DSP Mini-Project:
An Automatic Speaker Recognition System

http://www.ifp.uiuc.edu/~minhdo/teaching/speaker_recognition

1 Overview

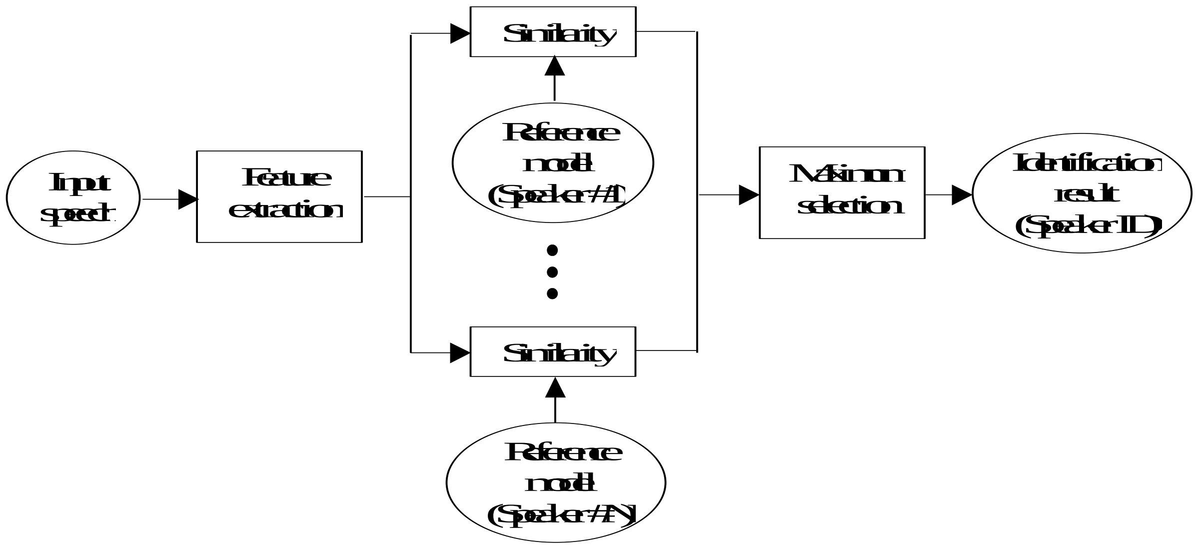
Speaker recognition is the process of automatically recognizing who is speaking on the basis of individual information included in speech waves. This technique makes it possible to use the speaker's voice to verify their identity and control access to services such as voice dialing, banking by telephone,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

This document describes how to build a simple, yet complete and *representative automatic speaker recognition system*. Such a speaker recognition system has potential in many security applications. For example, users have to speak a PIN (Personal Identification Number) in order to gain access to the laboratory door, or users have to speak their credit card number over the telephone line to verify their identity. By checking the voice characteristics of the input utterance, using an automatic speaker recognition system similar to the one that we will describe, the system is able to add an extra level of security.

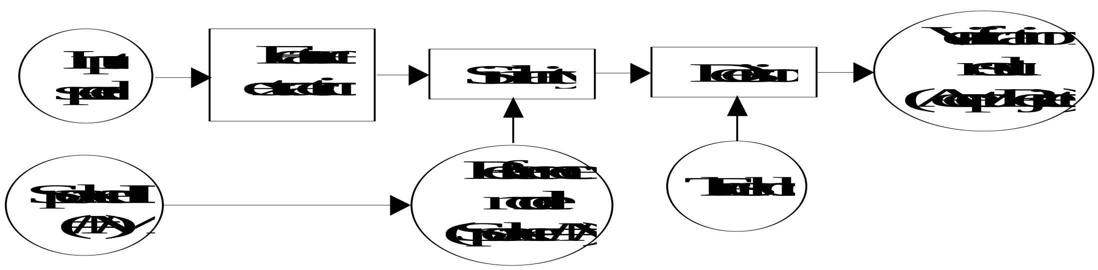
2 Principles of Speaker Recognition

Speaker recognition can be classified into identification and verification. *Speaker 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. The system that we will describe is classified as *text-independent speaker identification* system since its task is to identify the person who speaks regardless of what is saying.

