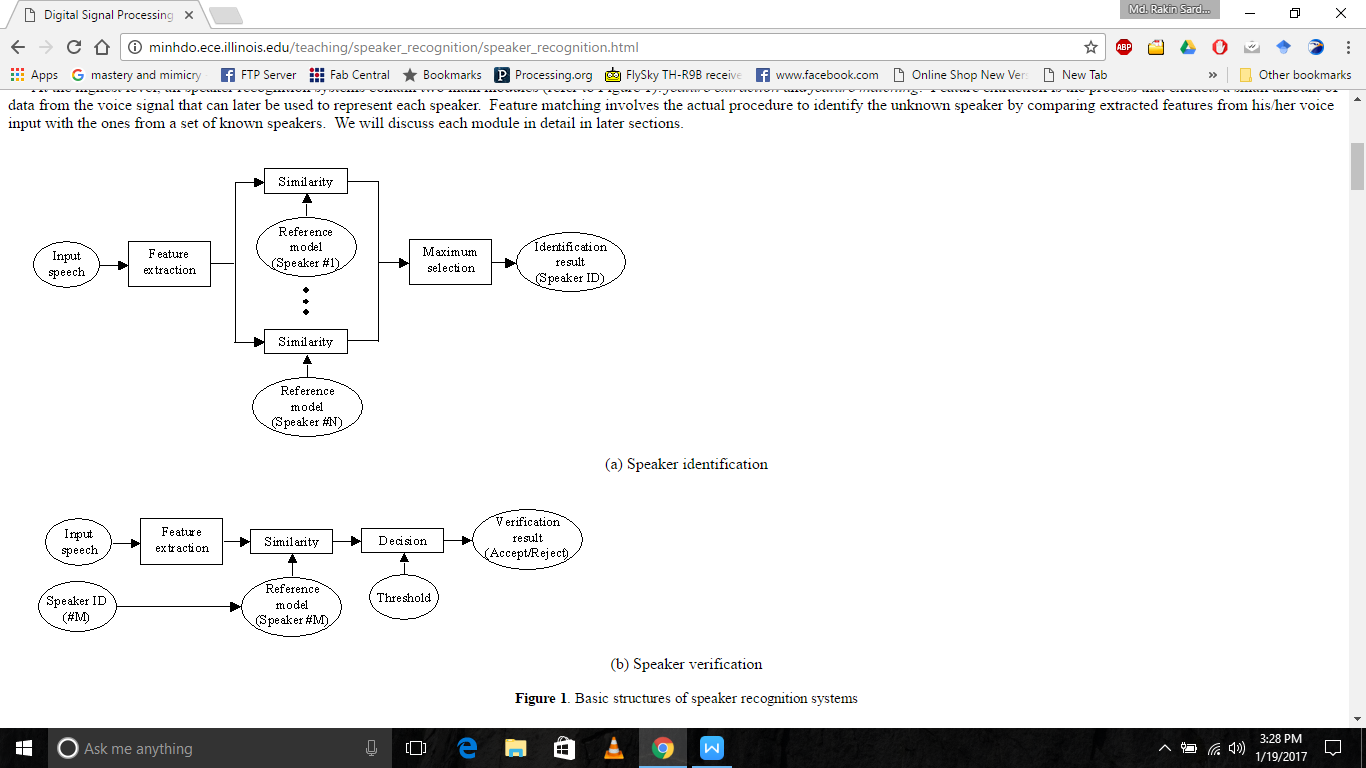
**Classroom attendance system using voice recognition**

**Block diagram**

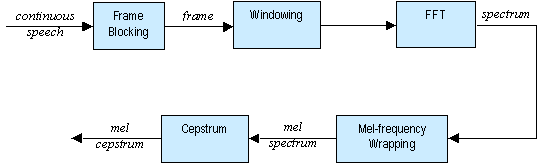
## *STEPS ARE:*

1. *Save the speech data in a specific folder with naming convention as s1, s2 etc. (NOTE: the file length is advised to be of 1 sec length. Current length is defined as such)*
2. *Open the speech\_recognition.m file and run it*
3. *Select the folder where you have the recorded files*
4. *Press save button beside it*
5. *Next enter the no. Of audio files in your recorded database*
6. *Press save button beside it*
7. *Then press train*
8. *After that you can run the program, record on spot and if matches it will give you output as matched with the student ID*

*Reference study:*

## *Mel-frequency cepstrum coefficients processor*

A block diagram of the structure of an MFCC processor is given in Figure 3.  The speech input is typically recorded at a sampling rate above 10000 Hz.  This sampling frequency was chosen to minimize the effects of *aliasing* in the analog-to-digital conversion.  These sampled signals can capture all frequencies up to 5 kHz, which cover most energy of sounds that are generated by humans.  As been discussed previously, the main purpose of the MFCC processor is to mimic the behavior of the human ears.  In addition, rather than the speech waveforms themselves, MFFC’s are shown to be less susceptible to mentioned variations.



