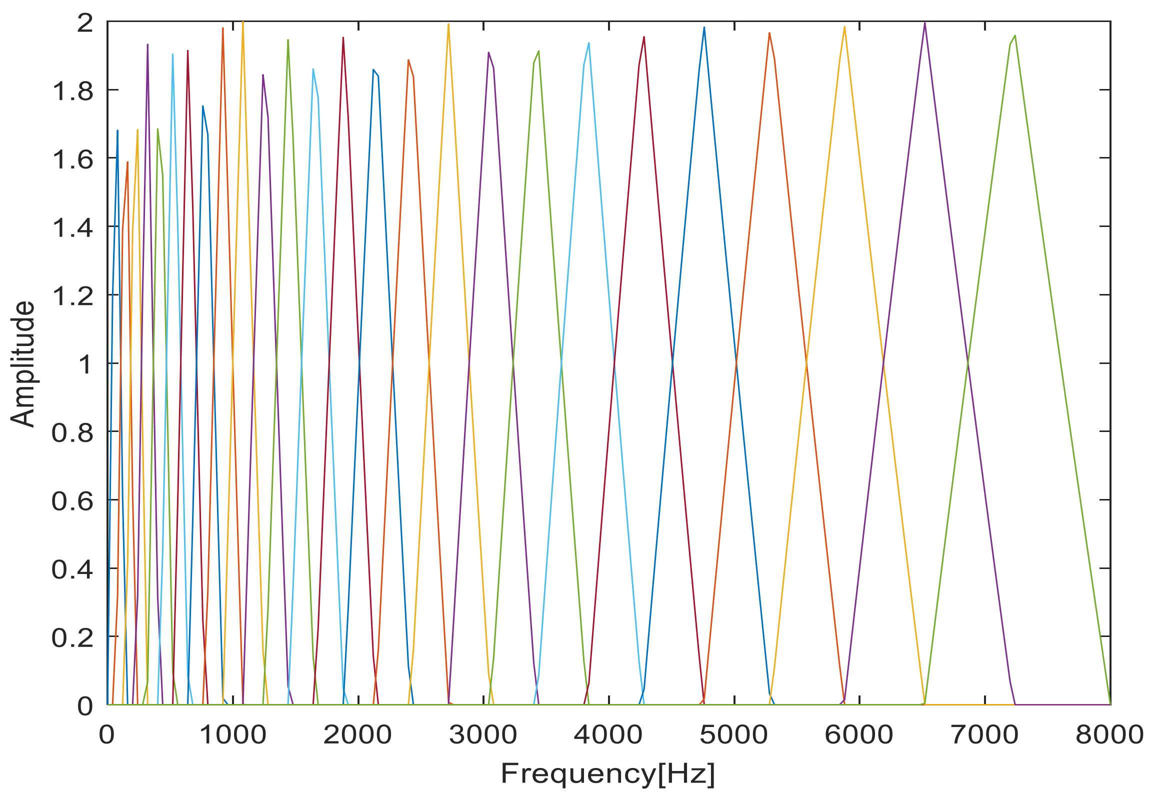


Figure 10: Mel Bank Filters

In this filter bank, 10 filters were spaced linearly below 1000 Hz and other filters were spaced logarithmically above 1000 Hz. This was done because, the mapping between frequency in Hertz and the mel scale is linear below 1000 Hz, and it is logarithmic above 1000 Hz [37]. Then, the log of all mel spectrum values was calculated as feature estimates. The result is less sensitivity to input variations.

The Cepstrum:

When a glottal source waveform of a specific fundamental frequency is passed through the vocal tract, a speech waveform is created. All the characteristics of this glottal pulse are not important for speech recognition. The exact position of the vocal tract is the most important information. Cepstrum is a tool that can be used to accomplish this. Cepstrum is defined as the inverse DFT of the log magnitude of the DFT of a signal.

In the proposed system, 26 mel bank filters were used for calculating 13 coefficients. Mel Frequency Cepstral Coefficients (MFCC) were calculated using Equation (30).

$$Cepstrum = dct\left[log\left(abs\big(X(k)\big)\right)\right]$$

-------(30)

Here, DCT was used instead of IFFT for computational efficiency. The MFCCs were computed by integrating spectral coefficients in each triangular frequency. The 26 values correspond to 26 filters accordingly. As observed in [43] [48], the recognition performance obtained after using first 10 to 13 coefficients was better than any other sub-band frequency range. Also, use of DCT decorrelates the features [46] and most of the information contained in the signal is accumulated in DCT at lower order coefficients [46]. The excitation information or the periodicity in the audio waveform is represented by higher order DCT coefficients, whereas, in this experiment, vocal tract shape or smooth spectral shape were more important [47]. Consequently, for MFCC extraction in this system, only the first 13 values were used [44] [45]. Figure 11 shows the plot of magnitude spectrum of a frame data and magnitude spectrum of MFCC coefficients calculated for the same frame.

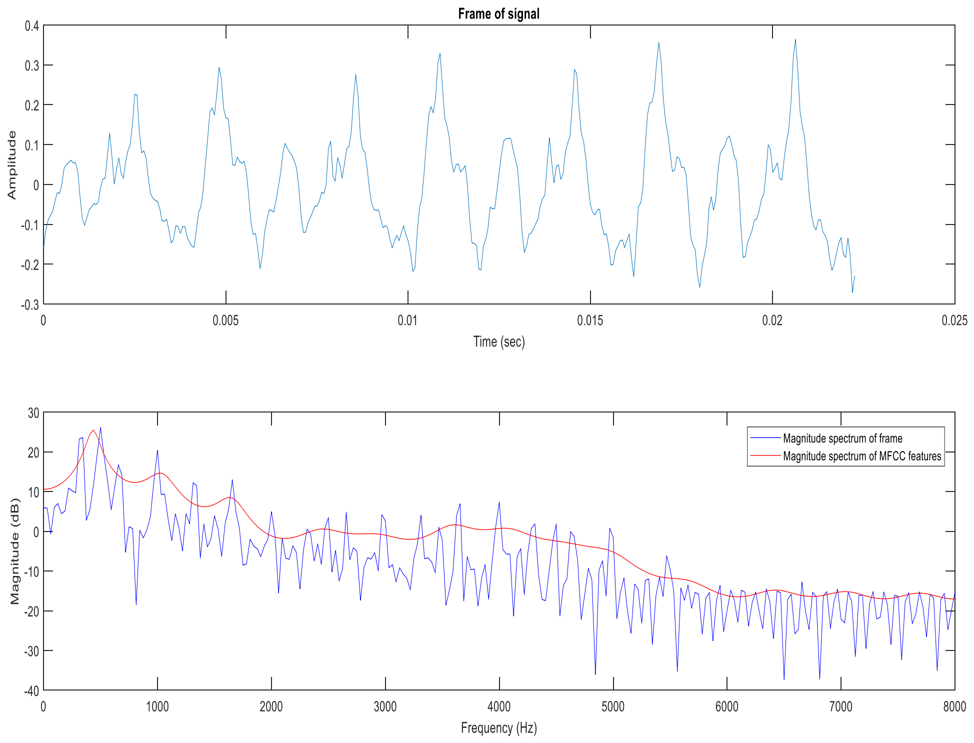


Figure 11: Magnitude spectrum of frame and MFCC features

From these 13 values of MFCC coefficients, delta and delta-delta coefficients were calculated using Equation (31).

$$d_t = \frac{\sum_{n=1}^{N} n(C_{t+n} - C_{t-n})}{2\sum_{n=1}^{N} n^2}$$

-------(31)

Here, d_t is the delta coefficient, t is the frame, C_{t+n} and C_{t-n} are the static coefficients. Value of N was selected as 2. Delta-delta coefficients were calculated using the same formula but by applying it on delta coefficients. In this way, for every frame of every speech sample, 39 coefficients (13 MFCC + 13 Delta + 13 Delta-delta) were calculated.

3.a.3 Dynamic Time Warping:

Dynamic Time Warping is a distance algorithm which is used for locally stretching or shrinking the time series before applying the classification technique [38].

There are two-time series, Q and C. Where, $Q = q_1, q_2, \ldots, q_i, \ldots, q_n$ = Query sequence. And, $C = c_1, c_2, \ldots, c_j, \ldots, c_m$ = Candidate sequence. Here, n and m are the number of frames in Q and C respectively. All the frames in both the sequences contain equal number of features in it.

To align these sequences, a matrix of size n by m is constructed such that ith and jth element of matrix contains the distance $d(q_i, c_j)$ between the two points, q_i and c_j. This distance is calculated using Equation (32).

$$\mathrm{d}(q_i, c_j) = (q_i - c_j)$$

-------(32)

The distance matrix is calculated using Equation (33):

$$D(\mathrm{i},\mathrm{j}) = d(\mathrm{i},\mathrm{j}) + \min\{\, D(\mathrm{i},\mathrm{j}-1)\,, D(\mathrm{i}-1,\mathrm{j})\,, D(\mathrm{i}-1,\mathrm{j}-1)\,\}$$

$$d(\mathrm{i},\mathrm{j}) = \text{Euclidean Distance} = |\text{query(i)} - \text{reference(j)}| = \text{Local Distance}$$

$$D(\mathrm{i},\mathrm{j}) = \text{Global Distance} = \text{Total Summation of all the local distances}$$

-------(33)

Here, each matrix element, (i,j) belongs to alignment between the points q_i and c_j.

Warping Path:

The set of matrix elements that defines the mapping between Q and C is called as the warping path, W. The kth element of W is defined as, $W_k = (i, j)$ [38]. So, we have,

$W = w_1, w_2, \ldots, w_k \ldots, \; w_K \; \max(m,n) \leq K < m+n-1$

After finding this warping matrix, the optimal warping path is searched. This warping path is related to following constraints [38]:

1. Boundary conditions: $w_1 = (1,1)$ and $w_k = (m,n)$, the warping path should start and finish in the diagonally opposite corner cells of the matrix.
2. Continuity: If, $w_k = (a,b)$ then, $W_{k-1} = (a',b')$. Here, $a-a' \leq 1$ and $b-b' \geq 0$. The allowable steps in the warping path are restricted to adjacent cells because of this constraint.
3. Monotonicity: If, $w_k = (a,b)$ then, $W_{k-1} = (a',b')$. Here, $a-a' \geq 0$ and $b-b' \geq 0$. The points in W are forced to be monotonically spaced in time using this constraint.

These conditions are satisfied by many warping paths but the path which minimizes the warping cost is calculated. This cost is calculated using Equation (34):

$$\mathrm{DTW}(q, C) = \min \left\{ \sqrt{\sum_{k=1}^{k} w_k} \Big/ k \right.$$

-------(34)

In the implemented system, this dynamic time warping was carried out after calculating the MFCC features of audio samples in the training dataset. The number of frames extracted after silence removal and end-point detection were different for all the audio samples. Hence, the number of MFCC features calculated for every audio sample were also different. A cell matrix was formed with all the number of frames related to audio samples with utterance ‘zero’. Then, the median of these number of frames was calculated. The audio sample which was having the number of frames equal to this median, was selected as the reference sample. All the other audio samples with utterance ‘zero’ were warped against this reference sample. Same procedure was carried out on all the other digits from ‘one’ to ‘nine’. In this way, in total ten (one matrix for each digit) training feature matrices were formed.

3.b Visual Feature Extraction:

The frame rate of all the videos collected was 25 frames per second. Given that each video has a duration of 2 seconds, each video was comprised of 50 frames.

3.b.1 Region of Interest(ROI) detection and lip localization:

For Region of Interest(ROI) detection and lip localization, the Viola-Jones algorithm was used. As explained previously, this algorithm works very efficiently for detecting faces and face components based on Haar-like features. These features are comprised of two edge features which are line features and rectangle features [15]. In this algorithm, the images are classified using simple features instead of pixels directly because, such systems are faster than pixel-based systems [15]. As explained in [15], there are three types of features used, two-rectangle features, three-rectangle features, and four-rectangle features. The value of the two-rectangle features is computed by taking the difference between the sum of pixels within two rectangles. The value of three rectangle features is calculated by taking difference between the sum of two outside rectangles and sum in the center rectangle. The value of four-rectangle feature is calculated by taking difference between diagonal pairs of rectangles.

These rectangle features are calculated using integral image. Equation (35) is used for calculating integral image at location (*x*,*y*):

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

-------(35)

Where, *ii*(*x*,*y*) is the integral image and *i*(*x*, *y*) is the original image. Equations (36) and (37) are used for calculating the integral image with one pass of original image.

$$s(x,y) = s(x,y-1) + i(x,y)$$

-------(36)

$$ii(x,y) = ii(x-1,y) + s(x,y)$$

-------(37)

Here, s(x,y) is the cumulative row sum, s(x,-1) = 0 and ii(-1,y) = 0.

After these rectangle features are calculated, the AdaBoost learning algorithm is used for classification. This was used for boosting the performance of simple learning algorithm (also called weak learning algorithm) [15]. Out of many rectangle features calculated from an image, very small number of features should be combined to get an efficient classification. Weak learning algorithm is good for completing this task [15]. The equation for weak classifier is as follows:

$$h_j(x) = \begin{cases} 1 & if\ p_j f_j(x) < p_j \theta_j \\ 0 & otherwise \end{cases}$$

-------(38)

Where, $h_j(x)$ is the weak classifier, f_j is the feature, p_j is the parity indicating direction of inequality sign, and θ_j is the threshold. The optimal threshold classification function is determined by the weak learner. In this way, a single rectangle feature is selected by this algorithm, which is best for separating positive and negative examples.

This ROI detection and lip localization technique was applied on all the 50 frames formed. All the 50 lip images extracted from every video were resized to 64 x 64. Feature extraction was then carried out on all these images.

3.b.2 Feature Extraction:

Many popular algorithms used for visual feature extraction have been discussed in the previous sections. The performance of all these algorithms was compared in [34] and [14]. Experimental results found that the image transform based visual features work significantly better than lip contour-based features [34]. This is because most of the speech-reading information is present in the oral cavity which cannot be captured by lip contours. When the image transform based algorithms were compared together, it was found that the DCT based technique works better than other techniques. Table 1 shows the overall summery [14]. As mentioned in the table, model based, or lip contour-based techniques are mainly useful when translational, rotational or scaling invariance is present within the database. Still their performance is low as compared to image pixel-based techniques. Considering all these things, the DCT based technique was used for visual feature extraction purpose in the proposed system.

Model	Example	Distortion	Lip Extraction	Performance	Limitation
Image-Based	DCT, DWT, PCA	Real Environment	Visual only	High DCT is better than others	Restricted to illumination, mouth rotation, dimensionality
Model-Based	ACM (snakes), ASM, AAM Deformable templates	Noise and channel	Acoustic and visual	Low but robust, invariant to translation, rotation, scaling and illumination	Inner outer lip contours, color of skin and lip matched, computationally expensive

Table 1: Comparison of Feature Extraction Techniques for Lip-Reading [14].

The ROI image of size 64x64 was divided into 16 non-overlapping blocks of 16x16 and then DCT was applied to each of these blocks. Sixteen transform coefficients were calculated using Equation (39), where *S* is a size *n* x *m* image, *u* = 0, 1, 2, 3…, *n* and *v* = 0, 1, 2, 3…., *m*.

$$S(u,v) = \frac{2}{\sqrt{nm}} C(u)C(v) \sum_{y=0}^{m-1} \sum_{x=0}^{n-1} S(x,y) \cos\frac{2(x+1)u\pi}{2n} \cos\frac{2(y+1)u\pi}{2m}$$

-------(39)

And, $C(u) = 2^{-1/2}$ for $u = 0$

$= 1$ otherwise.

Then energy coefficient $E(i)$ for each block is computed using Equation (40).

$$E(i) = \sum_{j=1}^{16} \sum_{k=1}^{16} S(j,k)$$

-------(40)

Where, i = 1, 2, 3, 4…, 16 and S is the transform coefficient. In this way, 800 coefficients (16*50) were calculated from every video. All these energy coefficients were stored in one vector, which was the targeted vector of visual features.

3.c Fusion and Classification:

The feature vector created from the videos consisted of 800 coefficients, and the feature vector created form audio samples consisted of 39 coefficients per frame of audio every sample. These feature vectors were concatenated to form a vector of features for every video in the database.

The Support Vector Machine (SVM) was used for classification purpose. This method was chosen due to its mathematical simplicity, and it avoids over-fitting [41]. As mentioned in [41], SVM is also well suited to classifying the MFCC features. In case of binary SVM, the optimal hyperplane is found such that it divides the two classes [36]. The distance between the data points and the hyperplane, which is refered to as the margin, is calculated. After hyperplane and margins are calculated, the class is chosen such that, it classifies the test datum with maximum margin [36].

Consider, training vector, $x_i \epsilon R^d$, $i = 1,\ldots l,$ and the label vector, y ϵ {1,-1}^{1}

Then, this SVM technique requires to find an optimum solution for Equation (41):

$$\underset{w\epsilon H, b\epsilon R, \xi_i \epsilon R}{min} \quad \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i$$

$$\text{subject to} \quad y_i(w^T \varphi(x_i) + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0, \mathrm{i} = 1, \ldots, \mathrm{i}$$

-------(41)

Here, w is the weight vector, C is the regularization constant and φ is the mapping function used for projecting the training data onto a suitable feature space H. This optimization problem is solved using equation (42):

$$\min_{\alpha \epsilon R} \quad \frac{1}{2}\alpha^T(\boldsymbol{K} \bullet (\boldsymbol{y}\boldsymbol{y}^T))\,\alpha - \boldsymbol{e}^T\alpha$$

$$subject\ to \quad \mathbf{0} \le \alpha \le C_{\boldsymbol{e}}, y^T\alpha = 0$$

-------(42)

Here $\boldsymbol{e}$ is a vector of all 1s, $\boldsymbol{K}$ is the kernel matrix and $\bullet$ is called as the Hadamard-Schur product [36].

To apply this binary SVM classifier on multiple classes, an approach called One-Versus-Rest was used in the implemented system. In this approach, SVM constructs, K separate classifiers for k-class classification. Accordingly, in SVMs, the *j*th classifier yields the decision function expressed by Equation (43):

$$f_j(\mathrm{x}) = w_j^T\varphi(x) + b_j$$

-------(43)

Here, W_j and b_j are hyperplane parameters calculated during *j*th classifier, and $f_j(\mathrm{x})$ is the distance between *x* and margin of classifier *j*. During multiclass classification, observation is assigned to that class *j** which produces largest value in all the M classifiers [39]. Equation (44) is expressed for explaining this concept:

$$j* = \arg \overset{max}{j = 1 \ldots M} f_j(\mathrm{x}) = \arg \overset{max}{j = 1 \ldots M} w_j^T \varphi(x) + b_j$$

-------(44)

While carrying out the classification, every test sample was first time warped against all ten (0 to 9) the reference samples one by one. Then these ten-time warped versions of the test sample were classified against all the ten-training feature (0 to 9) matrices. After carrying out these ten classifications, in total ten values of j* were calculated based on all individual digit (0-9) classifications. The maximum of these ten j* values was used to determine the best match.

3.d. Testing:

As discussed in the introduction, the aim of this research is to compare the results of two standalone systems (A Lip Reading System and a Speech Recognition System) with a feature-level fusion-based system (An Audio-Visual Speech Recognition System). The database, which was used for this comparison is called ‘vVISWa’ (Visual Vocabulary of Independent Standard Words) [35] database. Another database which was used for the comparison was collected in the Grand Valley State University’s campus. Both these databases are comprised of frontal profile utterances of numerals/digits. The duration of all the videos collected was approximately 2 seconds.

The experiments were carried out using three combinations of these two databases. In the first combination, five male participants and five female participants were included from the ‘vVISWa’ database. Four utterances were recorded from these ten participants for each digit. Out of these four utterances, two utterances per person per digit were included in the training dataset, and the other two utterances were included in the testing dataset. In this way, both training and testing datasets consisted of two hundred videos (10*2*10) each.

In the second combination, videos collected from three male participants and two female participants were included in the training dataset. The testing dataset comprised of the videos collected from two male participants and three female participants. These participants in the testing dataset were different than that of the training dataset, but all of them belonged to the ‘vVISWa’ database. Just like that of combination one, two utterances per digit per person were included in combination two database. In this way, both training and testing datasets consisted of hundred videos (10*2*5) each.

In the third combination, the same training dataset of two hundred videos from the combination one was used. The videos collected from the Grand Valley State University's campus were included in the testing dataset. The participants included in this testing dataset were belonging to different nationalities and different ethnicities. The algorithm was tested against this diversified dataset to check the robustness of the system. This testing dataset consisted of five male participants and five female participants. One utterance per person per digit was recorded from these university campus participants. In this way, the testing dataset of combination three consisted of hundred videos (10*1*10).

To compare the performances of the stand-alone speech recognition system and the fusion based Audio-visual speech recognition system, a statistical analysis method called McNemar's test is used [42]. It is performed in Microsoft Excel using the results data obtained.

For performing this test, four values were calculated using Equation (45) based on data obtained from the results.

$$e_{00} = Number\ of\ examples\ misclassified\ by\ both\ system\ 1\ and\ system\ 2$$

$$e_{01} = Number\ of\ examples\ misclassified\ by\ system\ 1\ but\ not\ by\ system\ 2$$

$$e_{10} = Number\ of\ examples\ misclassified\ by\ system\ 2\ but\ not\ by\ system\ 1$$

$$e_{11} = Number\ of\ examples\ correctly\ classified\ by\ both\ systm\ 1\ and\ system\ 2$$

-------(45)

For this experiment, system 1 is the stand-alone speech recognition system, and system 2 is the fusion based audio-visual speech recognition system. After computing these four values, the value of the χ^2 test statistic is calculated using equation (46).

$$\chi^2 = \frac{(|e_{01} - e_{10}| - 1)^2}{e_{01} + e_{10}}$$

-------(46)

The value of the χ^2 test statistic is then compared with a χ^2 distribution with 1 degree of freedom. Based on this comparison, the null hypothesis is either rejected or accepted. In equations (47) and (48), μ_0 and μ_1 denote theoretical outcomes of the two classification systems. The null hypothesis, as shown in equation (47), is that both classifiers are the same. To accept this null hypothesis, the test statistic calculated in equation (46) would need to be greater than that of the distribution for the corresponding confidence level. Table 2 shows values calculated from a χ^2 distribution with 1 degree of freedom, which are used for comparison with the test statistic.

$$H_0 : \mu_0 = \mu_1$$

-------(47)

$$H_1 : \mu_0 \neq \mu_1$$

-------(48)

For example, if the value of the test statistic calculated with equation (46) was 3.40, then the null hypothesis could be rejected with a confidence of 93%. This is because 3.40 is greater than $\chi^2(\alpha, dof) = \chi^2(0.07,1) = 3.283$, where $\alpha = 1 - confidence\ interval$.