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. We will discuss each module in detail in later sections.



(a) Speaker identification



(b) Speaker verification

Figure 1. Basic structures of speaker recognition systems

All speaker recognition systems have to serve two distinguished phases. The first one is referred to the enrolment or training phase, while the second one is referred to as the operational 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. In the *testing phase*, the input speech is matched with stored reference model(s) and a recognition decision is made.

Speaker recognition is a difficult task. 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 is the speaker himself/herself. 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, and so on. 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).

3 Speech Feature Extraction

3.1 Introduction

The purpose of this module is to convert the speech waveform, using digital signal processing (DSP) tools, to a set of features (at a considerably lower information rate) for further analysis. 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 msec), 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.

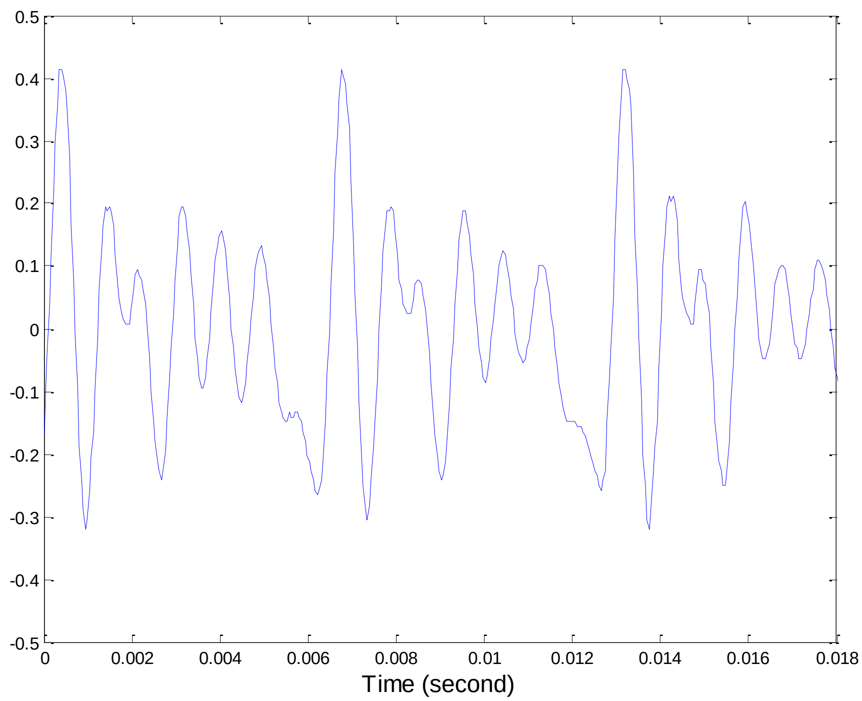


Figure 2. Example of speech signal

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 will be described in this paper.

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.

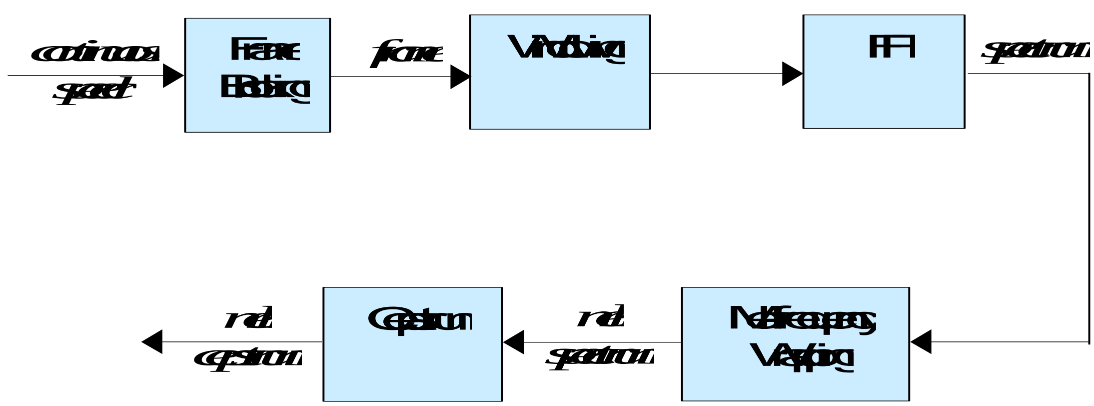


Figure 3. Block diagram of the MFCC processor

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 $w(n),\ 0 \le n \le N-1$, where *N* is the number of samples in each frame, then the result of windowing is the signal