**Figure 3**. Block diagram of the MFCC processor

### 3.2.1     Frame Blocking

In this step the continuous speech signal is blocked into frames of *N*  samples, with adjacent frames being separated by *M* (*M < N*).  The first frame consists of the first *N* samples.  The second frame begins *M* samples after the first frame, and overlaps it by *N - M* samples and so on.  This process continues until all the speech is accounted for within one or more frames.  Typical values for *N* and *M* are *N* = 256 (which is equivalent to ~ 30 msec windowing and facilitate the fast radix-2 FFT) and *M* = 100.

### 3.2.2     Windowing

The next step in the processing is to window each individual frame so as to minimize the signal discontinuities at the beginning and end of each frame.  The concept here is to minimize the spectral distortion by using the window to taper the signal to zero at the beginning and end of each frame.  If we define the window as IMG_257, where *N* is the number of samples in each frame, then the result of windowing is the signal

IMG_258

Typically the *Hamming* window is used, which has the form:

IMG_259

### 3.2.3     Fast Fourier Transform (FFT)

The next processing step is the Fast Fourier Transform, which converts each frame of *N* samples from the time domain into the frequency domain.  The FFT is a fast algorithm to implement the Discrete Fourier Transform (DFT), which is defined on the set of *N*samples {*xn*}, as follow:

IMG_260

In general *Xk*’s are complex numbers and we only consider their absolute values (frequency magnitudes).  The resulting sequence {*Xk*} is interpreted as follow: positive frequencies IMG_261 correspond to values IMG_262, while negative frequencies IMG_263 correspond to IMG_264.  Here, *Fs* denotes the sampling frequency.

The result after this step is often referred to as *spectrum* or *periodogram*.

### 3.2.4     Mel-frequency Wrapping

As mentioned above, psychophysical studies have shown that human perception of the frequency contents of sounds for speech signals does not follow a linear scale.  Thus for each tone with an actual frequency, *f*, measured in Hz, a subjective pitch is measured on a scale called the ‘mel’ scale.  The *mel-frequency* scale is a linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz.

|  |
| --- |
|  |
|  | IMG_265 |

**Figure 4**. An example of mel-spaced filterbank

One approach to simulating the subjective spectrum is to use a filter bank, spaced uniformly on the mel-scale (see Figure 4).  That filter bank has a triangular bandpass frequency response, and the spacing as well as the bandwidth is determined by a constant mel frequency interval.  The number of mel spectrum coefficients, *K*, is typically chosen as 20.  Note that this filter bank is applied in the frequency domain, thus it simply amounts to applying the triangle-shape windows as in the Figure 4 to the spectrum.  A useful way of thinking about this mel-wrapping filter bank is to view each filter as a histogram bin (where bins have overlap) in the frequency domain.

### 3.2.5     Cepstrum

In this final step, we convert the log mel spectrum back to time.  The result is called the mel frequency cepstrum coefficients (MFCC).  The cepstral representation of the speech spectrum provides a good representation of the local spectral properties of the signal for the given frame analysis. Because the mel spectrum coefficients (and so their logarithm) are real numbers, we can convert them to the time domain using the Discrete Cosine Transform (DCT).  Therefore if we denote those mel power spectrum coefficients that are the result of the last step are IMG_266, we can calculate the MFCC's, IMG_267 as

IMG_268

Note that we exclude the first component, IMG_269 from the DCT since it represents the mean value of the input signal, which carried little speaker specific information.

By applying the procedure described above, for each speech frame of around 30msec with overlap, a set of mel-frequency cepstrum coefficients is computed.  These are result of a cosine transform of the logarithm of the short-term power spectrum expressed on a mel-frequency scale.  This set of coefficients is called an *acoustic vector*.  Therefore each input utterance is transformed into a sequence of acoustic vectors.   In the next section we will see how those acoustic vectors can be used to represent and recognize the voice characteristic of the speaker.

# Feature Matching

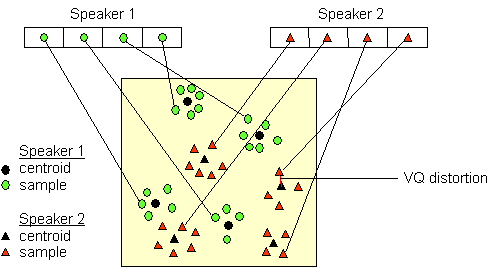
## *4.1*       *Overview*

The problem of speaker recognition belongs to a much broader topic in scientific and engineering so called *pattern recognition*.  The goal of pattern recognition is to classify objects of interest into one of a number of categories or classes.  The objects of interest are generically called *patterns* and in our case are sequences of acoustic vectors that are extracted from an input speech using the techniques described in the previous section.  The classes here refer to individual speakers.  Since the classification procedure in our case is applied on extracted features, it can be also referred to as *feature matching*.

