**Speaker Recognition in MATLAB**

**By Hamdi Boukamcha**

**Speaker/Voice-Command Recognition in MATLAB**

**1- OVERVIEW:**

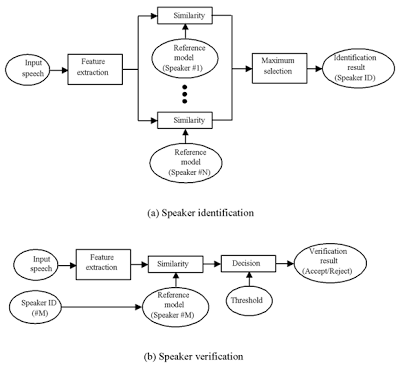
Speaker recognition is the process of automatically recognizing who is speaking on the basis of individual information included in speech waves. This technique makes it possible to use the speaker’s voice to verify their identity and control access to services such as voice dialing, banking by telephone, telephone shopping, database access services, information services, voice mail, security control for confidential Information areas and remote access to computers.

The goal of this project is to build a simple, yet complete and representative automatic speaker recognition system. Due to the limited space, we will only test our system on a mall speech database. But one can have many database files for training the system; the more files one train/teach to the system, the more accuracy is achieved. Here we’ll use 11 samples for each command to be recognized; hence total 99 files are being trained as there are 9 different commands.

**2- Principles of Speaker/Voice Recognition**

Speaker recognition can be classified into identification and verification. Speaker/Voice identification is the process of determining which registered speaker provides a given utterance. Speaker verification, on the other hand, is the process of accepting or rejecting the identity claim of a speaker. Figure 1 shows the basic structures of speaker identification and verification systems.

At the highest level, all speaker recognition systems contain two main modules (refer to Figure 1): feature extraction and feature matching. Feature extraction is the process that extracts a small amount of data from the voice signal that can later be used to represent each speaker. Feature matching involves the actual procedure to identify the unknown speaker by comparing extracted features from his/her voice input with the ones from a set of known speakers.

[](http://4.bp.blogspot.com/-22bUO_I2v1U/UGFSPaRipgI/AAAAAAAAAII/yCEyWy_LuXU/s1600/6.PNG)

**Figure 1.** Basic structures of speaker recognition systems

All speaker recognition systems have to serve two distinguish-phases. The first one is referred to the enrollment sessions or training phase while the second one is referred to as the operation sessions or testing phase. In the training phase, each registered speaker has to provide samples of their speech so that the system can build or train a reference model for that speaker. In case of speaker verification systems, in addition, a speaker-specific threshold is also computed from the training samples.

During the testing (operational) phase (see Figure 1), the input speech is matched with stored reference model(s) and recognition decision is made.

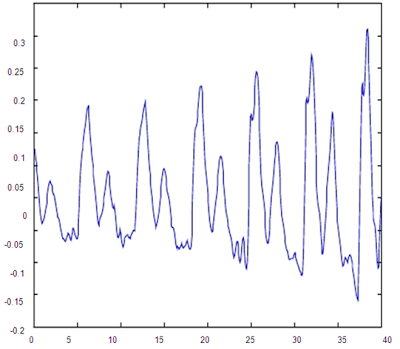
Speaker recognition is a difficult task and it is still an active research area. Automatic speaker recognition works based on the premise that a person’s speech exhibits characteristics that are unique to the speaker. However this task has been challenged by the highly variant of input speech signals. The principle source of variance comes from the speakers themselves. Speech signals in training and testing sessions can be greatly different due to many facts such as people voice change with time, health conditions (e.g. the speaker has a cold), speaking rates, etc. There are also other factors, beyond speaker variability, that present a challenge to speaker recognition technology. Examples of these are acoustical noise and variations in recording environments (e.g. speaker uses different telephone handsets/microphones).

**3- Speech Feature Extraction**

**3.1- Introduction**

The purpose of this module is to convert the speech waveform to some type of parametric representation (at a considerably lower information rate) for further analysis and processing. This is often referred as the signal-processing front end.

The speech signal is a slowly timed varying signal (it is called quasi-stationary). An example of speech signal is shown in Figure 2. When examined over a sufficiently short period of time (between 5 and 100 ms), its characteristics are fairly stationary. However, over long periods of time (on the order of 1/5 seconds or more) the signal characteristic change to reflect the different speech sounds being spoken. Therefore, short-time spectral analysis is the most common way to characterize the speech signal.