$$y_l(n) = x_l(n)w(n), \quad 0 \le n \le N-1$$

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

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \quad 0 \le n \le N-1$$

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 $\{x_n\}$, as follow:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi kn/N}, \quad k = 0,1,2,\ldots,N-1$$

In general X_k's are complex numbers and we only consider their absolute values (frequency magnitudes). The resulting sequence $\{X_k\}$ is interpreted as follow: positive frequencies $0 \le f < F_s/2$ correspond to values $0 \le n \le N/2-1$, while negative frequencies $-F_s/2 < f < 0$ correspond to $N/2+1 \le n \le N-1$. Here, F_s denotes the sampling frequency.

The result after this step is often referred to as *spectrum* or *period-o-gram*.

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.

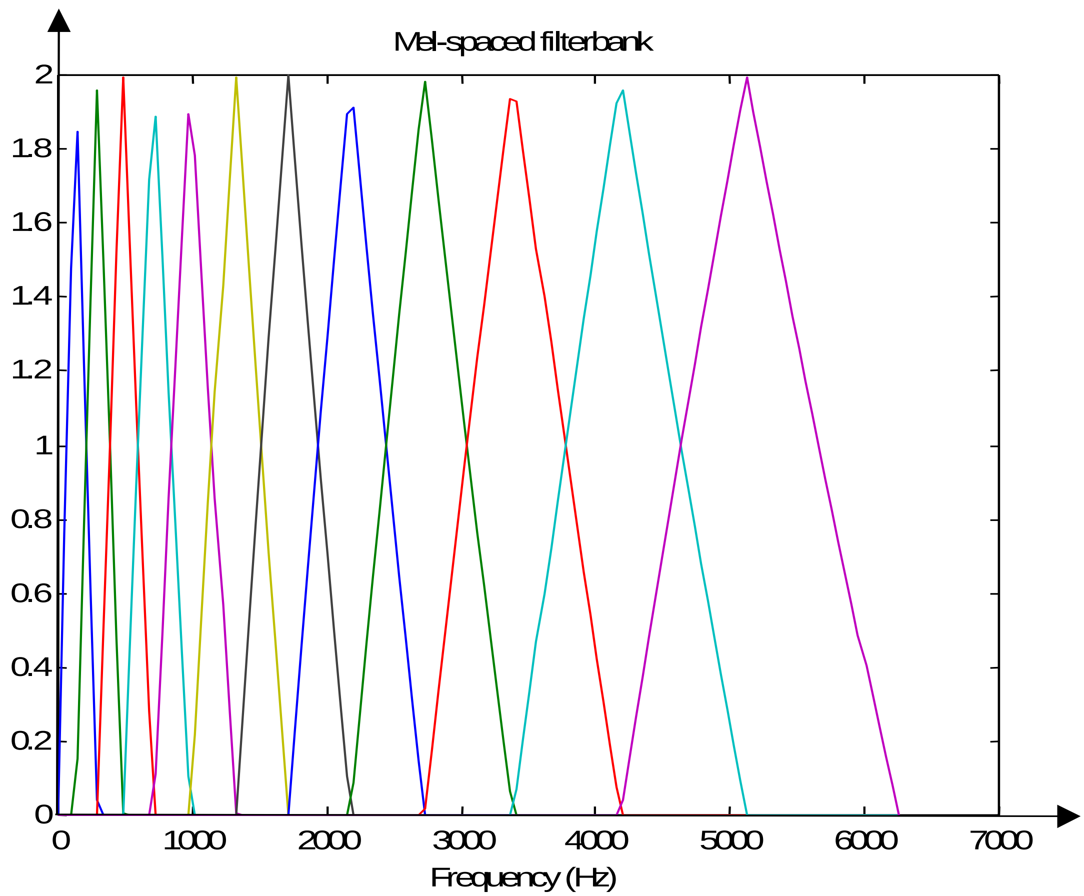


Figure4. An example of mel-spaced filter bank

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 band pass 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 $\tilde{S}_0, k = 0, 2, \ldots, K-1$, we can calculate the MFCC's, $\tilde{c}_n$, as