Confidence Level	**Alpha**	**Value of** χ^2 Distribution
80%	0.20	1.6424
81%	0.19	1.7176
82%	0.18	1.7976
83%	0.17	1.8829
84%	0.16	1.9742
85%	0.15	2.0723
86%	0.14	2.1780
87%	0.13	2.2925
88%	0.12	2.4173
89%	0.11	2.5542
90%	0.10	2.7055
91%	0.09	2.8744
92%	0.08	3.0649
93%	0.07	3.2830
94%	0.06	3.5374
95%	0.05	3.8415
96%	0.04	4.2179
97%	0.03	4.7093
98%	0.02	5.4119
99%	0.01	6.6349

Table 2: Confidence levels and corresponding values of χ^2 Distribution.

4. Results:

4.a. Speech Recognition results:

As discussed in the methodology section, before carrying out the feature extraction on the audio samples, silence removal and end-point detection algorithm was implemented on every sample. Figure 12 shows the speech signal waveform before silence removal and end-point detection, and Figure 13 shows the speech signal waveform after silence removal and end-point detection.

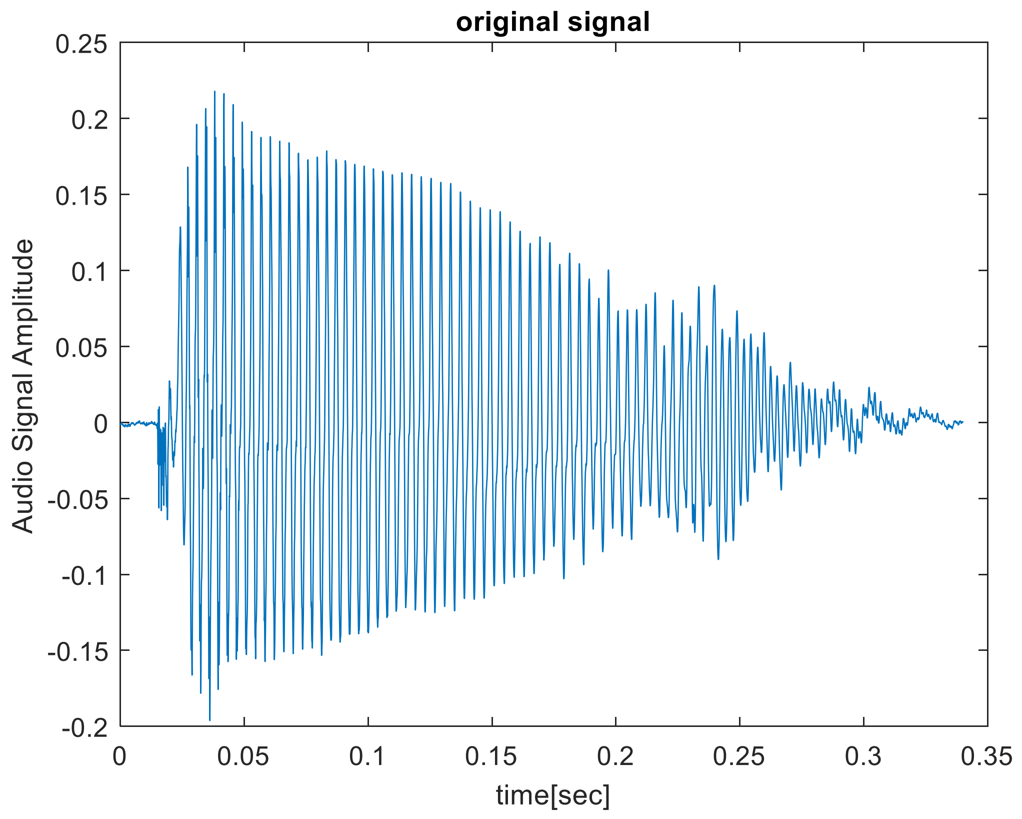


Figure 12: Speech signal waveform before silence removal and end-point detection

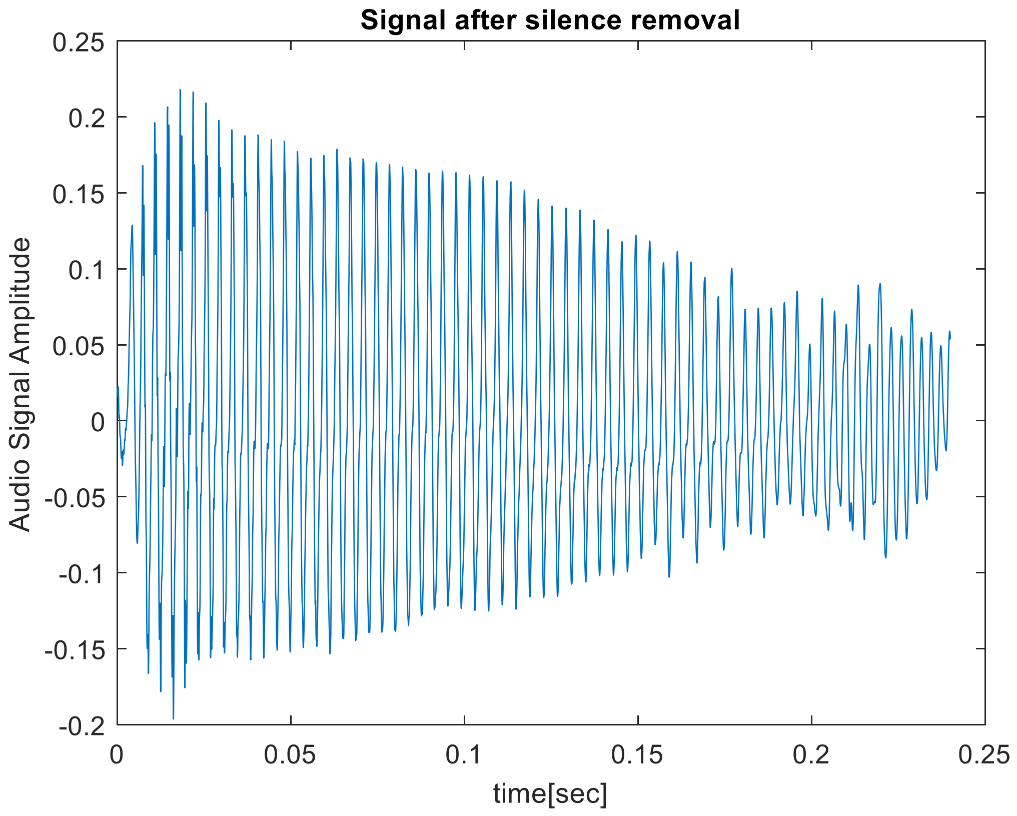


Figure 13: Speech signal waveform after silence removal and end-point detection

After implementing the silence removal and end-point detection algorithm, feature extraction was carried out on the signals. As explained in the methodology section, first step in the MFCC feature extraction was pre-emphasis. Figure 14 shows the waveform obtained after implementing the pre-emphasis step on the silence removed and end-point detected audio signal.

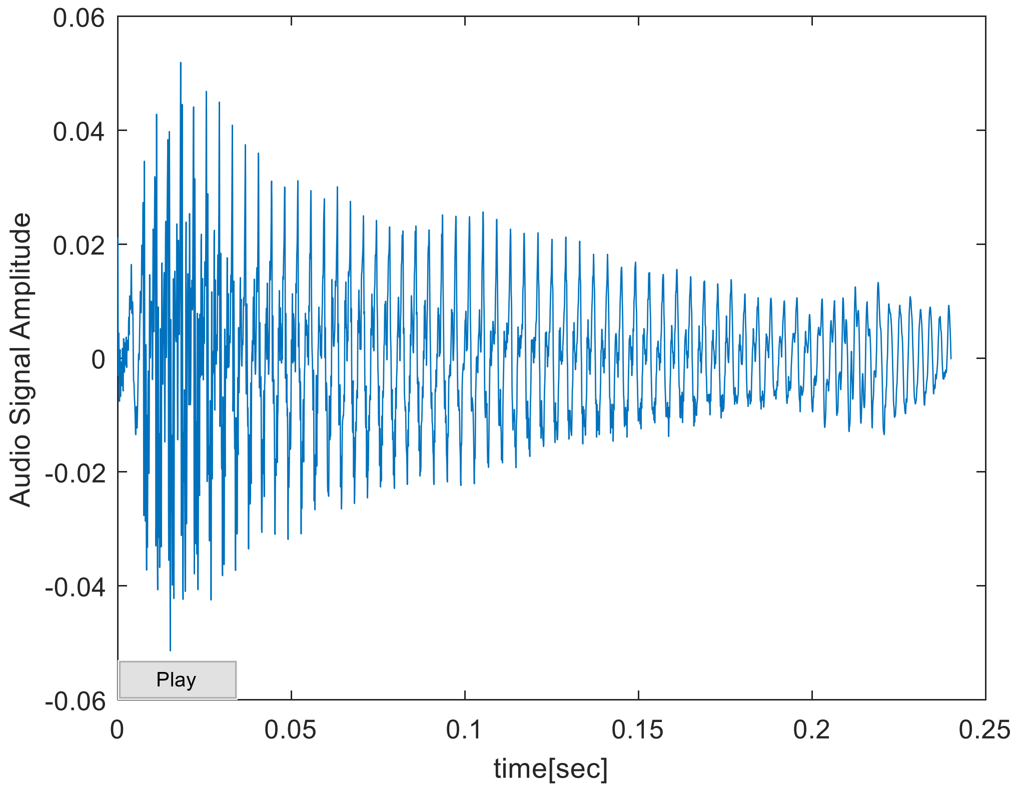


Figure 14: Effect of pre-emphasis

When the audio sample after pre-emphasis step was heard, it was observed that, it sounded sharper with a lower volume.

After pre-emphasis, steps of frame-blocking and windowing were implemented. For keeping the continuity in the first and last points in the frame, all the frames of signal were multiplied with a hamming window.

After windowing, FFT of the audio sample was calculated for extracting the spectral information from the windowed signal. When FFT is performed on a frame of signal, it is assumed that the signal within the frame is periodic and continuous. However, if the signal within the frame is not periodic and continuous, it can introduce some undesirable effects in the frequency response. Hence, each frame was multiplied by hamming window before implementing the FFT step. Figure 15 shows this effect of windowing on FFT.

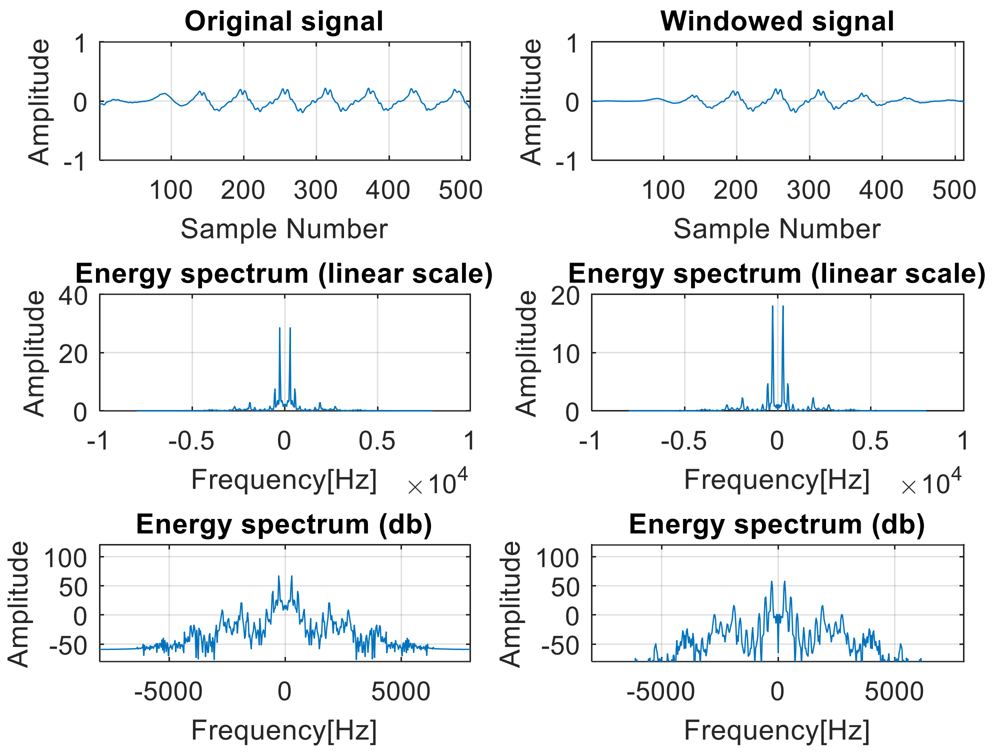
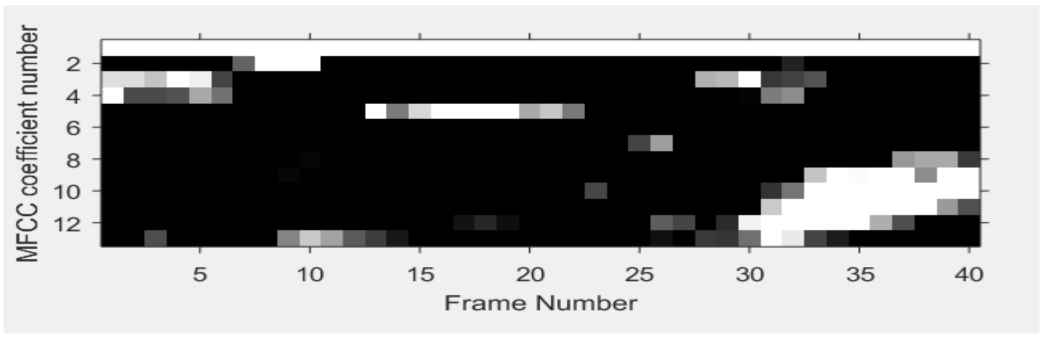


Figure 15: Effect of windowing on FFT

As shown in Figure 15, the peak in the frequency response of windowed signal was sharper and more distinct. It can also be seen that, the amplitude of side lobes in case of FFT of windowed signal was much lesser than that of original signal.

After this step, the frequency output of FFT was warped onto the mel scale for MFCC feature extraction.

Figure 16 shows the plot of MFCC coefficients for the audio sample of digits zero and one collected from one of the participants. Here the X-axis of every image indicates the different frame number and Y-axis denotes the MFCC coefficient number. The intensity of a pixel located at a point (x, y) indicates the value of MFCC coefficient number 'y' for a frame 'x'.



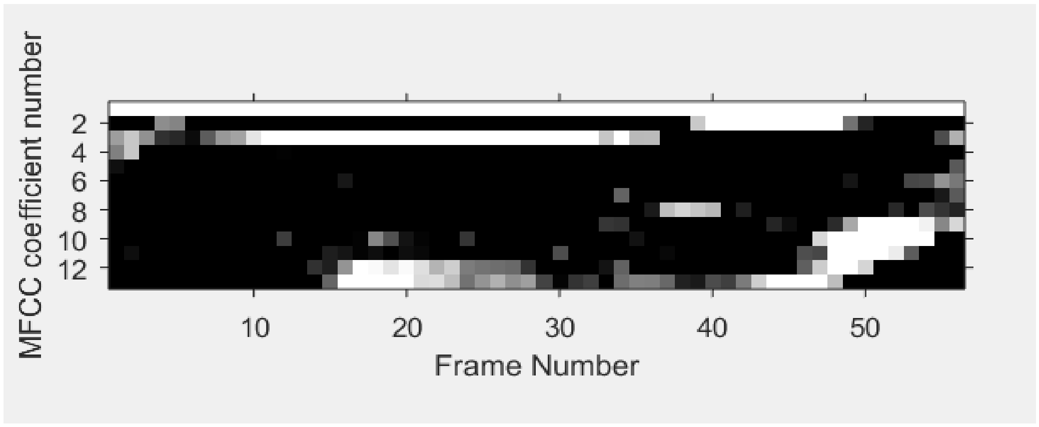


Figure 16: MFCC coefficients for utterance zero and one

As discussed in the methodology, Dynamic Time Warping was used after MFCC feature extraction for time alignment of signals. Figure 17 shows the result of an audio sample related to digit 'two' time warped against the reference signal related to digit 'two.' Figure 18 shows the result of an audio sample related to digit 'four' time warped against the same reference signal related to digit 'two.' For displaying these results, energies in the frequency domain for all the frames in both the signals were plotted against corresponding frame number. These energies of signals were plotted before and after implementing DTW.

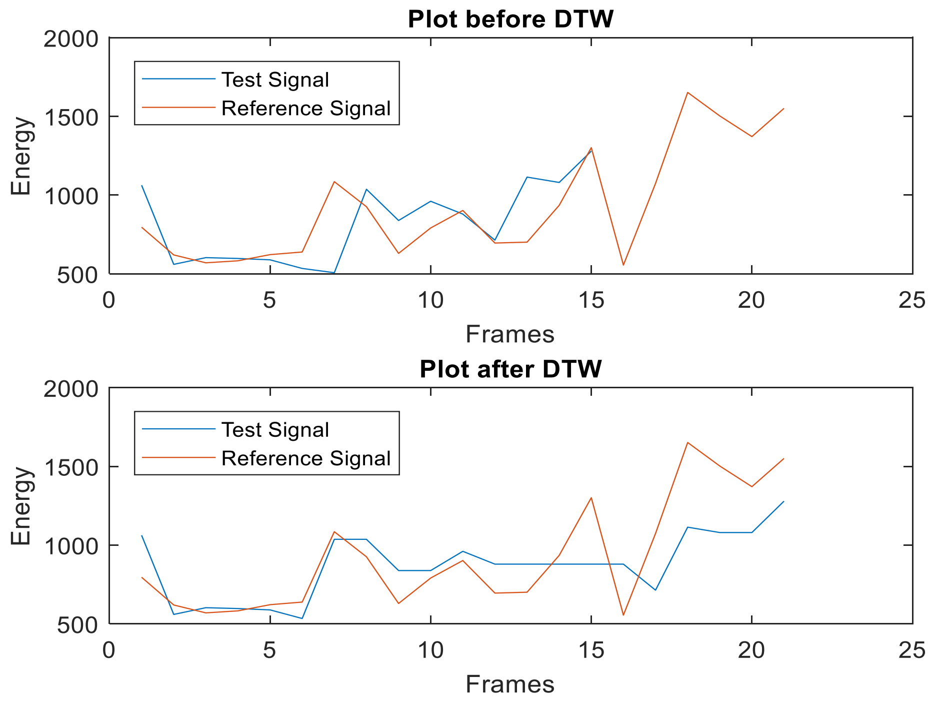


Figure 17: Energy plots before and after DTW [test signal of digit ‘two’, reference signal of digit ‘two’]

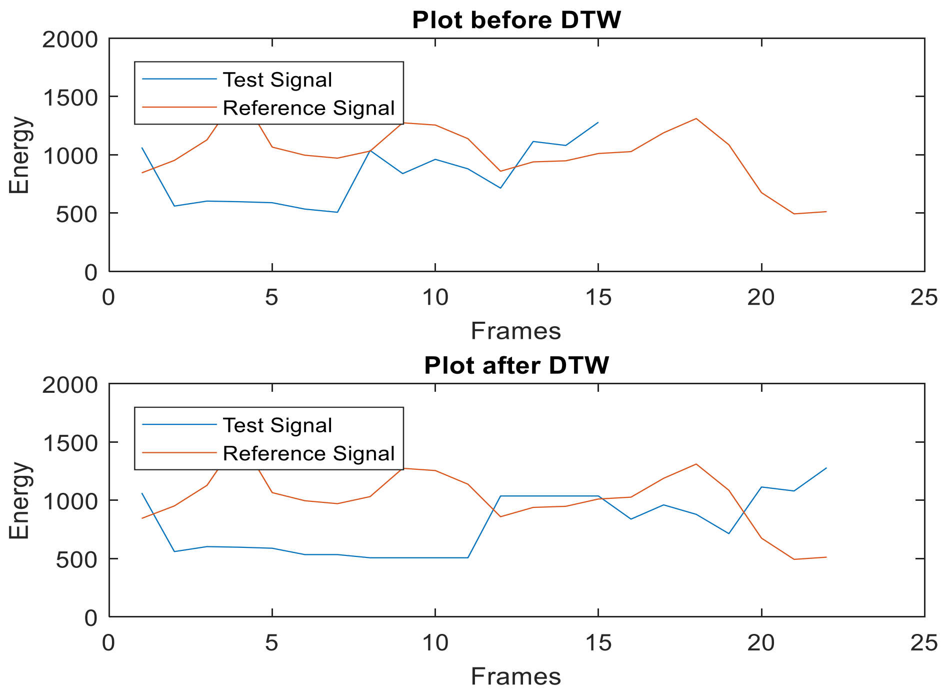


Figure 18: Energy plots before and after DTW [test signal of digit ‘four’, reference signal of digit ‘two’]

As shown in Figure 17 and Figure 18, the number of frames in both test signal and reference signal are different than each other. Both these signals were aligned together using Dynamic Time Warping technique such that the Euclidean distance between their points is the smallest. It can be observed in the plots obtained after DTW, that the number of frames in both the signals after are equal. Figure 19 shows the Dynamic Time Warping path plotted on the local similarity matrix for these two signals.

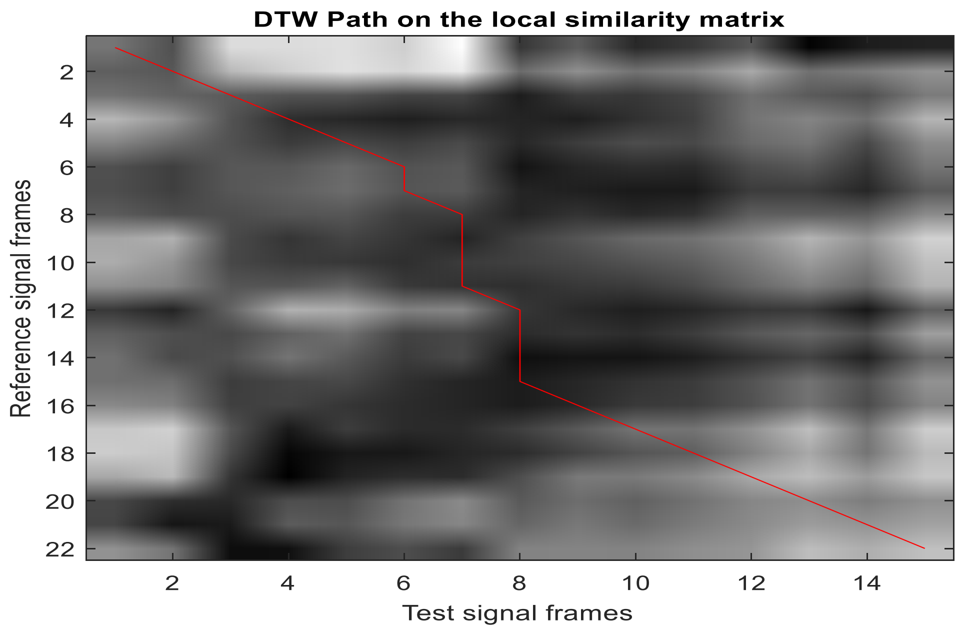


Figure 19: Dynamic Time Warping path

4.b. Lip Reading results:

Figure 20 and Figure 21 show the results obtained after applying the Viola-Jones technique for region of interest detection on the videos. The image shown in figure 19 is the 20th frame of a video. This video belongs to the database collected in the Grand Valley State University campus. As displayed in the image, face, eyes, nose and lips were detected as the regions of interest. Out of these regions, the lips area was taken out for further processing. Figure 20 shows the resized image of a lip from the same 20th frame of video.