[](http://2.bp.blogspot.com/-HWmjDkgPCDQ/UGFSEmJNJuI/AAAAAAAAAIA/Eh2nDWG6esM/s1600/5.PNG)

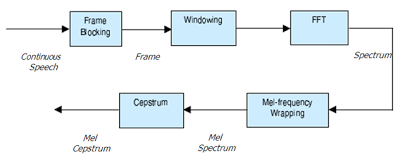
**Figure 2.** An example of speech signal

A wide range of possibilities exist for parametrically representing the speech signal for the speaker recognition task, such as Linear Prediction Coding (LPC), Mel-Frequency Cepstrum Coefficients (MFCC), and others. MFCC is perhaps the best known and most popular, and these will be used in this project.

MFCC’s are based on the known variation of the human ear’s critical bandwidths with frequency; filters spaced linearly at low frequencies and logarithmically at high frequencies have been used to capture the phonetically important characteristics of speech. This is expressed in the Mel-frequency scale, which is linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz. The process of computing MFCCs is described in more detail next.

**3.2- 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.

[](http://2.bp.blogspot.com/-RRyVPycykvg/UGFR5hi3vuI/AAAAAAAAAH4/8j1lg-TU7F4/s1600/4.PNG)

**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. Similarly, the third frame begins 2M samples after the first frame (or M samples after the second frame) and overlaps it by N - 2M samples. 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 ms 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 − 1, where N is the number of samples in each ≤ N ≤ n the window as w(n), 0 frame, then the result of windowing is the signal.

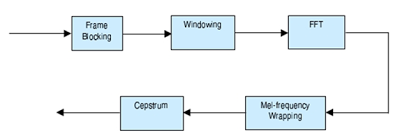
*yl* (*n*) *xl* (*n*)*w*(*n*), 0≤ *n*≤ *N*− 1

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

*w*(*n*) 0.54− 0.46 cos, 0≤ *n*≤ *N*− 1

**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:

[](http://4.bp.blogspot.com/-kIy7lYcL2dI/UGFRrc6AgqI/AAAAAAAAAHw/m3pirVGl3do/s1600/3.PNG)

Equations for Xn DFT

The result obtained after this step is often referred to as signal’s Spectrum or Periodogram.

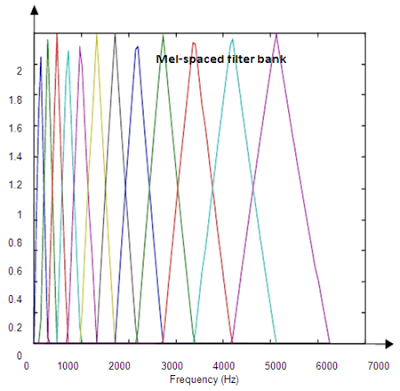
**3.2.4- Mel-frequency wrapping**

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 linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz. As a reference point, the pitch of a 1 kHz tone, 40 dB above the perceptual hearing threshold, is defined as 1000 Mel. Therefore we can use the following approximate formula to compute the Mel for a given frequency f in Hz:

Mel (f) = 2595 \* log10 (1 + f/700)

One approach to simulating the subjective spectrum is to use a filter bank, one filter for each desired Mel-frequency component (see Figure 4). That filter bank has a triangular band-pass frequency response, and the spacing as well as the bandwidth is determined by a constant Mel-frequency interval. The ) thus consists of the output power of these filters⎤modified spectrum of S( ) is the input. The number of Mel spectrum coefficients, K, is⎤when S( typically chosen as 20.

Note that this filter bank is applied in the frequency domain; therefore it simply amounts to taking those triangle-shape windows in the Figure 4 on 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.

[](http://1.bp.blogspot.com/-VHKt-T35oMk/UGFRerYLCcI/AAAAAAAAAHo/OKQ44UlWKWI/s1600/2.PNG)

**Figure 4.** An example of Mel-spaced filter-bank

**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).

**3.3- Summary**

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.

**4- Feature Matching**

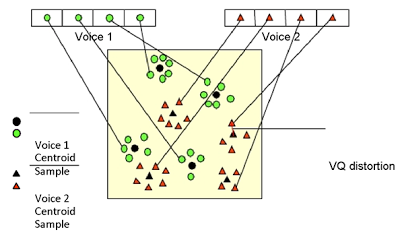
**4.1- Introduction**

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. This is exactly our case since during the training session, we label each input voice with the ID (s1 to sn). 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 includes 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 codeword is called a codebook.