$$\tilde{c}_n = \sum_{k=1}^{K} (\log \tilde{S}_k) \cos\left[n \left(k - \frac{1}{2} \right) \frac{\pi}{K} \right], \qquad n = 0, 1, \ldots, K\text{-}1$$

Note that we exclude the first component, $\tilde{c}_0$, from the DCT since it represents the mean value of the input signal, which carried little speaker specific information.

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 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.

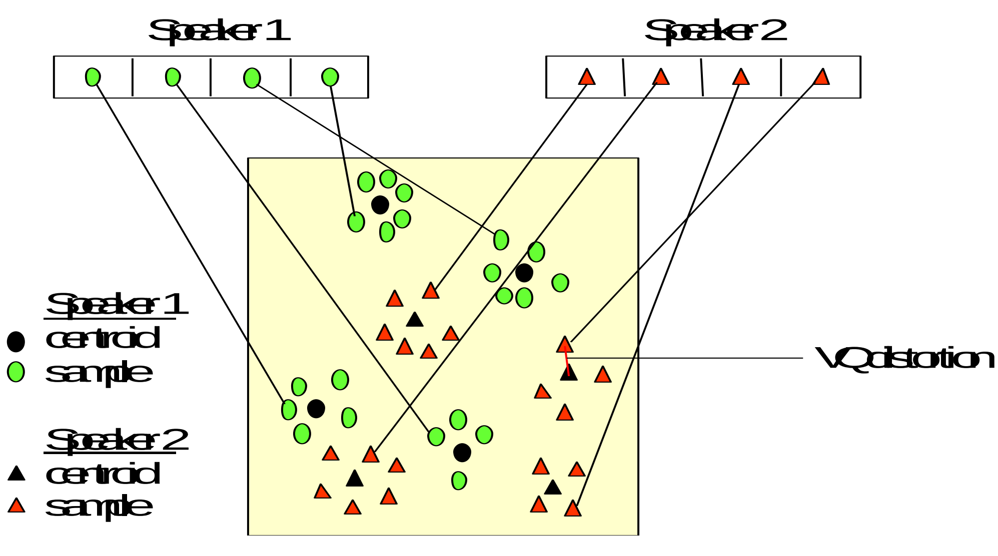


Figure 5. Conceptual diagram illustrating vector quantization codebook formation. One speaker can be discriminated from another based of the location of centroids. (Adapted from Song et al., 1987)

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 $\mathbf{y}_n$ according to the rule

$$\mathbf{y}_n^+ = \mathbf{y}_n(1+\varepsilon)$$
$$\mathbf{y}_n^- = \mathbf{y}_n(1-\varepsilon)$$

   where *n* varies from 1 to the current size of the codebook, and ε is a splitting parameter (we choose ε =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.