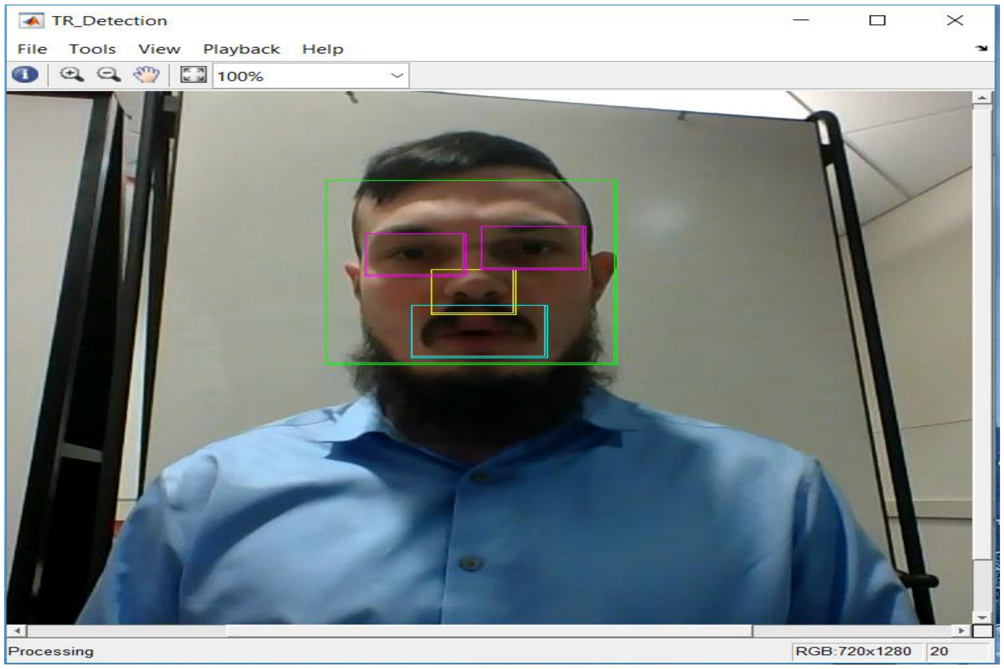


Figure 20: ROI detection in 20th frame of a video from GVSU campus Database

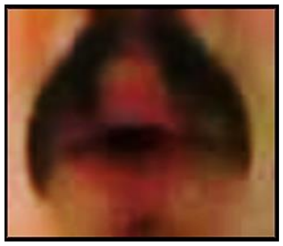


Figure 21: Resized extract of lip image from 20th frame of a video

Figure 22 shows the result obtained after applying the Viola-Jones technique for region of interest detection on the video belonging to the vVISWA database. As seen the Figure 21, all the videos belonging to vVISWA database were collected by keeping black background, whereas, the videos collected in the university campus had some other objects in the background along with the background of a white board. Irrespective of variable background, the Viola-Jones technique was able to detect the region of interest very efficiently on all the videos. It was also able to overcome the occlusions of facial hair and glasses on faces of some participants. The participants in the videos collected in the university campus are more diverse, hence face colors of participants in this database were different. The algorithm was still able to detect the ROIs correctly.

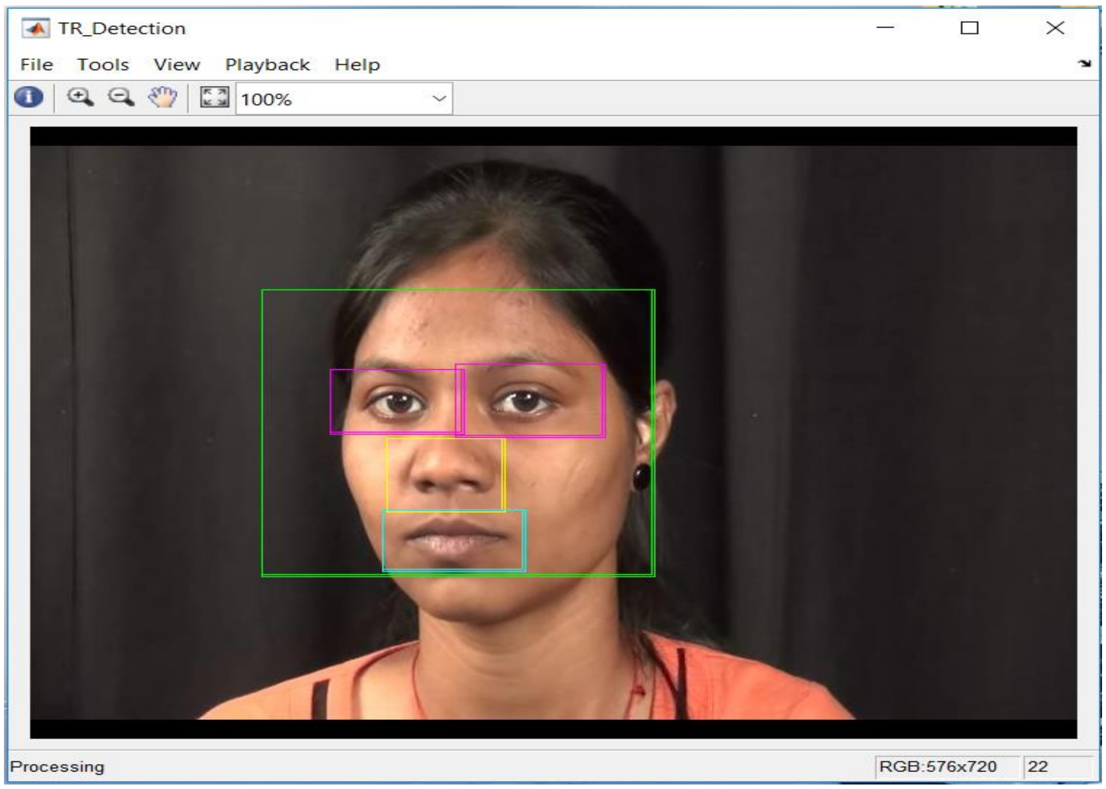


Figure 22: ROI detection in 24th frame of a video from vVISWA Database

The results obtained from all these three combinations of databases are as shown in Table 3:

Combination	Database	Visual only Recognition using DCT Features	Audio only Recognition using MFCC Features	Audio-Visual Speech Recognition
1.	vVISWA Dataset of 5 Males and 5 Females	53%	93%	96%
2.	vVISWA Dataset with different people in Training (3 Males, 2 Females) and Testing (2 Males and 3 Females) dataset	41%	87%	92%
3.	Training with vVISWA dataset of 10 people and testing with database collected in GVSU campus (10 people)	36%	78%	86%

Table 3: Results

As shown in Table 3, the results obtained from research work indicate significant improvement in the recognition rates due to fusion of audio and visual features. The results obtained from the algorithm using database combinations 1 and 2 are better than that of combination 3. Irrespective of the fact that ROI detection worked efficiently on the combination 3 participants with different races and face colors, recognition rates of the system were less than the other two combinations. The recognition rate of audio only stand-alone system was also less in combination 3. This is mainly because participants had different accents due to their different nationality or ethnicity. The lip movement done by these participants was also very different than the participants in combination 1 and 2.

	e_{00}	e_{01}	e_{10}	e_{11}	**Chi-Square value**
Database Combination 1	2	12	6	178	1.9444
Database Combination 2	5	8	3	79	2.1818
Database Combination 3	5	17	9	64	2.4231
Total of three combinations	12	37	18	321	6.5455

Table 4: McNemar’s Test Results

The performance of the two different classification systems were compared using the McNemar’s Test as described in the testing section of methodology, the results of which are shown in Table 4. From these results, it can be concluded that the performance of the stand-alone speech recognition system and the fusion based audio-visual speech recognition system were significantly different with a 98% confidence level. As a total, the fusion based audio-visual speech system correctly classified 37 utterances that were incorrectly characterized by the audio-only system. This is more than double the 18 utterances that were incorrectly characterized by the fusion system but correctly characterized by the audio only. Based on these values of $e_{01}$and e_{10}, it can be concluded that, the performance of fusion based audio-visual speech recognition system was better than standalone speech recognition system. The two different systems can also be analyzed based on performance on the three different databases (all with different characteristics that make them more or less challenging to characterize). For these subsets of results, it is more difficult to show that the two systems are different with at most 88% confidence and as little as 83% confidence. This is due to the limited number of outcomes that varied between the two classifiers.

5. Conclusion:

In this thesis, an audio-visual speech recognition system was implemented and tested using the videos of numeral digits. The videos were collected from participants belonging to different ethnicities to test the robustness of the system. The performance of the system was compared with two stand-alone systems (audio only and visual only recognition systems). The MFCC features were extracted from the audio samples and the DCT features were extracted from the lip movements in videos for classification purpose.

From the details discussed in the results section, it can be concluded that the implemented system was able to deliver a better recognition rate with the fusion of audio and video features as compared to the two stand-alone systems. If the performances of two stand-alone systems are compared together in all the three database combinations, the recognition rate of audio only speech recognition system was better than visual only recognition system.

In future work the performance in noisy environments may by combining other human modalities such as iris movements and hand gestures. A Support Vector Machine was used for classification. It was observed that, most of the execution time taken by MATLAB was mainly for carrying out the classification step. This execution time and overall performance of the system could be further tested using other computationally complex classification methods such as Neural Network based Hidden Markov Model or Random Forest Classifier.

This system was built and tested to recognize the numeral digits. This work can be further extended for applications such as voice-activated computers or voice-controlled driver's assistant systems in which more commands are required in the vocabulary. The database would then be required to be increased from digits to words or full sentences/phrases for that purpose.