Furthermore, if there exists some set of patterns that the individual classes of which are already known, then one has a problem in *supervised pattern recognition*.  These patterns comprise the *training set*and are used to derive a classification algorithm.  The remaining patterns are then used to test the classification algorithm; these patterns are collectively referred to as the *test set*.  If the correct classes of the individual patterns in the test set are also known, then one can evaluate the performance of the algorithm.

The state-of-the-art in feature matching techniques used in speaker recognition include Dynamic Time Warping (DTW), Hidden Markov Modeling (HMM), and Vector Quantization (VQ).  In this project, the VQ approach will be used, due to ease of implementation and high accuracy.  VQ is a process of mapping vectors from a large vector space to a finite number of regions in that space.  Each region is called a *cluster* and can be represented by its center called a *codeword*.  The collection of all codewords is called a *codebook*.

Figure 5 shows a conceptual diagram to illustrate this recognition process.  In the figure, only two speakers and two dimensions of the acoustic space are shown.  The circles refer to the acoustic vectors from the speaker 1 while the triangles are from the speaker 2.  In the training phase, using the clustering algorithm described in Section 4.2,  a *speaker-specific* VQ codebook is generated for each known speaker by clustering his/her training acoustic vectors.  The result codewords (centroids) are shown in Figure 5 by black circles and black triangles for speaker 1 and 2, respectively.  The distance from a vector to the closest codeword of a codebook is called a VQ-distortion.  In the recognition phase, an input utterance of an unknown voice is “vector-quantized” using each trained codebook and the *total VQ distortion* is computed.  The speaker corresponding to the VQ codebook with smallest total distortion is identified as the speaker of the input utterance.



**Figure 5**. Conceptual diagram illustrating vector quantization codebook formation.

One speaker can be discriminated from another based of the location of centroids.

(Adapted from Song et al., 1987)

## *4.2*       *Clustering the Training Vectors*

After the enrolment session, the acoustic vectors extracted from input speech of each speaker provide a set of training vectors for that speaker.  As described above, the next important step is to build a speaker-specific VQ codebook for each speaker using those training vectors.  There is a well-know algorithm, namely LBG algorithm [Linde, Buzo and Gray, 1980], for clustering a set of *L* training vectors into a set of *M* codebook vectors.  The algorithm is formally implemented by the following recursive procedure:

1.     Design a 1-vector codebook; this is the centroid of the entire set of training vectors (hence, no iteration is required here).

2.     Double the size of the codebook by splitting each current codebook **y***n* according to the rule

IMG_257

IMG_258

where *n* varies from 1 to the current size of the codebook, and IMG_259is a splitting parameter (we choose IMG_260=0.01).

3.     Nearest-Neighbor Search: for each training vector, find the codeword in the current codebook that is closest (in terms of similarity measurement), and assign that vector to the corresponding cell (associated with the closest codeword).

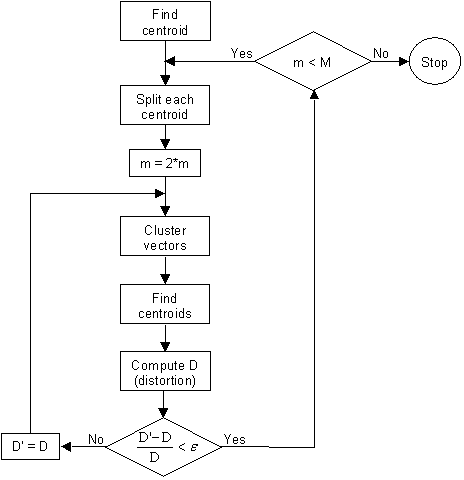
4.     Centroid Update: update the codeword in each cell using the centroid of the training vectors assigned to that cell.

5.     Iteration 1: repeat steps 3 and 4 until the average distance falls below a preset threshold

6.     Iteration 2: repeat steps 2, 3 and 4 until a codebook size of *M* is designed.

Intuitively, the LBG algorithm designs an *M*-vector codebook in stages.  It starts first by designing a 1-vector codebook, then uses a splitting technique on the codewords to initialize the search for a 2-vector codebook, and continues the splitting process until the desired *M*-vector codebook is obtained.

Figure 6 shows, in a flow diagram, the detailed steps of the LBG algorithm.  “*Cluster vectors*” is the nearest-neighbor search procedure which assigns each training vector to a cluster associated with the closest codeword.  “*Find centroids*” is the centroid update procedure.  “*Compute D (distortion)*” sums the distances of all training vectors in the nearest-neighbor search so as to determine whether the procedure has converged.



**Figure 6**. Flow diagram of the LBG algorithm