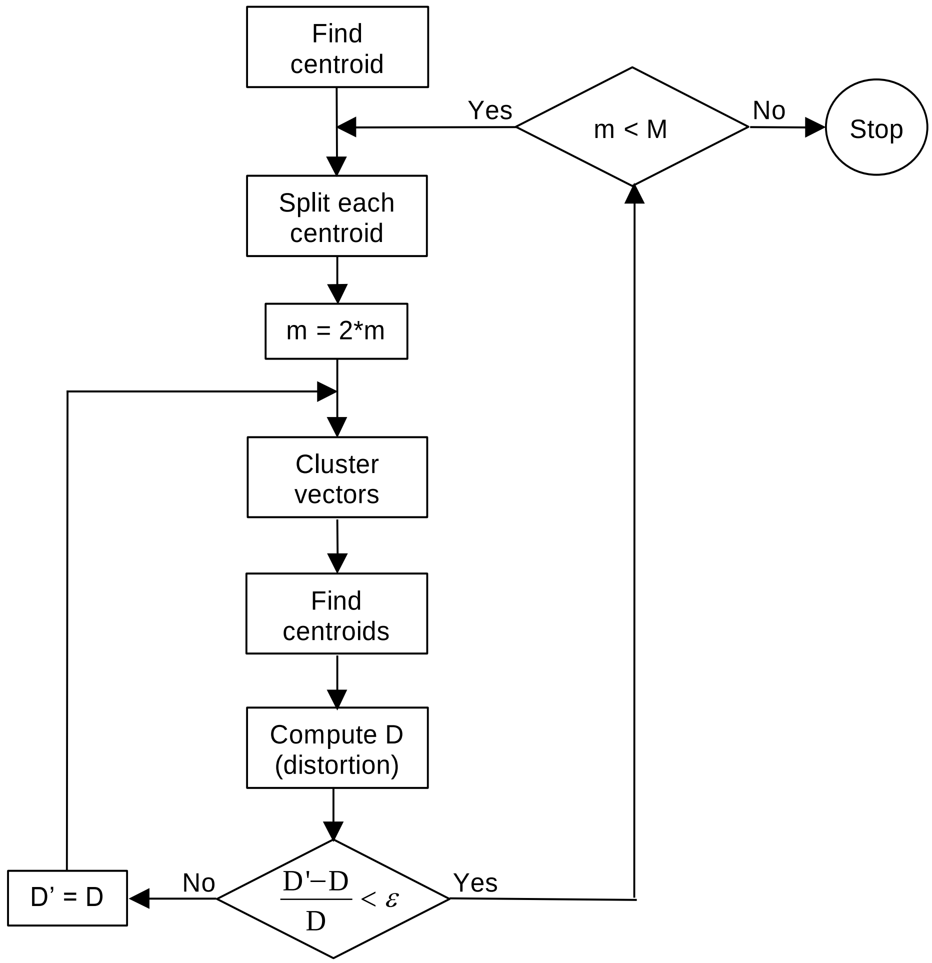


Figure 6. Flow diagram of the LBG algorithm (Adapted from Rabiner and Juang, 1993)

5 Project

As stated before, in this project we will experiment with the building and testing of an automatic speaker recognition system. In order to build such a system, one have to go through the steps that were described in previous sections. The most convenient platform for this is the **Matlab** environment since many of the above tasks were already implemented in Matlab. The project Web page given at the beginning provides a test database and several helper functions to ease the development process. We supplied you with two utility functions: `melfb` and `disteu`; and two main functions: `train` and `test`. Download all of these files from the project Web page 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 and check your understanding by answering all the questions.

5.1 Speech Data

Down load the ZIP file of the speech database from the project Web page. After unzipping the file correctly, you will find two folders, TRAIN and TEST, each contains 8 files, named: S1.WAV, S2.WAV, …, S8.WAV; each is labeled after the ID of the speaker. These files were recorded in Microsoft WAV format. In Windows systems, you can listen to the recorded sounds by double clicking into the files.

Our goal is to train a voice model (or more specific, a VQ codebook in the MFCC vector space) for each speaker S1 - S8 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) speaker. Next, in the testing phase, the system will be able to identify the (assumed unknown) speaker of each sound file in the TEST folder.

Question 1: *Play each sound file in the TRAIN folder. Can you distinguish the voices of the eight speakers in the database? Now play each sound in the TEST folder in a random order without looking at the file name (pretending that you do not known the speaker) and try to identify the speaker using your knowledge of their voices that you just learned from the TRAIN folder. This is exactly what the computer will do in our system. What is your (human performance) recognition rate? Record this result so that it could be later on compared against the computer performance of our system.*

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 are: `wavread`, `hamming`, `fft`, `dct` and `melfb` (supplied function). Type `help function_name` at the Matlab prompt for more information about these functions.