Appendix A:

```
%-------Vikrant Satish Acharya--------------------------------------------
%-------EGR 696: Master’s Thesis: Fusion of audio and visual information for
implementing improved speech recognition system.-----------
%-------fusion.m---------------------------------------------------------

clear all;
close all;
clc;

%-----------------------------Speech-Training-----------------------------------

Testing_Words = {'Zero','One','Two','Three','Four','Five','Six','Seven','Eight','Nine'};
%Sample Frequency
fs=16000;
Thresh=0.3;
TR_row=0;
for TR_p=1:200
  %for TR_q=1:5
   TR_row = TR_row + 1;
   TR_filename='E:\VIKRANT\GVSU\Thesis(EGR 695)\codes\Speech
Recognition\Demo2\Train\';
   %TR_filename='E:\VIKRANT\GVSU\Thesis(EGR 695)\codes\Speech
Recognition\Code Pieces\';
   [TR_x,Fs]=audioread(strcat(TR_filename,num2str(TR_p),'.wav'));
   TR_samplePerFrame=floor(fs/100);
TR_bgSampleCount=floor(fs/5); %according to formula, 1600 sample needed for 8 khz
%calculation of mean and std
TR_bgSample=[];
for i=1:1:TR_bgSampleCount
    TR_bgSample=[TR_bgSample TR_x(i)];
end
TR_meanvalue=mean(TR_bgSample);
TR_std_dev=std(TR_bgSample);
%identify voiced or not for each value
for i=1:1:length(TR_x)
   if(abs(TR_x(i)-TR_meanvalue)/TR_std_dev > Thresh)
       TR_voiced(i)=1;
   else
       TR_voiced(i)=0;
   end
end
```

```
%identify voiced or not for each frame
%discard insufficient samples of last frame
TR_useful_samples=length(TR_x)-mod(length(TR_x),TR_samplePerFrame);
TR_frameCount=TR_useful_samples/TR_samplePerFrame;
TR_voiced_frameCount=0;
for i=1:1:TR_frameCount
  TR_cVoiced=0;
  TR_cUnVoiced=0;
  for j=i*TR_samplePerFrame-TR_samplePerFrame+1:1:(i*TR_samplePerFrame)
     if(TR_voiced(j)==1)
        TR_cVoiced=(TR_cVoiced+1);
     else
        TR_cUnVoiced=TR_cUnVoiced+1;
     end
  end
%mark frame for voiced/unvoiced
  if(TR_cVoiced>TR_cUnVoiced)
     TR_voiced_frameCount=TR_voiced_frameCount+1;
     TR_voicedUnvoiced(i)=1;
  else
     TR_voicedUnvoiced(i)=0;
  end
end
TR_reqd_signal=[];
for i=1:1:TR_frameCount
   if(TR_voicedUnvoiced(i)==1)
   for j=i*TR_samplePerFrame-TR_samplePerFrame+1:1:(i*TR_samplePerFrame)
         TR_reqd_signal= [TR_reqd_signal TR_x(j)];
   end
   end
end
  TR_data{TR_row} = TR_reqd_signal;
TR_signals={};
TR_iter = 0;
TR_signals=TR_data';
TR_Data = mfcc1(TR_signals,Testing_Words,16000);
%end
zero = TR_Data(1:10:191);
zero_x1 = zero(14);
zero_x2 = cell2mat(zero_x1);
zero_ref = (zero_x2)'; %ref
for rw = 1:3
   zero1 = zero(rw);
```

```
    zero2 = cell2mat(zero1);
    zero_test = (zero2)';
    SM = simmx(abs(zero_ref),abs(zero_test));
    [p,q,C] = dp(1-SM);
    C(size(C,1),size(C,2));
    zero_testi1 = zeros(1, size(zero_ref,2));
    for i = 1:length(zero_testi1); zero_testi1(i) = q(min(find(p >= i))); end
    zero_testx = pvsample(zero_test, zero_testi1-1, 128);
    zero4 = zero_testx(:);
    zero5 = (zero4)';
    zero_Feature_Vector(rw,:) = zero5(:);
end
one = TR_Data(2:10:192);
one_x1 = one(7);
one_x2 = cell2mat(one_x1);
one_ref = (one_x2)'; %ref
for rw = 1:3
    one1 = one(rw);
    one2 = cell2mat(one1);
    one_test = (one2)';
    SM = simmx(abs(one_ref),abs(one_test));
    [p,q,C] = dp(1-SM);
    C(size(C,1),size(C,2));
    one_testi1 = zeros(1, size(one_ref,2));
    for i = 1:length(one_testi1); one_testi1(i) = q(min(find(p >= i))); end
    one_testx = pvsample(one_test, one_testi1-1, 128);
    one4 = one_testx(:);
    one5 = (one4)';
    one_Feature_Vector(rw,:) = one5(:);
end
two = TR_Data(3:10:193);
two_x1 = two(9);
two_x2 = cell2mat(two_x1);
two_ref = (two_x2)'; %ref
for rw = 1:3
    two1 = two(rw);
    two2 = cell2mat(two1);
    two_test = (two2)';
    SM = simmx(abs(two_ref),abs(two_test));
    [p,q,C] = dp(1-SM);
    C(size(C,1),size(C,2));
    two_testi1 = zeros(1, size(two_ref,2));
    for i = 1:length(two_testi1); two_testi1(i) = q(min(find(p >= i))); end
    two_testx = pvsample(two_test, two_testi1-1, 128);
    two4 = two_testx(:);
    two5 = (two4)';
    two_Feature_Vector(rw,:) = two5(:);
end
three = TR_Data(4:10:194);
three_x1 = three(11);
three_x2 = cell2mat(three_x1);
three_ref = (three_x2)'; %ref
for rw = 1:3
    three1 = three(rw);
    three2 = cell2mat(three1);
    three_test = (three2)';
    SM = simmx(abs(three_ref),abs(three_test));
    [p,q,C] = dp(1-SM);
    C(size(C,1),size(C,2));
    three_testi1 = zeros(1, size(three_ref,2));
    for i = 1:length(three_testi1); three_testi1(i) = q(min(find(p >= i))); end
    three_testx = pvsample(three_test, three_testi1-1, 128);
    three4 = three_testx(:);
    three5 = (three4)';
    three_Feature_Vector(rw,:) = three5(:);
end
four = TR_Data(5:10:195);
four_x1 = four(12);
four_x2 = cell2mat(four_x1);
four_ref = (four_x2)'; %ref
for rw = 1:3
    four1 = four(rw);
    four2 = cell2mat(four1);
    four_test = (four2)';
    SM = simmx(abs(four_ref),abs(four_test));
    [p,q,C] = dp(1-SM);
    C(size(C,1),size(C,2));
    four_testi1 = zeros(1, size(four_ref,2));
    for i = 1:length(four_testi1); four_testi1(i) = q(min(find(p >= i))); end
    four_testx = pvsample(four_test, four_testi1-1, 128);
    four4 = four_testx(:);
    four5 = (four4)';
    four_Feature_Vector(rw,:) = four5(:);
end
five = TR_Data(6:10:196);
five_x1 = five(8);
```

```
five_x2 = cell2mat(five_x1);
five_ref = (five_x2)'; %ref
for rw = 1:3
    five1 = five(rw);
    five2 = cell2mat(five1);
    five_test = (five2)';
    SM = simmx(abs(five_ref),abs(five_test));
    [p,q,C] = dp(1-SM);
    C(size(C,1),size(C,2));
    five_testi1 = zeros(1, size(five_ref,2));
    for i = 1:length(five_testi1); five_testi1(i) = q(min(find(p >= i))); end
    five_testx = pvsample(five_test, five_testi1-1, 128);
    five4 = five_testx(:);
    five5 = (five4)';
    five_Feature_Vector(rw,:) = five5(:);
end
six = TR_Data(7:10:197);
six_x1 = six(5);
six_x2 = cell2mat(six_x1);
six_ref = (six_x2)'; %ref
for rw = 1:3
    six1 = six(rw);
    six2 = cell2mat(six1);
    six_test = (six2)';
    SM = simmx(abs(six_ref),abs(six_test));
    [p,q,C] = dp(1-SM);
    C(size(C,1),size(C,2));
    six_testi1 = zeros(1, size(six_ref,2));
    for i = 1:length(six_testi1); six_testi1(i) = q(min(find(p >= i))); end
    six_testx = pvsample(six_test, six_testi1-1, 128);
    six4 = six_testx(:);
    six5 = (six4)';
    six_Feature_Vector(rw,:) = six5(:);
end
seven = TR_Data(8:10:198);
seven_x1 = seven(12);
seven_x2 = cell2mat(seven_x1);
seven_ref = (seven_x2)'; %ref
for rw = 1:3
    seven1 = seven(rw);
    seven2 = cell2mat(seven1);
    seven_test = (seven2)';
    SM = simmx(abs(seven_ref),abs(seven_test));
    [p,q,C] = dp(1-SM);
    C(size(C,1),size(C,2));
    seven_testi1 = zeros(1, size(seven_ref,2));
    for i = 1:length(seven_testi1); seven_testi1(i) = q(min(find(p >= i))); end
    seven_testx = pvsample(seven_test, seven_testi1-1, 128);
    seven4 = seven_testx(:);
    seven5 = (seven4)';
    seven_Feature_Vector(rw,:) = seven5(:);
end
eight = TR_Data(9:10:199);
eight_x1 = eight(10);
eight_x2 = cell2mat(eight_x1);
eight_ref = (eight_x2)'; %ref
for rw = 1:3
    eight1 = eight(rw);
    eight2 = cell2mat(eight1);
    eight_test = (eight2)';
    SM = simmx(abs(eight_ref),abs(eight_test));
    [p,q,C] = dp(1-SM);
    C(size(C,1),size(C,2));
    eight_testi1 = zeros(1, size(eight_ref,2));
    for i = 1:length(eight_testi1); eight_testi1(i) = q(min(find(p >= i))); end
    eight_testx = pvsample(eight_test, eight_testi1-1, 128);
    eight4 = eight_testx(:);
    eight5 = (eight4)';
    eight_Feature_Vector(rw,:) = eight5(:);
end
nine = TR_Data(10:10:200);
nine_x1 = nine(17);
nine_x2 = cell2mat(nine_x1);
nine_ref = (nine_x2)'; %ref
for rw = 1:3
    nine1 = nine(rw);
    nine2 = cell2mat(nine1);
    nine_test = (nine2)';
    SM = simmx(abs(nine_ref),abs(nine_test));
    [p,q,C] = dp(1-SM);
    C(size(C,1),size(C,2));
    nine_testi1 = zeros(1, size(nine_ref,2));
    for i = 1:length(nine_testi1); nine_testi1(i) = q(min(find(p >= i))); end
    nine_testx = pvsample(nine_test, nine_testi1-1, 128);
    nine4 = nine_testx(:);
    nine5 = (nine4)';
    nine_Feature_Vector(rw,:) = nine5(:);
end


%--------------------------------Lip-Training--------------------------------
TRLP_subject=0;
TRLP_digit=0;

for TRLP_subject = 1:200
    %for TRLP_digit = 1:3

TRLP_Video = strcat('E:\VIKRANT\GVSU\Thesis(EGR
695)\codes\Fusion\Training\',num2str(TRLP_subject),'.avi');
%Create a VideoFileReader Object
TRLP_videoFReader = vision.VideoFileReader(TRLP_Video);

%Create a VideoPlayer Object
TRLP_videoPlayer = vision.VideoPlayer;

%Define size of VideoPlayer and Create Object to Display Multiple Video At a time
TRLP_WindowSize = [600 500];
TRLP_originalVideo = vision.VideoPlayer('Name', 'TRLP_Original');
TRLP_originalVideo.Position = [20 TRLP_originalVideo.Position(2)
TRLP_WindowSize];

TRLP_detectorVideo = vision.VideoPlayer('Name','TRLP_Detection');
TRLP_detectorVideo.Position = [200 TRLP_detectorVideo.Position(2)
TRLP_WindowSize];

TRLP_extractVideo = vision.VideoPlayer('Name','MouthExtract');
TRLP_extractVideo.Position = [400 TRLP_extractVideo.Position(2)
TRLP_WindowSize];
% noOfFrame = videoFReader.NumberOfFrames;
% disp(noOfFrame);

%Define Counter as 0 for Giving Image Name
TRLP_fcnt=0;
mkdir('Frames');
%Read Only First 40 Frame From Video
while ~isDone(TRLP_videoFReader)

    %counter Increment
    TRLP_fcnt =TRLP_fcnt+1;
    %Create A path of the Image to Write
    TRLP_filename = strcat('E:\VIKRANT\GVSU\Thesis(EGR
695)\codes\Fusion\TR_Detected\Frames\',num2str(TRLP_subject),'_',num2str(TRLP_dig
it),'_',num2str(TRLP_fcnt),'.jpg');

    %Read Frame from Video
    TRLP_videoFrame=step(TRLP_videoFReader);

    %videoFrame = imresize(videoFrame,[420 380]);
    %write a Frame into the Images/Frame Folder
    imwrite(TRLP_videoFrame,TRLP_filename,'jpg');
end

%initialize counter is 0
TRLP_cnt=0;

%Call the Function buildDetector
TRLP_detector = buildDetector();
TRLP_aa=[];

%Read the Frame from Given Path and Perform Some Operation On It.
for i=1:TRLP_fcnt
   %increment the Counter
   TRLP_cnt = TRLP_cnt + 1;

   %Set the Path of Frame to Get
   TRLP_getFrame = strcat('E:\VIKRANT\GVSU\Thesis(EGR
695)\codes\Fusion\TR_Detected\Frames\',num2str(TRLP_subject),'_',num2str(TRLP_dig
it),'_',num2str(i),'.jpg');

   %Read the Frames from Given path
   TRLP_frame = imread(TRLP_getFrame);

   TRLP_Rframe=step(TRLP_videoFReader);
   %frame = imresize(frame,[480 720]);
   %Call the Function detectFaceParts send paramert (detector,Image,width
   %of Boundry Box)
   [TRLP_bbox TRLP_bbimg TRLP_faces TRLP_bbfaces] =
detectFaceParts(TRLP_detector,TRLP_frame,2);
%    bbox = detectSideFaceParts(detector,frame,2);

   %Playing a Original Video
   step(TRLP_originalVideo,TRLP_frame);
   %----------------------------End Original Video --------------------
```

```
  %Create a Path to Store a Result Image into the Folder Name as
  %DetectedFrame
  TRLP_filename1 = strcat('E:\VIKRANT\GVSU\Thesis(EGR
695)\codes\Fusion\TR_Detected\Detected
Frames\',num2str(TRLP_subject),'_',num2str(TRLP_digit),'_',num2str(TRLP_cnt),'.jpg');

  %write a Frame into the Images/DetectedFrame Folder
  imwrite(TRLP_bbimg,TRLP_filename1,'jpg');

  %Playing a Result Video
  step(TRLP_detectorVideo,TRLP_bbimg);
  %--------------End Facial Feature Detected Video ------------------

  TRLP_aa = [TRLP_aa ; TRLP_bbox(:,13:16)];

  %mouth Feature Extration
  TRLP_featurePoint = TRLP_bbox(:,13:16);
  %disp(featurePoint);
  if TRLP_featurePoint(1,1)==0
     TRLP_featurePoint = TRLP_aa(1,:);
  end

  %Insrease the size of Bbox
  TRLP_featurePoint = [TRLP_featurePoint(1,1) TRLP_featurePoint(1,2)-5
TRLP_featurePoint(1,3) TRLP_featurePoint(1,4)+3];

  disp(TRLP_featurePoint);
  %Crop specific Area of Mouth using Crop function
  TRLP_extractFrame = imcrop(TRLP_frame,TRLP_featurePoint);

  TRLP_filename3 = strcat('E:\VIKRANT\GVSU\Thesis(EGR
695)\codes\Fusion\TR_Detected\Extract\',num2str(TRLP_subject),'_',num2str(TRLP_dig
it),'_',num2str(TRLP_cnt),'.jpg');
  %Write all the Images into the folder
  imwrite(TRLP_extractFrame,TRLP_filename3,'jpg');


  %Make Same Width and Height of all crop Images
  TRLP_resizeFrame = imresize(TRLP_extractFrame,[64 64]);

  %Create a Path to Store the Mouth Images
```

```
   TRLP_filename2 = strcat('E:\VIKRANT\GVSU\Thesis(EGR
695)\codes\Fusion\TR_Detected\Resize
Extract\MouthExtract',num2str(TRLP_subject),'_',num2str(TRLP_digit),'_',num2str(TRL
P_cnt),'.jpg');

   %Write all the Images into the folder
   imwrite(TRLP_resizeFrame,TRLP_filename2,'jpg');

   %Play Mouth Extract Frame into Video
   %step(extractVideo,resizeFrame);
   %------------------End Mouth Extracted Video ----------------------
end
end

rmdir('Frames','s');
%Realease all the VideoFile and VideoPlayer Object
release(TRLP_originalVideo);
release(TRLP_detectorVideo);
release(TRLP_videoFReader);

%------------------------------DCT--------------------------------------
%---Calculation of 800 DCT coeffients for all 50 frames of the input videos
%---using 16*16 block-----------------------------------------------------
TRLP_iter = 0;
 for TRLP_subject = 1:200
    %for TRLP_digit = 0:9
       for fr = 1:50
    TRLP_iter = TRLP_iter+1;
    TRLP_resize_extract = strcat('E:\VIKRANT\GVSU\Thesis(EGR
695)\codes\Fusion\TR_Detected\Resize
Extract\MouthExtract',num2str(TRLP_subject),'_',num2str(fr),'.jpg');
    TRLP_f = imread(TRLP_resize_extract);
    TRLP_f = rgb2gray(TRLP_f);
    TRLP_f = im2double(TRLP_f);
    TRLP_T = dctmtx(16);
    TRLP_dct = @(block_struct) TRLP_T * block_struct.data * TRLP_T';
    TRLP_B1 = blockproc(TRLP_f,[16 16],TRLP_dct);
    TRLP_fun = @(block_struct) sum(block_struct.data(:));
    TRLP_B2 = blockproc(TRLP_B1,[16 16],TRLP_fun);
    TRLP_DCT_Feature = cell(1,160000);
    TRLP_D{TRLP_iter} = TRLP_B2(:)';
    TRLP_DCT_Feature = cat(2,TRLP_D{:});
       end
    %end
 end

TRLP_DCT_Feature = reshape(TRLP_DCT_Feature,[],200);
TRLP_DCT_Feature = TRLP_DCT_Feature';
%----------------------------------------------------------------------

%-----------------------Normalisation----------------------------------
TRLP_DCT_Feature_Vector = normalise(TRLP_DCT_Feature);
%----------------------------------------------------------------------


%-------------------------Fusion---------------------------------------
%-------------Fusion of MFCC and DCT Training feature vectors-----------
TRFusion_Feature_Vector = horzcat(TRSP_MFCC_Feature,TRLP_DCT_Feature);
%----------------------------------------------------------------------


%----------------------------Speech-Testing----------------------------------

Testing_Words = {'Zero','One','Two','Three','Four','Five','Six','Seven','Eight','Nine'};
%Sample Frequency
fs=16000;
Thresh=0.3;
TE_row=0;
for TE_p=1:200
 %for TE_q=0:9
   TE_row = TE_row + 1;
   TE_filename='E:\VIKRANT\GVSU\Thesis(EGR 695)\codes\Speech
Recognition\Demo2\Test\';
   [TE_x,TE_fs]=audioread(strcat(TE_filename,num2str(TE_p),'.wav'));
   TE_samplePerFrame=floor(TE_fs/100);
TE_bgSampleCount=floor(TE_fs/5); %according to formula, 1600 sample needed for 8
khz
%calculation of mean and std
TE_bgSample=[];
for i=1:1:TE_bgSampleCount
    TE_bgSample=[TE_bgSample TE_x(i)];
end
TE_meanvalue=mean(TE_bgSample);
TE_std_dev=std(TE_bgSample);
%identify voiced or not for each value
for i=1:1:length(TE_x)
```

```
   if(abs(TE_x(i)-TE_meanvalue)/TE_std_dev > Thresh)
       TE_voiced(i)=1;
   else
       TE_voiced(i)=0;
   end
end
%identify voiced or not for each frame
%discard insufficient samples of last frame
TE_useful_samples=length(TE_x)-mod(length(TE_x),TE_samplePerFrame);
TE_frameCount=TE_useful_samples/TE_samplePerFrame;
TE_voiced_frameCount=0;
for i=1:1:TE_frameCount
   TE_cVoiced=0;
   TE_cUnVoiced=0;
   for j=i*TE_samplePerFrame-TE_samplePerFrame+1:1:(i*TE_samplePerFrame)
       if(TE_voiced(j)==1)
           TE_cVoiced=(TE_cVoiced+1);
       else
           TE_cUnVoiced=TE_cUnVoiced+1;
       end
   end
%mark frame for voiced/unvoiced
   if(TE_cVoiced>TE_cUnVoiced)
       TE_voiced_frameCount=TE_voiced_frameCount+1;
       TE_voicedUnvoiced(i)=1;
   else
       TE_voicedUnvoiced(i)=0;
   end
end
TE_reqd_signal=[];
for i=1:1:TE_frameCount
    if(TE_voicedUnvoiced(i)==1)
    for j=i*TE_samplePerFrame-TE_samplePerFrame+1:1:(i*TE_samplePerFrame)
            TE_reqd_signal= [TE_reqd_signal TE_x(j)];
    end
    end
end
t1 = (0:length(TE_x)-1)/fs;
    t1 = (0:length(TE_x)-1)/fs;
    figure(); plot(t1,TE_x)
    title('original signal');
    xlabel('time[sec]')
    ylabel('Audio Signal Amplitude')
    t2 = (0:length(TE_reqd_signal)-1)/fs;
    figure(); plot(t2, TE_reqd_signal);
    title('Signal after silence removal');
    xlabel('time[sec]')
    ylabel('Audio Signal Amplitude')
    fs = 16000;
n=512;
t=(1:n)'/fs;
startIndex=500;
endIndex=startIndex+n-1;
original=TE_x(startIndex:endIndex);
windowed=original.*hamming(n);
[mag1, phase1, freq1]=fftTwoSide(original, fs);
[mag2, phase2, freq2]=fftTwoSide(windowed, fs);
figure();
subplot(3,2,1); plot(original); grid on; axis([-inf inf -1 1]); title('Original
signal');xlabel('Sample Number');ylabel('Amplitude');
subplot(3,2,2); plot(windowed); grid on; axis([-inf inf -1 1]); title('Windowed
signal');xlabel('Sample Number');ylabel('Amplitude');
subplot(3,2,3); plot(freq1, mag1); grid on; title('Energy spectrum (linear
scale)');xlabel('Frequency[Hz]');ylabel('Amplitude');
subplot(3,2,4); plot(freq2, mag2); grid on; title('Energy spectrum (linear
scale)');xlabel('Frequency[Hz]');ylabel('Amplitude');
subplot(3,2,5); plot(freq1, 20*log(mag1)); grid on; axis([-inf inf -80 120]); title('Energy
spectrum (db)');xlabel('Frequency[Hz]');ylabel('Amplitude');
subplot(3,2,6); plot(freq2, 20*log(mag2)); grid on; axis([-inf inf -80 120]); title('Energy
spectrum (db)');xlabel('Frequency[Hz]');ylabel('Amplitude');
TE_data{TE_row} = TE_reqd_signal;
TE_signals={};
TE_signals=TE_data';
TE_Data = mfcc1(TE_signals,Testing_Words,16000);
end
for rw = 1:1
    TE_Data1_0 = TE_Data(rw);
    TE_Data2_0 = cell2mat(TE_Data1_0);
    TE_Data3_0 = (TE_Data2_0)';
    SM = simmx(abs(zero_ref),abs(TE_Data3_0));
    figure(),subplot(121)
    imagesc(SM)
    colormap(1-gray)
    [p,q,C] = dp(1-SM);
    hold on; plot(q,p,'r'); hold off
    subplot(122)
    imagesc(C)
    hold on; plot(q,p,'r'); hold off
    C(size(C,1),size(C,2));
    TE_Data3_0i1 = zeros(1, size(zero_ref,2));
    for i = 1:length(TE_Data3_0i1); TE_Data3_0i1(i) = q(min(find(p >= i))); end
    TE_Data3_0x = pvsample(TE_Data3_0, TE_Data3_0i1-1, 128);
    TE_Feature_Vector_0(rw,:) = TE_Data3_0x(:);

    energy1 = [];
    coeff = TE_Data3_0(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy1(col) = Total_ene;
    end
    energy2 = [];
    coeff = zero_ref(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy2(col) = Total_ene;
    end
    energy3 = [];
    coeff = TE_Data3_0x(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy3(col) = Total_ene;
    end
figure(), plot(energy1);
title('Plot before DTW');
xlabel('Frames');ylabel('Energy');
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
figure(), plot(energy3);
title('Plot after DTW');
xlabel('Frames');ylabel('Energy');
```

```
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
end
for rw = 1:1
    TE_Data1_1 = TE_Data(rw);
    TE_Data2_1 = cell2mat(TE_Data1_1);
    TE_Data3_1 = (TE_Data2_1)';
    SM = simmx(abs(one_ref),abs(TE_Data3_1));
    figure(),subplot(121)
    imagesc(SM)
    colormap(1-gray)
    [p,q,C] = dp(1-SM);
    hold on; plot(q,p,'r'); hold off
    subplot(122)
    imagesc(C)
    hold on; plot(q,p,'r'); hold off
    C(size(C,1),size(C,2));
    TE_Data3_1i1 = zeros(1, size(one_ref,2));
    for i = 1:length(TE_Data3_1i1); TE_Data3_1i1(i) = q(min(find(p >= i))); end
    TE_Data3_1x = pvsample(TE_Data3_1, TE_Data3_1i1-1, 128);
    TE_Feature_Vector_1(rw,:) = TE_Data3_1x(:);

    energy1 = [];
    coeff = TE_Data3_1(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy1(col) = Total_ene;
    end
    energy2 = [];
    coeff = one_ref(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy2(col) = Total_ene;
    end
    energy3 = [];
    coeff = TE_Data3_1x(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy3(col) = Total_ene;
    end
figure(), plot(energy1);
title('Plot before DTW');
xlabel('Frames');ylabel('Energy');
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
figure(), plot(energy3);
title('Plot after DTW');
xlabel('Frames');ylabel('Energy');
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
end
for rw = 1:1
    TE_Data1_2 = TE_Data(rw);
    TE_Data2_2 = cell2mat(TE_Data1_2);
    TE_Data3_2 = (TE_Data2_2)';
    SM = simmx(abs(two_ref),abs(TE_Data3_2));
    figure(),subplot(121)
    imagesc(SM)
    colormap(1-gray)
    [p,q,C] = dp(1-SM);
    hold on; plot(q,p,'r'); hold off
    subplot(122)
    imagesc(C)
    hold on; plot(q,p,'r'); hold off
    C(size(C,1),size(C,2));
    TE_Data3_2i1 = zeros(1, size(two_ref,2));
    for i = 1:length(TE_Data3_2i1); TE_Data3_2i1(i) = q(min(find(p >= i))); end
    TE_Data3_2x = pvsample(TE_Data3_2, TE_Data3_2i1-1, 128);
    TE_Feature_Vector_2(rw,:) = TE_Data3_2x(:);

    energy1 = [];
    coeff = TE_Data3_2(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy1(col) = Total_ene;
    end
    energy2 = [];
    coeff = two_ref(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy2(col) = Total_ene;
    end
    energy3 = [];
    coeff = TE_Data3_2x(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy3(col) = Total_ene;
    end
figure(), plot(energy1);
title('Plot before DTW');
xlabel('Frames');ylabel('Energy');
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
figure(), plot(energy3);
title('Plot after DTW');
xlabel('Frames');ylabel('Energy');
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
end
for rw = 1:1
    TE_Data1_3 = TE_Data(rw);
    TE_Data2_3 = cell2mat(TE_Data1_3);
    TE_Data3_3 = (TE_Data2_3)';
    SM = simmx(abs(three_ref),abs(TE_Data3_3));
    figure(),subplot(121)
    imagesc(SM)
    colormap(1-gray)
    [p,q,C] = dp(1-SM);
    hold on; plot(q,p,'r'); hold off
    subplot(122)
    imagesc(C)
    hold on; plot(q,p,'r'); hold off
    C(size(C,1),size(C,2));
    TE_Data3_3i1 = zeros(1, size(three_ref,2));
    for i = 1:length(TE_Data3_3i1); TE_Data3_3i1(i) = q(min(find(p >= i))); end
    TE_Data3_3x = pvsample(TE_Data3_3, TE_Data3_3i1-1, 128);
    TE_Feature_Vector_3(rw,:) = TE_Data3_3x(:);

    energy1 = [];
    coeff = TE_Data3_3(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy1(col) = Total_ene;
    end
    energy2 = [];
    coeff = three_ref(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy2(col) = Total_ene;
    end
    energy3 = [];
    coeff = TE_Data3_3x(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy3(col) = Total_ene;
    end
figure(), plot(energy1);
title('Plot before DTW');
xlabel('Frames');ylabel('Energy');
```

```
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
figure(), plot(energy3);
title('Plot after DTW');
xlabel('Frames');ylabel('Energy');
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
end
for rw = 1:1
   TE_Data1_4 = TE_Data(rw);
   TE_Data2_4 = cell2mat(TE_Data1_4);
   TE_Data3_4 = (TE_Data2_4)';
   SM = simmx(abs(four_ref),abs(TE_Data3_4));
   figure()%subplot(121)
   imagesc(SM)
   colormap(1-gray)
   [p,q,C] = dp(1-SM);
   hold on; plot(q,p,'r'); hold off
%     title('DTW Path on the local similarity matrix');
%     xlabel('Test signal frames');ylabel('Reference signal frames');
   subplot(122)
   imagesc(C)
   hold on; plot(q,p,'r'); hold off
   C(size(C,1),size(C,2));
   TE_Data3_4i1 = zeros(1, size(four_ref,2));
   for i = 1:length(TE_Data3_4i1); TE_Data3_4i1(i) = q(min(find(p >= i))); end
   TE_Data3_4x = pvsample(TE_Data3_4, TE_Data3_4i1-1, 128);
   TE_Feature_Vector_4(rw,:) = TE_Data3_4x(:);

   energy1 = [];
   coeff = TE_Data3_4(1:13,:);
   for col = 1:size(coeff,2)
     Fourier = fft(coeff(:,col));
     ene = Fourier.*conj(Fourier);
     Total_ene = sum(ene);
     energy1(col) = Total_ene;
   end
```

```
    energy2 = [];
    coeff = four_ref(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy2(col) = Total_ene;
    end
    energy3 = [];
    coeff = TE_Data3_4x(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy3(col) = Total_ene;
    end
figure(), plot(energy1);
title('Plot before DTW');
xlabel('Frames');ylabel('Energy');
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
figure(), plot(energy3);
title('Plot after DTW');
xlabel('Frames');ylabel('Energy');
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
end
for rw = 1:1
    TE_Data1_5 = TE_Data(rw);
    TE_Data2_5 = cell2mat(TE_Data1_5);
    TE_Data3_5 = (TE_Data2_5)';
    SM = simmx(abs(five_ref),abs(TE_Data3_5));
    figure(),subplot(121)
    imagesc(SM)
    colormap(1-gray)
    [p,q,C] = dp(1-SM);
    hold on; plot(q,p,'r'); hold off
    subplot(122)
    imagesc(C)
    hold on; plot(q,p,'r'); hold off
    C(size(C,1),size(C,2));
    TE_Data3_5i1 = zeros(1, size(five_ref,2));
    for i = 1:length(TE_Data3_5i1); TE_Data3_5i1(i) = q(min(find(p >= i))); end
    TE_Data3_5x = pvsample(TE_Data3_5, TE_Data3_5i1-1, 128);
    TE_Feature_Vector_5(rw,:) = TE_Data3_5x(:);

    energy1 = [];
    coeff = TE_Data3_5(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy1(col) = Total_ene;
    end
    energy2 = [];
    coeff = five_ref(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy2(col) = Total_ene;
    end
    energy3 = [];
    coeff = TE_Data3_5x(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy3(col) = Total_ene;
    end
figure(), plot(energy1);
title('Plot before DTW');
xlabel('Frames');ylabel('Energy');
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
figure(), plot(energy3);
title('Plot after DTW');
```

```
xlabel('Frames');ylabel('Energy');
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
end
for rw = 1:1
    TE_Data1_6 = TE_Data(rw);
    TE_Data2_6 = cell2mat(TE_Data1_6);
    TE_Data3_6 = (TE_Data2_6)';
    SM = simmx(abs(six_ref),abs(TE_Data3_6));
    figure(),subplot(121)
    imagesc(SM)
    colormap(1-gray)
    [p,q,C] = dp(1-SM);
    hold on; plot(q,p,'r'); hold off