Figure 5 shows a conceptual diagram to illustrate this recognition process. In the figure, only two voices and two dimensions of the acoustic space are shown. The circles refer to the acoustic vectors from the voice 1 while the triangles are from the voice 2. In the training phase, a speaker-specific VQ codebook is generated for each known speaker by clustering his/her training acoustic vectors. The result codeword (Centroids) are shown in Figure 5 by black circles and black triangles for voices 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.

[](http://4.bp.blogspot.com/-ejAezjt_Cds/UGFRQkPpMFI/AAAAAAAAAHg/s0KgZrku_yw/s1600/1.PNG)

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

(One voice can be discriminated from another based of the location of Centroids)

**5- Implementation**

As stated above, in this project we will experience the building and testing of an automatic voice recognition system. In order to implement such a system, one must go through several steps which were described in details in previous sections. Note that many of the above tasks are already implemented in Matlab. Furthermore, to ease the development process, we supply with two utility functions: melfb and disteu; and two main functions: train and test. Download all of those files into your working folder. The first two files can be treated as a black box, but the later two needs to be thoroughly understood. In fact, your tasks are to write two missing functions: mfcc and vqlbg, which will be called from the given main functions. In order to accomplish that, follow each step in this section carefully.

**5.1- Speech Data**

Our goal is to train a voice model (or more specific, a VQ codebook in the MFCC vector space) for each voice commands S1 – S(n) using the corresponding sound file in the TRAIN folder. After this training step, the system would have knowledge of the voice characteristic of each (known) voice of the speaker. Next, in the testing phase, the system will be able to identify the (assumed unknown) voice/speaker of each sound file, recorded and saved in the TEST folder (only last live sound).

**5.2- Speech Processing**

In this phase you are required to write a Matlab function that reads a sound file and turns it into a sequence of MFCC (acoustic vectors) using the speech processing steps described previously. Many of those tasks are already provided by either standard or our supplied Matlab functions. The Matlab functions that you would need to use are: wavread, hamming, fft, dct and melfb (supplied function). Type- help function-name at the Matlab prompts for more information about a function.

Now cut the speech signal (a vector) into frames with overlap (refer to the frame section in the theory part). The result is a matrix where each column is a frame of N samples from original speech signal. Applying the steps for Windowing and FFT to transform the signal into the frequency domain; this process is used in many different applications and is referred in literature as Windowed Fourier Transform (WFT) or Short-Time Fourier Transform (STFT). The result is often called as the spectrum or Periodogram.

The last step in speech processing is converting the power spectrum into Mel-frequency Cepstrum coefficients. The supplied function melfb facilitates this task.

Finally, put all the pieces together into a single Matlab function, mfcc, which performs the MFCC processing.

**5.3-Vector Quantization**

The result of the last section is that we transform speech signals into vectors in an acoustic space. In this section, we will apply the VQ-based pattern recognition technique to build speaker reference models from those vectors in the training phase and then can identify any sequences of acoustic vectors uttered by unknown speakers.

Now write a Matlab function, vqlbg that trains a VQ codebook using the LGB algorithm described before. Use the supplied utility function disteu to compute the pair wise Euclidean distances between the codeword and training vectors in the iterative process.

**5.4- Simulation and Evaluation**

**1**     Open MATLAB command window, and set the working folder path.

**2**     Type as:

>> clear all                  (press enter)

>> close all                  (press enter)

>> setupfile                 (press enter)

This will setup all the prerequisites needed for the Voice recognition system to RUN properly

**3**      Then type as:

>> UUI                         (press enter)

This will open a GUI, (Read messages on the very bottom of the GUI, if required)

**4**     Press Record Button for recording samples; after clicking this button, command window will get focused, please follow the messages during recording on main window. (Complete recording is compulsory/must).

**5**      After recording all the 99 sample voices, click on teaching button. Teaching of 99 recorded voices will automatically start on command window.

**6**      Last but not least, Hit "Press & Speak" button. In return, you'll see the circular-object in motion on the plot axes. All the motions for the object have been programmed in polar coordinates.

**7**      Continue saying your desired commands just after pressing "Press & Speak" button.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**NOTE:** If you have pre-recorded files, then name these files as:

s1.wav      -           s11.wav           >> for UP

s12           -           s22                   >> DOWN

s23           -           s33                   >> RIGHT3

s34           -           s44                   >> RIGHT5

s45           -           s55                   >> RIGHT10

s56           -           s66                   >> LEFT3

s67           -           s77                   >> LEFT5

s78           -           s88                   >> LEFT10

s89           -           s99                   >> SHOOT

And place/copy in "train" folder. In this case you only need to teach these files.

**PS: In case of any query, please feel free to contact us: hamdouchhd@hotmail.com**

http://matlab-recognition-code.com