Question 2: *Read a sound file into Matlab. Check it by playing the sound file in Matlab using the function:* `sound`*. What is the sampling rate? What is the highest frequency that the recorded sound can capture with fidelity? With that sampling rate, how many msecs of actual speech are contained in a block of 256 samples?*

Plot the signal to view it in the time domain. It should be obvious that the raw data in the time domain has a very high amount of data and it is difficult for analyzing the voice characteristic. So the motivation for this step (speech feature extraction) should be clear now!

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 "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*.

<u>Question 3:</u> *After successfully running the preceding process, what is the interpretation of the result? Compute the power spectrum and plot it out using the* `imagesc` *command. Note that it is better to view the power spectrum on the log scale. Locate the region in the plot that contains most of the energy. Translate this location into the actual ranges in time (msec) and frequency (in Hz) of the input speech signal.*

<u>Question 4:</u> *Compute and plot the power spectrum of a speech file using different frame size: for example N = 128, 256 and 512. In each case, set the frame increment M to be about N/3. Can you describe and explain the differences among those spectra?*

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

<u>Question 5:</u> *Type* `help melfb` *at the Matlab prompt for more information about this function. Follow the guidelines to plot out the mel-spaced filter bank. What is the behavior of this filter bank? Compare it with the theoretical part.*

<u>Question 6:</u> *Compute and plot the spectrum of a speech file before and after the mel-frequency wrapping step. Describe and explain the impact of the* `melfb` *program.*

Finally, complete the "Cepstrum" step and put all 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.

<u>Question 7:</u> *To inspect the acoustic space (MFCC vectors) we can pick any two dimensions (say the 5^{th} and the 6^{th}) and plot the data points in a 2D plane. Use acoustic vectors of two different speakers and plot data points in two different colors. Do the data regions from the two speakers overlap each other? Are they in clusters?*

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 pairwise Euclidean distances between the codewords and training vectors in the iterative process.

<u>Question 8:</u> *Plot the resulting VQ codewords after function* `vqlbg` *using the same two dimensions over the plot of the previous question. Compare the result with Figure 5.*

5.4 Simulation and Evaluation

Now is the final part! Use the two supplied programs: `train` and `test` (which require two functions `mfcc` and `vqlbg` that you just complete) to simulate the training and testing procedure in speaker recognition system, respectively.

<u>Question 9:</u> *What is recognition rate our system can perform? Compare this with the human performance. For the cases that the system makes errors, re-listen to the speech files and try to come up with some explanations.*

<u>Question 10:</u> *You can also test the system with your own speech files. Use the Window's program Sound Recorder to record more voices from yourself and your friends. Each new speaker needs to provide one speech file for training and one for testing. Can the system recognize your voice? Enjoy!*

References

[1] L.R. Rabiner and B.H. Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, Englewood Cliffs, N.J., 1993.

[2] L.R Rabiner and R.W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, N.J., 1978.

[3] S.B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences", *IEEE Transactions on Acoustics, Speech, Signal Processing*, Vol. ASSP-28, No. 4, August 1980.

[4] Y. Linde, A. Buzo & R. Gray, "An algorithm for vector quantizer design", *IEEE Transactions on Communications*, Vol. 28, pp.84-95, 1980.

[5] S. Furui, "Speaker independent isolated word recognition using dynamic features of speech spectrum", *IEEE Transactions on Acoustic, Speech, Signal Processing*, Vol. ASSP-34, No. 1, pp. 52-59, February 1986.

[6] S. Furui, "An overview of speaker recognition technology", *ESCA Workshop on Automatic Speaker Recognition, Identification and Verification*, pp. 1-9, 1994.

[7] F.K. Song, A.E. Rosenberg and B.H. Juang, "A vector quantisation approach to speaker recognition", *AT&T Technical Journal*, Vol. 66-2, pp. 14-26, March 1987.

[8] `comp.speech` Frequently Asked Questions WWW site, http://svr-www.eng.cam.ac.uk/comp.speech/