    subplot(122)
    imagesc(C)
    hold on; plot(q,p,'r'); hold off
    C(size(C,1),size(C,2));
    TE_Data3_6i1 = zeros(1, size(six_ref,2));
    for i = 1:length(TE_Data3_6i1); TE_Data3_6i1(i) = q(min(find(p >= i))); end
    TE_Data3_6x = pvsample(TE_Data3_6, TE_Data3_6i1-1, 128);
    TE_Feature_Vector_6(rw,:) = TE_Data3_6x(:);

    energy1 = [];
    coeff = TE_Data3_6(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy1(col) = Total_ene;
    end
    energy2 = [];
    coeff = six_ref(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy2(col) = Total_ene;
    end
    energy3 = [];
    coeff = TE_Data3_6x(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy3(col) = Total_ene;
    end
figure(), plot(energy1);
title('Plot before DTW');
xlabel('Frames');ylabel('Energy');
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
figure(), plot(energy3);
title('Plot after DTW');
xlabel('Frames');ylabel('Energy');
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
end
for rw = 1:1
    TE_Data1_7 = TE_Data(rw);
    TE_Data2_7 = cell2mat(TE_Data1_7);
    TE_Data3_7 = (TE_Data2_7)';
    SM = simmx(abs(seven_ref),abs(TE_Data3_7));
    figure(),subplot(121)
    imagesc(SM)
    colormap(1-gray)
    [p,q,C] = dp(1-SM);
    hold on; plot(q,p,'r'); hold off
    subplot(122)
    imagesc(C)
    hold on; plot(q,p,'r'); hold off
    C(size(C,1),size(C,2));
    TE_Data3_7i1 = zeros(1, size(seven_ref,2));
    for i = 1:length(TE_Data3_7i1); TE_Data3_7i1(i) = q(min(find(p >= i))); end
    TE_Data3_7x = pvsample(TE_Data3_7, TE_Data3_7i1-1, 128);
    TE_Feature_Vector_7(rw,:) = TE_Data3_7x(:);
    energy1 = [];
    coeff = TE_Data3_7(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy1(col) = Total_ene;
    end
    energy2 = [];
    coeff = seven_ref(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy2(col) = Total_ene;
    end
    energy3 = [];
    coeff = TE_Data3_7x(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy3(col) = Total_ene;
    end
figure(), plot(energy1);
title('Plot before DTW');
xlabel('Frames');ylabel('Energy');
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
figure(), plot(energy3);
title('Plot after DTW');
xlabel('Frames');ylabel('Energy');
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
end
for rw = 1:1
    TE_Data1_8 = TE_Data(rw);
    TE_Data2_8 = cell2mat(TE_Data1_8);
    TE_Data3_8 = (TE_Data2_8)';
    SM = simmx(abs(eight_ref),abs(TE_Data3_8));
    figure(),subplot(121)
    imagesc(SM)
    colormap(1-gray)
    [p,q,C] = dp(1-SM);
    hold on; plot(q,p,'r'); hold off
    subplot(122)
    imagesc(C)
    hold on; plot(q,p,'r'); hold off
    C(size(C,1),size(C,2));
    TE_Data3_8i1 = zeros(1, size(eight_ref,2));
    for i = 1:length(TE_Data3_8i1); TE_Data3_8i1(i) = q(min(find(p >= i))); end
    TE_Data3_8x = pvsample(TE_Data3_8, TE_Data3_8i1-1, 128);
    TE_Feature_Vector_8(rw,:) = TE_Data3_8x(:);

    energy1 = [];
    coeff = TE_Data3_8(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy1(col) = Total_ene;
    end
    energy2 = [];
    coeff = eight_ref(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy2(col) = Total_ene;
    end
    energy3 = [];
    coeff = TE_Data3_8x(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy3(col) = Total_ene;
    end
figure(), plot(energy1);
title('Plot before DTW');
```

```
xlabel('Frames');ylabel('Energy');
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
figure(), plot(energy3);
title('Plot after DTW');
xlabel('Frames');ylabel('Energy');
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
end
for rw = 1:1
    TE_Data1_9 = TE_Data(rw);
    TE_Data2_9 = cell2mat(TE_Data1_9);
    TE_Data3_9 = (TE_Data2_9)';
    SM = simmx(abs(nine_ref),abs(TE_Data3_9));
    figure(),subplot(121)
    imagesc(SM)
    colormap(1-gray)
    [p,q,C] = dp(1-SM);
    hold on; plot(q,p,'r'); hold off
    subplot(122)
    imagesc(C)
    hold on; plot(q,p,'r'); hold off
    C(size(C,1),size(C,2));
    TE_Data3_9i1 = zeros(1, size(nine_ref,2));
    for i = 1:length(TE_Data3_9i1); TE_Data3_9i1(i) = q(min(find(p >= i))); end
    TE_Data3_9x = pvsample(TE_Data3_9, TE_Data3_9i1-1, 128);
    TE_Feature_Vector_9(rw,:) = TE_Data3_9x(:);

    energy1 = [];
    coeff = TE_Data3_9(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy1(col) = Total_ene;
    end
    energy2 = [];
    coeff = nine_ref(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy2(col) = Total_ene;
    end
    energy3 = [];
    coeff = TE_Data3_9x(1:13,:);
    for col = 1:size(coeff,2)
      Fourier = fft(coeff(:,col));
      ene = Fourier.*conj(Fourier);
      Total_ene = sum(ene);
      energy3(col) = Total_ene;
    end
figure(), plot(energy1);
title('Plot before DTW');
xlabel('Frames');ylabel('Energy');
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
figure(), plot(energy3);
title('Plot after DTW');
xlabel('Frames');ylabel('Energy');
hold on
plot(energy2);
xlabel('Frames');ylabel('Energy');
legend('Test Signal','Reference Signal')
hold off
end


%---------------------------------Lip-Testing-----------------------------
TELP_subject=0;
TELP_digit=0;

for TELP_subject = 1:200
    %for TELP_digit = 1:3

TELP_Video = strcat('E:\VIKRANT\GVSU\Thesis(EGR
695)\codes\Fusion\Testing\',num2str(TELP_subject),'.avi');
```

```
%Create a VideoFileReader Object
TELP_videoFReader = vision.VideoFileReader(TELP_Video);

%Create a VideoPlayer Object
TELP_videoPlayer = vision.VideoPlayer;

%Define size of VideoPlayer and Create Object to Display Multiple Video At a time
TELP_WindowSize = [600 500];
TELP_originalVideo = vision.VideoPlayer('Name', 'TELP_Original');
TELP_originalVideo.Position = [20 TELP_originalVideo.Position(2)
TELP_WindowSize];

TELP_detectorVideo = vision.VideoPlayer('Name','TELP_Detection');
TELP_detectorVideo.Position = [200 TELP_detectorVideo.Position(2)
TELP_WindowSize];

TELP_extractVideo = vision.VideoPlayer('Name','MouthExtract');
TELP_extractVideo.Position = [400 TELP_extractVideo.Position(2)
TELP_WindowSize];
% noOfFrame = videoFReader.NumberOfFrames;
% disp(noOfFrame);

%Define Counter as 0 for Giving Image Name
TELP_fcnt=0;
mkdir('Frames');
%Read Only First 40 Frame From Video
while ~isDone(TELP_videoFReader)

   %counter Increment
   TELP_fcnt =TELP_fcnt+1;

   %Create A path of the Image to Write
   TELP_filename = strcat('E:\VIKRANT\GVSU\Thesis(EGR
695)\codes\Fusion\TE_Detected\Frames\',num2str(TELP_subject),'_',num2str(TELP_digi
t),'_',num2str(TELP_fcnt),'.jpg');

   %Read Frame from Video
   TELP_videoFrame=step(TELP_videoFReader);

   %videoFrame = imresize(videoFrame,[420 380]);
   %write a Frame into the Images/Frame Folder
   imwrite(TELP_videoFrame,TELP_filename,'jpg');
end
```

```
%initialize counter is 0
TELP_cnt=0;

%Call the Function buildDetector
TELP_detector = buildDetector();
TELP_aa=[];

%Read the Frame from Given Path and Perform Some Operation On It.
for i=1:TELP_fcnt
  %increment the Counter
  TELP_cnt = TELP_cnt + 1;

  %Set the Path of Frame to Get
  TELP_getFrame = strcat('E:\VIKRANT\GVSU\Thesis(EGR
695)\codes\Fusion\TE_Detected\Frames\',num2str(TELP_subject),'_',num2str(TELP_digi
t),'_',num2str(i),'.jpg');

  %Read the Frames from Given path
  TELP_frame = imread(TELP_getFrame);

  TELP_Rframe=step(TELP_videoFReader);
  %frame = imresize(frame,[480 720]);
  %Call the Function detectFaceParts send paramert (detector,Image,width
  %of Boundry Box)
  [TELP_bbox TELP_bbimg TELP_faces TELP_bbfaces] =
detectFaceParts(TELP_detector,TELP_frame,2);
%    bbox = detectSideFaceParts(detector,frame,2);

  %Playing a Original Video
  step(TELP_originalVideo,TELP_frame);
  %-----------------------------End Original Video ---------------------


  %Create a Path to Store a Result Image into the Folder Name as
  %DetectedFrame
  TELP_filename1 = strcat('E:\VIKRANT\GVSU\Thesis(EGR
695)\codes\Fusion\TE_Detected\Detected
Frames\',num2str(TELP_subject),'_',num2str(TELP_digit),'_',num2str(TELP_cnt),'.jpg');

  %write a Frame into the Images/DetectedFrame Folder
  imwrite(TELP_bbimg,TELP_filename1,'jpg');

  %Playing a Result Video
  step(TELP_detectorVideo,TELP_bbimg);
  %--------------End Facial Feature Detected Video ------------------

  TELP_aa = [TELP_aa ; TELP_bbox(:,13:16)];

  %mouth Feature Extration
  TELP_featurePoint = TELP_bbox(:,13:16);
  %disp(featurePoint);
  if TELP_featurePoint(1,1)==0
     TELP_featurePoint = TELP_aa(1,:);
  end

  %Insrease the size of Bbox
  TELP_featurePoint = [TELP_featurePoint(1,1) TELP_featurePoint(1,2)-5
TELP_featurePoint(1,3) TELP_featurePoint(1,4)+3];

  disp(TELP_featurePoint);
  %Crop specific Area of Mouth using Crop function
  TELP_extractFrame = imcrop(TELP_frame,TELP_featurePoint);

  TELP_filename3 = strcat('E:\VIKRANT\GVSU\Thesis(EGR
695)\codes\Fusion\TE_Detected\Extract\',num2str(TELP_subject),'_',num2str(TELP_digi
t),'_',num2str(TELP_cnt),'.jpg');
  %Write all the Images into the folder
  imwrite(TELP_extractFrame,TELP_filename3,'jpg');


  %Make Same Width and Height of all crop Images
  TELP_resizeFrame = imresize(TELP_extractFrame,[64 64]);

  %Create a Path to Store the Mouth Images
  TELP_filename2 = strcat('E:\VIKRANT\GVSU\Thesis(EGR
695)\codes\Fusion\TE_Detected\Resize
Extract\MouthExtract',num2str(TELP_subject),'_',num2str(TELP_digit),'_',num2str(TEL
P_cnt),'.jpg');

  %Write all the Images into the folder
  imwrite(TELP_resizeFrame,TELP_filename2,'jpg');

  %Play Mouth Extract Frame into Video
  %step(extractVideo,resizeFrame);
  %------------------End Mouth Extracted Video -----------------------
end
end
```

```
rmdir('Frames','s');
%Realease all the VideoFile and VideoPlayer Object
release(TELP_originalVideo);
release(TELP_detectorVideo);
release(TELP_videoFReader);

%-------------------------------DCT---------------------------------------
TELP_iter = 0;
 for TELP_subject = 1:200
    %for TELP_digit = 0:9
        for fr = 1:50
    TELP_iter = TELP_iter+1;
    TELP_resize_extract = strcat('E:\VIKRANT\GVSU\Thesis(EGR
695)\codes\Fusion\TE_Detected\Resize
Extract\MouthExtract',num2str(TELP_subject),'_',num2str(fr),'.jpg');
    TELP_f = imread(TELP_resize_extract);
    TELP_f = rgb2gray(TELP_f);
    TELP_f = im2double(TELP_f);
    TELP_T = dctmtx(16);
    TELP_dct = @(block_struct) TELP_T * block_struct.data * TELP_T';
    TELP_B1 = blockproc(TELP_f,[16 16],TELP_dct);
    TELP_fun = @(block_struct) sum(block_struct.data(:));
    TELP_B2 = blockproc(TELP_B1,[16 16],TELP_fun);
    TELP_DCT_Feature = cell(1,160000);
    TELP_D{TELP_iter} = TELP_B2(:)';
    TELP_DCT_Feature = cat(2,TELP_D{:});
        end
    %end
 end

TELP_DCT_Feature = reshape(TELP_DCT_Feature,[],200);
TELP_DCT_Feature = TELP_DCT_Feature';
%-----------------------------------------------------------------------

%-------------------------Normalisation------------------------------------
TELP_DCT_Feature_Vector = normalise(TELP_DCT_Feature);
%-----------------------------------------------------------------------


%---------------------------Fusion-----------------------------------------
%--------------Fusion of MFCC and DCT Training feature vectors-------------
TEFusion_Feature_Vector = horzcat(TESP_MFCC_Feature,TELP_DCT_Feature);
%-----------------------------------------------------------------------
```

```
%-------------------------------Label of Class----------------------------
%Formation of Labels Matrix for Training Feature Vector from 0 to 9--------
labels=[];
val=0;
%cnt = 0;
for i=1:size(TRFusion_Feature_Vector,1)
%cnt=cnt+1;
   labels(1,i)=val;
   val=val+1;
   if val>9
      val=0;
   end
end
class = (labels)';
%------------------------------------------------------------------------


%[index] = multisvm(Zero_Feature_Vector,class,TE_Feature_Vector_0)

Model_0=svm.train(abs(zero_Feature_Vector),class);
[predict_0,matrix_0]=svm.predict(Model_0,abs(TE_Feature_Vector_0));
maximum(1,:) = max(matrix_0);

Model_1=svm.train(abs(one_Feature_Vector),class);
[predict_1,matrix_1]=svm.predict(Model_1,abs(TE_Feature_Vector_1));
maximum(2,:) = max(matrix_1);

Model_2=svm.train(abs(two_Feature_Vector),class);
[predict_2,matrix_2]=svm.predict(Model_2,abs(TE_Feature_Vector_2));
maximum(3,:) = max(matrix_2);

Model_3=svm.train(abs(three_Feature_Vector),class);
[predict_3,matrix_3]=svm.predict(Model_3,abs(TE_Feature_Vector_3));
maximum(4,:) = max(matrix_3);

Model_4=svm.train(abs(four_Feature_Vector),class);
[predict_4,matrix_4]=svm.predict(Model_4,abs(TE_Feature_Vector_4));
maximum(5,:) = max(matrix_4);

Model_5=svm.train(abs(five_Feature_Vector),class);
[predict_5,matrix_5]=svm.predict(Model_5,abs(TE_Feature_Vector_5));
maximum(6,:) = max(matrix_5);
```

```
Model_6=svm.train(abs(six_Feature_Vector),class);
[predict_6,matrix_6]=svm.predict(Model_6,abs(TE_Feature_Vector_6));
maximum(7,:) = max(matrix_6);

Model_7=svm.train(abs(seven_Feature_Vector),class);
[predict_7,matrix_7]=svm.predict(Model_7,abs(TE_Feature_Vector_7));
maximum(8,:) = max(matrix_7);

Model_8=svm.train(abs(eight_Feature_Vector),class);
[predict_8,matrix_8]=svm.predict(Model_8,abs(TE_Feature_Vector_8));
maximum(9,:) = max(matrix_8);

Model_9=svm.train(abs(nine_Feature_Vector),class);
[predict_9,matrix_9]=svm.predict(Model_9,abs(TE_Feature_Vector_9));
maximum(10,:) = max(matrix_9);

%scores(TE_row,:) = maximum;
[digit_score,Digit] = max(maximum);
%clear maximum;

%maximum
Recognized_Digit(TE_row,:) = Digit-1;
%Recognized_Digit = Digit-1
end


function [p,q,D] = dp(M)
% [p,q] = dp(M)
%    Use dynamic programming to find a min-cost path through matrix M.
%    Return state sequence in p,q
% 2003-03-15 dpwe@ee.columbia.edu

% Copyright (c) 2003 Dan Ellis <dpwe@ee.columbia.edu>
% released under GPL - see file COPYRIGHT

[r,c] = size(M);

% costs
D = zeros(r+1, c+1);
D(1,:) = NaN;
D(:,1) = NaN;
D(1,1) = 0;
D(2:(r+1), 2:(c+1)) = M;
```

```
% traceback
phi = zeros(r,c);

for i = 1:r;
  for j = 1:c;
    [dmax, tb] = min([D(i, j), D(i, j+1), D(i+1, j)]);
    D(i+1,j+1) = D(i+1,j+1)+dmax;
    phi(i,j) = tb;
  end
end

% Traceback from top left
i = r;
j = c;
p = i;
q = j;
while i > 1 & j > 1
  tb = phi(i,j);
  if (tb == 1)
    i = i-1;
    j = j-1;
  elseif (tb == 2)
    i = i-1;
  elseif (tb == 3)
    j = j-1;
  else
    error;
  end
  p = [i,p];
  q = [j,q];
end

% Strip off the edges of the D matrix before returning
D = D(2:(r+1),2:(c+1));


function [f,t]=enframe(x,win,inc)
%ENFRAME split signal up into (overlapping) frames: one per row.
[F,T]=(X,WIN,INC)
%
%   F = ENFRAME(X,LEN) splits the vector X(:) up into
%   frames. Each frame is of length LEN and occupies
%   one row of the output matrix. The last few frames of X
%   will be ignored if its length is not divisible by LEN.
%   It is an error if X is shorter than LEN.
%
%   F = ENFRAME(X,LEN,INC) has frames beginning at increments of INC
%   The centre of frame I is X((I-1)*INC+(LEN+1)/2) for I=1,2,...
%   The number of frames is fix((length(X)-LEN+INC)/INC)
%
%   F = ENFRAME(X,WINDOW) or ENFRAME(X,WINDOW,INC) multiplies
%   each frame by WINDOW(:)
%
%   The second output argument, T, gives the time in samples at the centre
%   of each frame. T=i corresponds to the time of sample X(i).
%
% Example of frame-based processing:
%          INC=20                                                  % set frame increment
%          NW=INC*2                                                     % oversample by a factor of 2 (4
is also often used)
%          S=cos((0:NW*7)*6*pi/NW);                              % example input signal
%          W=sqrt(hamming(NW+1)); W(end)=[];      % sqrt hamming window of period
NW
%          F=enframe(S,W,INC);                            % split into frames
%          ... process frames ...
%          X=overlapadd(F,W,INC);                      % reconstitute the time waveform (omit
"X=" to plot waveform)

%      Copyright (C) Mike Brookes 1997
%      Version: $Id: enframe.m,v 1.7 2009/11/01 21:08:21 dmb Exp $
%
%   VOICEBOX is a MATLAB toolbox for speech processing.
%   Home page: http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This program is free software; you can redistribute it and/or modify
%   it under the terms of the GNU General Public License as published by
%   the Free Software Foundation; either version 2 of the License, or
%   (at your option) any later version.
%
%   This program is distributed in the hope that it will be useful,
%   but WITHOUT ANY WARRANTY; without even the implied warranty of
%   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%   GNU General Public License for more details.
```

```
%
%   You can obtain a copy of the GNU General Public License from
%   http://www.gnu.org/copyleft/gpl.html or by writing to
%   Free Software Foundation, Inc.,675 Mass Ave, Cambridge, MA 02139, USA.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nx=length(x(:));
nwin=length(win);
if (nwin == 1)
   len = win;
else
   len = nwin;
end
if (nargin < 3)
   inc = len;
end
nf = fix((nx-len+inc)/inc);
f=zeros(nf,len);
indf= inc*(0:(nf-1)).';
inds = (1:len);
f(:) = x(indf(:,ones(1,len))+inds(ones(nf,1),:));
if (nwin > 1)
    w = win(:)';
    f = f .* w(ones(nf,1),:);
end
if nargout>1
    t=(1+len)/2+indf;
end


function [mag, phase, freq, powerDb]=fftTwoSide(signal, fs, plotOpt)
% fftTwoSide: Two-sided FFT for real/complex signals
%   Usage: [mag, phase, freq, powerDb]=fftTwoSide(signal, fs)

%   Roger Jang, 20060411

if nargin<1, selfdemo; return; end
if nargin<2, fs=1; end
if nargin<3, plotOpt=0; end

N = length(signal);     % ??
freqStep = fs/N;        % ?????????
```

```
time = (0:N-1)/fs;      % ???????
z = fft(signal);        % Spectrum
freq = freqStep*(-N/2:N/2-1);   % ???????
z = fftshift(z);        % ?????????
mag=abs(z);         % Magnitude
phase=unwrap(angle(z));     % Phase
powerDb=20*log(mag+eps);    % Power in db

if plotOpt
   % ====== Plot time-domain signals
   subplot(3,1,1);
   plot(time, signal, '.-');
   title(sprintf('Input signals (fs=%d)', fs));
   xlabel('Time (seconds)'); ylabel('Amplitude'); axis tight
   % ====== Plot spectral power
   subplot(3,1,2);
   plot(freq, powerDb, '.-'); grid on
   title('Power spectrum');
   xlabel('Frequency (Hz)'); ylabel('Power (db)'); axis tight
   % ====== Plot phase
   subplot(3,1,3);
   plot(freq, phase, '.-'); grid on
   title('Phase');
   xlabel('Frequency (Hz)'); ylabel('Phase (Radian)'); axis tight
end


function mel = frq2mel(frq)
%FRQ2ERB  Convert Hertz to Mel frequency scale MEL=(FRQ)
%   mel = frq2mel(frq) converts a vector of frequencies (in Hz)
%   to the corresponding values on the Mel scale which corresponds
%   to the perceived pitch of a tone

%   The relationship between mel and frq is given by:
%
%   m = ln(1 + f/700) * 1000 / ln(1+1000/700)
%
%   This means that m(1000) = 1000
%
%   References:
%
%     [1] S. S. Stevens & J. Volkman "The relation of pitch to
%       frequency", American J of Psychology, V 53, p329 1940
%     [2] C. G. M. Fant, "Acoustic description & classification
%       of phonetic units", Ericsson Tchnics, No 1 1959
%       (reprinted in "Speech Sounds & Features", MIT Press 1973)
%     [3] S. B. Davis & P. Mermelstein, "Comparison of parametric
%       representations for monosyllabic word recognition in
%       continuously spoken sentences", IEEE ASSP, V 28,
%       pp 357-366 Aug 1980
%     [4] J. R. Deller Jr, J. G. Proakis, J. H. L. Hansen,
%       "Discrete-Time Processing of Speech Signals", p380,
%       Macmillan 1993
%     [5] HTK Reference Manual p73
%



%      Copyright (C) Mike Brookes 1998
%      Version: $Id: frq2mel.m,v 1.5 2009/12/30 10:30:05 dmb Exp $
%
%   VOICEBOX is a MATLAB toolbox for speech processing.
%   Home page: http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This program is free software; you can redistribute it and/or modify
%   it under the terms of the GNU General Public License as published by
%   the Free Software Foundation; either version 2 of the License, or
%   (at your option) any later version.
%
%   This program is distributed in the hope that it will be useful,
%   but WITHOUT ANY WARRANTY; without even the implied warranty of
%   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%   GNU General Public License for more details.
%
%   You can obtain a copy of the GNU General Public License from
%   http://www.gnu.org/copyleft/gpl.html or by writing to
%   Free Software Foundation, Inc.,675 Mass Ave, Cambridge, MA 02139, USA.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mel = sign(frq).*log(1+abs(frq)/700)*1127.01048;
if ~nargout
    plot(frq,mel,'-x');
    xlabel(['Frequency (' xticksi 'Hz)']);
    ylabel(['Frequency (' yticksi 'Mel)']);
end


function x = istft(d, ftsize, w, h)
% X = istft(D, F, W, H)                   Inverse short-time Fourier transform.
%   Performs overlap-add resynthesis from the short-time Fourier transform
%   data in D.  Each column of D is taken as the result of an F-point
%   fft; each successive frame was offset by H points (default
%   W/2, or F/2 if W==0). Data is hann-windowed at W pts, or
%       W = 0 gives a rectangular window (default);
%       W as a vector uses that as window.
%       This version scales the output so the loop gain is 1.0 for
%       either hann-win an-syn with 25% overlap, or hann-win on
%       analysis and rect-win (W=0) on synthesis with 50% overlap.
% dpwe 1994may24.  Uses built-in 'ifft' etc.
% $Header: /home/empire6/dpwe/public_html/resources/matlab/pvoc/RCS/istft.m,v 1.5
2010/08/12 20:39:42 dpwe Exp $

if nargin < 2; ftsize = 2*(size(d,1)-1); end
if nargin < 3; w = 0; end
if nargin < 4; h = 0; end  % will become winlen/2 later

s = size(d);
if s(1) ~= (ftsize/2)+1
  error('number of rows should be fftsize/2+1')
end
cols = s(2);

if length(w) == 1
  if w == 0
    % special case: rectangular window
    win = ones(1,ftsize);
  else
    if rem(w, 2) == 0   % force window to be odd-len
      w = w + 1;
    end
    halflen = (w-1)/2;
    halff = ftsize/2;
    halfwin = 0.5 * ( 1 + cos( pi * (0:halflen)/halflen));
    win = zeros(1, ftsize);
    acthalflen = min(halff, halflen);
    win((halff+1):(halff+acthalflen)) = halfwin(1:acthalflen);
    win((halff+1):-1:(halff-acthalflen+2)) = halfwin(1:acthalflen);
    % 2009-01-06: Make stft-istft loop be identity for 25% hop
    win = 2/3*win;
  end
else
  win = w;
end

w = length(win);
% now can set default hop
if h == 0
  h = floor(w/2);
end

xlen = ftsize + (cols-1)*h;
x = zeros(1,xlen);

for b = 0:h:(h*(cols-1))
  ft = d(:,1+b/h)';
  ft = [ft, conj(ft([((ftsize/2)):-1:2]))];
  px = real(ifft(ft));
  x((b+1):(b+ftsize)) = x((b+1):(b+ftsize))+px.*win;
end;


function frq = mel2frq(mel)
%MEL2FRQ  Convert Mel frequency scale to Hertz FRQ=(MEL)
%   frq = mel2frq(mel) converts a vector of Mel frequencies
%   to the corresponding real frequencies.
%   The Mel scale corresponds to the perceived pitch of a tone

%   The relationship between mel and frq is given by:
%
%   m = ln(1 + f/700) * 1000 / ln(1+1000/700)
%
%   This means that m(1000) = 1000
%
%   References:
%
%     [1] S. S. Stevens & J. Volkman "The relation of pitch to
%       frequency", American J of Psychology, V 53, p329 1940
%     [2] C. G. M. Fant, "Acoustic description & classification
%       of phonetic units", Ericsson Tchnics, No 1 1959
%      (reprinted in "Speech Sounds & Features", MIT Press 1973)
%    [3] S. B. Davis & P. Mermelstein, "Comparison of parametric
%      representations for monosyllabic word recognition in
%      continuously spoken sentences", IEEE ASSP, V 28,
%      pp 357-366 Aug 1980
%    [4] J. R. Deller Jr, J. G. Proakis, J. H. L. Hansen,
%      "Discrete-Time Processing of Speech Signals", p380,
%      Macmillan 1993
%    [5] HTK Reference Manual p73
%



%     Copyright (C) Mike Brookes 1998
%     Version: $Id: mel2frq.m,v 1.5 2009/12/30 10:30:25 dmb Exp $
%
%   VOICEBOX is a MATLAB toolbox for speech processing.
%   Home page: http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This program is free software; you can redistribute it and/or modify
%   it under the terms of the GNU General Public License as published by
%   the Free Software Foundation; either version 2 of the License, or
%   (at your option) any later version.
%
%   This program is distributed in the hope that it will be useful,
%   but WITHOUT ANY WARRANTY; without even the implied warranty of
%   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%   GNU General Public License for more details.
%
%   You can obtain a copy of the GNU General Public License from
%   http://www.gnu.org/copyleft/gpl.html or by writing to
%   Free Software Foundation, Inc.,675 Mass Ave, Cambridge, MA 02139, USA.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

frq=700*sign(mel).*(exp(abs(mel)/1127.01048)-1);
if ~nargout
    plot(mel,frq,'-x');
    xlabel(['Frequency (' xticksi 'Mel)']);
    ylabel(['Frequency (' yticksi 'Hz)']);
end
```

```
function [x,mc,mn,mx]=melbankm(p,n,fs,fl,fh,w)
%MELBANKM determine matrix for a mel/erb/bark-spaced filterbank
[X,MN,MX]=(P,N,FS,FL,FH,W)
%
% Inputs:
%       p   number of filters in filterbank or the filter spacing in k-mel/bark/erb
[ceil(4.6*log10(fs))]
%       n   length of fft
%       fs  sample rate in Hz
%       fl  low end of the lowest filter as a fraction of fs [default = 0]
%       fh  high end of highest filter as a fraction of fs [default = 0.5]
%       w   any sensible combination of the following:
%             'b' = bark scale instead of mel
%             'e' = erb-rate scale
%             'l' = log10 Hz frequency scale
%             'f' = linear frequency scale
%
%             'c' = fl/fh specify centre of low and high filters
%             'h' = fl/fh are in Hz instead of fractions of fs
%             'H' = fl/fh are in mel/erb/bark/log10
%
%             't' = triangular shaped filters in mel/erb/bark domain (default)
%             'n' = hanning shaped filters in mel/erb/bark domain
%             'm' = hamming shaped filters in mel/erb/bark domain
%
%             'z' = highest and lowest filters taper down to zero [default]
%             'y' = lowest filter remains at 1 down to 0 frequency and
%                   highest filter remains at 1 up to nyquist freqency
%
%             'u' = scale filters to sum to unity
%
%             's' = single-sided: do not double filters to account for negative frequencies
%
%             'g' = plot idealized filters [default if no output arguments present]
%
% Note that the filter shape (triangular, hamming etc) is defined in the mel (or erb etc)
domain.
% Some people instead define an asymmetric triangular filter in the frequency domain.
%
%              If 'ty' or 'ny' is specified, the total power in the fft is preserved.
%
```

```
% Outputs:  x     a sparse matrix containing the filterbank amplitudes
%                 If the mn and mx outputs are given then size(x)=[p,mx-mn+1]
%                 otherwise size(x)=[p,1+floor(n/2)]
%                 Note that the peak filter values equal 2 to account for the power
%                 in the negative FFT frequencies.
%           mc    the filterbank centre frequencies in mel/erb/bark
%           mn    the lowest fft bin with a non-zero coefficient
%           mx    the highest fft bin with a non-zero coefficient
%                 Note: you must specify both or neither of mn and mx.
%
% Examples of use:
%
% (a) Calcuate the Mel-frequency Cepstral Coefficients
%
%       f=rfft(s);                  % rfft() returns only 1+floor(n/2) coefficients
%       x=melbankm(p,n,fs);         % n is the fft length, p is the number of filters wanted
%       z=log(x*abs(f).^2);         % multiply x by the power spectrum
%       c=dct(z);                   % take the DCT
%
% (b) Calcuate the Mel-frequency Cepstral Coefficients efficiently
%
%       f=fft(s);                        % n is the fft length, p is the number of filters wanted
%       [x,mc,na,nb]=melbankm(p,n,fs);   % na:nb gives the fft bins that are needed
%       z=log(x*(f(na:nb)).*conj(f(na:nb)));
%
% (c) Plot the calculated filterbanks
%
%      plot((0:floor(n/2))*fs/n,melbankm(p,n,fs)')   % fs=sample frequency
%
% (d) Plot the idealized filterbanks (without output sampling)
%
%      melbankm(p,n,fs);
%
% References:
%
% [1] S. S. Stevens, J. Volkman, and E. B. Newman. A scale for the measurement
%     of the psychological magnitude of pitch. J. Acoust Soc Amer, 8: 185–19, 1937.
% [2] S. Davis and P. Mermelstein. Comparison of parametric representations for
%     monosyllabic word recognition in continuously spoken sentences.
%     IEEE Trans Acoustics Speech and Signal Processing, 28 (4): 357–366, Aug. 1980.


%      Copyright (C) Mike Brookes 1997-2009
%      Version: $Id: melbankm.m,v 1.11 2010/01/02 20:02:22 dmb Exp $
%
%   VOICEBOX is a MATLAB toolbox for speech processing.
%   Home page: http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This program is free software; you can redistribute it and/or modify
%   it under the terms of the GNU General Public License as published by
%   the Free Software Foundation; either version 2 of the License, or
%   (at your option) any later version.
%
%   This program is distributed in the hope that it will be useful,
%   but WITHOUT ANY WARRANTY; without even the implied warranty of
%   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%   GNU General Public License for more details.
%
%   You can obtain a copy of the GNU General Public License from
%   http://www.gnu.org/copyleft/gpl.html or by writing to
%   Free Software Foundation, Inc.,675 Mass Ave, Cambridge, MA 02139, USA.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Note "FFT bin_0" assumes DC = bin 0 whereas "FFT bin_1" means DC = bin 1

if nargin < 6
    w='tz'; % default options
    if nargin < 5
        fh=0.5; % max freq is the nyquist
        if nargin < 4
            fl=0; % min freq is DC
        end
    end
end
sfact=2-any(w=='s');   % 1 if single sided else 2
wr=' ';   % default warping is mel
for i=1:length(w)
    if any(w(i)=='lebf');
        wr=w(i);
    end
end
if any(w=='h') || any(w=='H')
    mflh=[fl fh];
```

```
else
   mflh=[fl fh]*fs;
end
if ~any(w=='H')
   switch wr
               case 'f'      % no transformation
      case 'l'
         if fl<=0
            error('Low frequency limit must be >0 for l option');
         end
         mflh=log10(mflh);      % convert frequency limits into log10 Hz
      case 'e'
         mflh=frq2erb(mflh);      % convert frequency limits into erb-rate
      case 'b'
         mflh=frq2bark(mflh);      % convert frequency limits into bark
      otherwise
         mflh=frq2mel(mflh);      % convert frequency limits into mel
   end
end
melrng=mflh*(-1:2:1)';         % mel range
fn2=floor(n/2);     % bin index of highest positive frequency (Nyquist if n is even)
if isempty(p)
   p=ceil(4.6*log10(fs));         % default number of filters
end
if any(w=='c')              % c option: specify fiter centres not edges
if p<1
   p=round(melrng/(p*1000))+1;
end
melinc=melrng/(p-1);
mflh=mflh+(-1:2:1)*melinc;
else
   if p<1
   p=round(melrng/(p*1000))-1;
end
melinc=melrng/(p+1);
end

%
% Calculate the FFT bins corresponding to [filter#1-low filter#1-mid filter#p-mid
filter#p-high]
%
switch wr
      case 'f'
```

```
        blim=(mflh(1)+[0 1 p p+1]*melinc)*n/fs;
    case 'l'
        blim=10.^(mflh(1)+[0 1 p p+1]*melinc)*n/fs;
    case 'e'
        blim=erb2frq(mflh(1)+[0 1 p p+1]*melinc)*n/fs;
    case 'b'
        blim=bark2frq(mflh(1)+[0 1 p p+1]*melinc)*n/fs;
    otherwise
        blim=mel2frq(mflh(1)+[0 1 p p+1]*melinc)*n/fs;
end
mc=mflh(1)+(1:p)*melinc;    % mel centre frequencies
b1=floor(blim(1))+1;            % lowest FFT bin_0 required might be negative)
b4=min(fn2,ceil(blim(4))-1);    % highest FFT bin_0 required
%
% now map all the useful FFT bins_0 to filter1 centres
%
switch wr
        case 'f'
        pf=((b1:b4)*fs/n-mflh(1))/melinc;
    case 'l'
        pf=(log10((b1:b4)*fs/n)-mflh(1))/melinc;
    case 'e'
        pf=(frq2erb((b1:b4)*fs/n)-mflh(1))/melinc;
    case 'b'
        pf=(frq2bark((b1:b4)*fs/n)-mflh(1))/melinc;
    otherwise
        pf=(frq2mel((b1:b4)*fs/n)-mflh(1))/melinc;
end
%
%  remove any incorrect entries in pf due to rounding errors
%
if pf(1)<0
    pf(1)=[];
    b1=b1+1;
end
if pf(end)>=p+1
    pf(end)=[];
    b4=b4-1;
end
fp=floor(pf);                  % FFT bin_0 i contributes to filters_1 fp(1+i-b1)+[0 1]
pm=pf-fp;                      % multiplier for upper filter
k2=find(fp>0,1);   % FFT bin_1 k2+b1 is the first to contribute to both upper and lower
filters
```

```
k3=find(fp<p,1,'last');  % FFT bin_1 k3+b1 is the last to contribute to both upper and
lower filters
k4=numel(fp); % FFT bin_1 k4+b1 is the last to contribute to any filters
if isempty(k2)
    k2=k4+1;
end
if isempty(k3)
    k3=0;
end
if any(w=='y')          % preserve power in FFT
    mn=1; % lowest fft bin required (1 = DC)
    mx=fn2+1; % highest fft bin required (1 = DC)
    r=[ones(1,k2+b1-1) 1+fp(k2:k3) fp(k2:k3) repmat(p,1,fn2-k3-b1+1)]; % filter
number_1
    c=[1:k2+b1-1 k2+b1:k3+b1 k2+b1:k3+b1 k3+b1+1:fn2+1]; % FFT bin1
    v=[ones(1,k2+b1-1) pm(k2:k3) 1-pm(k2:k3) ones(1,fn2-k3-b1+1)];
else
    r=[1+fp(1:k3) fp(k2:k4)]; % filter number_1
    c=[1:k3 k2:k4]; % FFT bin_1 - b1
    v=[pm(1:k3) 1-pm(k2:k4)];
    mn=b1+1; % lowest fft bin_1
    mx=b4+1;  % highest fft bin_1
end
if b1<0
    c=abs(c+b1-1)-b1+1;     % convert negative frequencies into positive
end
% end
if any(w=='n')
    v=0.5-0.5*cos(v*pi);      % convert triangles to Hanning
elseif any(w=='m')
    v=0.5-0.46/1.08*cos(v*pi);  % convert triangles to Hamming
end
if sfact==2  % double all except the DC and Nyquist (if any) terms
    msk=(c+mn>2) & (c+mn<n-fn2+2);  % there is no Nyquist term if n is odd
    v(msk)=2*v(msk);
end
%
% sort out the output argument options
%
if nargout > 2
    x=sparse(r,c,v);
    if nargout == 3     % if exactly three output arguments, then
        mc=mn;          % delete mc output for legacy code compatibility
        mn=mx;
    end
else
    x=sparse(r,c+mn-1,v,p,1+fn2);
end
if any(w=='u')
    sx=sum(x,2);
    x=x./repmat(sx+(sx==0),1,size(x,2));
end
%
% plot results if no output arguments or g option given
%
if ~nargout || any(w=='g') % plot idealized filters
    ng=201;     % 201 points
    me=mflh(1)+(0:p+1)'*melinc;
    switch wr
              case 'f'
            fe=me; % defining frequencies
            xg=repmat(linspace(0,1,ng),p,1).*repmat(me(3:end)-me(1:end-
2),1,ng)+repmat(me(1:end-2),1,ng);
        case 'l'
            fe=10.^me; % defining frequencies
            xg=10.^(repmat(linspace(0,1,ng),p,1).*repmat(me(3:end)-me(1:end-
2),1,ng)+repmat(me(1:end-2),1,ng));
        case 'e'
            fe=erb2frq(me); % defining frequencies
            xg=erb2frq(repmat(linspace(0,1,ng),p,1).*repmat(me(3:end)-me(1:end-
2),1,ng)+repmat(me(1:end-2),1,ng));
        case 'b'
            fe=bark2frq(me); % defining frequencies
            xg=bark2frq(repmat(linspace(0,1,ng),p,1).*repmat(me(3:end)-me(1:end-
2),1,ng)+repmat(me(1:end-2),1,ng));
        otherwise
            fe=mel2frq(me); % defining frequencies
            xg=mel2frq(repmat(linspace(0,1,ng),p,1).*repmat(me(3:end)-me(1:end-
2),1,ng)+repmat(me(1:end-2),1,ng));
    end

    v=1-abs(linspace(-1,1,ng));
    if any(w=='n')
        v=0.5-0.5*cos(v*pi);      % convert triangles to Hanning
    elseif any(w=='m')
        v=0.5-0.46/1.08*cos(v*pi);  % convert triangles to Hamming
    end
    v=v*sfact;  % multiply by 2 if double sided
    v=repmat(v,p,1);
    if any(w=='y')  % extend first and last filters
        v(1,xg(1,:)<fe(2))=sfact;
        v(end,xg(end,:)>fe(p+1))=sfact;
    end
    if any(w=='u') % scale to unity sum
        dx=(xg(:,3:end)-xg(:,1:end-2))/2;
        dx=dx(:,[1 1:ng-2 ng-2]);
        vs=sum(v.*dx,2);
        v=v./repmat(vs+(vs==0),1,ng)*fs/n;
    end
    plot(xg',v','b');
    set(gca,'xlim',[fe(1) fe(end)]);
%     xlabel(['Frequency ('xticksi 'Hz)']);
end

function c=melcepst(s,fs,w,nc,p,n,inc,fl,fh)
%MELCEPST Calculate the mel cepstrum of a signal C=(S,FS,W,NC,P,N,INC,FL,FH)
%
%
% Simple use: c=melcepst(s,fs)  % calculate mel cepstrum with 12 coefs, 256 sample
frames
%                c=melcepst(s,fs,'e0dD') % include log energy, 0th cepstral coef, delta and
delta-delta coefs
%
% Inputs:
%     s  speech signal
%     fs  sample rate in Hz (default 11025)
%     nc  number of cepstral coefficients excluding 0'th coefficient (default 12)
%     n   length of frame in samples (default power of 2 < (0.03*fs))
%     p   number of filters in filterbank (default: floor(3*log(fs)) = approx 2.1 per ocatave)
%     inc frame increment (default n/2)
%     fl  low end of the lowest filter as a fraction of fs (default = 0)
%     fh  high end of highest filter as a fraction of fs (default = 0.5)
%
%       w   any sensible combination of the following:
%
%             'R'  rectangular window in time domain
%             'N' Hanning window in time domain
%             'M' Hamming window in time domain (default)
%
```

```
%            't'  triangular shaped filters in mel domain (default)
%            'n'  hanning shaped filters in mel domain
%            'm'  hamming shaped filters in mel domain
%
%              'p' filters act in the power domain
%              'a' filters act in the absolute magnitude domain (default)
%
%             '0'  include 0'th order cepstral coefficient
%              'e'  include log energy
%              'd' include delta coefficients (dc/dt)
%              'D' include delta-delta coefficients (d^2c/dt^2)
%
%            'z'  highest and lowest filters taper down to zero (default)
%            'y'  lowest filter remains at 1 down to 0 frequency and
%                highest filter remains at 1 up to nyquist freqency
%
%             If 'ty' or 'ny' is specified, the total power in the fft is preserved.
%
% Outputs:  c     mel cepstrum output: one frame per row. Log energy, if requested, is the
%                first element of each row followed by the delta and then the delta-delta
%                coefficients.
%

% BUGS: (1) should have power limit as 1e-16 rather than 1e-6 (or possibly a better way
of choosing this)
%          and put into VOICEBOX
%      (2) get rdct to change the data length (properly) instead of doing it explicitly
(wrongly)

%      Copyright (C) Mike Brookes 1997
%      Version: $Id: melcepst.m,v 1.7 2009/10/19 10:20:32 dmb Exp $
%
%   VOICEBOX is a MATLAB toolbox for speech processing.
%   Home page: http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This program is free software; you can redistribute it and/or modify
%   it under the terms of the GNU General Public License as published by
%   the Free Software Foundation; either version 2 of the License, or
%   (at your option) any later version.
%
%   This program is distributed in the hope that it will be useful,
%   but WITHOUT ANY WARRANTY; without even the implied warranty of
%   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%   GNU General Public License for more details.
%
%   You can obtain a copy of the GNU General Public License from
%   http://www.gnu.org/copyleft/gpl.html or by writing to
%   Free Software Foundation, Inc.,675 Mass Ave, Cambridge, MA 02139, USA.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if nargin<2 fs=11025; end
if nargin<3 w='M'; end
if nargin<4 nc=13; end
if nargin<5 p=floor(3*log(fs)); end
if nargin<6 n=256; end   %pow2(floor(log2(0.03*fs))); end
if nargin<9
   fh=0.5;
   if nargin<8
     fl=0;
     if nargin<7
        inc=floor(n/2);
     end
  end
end

if length(w)==0
   w='M';
end
if any(w=='R')
   z=enframe(s,n,inc);
elseif any (w=='N')
   z=enframe(s,hanning(n),inc);
else
   z=enframe(s,hamming(n),inc);
end
f=rfft(z.');
% pl = (0:length(f)-1)*50/length(f);
% figure, plot(pl,abs(f));
% title('magnitude');
[m,a,b]=melbankm(p,n,fs,fl,fh,w);
%figure,plot((0:floor(n/2))*fs/n,melbankm(p,n,fs)');
%xlabel('Frequency[Hz]');ylabel('Amplitude');
pw=f(a:b,:).*conj(f(a:b,:));
```

```
pth=max(pw(:))*1E-20;
if any(w=='p')
   y=log(max(m*pw,pth));
else
   ath=sqrt(pth);
   y=log(max(m*abs(f(a:b,:)),ath));
end
c=rdct(y).';
nf=size(c,1);
nc=nc+1;
if p>nc
   c(:,nc+1:end)=[];
elseif p<nc
   c=[c zeros(nf,nc-p)];
end
if ~any(w=='0')
   c(:,1)=[];
   nc=nc-1;
end
if any(w=='e')
   c=[log(sum(pw)).' c];
   nc=nc+1;
end

% calculate derivative

if any(w=='D')
  vf=(4:-1:-4)/60;
  af=(1:-1:-1)/2;
  ww=ones(5,1);
  cx=[c(ww,:); c; c(nf*ww,:)];
  vx=reshape(filter(vf,1,cx(:)),nf+10,nc);
  vx(1:8,:)=[];
  ax=reshape(filter(af,1,vx(:)),nf+2,nc);
  ax(1:2,:)=[];
  vx([1 nf+2],:)=[];
  if any(w=='d')
     c=[c vx ax];
  else
     c=[c ax];
  end
elseif any(w=='d')
  vf=(4:-1:-4)/60;
  ww=ones(4,1);
  cx=[c(ww,:); c; c(nf*ww,:)];
  vx=reshape(filter(vf,1,cx(:)),nf+8,nc);
  vx(1:8,:)=[];
  c=[c vx];
end

if nargout<1
   [nf,nc]=size(c);
   t=((0:nf-1)*inc+(n-1)/2)/fs;
   ci=(1:nc)-any(w=='0')-any(w=='e');
   imh = imagesc(t,ci,c.');
   axis('xy');
   xlabel('Time (s)');
   ylabel('Mel-cepstrum coefficient');
    map = (0:63)'/63;
    colormap([map map map]);
    colorbar;
end


function [cepstral] = mfcc1(x,y,fs)

% Calculate mfcc's with a frequency(fs) and store in ceptral cell. Display
% y at a time when x is calculated
cepstral = cell(size(x,1),1);
for i = 1:size(x,1)
% disp(y(i,:))
nc = 13;
p = 26;
n = 400;
inc = 240;
cepstral{i} = melcepst(x{i},fs,'E0dD',nc,p,n,inc);
end

%MELCEPST Calculate the mel cepstrum of a signal C=(S,FS,W,NC,P,N,INC,FL,FH)
%s   speech signal
%     fs  sample rate in Hz (default 11025)
%     nc  number of cepstral coefficients excluding 0'th coefficient (default 12)
%     n   length of frame in samples (default power of 2 < (0.03*fs))
%     p   number of filters in filterbank (default: floor(3*log(fs)) = approx 2.1 per ocatave)
%     inc frame increment (default n/2)
%     fl  low end of the lowest filter as a fraction of fs (default = 0)
%     fh  high end of highest filter as a fraction of fs (default = 0.5)


function c = pvsample(b, t, hop)
% c = pvsample(b, t, hop)   Interpolate an STFT array according to the 'phase vocoder'
%     b is an STFT array, of the form generated by 'specgram'.
%     t is a vector of (real) time-samples, which specifies a path through
%     the time-base defined by the columns of b.  For each value of t,
%     the spectral magnitudes in the columns of b are interpolated, and
%     the phase difference between the successive columns of b is
%     calculated; a new column is created in the output array c that
%     preserves this per-step phase advance in each bin.
%     hop is the STFT hop size, defaults to N/2, where N is the FFT size
%     and b has N/2+1 rows.  hop is needed to calculate the 'null' phase
%     advance expected in each bin.
%     Note: t is defined relative to a zero origin, so 0.1 is 90% of
%     the first column of b, plus 10% of the second.
% 2000-12-05 dpwe@ee.columbia.edu
% $Header: /homes/dpwe/public_html/resources/matlab/dtw/../RCS/pvsample.m,v 1.3
2003/04/09 03:17:10 dpwe Exp $

if nargin < 3
  hop = 0;
end

[rows,cols] = size(b);

N = 2*(rows-1);

if hop == 0
  % default value
  hop = N/2;
end

% Empty output array
c = zeros(rows, length(t));

% Expected phase advance in each bin
dphi = zeros(1,N/2+1);
dphi(2:(1 + N/2)) = (2*pi*hop)./(N./(1:(N/2)));

% Phase accumulator
% Preset to phase of first frame for perfect reconstruction
% in case of 1:1 time scaling
```

```
ph = angle(b(:,1));

% Append a 'safety' column on to the end of b to avoid problems
% taking *exactly* the last frame (i.e. 1*b(:,cols)+0*b(:,cols+1))
b = [b,zeros(rows,1)];

ocol = 1;
for tt = t
  % Grab the two columns of b
  bcols = b(:,floor(tt)+[1 2]);
  tf = tt - floor(tt);
  bmag = (1-tf)*abs(bcols(:,1)) + tf*(abs(bcols(:,2)));
  % calculate phase advance
  dp = angle(bcols(:,2)) - angle(bcols(:,1)) - dphi';
  % Reduce to -pi:pi range
  dp = dp - 2 * pi * round(dp/(2*pi));
  % Save the column
  c(:,ocol) = bmag .* exp(j*ph);
  % Cumulate phase, ready for next frame
  ph = ph + dphi' + dp;
  ocol = ocol+1;
end


function MFCC_Feature_Vector = normalise(X)
mindata = min(X);
maxdata = max(X);
MFCC_Feature_Vector = bsxfun(@rdivide, bsxfun(@minus, X, mindata), maxdata -
mindata);


function y=rdct(x,n,a,b)
%RDCT     Discrete cosine transform of real data Y=(X,N,A,B)
% Data is truncated/padded to length N.
%
% This routine is equivalent to multiplying by the matrix
%
%   rdct(eye(n)) = diag([sqrt(2)*B/A repmat(2/A,1,n-1)]) * cos((0:n-1)'*(0.5:n)*pi/n)
%
% Default values of the scaling factors are A=sqrt(2N) and B=1 which
% results in an orthogonal matrix. Other common values are A=1 or N and/or B=1 or
sqrt(2).
% If b~=1 then the columns are no longer orthogonal.
```

```
%
% see IRDCT for the inverse transform

% BUG: in line 51 we should do chopping after transform and not before



%      Copyright (C) Mike Brookes 1998
%      Version: $Id: rdct.m,v 1.6 2007/05/04 07:01:39 dmb Exp $
%
%   VOICEBOX is a MATLAB toolbox for speech processing.
%   Home page: http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This program is free software; you can redistribute it and/or modify
%   it under the terms of the GNU General Public License as published by
%   the Free Software Foundation; either version 2 of the License, or
%   (at your option) any later version.
%
%   This program is distributed in the hope that it will be useful,
%   but WITHOUT ANY WARRANTY; without even the implied warranty of
%   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%   GNU General Public License for more details.
%
%   You can obtain a copy of the GNU General Public License from
%   http://www.gnu.org/copyleft/gpl.html or by writing to
%   Free Software Foundation, Inc.,675 Mass Ave, Cambridge, MA 02139, USA.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fl=size(x,1)==1;
if fl x=x(:); end
[m,k]=size(x);
if nargin<2 n=m;
end
if nargin<4 b=1;
   if nargin<3 a=sqrt(2*n);
   end
   end
if n>m x=[x; zeros(n-m,k)];
elseif n<m x(n+1:m,:)=[];
end
```

```
x=[x(1:2:n,:); x(2*fix(n/2):-2:2,:)];
z=[sqrt(2) 2*exp((-0.5i*pi/n)*(1:n-1))].';
y=real(fft(x).*z(:,ones(1,k)))/a;
y(1,:)=y(1,:)*b;
if fl y=y.'; end

function A = resize(B,R,C)
% A = resize(B,R,C)   Crop or zero-pad B to have R rows and C columns.
%   I'm sure this must already be provided, but how to know?
% dpwe 1995jan21

% Copyright (c) 1995 Dan Ellis <dpwe@ee.columbia.edu>
% released under GPL - see file COPYRIGHT

A = zeros(R,C);
[r,c] = size(B);

mr = min(r,R);
mc = min(c,C);

A(1:mr,1:mc) = B(1:mr, 1:mc);


function y=rfft(x,n,d)
%RFFT     Calculate the DFT of real data Y=(X,N,D)
% Data is truncated/padded to length N if specified.
%   N even: (N+2)/2 points are returned with
%           the first and last being real
%   N odd:  (N+1)/2 points are returned with the
%           first being real
% In all cases fix(1+N/2) points are returned
% D is the dimension along which to do the DFT



%      Copyright (C) Mike Brookes 1998
%      Version: $Id: rfft.m,v 1.7 2009/06/03 11:57:52 dmb Exp $
%
%   VOICEBOX is a MATLAB toolbox for speech processing.
%   Home page: http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%   This program is free software; you can redistribute it and/or modify
%   it under the terms of the GNU General Public License as published by
%   the Free Software Foundation; either version 2 of the License, or
%   (at your option) any later version.
%
%   This program is distributed in the hope that it will be useful,
%   but WITHOUT ANY WARRANTY; without even the implied warranty of
%   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%   GNU General Public License for more details.
%
%   You can obtain a copy of the GNU General Public License from
%   http://www.gnu.org/copyleft/gpl.html or by writing to
%   Free Software Foundation, Inc.,675 Mass Ave, Cambridge, MA 02139, USA.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

s=size(x);
if prod(s)==1
    y=x;
else
    if nargin <3 || isempty(d)
        d=find(s>1);
        d=d(1);
        if nargin<2
            n=s(d);
        end
    end
    if isempty(n)
        n=s(d);
    end
    y=fft(x,n,d);
    y=reshape(y,prod(s(1:d-1)),n,prod(s(d+1:end)));
    s(d)=1+fix(n/2);
    y(:,s(d)+1:end,:)=[];
    y=reshape(y,s);
end
```

```
function M = simmx(A,B)
% M = simmx(A,B)
%    calculate a sim matrix between specgram-like feature matrices A and B.
%    size(M) = [size(A,2) size(B,2)]; A and B have same #rows.
% 2003-03-15 dpwe@ee.columbia.edu

% Copyright (c) 2003 Dan Ellis <dpwe@ee.columbia.edu>
% released under GPL - see file COPYRIGHT

EA = sqrt(sum(A.^2));
EB = sqrt(sum(B.^2));

%ncA = size(A,2);
%ncB = size(B,2);
%M = zeros(ncA, ncB);
%for i = 1:ncA
%  for j = 1:ncB
%    % normalized inner product i.e. cos(angle between vectors)
%    M(i,j) = (A(:,i)'*B(:,j))/(EA(i)*EB(j));
%  end
%end

% this is 10x faster
M = (A'*B)./(EA'*EB);


classdef svm
    methods (Static)
        function Model=train(training, groupnames, varargin)
            %SVMTRAIN Train a support vector machine classifier
            %   SVMSTRUCT = SVMTRAIN(TRAINING, Y) trains a support vector
machine (SVM)
            %   classifier on data taken from two groups. TRAINING is a numeric matrix
            %   of predictor data. Rows of TRAINING correspond to observations; columns
            %   correspond to features. Y is a column vector that contains the known
            %   class labels for TRAINING. Y is a grouping variable, i.e., it can be a
            %   categorical, numeric, or logical vector; a cell vector of strings; or a
            %   character matrix with each row representing a class label (see help for
            %   groupingvariable). Each element of Y specifies the group the
            %   corresponding row of TRAINING belongs to. TRAINING and Y must have
the
            %   same number of rows. SVMSTRUCT contains information about the trained
            %   classifier, including the support vectors, that is used by SVMCLASSIFY
            %   for classification. SVMTRAIN treats NaNs, empty strings or 'undefined'
            %   values as missing values and ignores the corresponding rows in
            %   TRAINING and Y.
            %
            %   SVMSTRUCT = SVMTRAIN(TRAINING, Y, 'PARAM1',val1,
'PARAM2',val2, ...)
            %   specifies one or more of the following name/value pairs:
            %
            %     Name               Value
            %     'kernel_function'  A string or a function handle specifying the
            %                        kernel function used to represent the dot
            %                        product in a new space. The value can be one of
            %                        the following:
            %                        'linear'     - Linear kernel or dot product
            %                                       (default). In this case, SVMTRAIN
            %                                       finds the optimal separating plane
            %                                       in the original space.
            %                        'quadratic'  - Quadratic kernel
            %                        'polynomial' - Polynomial kernel with default
            %                                       order 3. To specify another order,
            %                                       use the 'polyorder' argument.
            %                        'rbf'        - Gaussian Radial Basis Function
            %                                       with default scaling factor 1. To
            %                                       specify another scaling factor,
            %                                       use the 'rbf_sigma' argument.
            %                        'mlp'        - Multilayer Perceptron kernel (MLP)
            %                                       with default weight 1 and default
            %                                       bias -1. To specify another weight
            %                                       or bias, use the 'mlp_params'
            %                                       argument.
            %                        function     - A kernel function specified using
            %                                       @(for example @KFUN), or an
            %                                       anonymous function. A kernel
            %                                       function must be of the form
            %
            %                                       function K = KFUN(U, V)
            %
            %                                       The returned value, K, is a matrix
            %                                       of size M-by-N, where M and N are
            %                                       the number of rows in U and V
            %                                       respectively.
            %
            %   'rbf_sigma'          A positive number specifying the scaling factor
            %                    in the Gaussian radial basis function kernel.
            %                    Default is 1.
            %
            %   'polyorder'        A positive integer specifying the order of the
            %                    polynomial kernel. Default is 3.
            %
            %   'mlp_params'        A vector [P1 P2] specifying the parameters of MLP
            %                    kernel.  The MLP kernel takes the form:
            %                    K = tanh(P1*U*V' + P2),
            %                    where P1 > 0 and P2 < 0. Default is [1,-1].
            %
            %   'method'           A string specifying the method used to find the
            %                    separating hyperplane. Choices are:
            %                    'SMO' - Sequential Minimal Optimization (SMO)
            %                           method (default). It implements the L1
            %                           soft-margin SVM classifier.
            %                    'QP'  - Quadratic programming (requires an
            %                           Optimization Toolbox license). It
            %                           implements the L2 soft-margin SVM
            %                           classifier. Method 'QP' doesn't scale
            %                           well for TRAINING with large number of
            %                           observations.
            %                    'LS'  - Least-squares method. It implements the
            %                           L2 soft-margin SVM classifier.
            %
            %   'options'          Options structure created using either STATSET or
            %                    OPTIMSET.
            %                    * When you set 'method' to 'SMO' (default),
            %                      create the options structure using STATSET.
            %                      Applicable options:
            %                      'Display'  Level of display output.  Choices
            %                               are 'off' (the default), 'iter', and
            %                               'final'. Value 'iter' reports every
            %                               500 iterations.
            %                      'MaxIter'  A positive integer specifying the
            %                               maximum number of iterations allowed.
            %                               Default is 15000 for method 'SMO'.
            %                    * When you set method to 'QP', create the
            %                      options structure using OPTIMSET. For details
            %                      of applicable options choices, see QUADPROG
            %                      options. SVM uses a convex quadratic program,
            %                      so you can choose the 'interior-point-convex'
            %                      algorithm in QUADPROG.
```

```
%
%  'tolkkt'            A positive scalar that specifies the tolerance
%                      with which the Karush-Kuhn-Tucker (KKT) conditions
%                      are checked for method 'SMO'. Default is
%                      1.0000e-003.
%
%  'kktviolationlevel'   A scalar specifying the fraction of observations
%                      that are allowed to violate the KKT conditions for
%                      method 'SMO'. Setting this value to be positive
%                      helps the algorithm to converge faster if it is
%                      fluctuating near a good solution. Default is 0.
%
%  'kernelcachelimit'    A positive scalar S specifying the size of the
%                      kernel matrix cache for method 'SMO'. The
%                      algorithm keeps a matrix with up to S * S
%                      double-precision numbers in memory. Default is
%                      5000. When the number of points in TRAINING
%                      exceeds S, the SMO method slows down. It's
%                      recommended to set S as large as your system
%                      permits.
%
%  'boxconstraint'       The box constraint C for the soft margin. C can be
%                      a positive numeric scalar or a vector of positive
%                      numbers with the number of elements equal to the
%                      number of rows in TRAINING.
%                      Default is 1.
%                      * If C is a scalar, it is automatically rescaled
%                        by N/(2*N1) for the observations of group one,
%                        and by N/(2*N2) for the observations of group
%                        two, where N1 is the number of observations in
%                        group one, N2 is the number of observations in
%                        group two. The rescaling is done to take into
%                        account unbalanced groups, i.e., when N1 and N2
%                        are different.
%                      * If C is a vector, then each element of C
%                        specifies the box constraint for the
%                        corresponding observation.
%
%   'autoscale'          A logical value specifying whether or not to
%                      shift and scale the data points before training.
%                      When the value is true, the columns of TRAINING
%                      are shifted and scaled to have zero mean unit
%                      variance. Default is true.
```

```
%
%   'showplot'         A logical value specifying whether or not to show
%                     a plot. When the value is true, SVMTRAIN creates a
%                     plot of the grouped data and the separating line
%                     for the classifier, when using data with 2
%                     features (columns). Default is false.
%
%   SVMSTRUCT is a structure having the following properties:
%
%   SupportVectors       Matrix of data points with each row corresponding
%                     to a support vector.
%                     Note: when 'autoscale' is false, this field
%                     contains original support vectors in TRAINING.
%                     When 'autoscale' is true, this field contains
%                     shifted and scaled vectors from TRAINING.
%   Alpha               Vector of Lagrange multipliers for the support
%                     vectors. The sign is positive for support vectors
%                     belonging to the first group and negative for
%                     support vectors belonging to the second group.
%   Bias               Intercept of the hyperplane that separates
%                     the two groups.
%                     Note: when 'autoscale' is false, this field
%                     corresponds to the original data points in
%                     TRAINING. When 'autoscale' is true, this field
%                     corresponds to shifted and scaled data points.
%   KernelFunction       The function handle of kernel function used.
%   KernelFunctionArgs   Cell array containing the additional arguments
%                     for the kernel function.
%   GroupNames           A column vector that contains the known
%                     class labels for TRAINING. Y is a grouping
%                     variable (see help for groupingvariable).
%   SupportVectorIndices A column vector indicating the indices of support
%                     vectors.
%   ScaleData            This field contains information about auto-scale.
%                     When 'autoscale' is false, it is empty. When
%                     'autoscale' is set to true, it is a structure
%                     containing two fields:
%                     shift       - A row vector containing the negative
%                                   of the mean across all observations
%                                   in TRAINING.
%                     scaleFactor - A row vector whose value is
%                                   1./STD(TRAINING).
%   FigureHandles        A vector of figure handles created by SVMTRAIN
%                    when 'showplot' argument is TRUE.
%
%   Example:
%      % Load the data and select features for classification
%      load fisheriris
%      X = [meas(:,1), meas(:,2)];
%      % Extract the Setosa class
%      Y = nominal(ismember(species,'setosa'));
%      % Randomly partitions observations into a training set and a test
%      % set using stratified holdout
%      P = cvpartition(Y,'Holdout',0.20);
%      % Use a linear support vector machine classifier
%      svmStruct = msvmtrain(X(P.training,:),Y(P.training),'showplot',true);
%      C = msvmclassify(svmStruct,X(P.test,:),'showplot',true);
%      errRate = sum(Y(P.test)~= C)/P.TestSize  %mis-classification rate
%      conMat = confusionmat(Y(P.test),C) % the confusion matrix
%
%   See also SVMCLASSIFY, NAIVEBAYES, CLASSREGTREE, CLASSIFY,
TREEBAGGER,
%          GROUPINGVARIABLE

%   Copyright 2004-2012 The MathWorks, Inc.


%   References:
%
%     [1] Cristianini, N., Shawe-Taylor, J An Introduction to Support
%         Vector Machines, Cambridge University Press, Cambridge, UK. 2000.
%         http://www.support-vector.net
%     [2] Kecman, V, Learning and Soft Computing,
%         MIT Press, Cambridge, MA. 2001.
%     [3] Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B.,
%         Vandewalle, J., Least Squares Support Vector Machines,
%         World Scientific, Singapore, 2002.
%     [4] J.C. Platt: A Fast Algorithm for Training  Support Vector
%         Machines,  Advances in Kernel Methods - Support Vector Learning,
%         MIT Press, 1998.
%     [5] J.C. Platt: Fast Training of Support Vector Machines using
%         Sequential Minimal Optimization Microsoft Research Technical
%         Report MSR-TR-98-14, 1998.
%     [6] http://www.kernel-machines.org/papers/tr-30-1998.ps.gz
%
%   SVMTRAIN(...,'KFUNARGS',ARGS) allows you to pass additional
```

```
        %   arguments to kernel functions.
        %
        %     Code is modified for multuclass svm
        %        by Er.Abbas Manthiri BE
        %     Email abbasmanthiribe@gmail.com
        %     Date:15-03-2017

        classInstance=unique(groupnames);
        svmValue=sum(classInstance);
        nsample=length(classInstance);
        if nsample>2
            model=cell(1,nsample);
            for i=1:nsample
                classx=groupnames;
                classx(classx==classInstance(i))=svmValue;
                classx(classx~=svmValue)=1;
                classx(classx==svmValue)=0;
                model{i}=svmtrain(training,classx,varargin{:});
                fprintf('Multi Class SVM Model for Class Instance %d ---
>\n',classInstance(i))
                disp(model{i})
            end
        else
            model=svmtrain(training,groupnames,varargin{:});
            fprintf('\nx Two class svm  Model--->\n')
            disp(model)
        end
        Model.model=model;
        Model.classInstance=classInstance;
        fprintf('\nTrain Model Completed\n')

    end

    function [output,matrix]=predict(Model,sample,varargin)
        %SVMCLASSIFY Classify data using a support vector machine
        %   GROUP = SVMCLASSIFY(SVMSTRUCT, TEST) classifies each row in
TEST using
        %   the support vector machine classifier structure SVMSTRUCT created
        %   using SVMTRAIN, and returns the predicted class level GROUP. TEST must
        %   have the same number of columns as the data used to train the
        %   classifier in SVMTRAIN. GROUP indicates the group to which each row of
        %   TEST is assigned.
        %
        %   GROUP = SVMCLASSIFY(...,'SHOWPLOT',true) plots the test data TEST 
on
        %   the figure created using the SHOWPLOT option in SVMTRAIN.
        %
        %   Example:
        %       % Load the data and select features for classification
        %       load fisheriris
        %       X = [meas(:,1), meas(:,2)];
        %       % Extract the Setosa class
        %       Y = nominal(ismember(species,'setosa'));
        %       % Randomly partitions observations into a training set and a test
        %       % set using stratified holdout
        %       P = cvpartition(Y,'Holdout',0.20);
        %       % Use a linear support vector machine classifier
        %       svmStruct = msvmtrain(X(P.training,:),Y(P.training),'showplot',true);
        %       C = msvmclassify(svmStruct,X(P.test,:),'showplot',true);
        %       err_rate = sum(Y(P.test)~= C)/P.TestSize % mis-classification rate
        %       conMat = confusionmat(Y(P.test),C) % the confusion matrix
        %
        %   See also SVMTRAIN, NAIVEBAYES, CLASSREGTREE, CLASSIFY, 
TREEBAGGER

        %   Copyright 2004-2012 The MathWorks, Inc.


        %   References:
        %
        %     [1] Cristianini, N., Shawe-Taylor, J An Introduction to Support
        %         Vector Machines, Cambridge University Press, Cambridge, UK. 2000.
        %         http://www.support-vector.net
        %     [2] Kecman, V, Learning and Soft Computing,
        %         MIT Press, Cambridge, MA. 2001.
        %     [3] Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B.,
        %         Vandewalle, J., Least Squares Support Vector Machines,
        %         World Scientific, Singapore, 2002.
        %
        %     Code is modified for multuclass svm
        %         by Er.Abbas Manthiri BE
        %     Email abbasmanthiribe@gmail.com
        %     Date:15-03-2017
        model=Model.model;
        classInstance=Model.classInstance;
        nsample=length(classInstance);
```

```
            if nsample>2
                numberOfSamples=size(sample,1);
                classRange=zeros(numberOfSamples,length(classInstance));
                for i=1:nsample
                    [~,threshold]=svm.svmclassify(model{i},sample,varargin{:});
                    classRange(:,i)=threshold;
                    fprintf('\nMulti Class SVM classify values Claculated  for Class Instance %d
',classInstance(i))
                end
                [~,index]=max(transpose(classRange));
                output=classInstance(index);
                matrix = classRange;
            else
                output=svm.svmclassify(model,sample,varargin{:});
            end
            fprintf('\n SVM Classification is completed\n')
        end

        function [outclass,val] = svmclassify(svmStruct,sample, varargin)

            % set defaults
            plotflag = false;

            % check inputs
            narginchk(2, Inf);

            % deal with struct input case
            if ~isstruct(svmStruct)
                error(message('stats:svmclassify:TwoInputsNoStruct'));
            end

            if ~isnumeric(sample) || ~ismatrix(sample)
                error(message('stats:svmclassify:BadSample'));
            end

            if size(sample,2)~=size(svmStruct.SupportVectors,2)
                error(message('stats:svmclassify:TestSizeMismatch'));
            end

            % deal with the various inputs
            if nargin > 2
                if rem(nargin,2) == 1
                    error(message('stats:svmclassify:IncorrectNumberOfArguments'));
                end
                okargs = {'showplot','-compilerhelper'};
                for j=1:2:nargin-2
                    pname = varargin{j};
                    pval = varargin{j+1};
                    k = find(strncmpi(pname, okargs,numel(pname)));
                    if isempty(k)
                        error(message('stats:svmclassify:UnknownParameterName', pname));
                    elseif length(k)>1
                        error(message('stats:svmclassify:AmbiguousParameterName', pname));
                    else
                        switch(k)
                            case 1 % plotflag ('SHOWPLOT')
                                plotflag = opttf(pval,okargs{k});
                            case 2 % help the compiler find required function handles by including
svmtrain
                                svmtrain(eye(2),[1 0]);
                        end
                    end
                end
            end

            groupnames = svmStruct.GroupNames;

            % check group is a vector -- though char input is special...
            if ~isvector(groupnames) && ~ischar(groupnames)
                error(message('stats:svmclassify:GroupNotVector'));
            end

            % grp2idx sorts a numeric grouping var ascending, and a string grouping
            % var by order of first occurrence
            [~,groupString,glevels] = grp2idx(groupnames);

            % do the classification
            if ~isempty(sample)
                % shift and scale the data if necessary:
                sampleOrig = sample;
                if ~isempty(svmStruct.ScaleData)
                    for c = 1:size(sample, 2)
                        sample(:,c) = svmStruct.ScaleData.scaleFactor(c) * ...
                            (sample(:,c) +  svmStruct.ScaleData.shift(c));
                    end
                end
```

```
%     try
[outclass,val] = svm.svmdecision(sample,svmStruct);
%     catch ME
%         error(message('stats:svmclassify:ClassifyFailed', ME.message));
%     end
if plotflag

    if isempty(svmStruct.FigureHandles)
        warning(message('stats:svmclassify:NoTrainingFigure'));

    else
        try
            hAxis = svmStruct.FigureHandles{1};
            hLines = svmStruct.FigureHandles{2};
            hSV = svmStruct.FigureHandles{3};
            % unscale the data for plotting purposes
            [~,hClassLines] = svmplotdata(sampleOrig,outclass,hAxis);
            trainingString = strcat(cellstr(groupString),' (training)');
            sampleString = strcat(cellstr(groupString),' (classified)');
            legend([hLines(1),hClassLines(1),hLines(2),hClassLines(2),hSV],...
                {trainingString{1},sampleString{1},...
                trainingString{2},sampleString{2},'Support Vectors'});
        catch ME
            warning(message('stats:svmclassify:DisplayFailed', ME.message));
        end
    end
end
outclass(outclass == -1) = 2;
unClassified = isnan(outclass);
outclass = glevels(outclass(~unClassified),:);
if any(unClassified)

    try
        outclass = statinsertnan(unClassified,outclass);
    catch ME
        if ~isequal(ME.identifier,'stats:statinsertnan:LogicalInput')
            rethrow(ME);
        else
            error(message('stats:svmclassify:logicalwithNaN'));
        end
    end
end
```

```
            else
                outclass = [];
            end

        end
        function [out,f] = svmdecision(Xnew,svm_struct)
            %SVMDECISION Evaluates the SVM decision function

            %   Copyright 2004-2012 The MathWorks, Inc.


            sv = svm_struct.SupportVectors;
            alphaHat = svm_struct.Alpha;
            bias = svm_struct.Bias;
            kfun = svm_struct.KernelFunction;
            kfunargs = svm_struct.KernelFunctionArgs;
            f = (feval(kfun,sv,Xnew,kfunargs{:})'*alphaHat(:)) + bias;
            out = sign(f);
            % points on the boundary are assigned to class 1
            out(out==0) = 1;
        end


        function [Model,predicted] = classify(Sample,class,SampleTest)
            Model=svm.train(Sample,class);
            predicted=svm.predict(Model,SampleTest);
        end
    end
end


% buildDetector: build face parts detector object
%
% detector = buildDetector( thresholdFace, thresholdParts, stdsize )
%
%Output parameter:
% detector: built detector object
%
%
%Input parameters:
% thresholdFace (optional): MergeThreshold for face detector (Default: 1)
% thresholdParts (optional): MergeThreshold for face parts detector (Default: 1)
% stdsize (optional): size of normalized face (Default: 176)
```

```
%
%
%Example:
% detector = buildDetector();
% img = imread('img.jpg');
% [bbbox bbimg] = detectFaceParts(detector,img);
%
%
%Version: 20120529

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Face Parts Detection:                    %
%                                          %
% Copyright (C) 2012 Masayuki Tanaka. All rights reserved. %
%             mtanaka@ctrl.titech.ac.jp        %
%                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function detector = buildDetector( thresholdFace, thresholdParts, stdsize )

if( nargin < 1 )
 thresholdFace = 1;
end

if( nargin < 2 )
 thresholdParts = 1;
end

if( nargin < 3 )
 stdsize = 176;
end

nameDetector = {'LeftEye'; 'RightEye'; 'Mouth'; 'Nose'; };
mins = [[12 18]; [12 18]; [15 25]; [15 18]; ];

detector.stdsize = stdsize;
detector.detector = cell(5,1);
for k=1:4
 minSize = int32([stdsize/5 stdsize/5]);
 minSize = [max(minSize(1),mins(k,1)), max(minSize(2),mins(k,2))];
 detector.detector{k} = vision.CascadeObjectDetector(char(nameDetector(k)),
'MergeThreshold', thresholdParts, 'MinSize', minSize);
```

```
end

detector.detector{5} = vision.CascadeObjectDetector('FrontalFaceCART',
'MergeThreshold', thresholdFace);


% detectFaceParts: detect faces with parts
%
% [bbox,bbX,faces,bbfaces] = detectFaceParts(detector,X,thick)
%
%Output parameters:
% bbox: bbox(:, 1: 4) is bounding box for face
%       bbox(:, 5: 8) is bounding box for left eye
%       bbox(:, 9:12) is bounding box for right eye
%       bbox(:,13:16) is bounding box for mouth
%       bbox(:,17:20) is bounding box for nose
%       please see the documentation of the computer vision toolbox for details of the
bounding box.
% bbX: image with found faces which are shown as boxes
% faces: found faces stored as cell array
% bbfaces: found faces with boxes stored as cell array
%
%
%Input parameters:
% detector: the detection object built by buildDetector
% X: image data which should be uint8
% thick(optional): thickness of bounding box (default:1)
%
%
%Example:
% detector = buildDetector();
% img = imread('img.jpg');
% [bbbox bbimg] = detectFaceParts(detector,img);
%
%
%Version: 20120529

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Face Parts Detection:                          %
%                                              %
% Copyright (C) 2012 Masayuki Tanaka. All rights reserved. %
%                  mtanaka@ctrl.titech.ac.jp             %
%                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [bbox,bbX,faces,bbfaces] = detectFaceParts(detector,X,thick)

if( nargin < 3 )
 thick = 1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% detect face
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Detect faces
bbox = step(detector.detector{5}, X);

bbsize = size(bbox);
partsNum = zeros(size(bbox,1),1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% detect parts
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nameDetector = {'LeftEye'; 'RightEye'; 'Mouth'; 'Nose'; };
mins = [[12 18]; [12 18]; [15 25]; [15 18]; ];

stdsize = detector.stdsize;

for k=1:4
 if( k == 1 )
  region = [1,int32(stdsize*2/3); 1, int32(stdsize*2/3)];
 elseif( k == 2 )
  region = [int32(stdsize/3),stdsize; 1, int32(stdsize*2/3)];
 elseif( k == 3 )
  region = [1,stdsize; int32(stdsize/3), stdsize];
 elseif( k == 4 )
  region = [int32(stdsize/5),int32(stdsize*4/5); int32(stdsize/3),stdsize];
 else
  region = [1,stdsize;1,stdsize];
 end

 bb = zeros(bbsize);
 for i=1:size(bbox,1)
  XX = X(bbox(i,2):bbox(i,2)+bbox(i,4)-1,bbox(i,1):bbox(i,1)+bbox(i,3)-1,:);
  XX = imresize(XX,[stdsize, stdsize]);
  XX = XX(region(2,1):region(2,2),region(1,1):region(1,2),:);
  b = step(detector.detector{k},XX);

  if( size(b,1) > 0 )
   partsNum(i) = partsNum(i) + 1;

   if( k == 1 )
    b = sortrows(b,1);
   elseif( k == 2 )
    b = flipud(sortrows(b,1));
   elseif( k == 3 )
    b = flipud(sortrows(b,2));
   elseif( k == 4 )
    b = flipud(sortrows(b,3));
   end

   ratio = double(bbox(i,3)) / double(stdsize);
   b(1,1) = int32( ( b(1,1)-1 + region(1,1)-1 ) * ratio + 0.5 ) + bbox(i,1);
   b(1,2) = int32( ( b(1,2)-1 + region(2,1)-1 ) * ratio + 0.5 ) + bbox(i,2);
   b(1,3) = int32( b(1,3) * ratio + 0.5 );
   b(1,4) = int32( b(1,4) * ratio + 0.5 );

   bb(i,:) = b(1,:);
  end
 end
 bbox = [bbox,bb];

 p = ( sum(bb') == 0 );
 bb(p,:) = [];
end


%%%%%%%%%%%%%%%%%%%%%%%%%% draw faces
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
bbox = [bbox,partsNum];
bbox(partsNum<=2,:)=[];

if( thick >= 0 )
 t = (thick-1)/2;
 t0 = -int32(ceil(t));
 t1 = int32(floor(t));
else
 t0 = 0;
 t1 = 0;
```

```
end

bbX = X;
boxColor = [[0,255,0]; [255,0,255]; [255,0,255]; [0,255,255]; [255,255,0]; ];
for k=5:-1:1
 shapeInserter =
vision.ShapeInserter('BorderColor','Custom','CustomBorderColor',boxColor(k,:));
 for i=t0:t1
  bb = int32(bbox(:,(k-1)*4+1:k*4));
%   bb(:,1:2) = bb(:,1:2)-i;
  bb(:,3:4) = bb(:,3:4)+i*2;
  bbX = step(shapeInserter, bbX, bb);
 end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%% faces
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if( nargout > 2 )
 faces = cell(size(bbox,1),1);
 boxfaces = cell(size(bbox,1),1);
 for i=1:size(bbox,1)
  faces{i,1} = X(bbox(i,2):bbox(i,2)+bbox(i,4)-1,bbox(i,1):bbox(i,1)+bbox(i,3)-1,:);
  bbfaces{i,1} = bbX(bbox(i,2):bbox(i,2)+bbox(i,4)-1,bbox(i,1):bbox(i,1)+bbox(i,3)-1,:);
 end
end
```

Appendix B:

Stand-alone Speech Recognition System results for combination 1 database:

20 utterances were tested for each participant. The entries highlighted in Green indicate correct recognition.

Participant 1	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	2
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9

Participant 2	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	3
	4	4
	5	3
	6	6
	7	7
	8	8
	9	9
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	8
	7	7
	8	8
	9	9

Participant 3	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9
	0	9
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9

Participant 4	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9
	0	0
	1	1
	2	8
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9

Participant 5	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9
	0	0
	1	9
	2	2
	3	5
	4	5
	5	5
	6	6
	7	7
	8	8
	9	9

Participant 6	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	0
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9

Participant 7	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	5
	9	9
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9

Participant 8	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9
	0	8
	1	0
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9

Participant 9	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9
	0	0
	1	1
	2	9
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9

Participant 10	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9
	0	9
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9

Stand-alone Lip Reading System results for combination 1 database:

Participant Number 1	Digit Spoken	Digit Recognized
	0	8
	1	1
	2	1
	3	3
	4	8
	5	5
	6	6
	7	0
	8	8
	9	0
	0	8
	1	1
	2	8
	3	3
	4	4
	5	8
	6	6
	7	8
	8	7
	9	9

Participant Number 2	Digit Spoken	Digit Recognized
	0	0
	1	5
	2	2
	3	8
	4	4
	5	8
	6	6
	7	8
	8	0
	9	9
	0	0
	1	8
	2	4
	3	3
	4	4
	5	5
	6	8
	7	7
	8	8
	9	9

Participant Number 3	Digit Spoken	Digit Recognized
	0	0
	1	8
	2	8
	3	3
	4	4
	5	0
	6	2
	7	7
	8	8
	9	4
	0	0
	1	1
	2	2
	3	2
	4	4
	5	8
	6	1
	7	7
	8	8
	9	9

Participant Number 4	Digit Spoken	Digit Recognized
	0	2
	1	1
	2	8
	3	8
	4	4
	5	5
	6	6
	7	7
	8	8
	9	2
	0	0
	1	8
	2	8
	3	3
	4	8
	5	5
	6	8
	7	7
	8	6
	9	3

Participant Number 5	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	5
	3	8
	4	4
	5	5
	6	3
	7	8
	8	8
	9	9
	0	0
	1	1
	2	8
	3	3
	4	5
	5	8
	6	6
	7	0
	8	8
	9	9

Participant Number 6	Digit Spoken	Digit Recognized
	0	0
	1	7
	2	2
	3	8
	4	8
	5	5
	6	8
	7	8
	8	8
	9	8
	0	8
	1	1
	2	5
	3	3
	4	8
	5	8
	6	6
	7	7
	8	8
	9	8

Participant Number 7	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	8
	3	3
	4	4
	5	5
	6	8
	7	8
	8	8
	9	8
	0	0
	1	8
	2	0
	3	8
	4	4
	5	0
	6	5
	7	7
	8	8
	9	9

Participant Number 8	Digit Spoken	Digit Recognized
	0	0
	1	5
	2	2
	3	7
	4	8
	5	5
	6	6
	7	8
	8	8
	9	1
	0	0
	1	8
	2	8
	3	1
	4	4
	5	5
	6	7
	7	5
	8	8
	9	1

Participant Number 9	Digit Spoken	Digit Recognized
	0	4
	1	1
	2	6
	3	3
	4	4
	5	5
	6	1
	7	7
	8	1
	9	9
	0	0
	1	1
	2	8
	3	1
	4	4
	5	3
	6	8
	7	7
	8	1
	9	9

Participant Number 10	Digit Spoken	Digit Recognized
	0	1
	1	1
	2	2
	3	8
	4	4
	5	6
	6	6
	7	8
	8	8
	9	8
	0	8
	1	8
	2	2
	3	3
	4	4
	5	5
	6	8
	7	8
	8	4
	9	9

Fusion-based Audio-Visual Speech Recognition System results for combination 1 database:

Participant 1	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	9
	7	7
	8	8
	9	9
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9

Participant 2	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9
	0	1
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9

Participant 3	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9

Participant 4	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9
	0	1
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9

Participant 5	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9
	0	0
	1	0
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9

Participant 6	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9
	0	9
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9

Participant 7	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	0

Participant 8	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9
	0	9
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9

Participant 9	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	8
	8	8
	9	9
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9

Participant 10	Digit Spoken	Digit Recognized
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9
	0	0
	1	1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9

References:

[1] Thiang, Implementation of Speech Recognition on MCS51 Microcontroller for Controlling Wheelchair, Electrical Engineering Department, Petra Christian University.

[2] Uvais Qidwai and Fatma Ibrahim, Arabic Speech-Controlled Wheelchair: A Fuzzy Scenario, Computer Science & Engineering Department, Qatar University

[3] Rashmi C R, Review of Algorithms and Applications in Speech Recognition System, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (4), 2014, 5258-5262.

[4] M.Tokuhira and Y.Ariki, Effectiveness of Kl-Transformation in Spectral Delta Expansion, Ryukoku University.

[5] Tetsuya Takiguchi, Yasuo Ariki, PCA-Based Speech Enhancement for Distorted Speech Recognition, Department of Computer and System Engineering, Kobe University, Japan.

[6] Utpal Bhattacharjee, A Comparative Study of LPCC And MFCC Features For The Recognition Of Assamese Phonemes, Department of Computer Science and Engineering, Rajiv Gandhi University, Arunachal Pradesh, India.

[7] L. R. Rabiner, R. W. Schafer, Digital Processing of Speech Signals.

[8] WANG Zhen-li, Bai Zhi-qiang, A Hybrid Method of Noise Robust Speech Recognition Based on Fractional Spectral Subtraction and Perceptual Linear Predictive, 2008 Congress on Image and Signal Processing.

[9] Lindasalwa Muda, Mumtaj Begam and I. Elamvazuthi, Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques Journal of Computing, Volume 2, Issue 3, March 2010.

[10] Talal Bin Amin, Iftekhar Mahmood, Speech Recognition Using Dynamic Time Warping, ICAST 2008 2nd International Conference on Advances in Space Technologies Islamabad, Pakistan, 29th – 30th November, 2008.

[11] D.B. Paul, Speech Recognition Using Hidden Markov Models, The Lincoln Laboratory Journal, Volume 3, Number 1 (l990)

[12] Jaume Padrell-Sendra, Dario Martin-Iglesias and Fernando Diaz-de-Maria, Support Vector Machines for Continuous Speech Recognition, 14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 4-8, 2006.

[13] Prashant Borde, Amarsinh Varpe, Ramesh Manza, Pravin Yannawar, Recognition of Isolated Words using Zernike and MFCC features for Audio Visual Speech Recognition, Dr. Babasaheb Ambedkar Marathwada University, Aurangabad (MS) India.

[14] Nahid Akhter, Amitabha Chakrabarty, A Survey-based Study on Lip Segmentation Techniques for Lip Reading Applications, Department of Computer Science and Engineering BRAC University, Dhaka, Bangladesh.

[15] Paul Viola Michael Jones, Rapid Object Detection using a Boosted Cascade of Simple Features, Accepted conference on computer vision and pattern recognition 2001.

[16] Ritesh Boda and M. Jasmine Pemeena Priyadarsini, Face Detection and Tracking Using Klt and Viola Jones, ARPN Journal of Engineering and Applied Sciences.

[17] Rein-Lien Hsu, Mohamed Abdel-Mottaleb, and Ani1 K. Jain, FACE DETECTION IN COLOR IMAGES.

[18] Sunil S. Morade, Suprava Patnaik, Visual Lip Reading using 3D-DCT and 3D-DWT and LSDA, International Journal of Computer Applications (0975 – 8887) Volume 136 – No.4, February 2016.

[19] Sunil Sudam Morade, Suprava Patnaik, Lip Reading Using DWT and LSDA.

[20] Xiaopeng Hong, Hongxun Yao, A PCA based visual DCT feature extraction method for lip reading, Conference Paper · December 2006.

[21] Liang Yaling, Yao Wenjuan, Du Minghui, Feature Extraction Based on LSDA for Lipreading, 2010 IEEE.

[22] He Jun, Zhang Hua, Research on Visual Speech Feature Extraction, 2009 International Conference on Computer Engineering and Technology.

[23] Michael Kass, Andrew Witkin, Demetri Terzopoulos, Snakes:Active Contour Models, International Journal of Computer Vision, 321-331 (1988).

[24] Iain Matthews, Tim Cootes, Stephen Cox, Richard Harvey, and J. Andrew Bangham, Auditory-Visual Speech Processing (AVSP'98), Lipreading Using Shape, Shading and Scale.

[25] Janet Finlay and Russell Beale, Neural Networks and Pattern Recognition in Human-Computer Interaction, Workshop at CHI'91, New Orleans, Louisiana, U. S. A

[26] Phillip Ian Wilson and Dr. John Fernandez, Facial Feature Detection using Haar Classifiers, Texas A&M University – Corpus Christi.

[27] Dave Marshall, 'The Discrete Cosine Transform', 2001. [Online]. Available: https://users.cs.cf.ac.uk/Dave.Marshall/Multimedia/node231.html

[28] Veton Z. Këpuska, Mohamed M. Eljhani, Brian H. Hight, Wake-Up-Word Feature Extraction on FPGA, Electrical & Computer Engineering Department, Florida Institute of Technology, Melbourne, USA.

[29] G. Saha1, Sandipan Chakroborty2, Suman Senapati3, A New Silence Removal and Endpoint Detection Algorithm for Speech and Speaker Recognition Applications, Department of

Electronics and Electrical Communication Engineering Indian Institute of Technology, Khragpur, Kharagpur-721 302, India.

[30] Bachu R.G., Kopparthi S., Adapa B., Barkana B.D., Separation of Voiced and Unvoiced using Zero crossing rate and Energy of the Speech Signal, Electrical Engineering Department School of Engineering, University of Bridgeport.

[31] Nirmesh J. Shah, Bhavik B. Vachhani, Hardik B. Sailor and Hemant A. Patil, Effectiveness of Plp-Based Phonetic Segmentation for Speech Synthesis, Dhirubhai Ambani Institute of Information and Communication Technology (DA-IICT), Gandhinagar-382007, India

[32] Namrata Dave, Feature Extraction Methods LPC, PLP and MFCC In Speech Recognition, G H Patel College of Engineering, Gujarat Technology University, INDIA

[33] Rajesh Mahanand Hegde, Fourier Transform Phase-Based Features for Speech Recognition, Department of Computer Science and Engineering Indian Institute of Technology Madras. July2005

[34] Gerasimos Potamianos, Hans Peter Graf, and Eric Cosatto, An Image Transform Approach for HMM Based Automatic Lipreading, Proceedings of the International Conference on Image Processing, Chicago, vol. III, pp. 173-177, 1998.

[35] Prashant Borde, Ramesh Manza, Bharti Gawali and Pravin Yannawar. Article: ‘vVISWa’ – A Multilingual Multi-Pose Audio Visual Database for Robust Human Computer Interaction. International Journal of Computer Applications 137(4):25-31, March 2016

[36] Zhe Wang and Xiangyang Xue, Chapter 2 Multi-Class Support Vector Machine.

[37] Urmila Shrawankar, Dr. Vilas Thakare, Techniques for Feature Extraction in Speech Recognition System: A Comparative Study, SGB Amravati University, Amravati

[38] Eamonn J. Keogh and Michael J. Pazzani, Scaling up Dynamic Time Warping to Massive Datasets, Department of Information and Computer Science University of California, Irvine, California 92697 USA.

[39] Ga¨el de Lannoy and Damien Fran¸cois and Michel Verleysen, Class-Specific Feature Selection for One-Against-All Multiclass SVMs, Universit´e catholique de Louvain Institute of Information and Communication Technologies, Electronics and Applied Mathematics Machine Learning Group

[40] Zeng Yumin, Wu Zhenyang, Combination of Pitch Synchronous Analysis and Fisher Criterion For Speaker Identification, Journal Of Electronics(China), November 2007.

[41] W. Astuti, A.M.Salma, A.M. Aibinu , R. Akmeliawati , Momoh Jimoh E.Salami, Automatic Arabic Recognition System based on Support Vector Machines (SVMs), International Islamic University Malaysia, Gombak, Selangor Darul Ehsan, Malaysia.

[42] Thomas G. Dietterich, Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms, Department of Computer Science, Oregon State University.

[43] Longbiao Wang, Shinji Ohtsuka, Seiichi Nakagawa, High Improvement of Speaker Identification and Verification By Combining MFCC And Phase Information, 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, 2009.

[44] Swapnanil Gogoi, Utpal Bhattacharjee, Vocal Tract Length Normalization and Sub Band Spectral Subtraction Based Robust Assamese Vowel Recognition System, Proceedings of the IEEE 2017 International Conference on Computing Methodologies and Communication (ICCMC).

[45] Yen-Lin Chiang, Yuan-Shan Lee,Wen-Chi Hsieh, and Jia-Ching Wang, Efficient and Portable Content-Based Music Retrieval System, 2014 IEEE International Conference on Orange Technologies, 2014.

[46] Taabish Gulzar, Anand Singh, Sandeep Sharma, Comparative Analysis of LPCC, MFCC and BFCC for the Recognition of Hindi Words using Artificial Neural Networks, International Journal of Computer Applications (0975 – 8887) Volume 101– No.12, September 2014.

[47] Joseph W. Picone, Signal Modeling Techniques in Speech Recognition, Proceedings of The IEEE, Vol. 81, No. 9, September 1993.

[48] Davis, S. Mermelstein, P. (1980) Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 28 No. 4, pp. 357-366

[49] Mark Greenwood, Andrew Kinghorn, Suving: Automatic Silence /Unvoiced/Voiced Classification of Speech, Department of Computer Science, University of Sheffield.