Design Of An Automatic Speaker Recognition System Using MFCC, Vector Quantization And LBG Algorithm

Prof. Ch.Srinivasa Kumar
Prof. and Head of department.
Electronics and communication
Nalanda Institute of technology
Guntur.,

Dr. P. Mallikarjuna Rao
Prof. , Andhra University.
Electronics and communicationVizag.

Abstract

The results of a case study carried out while developing an automatic speaker recognition system are presented in this paper. The Vector Quantization (VQ) approach is used for 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. After the enrolment session, the acoustic vectors extracted from input speech of a speaker provide a set of training vectors. LBG algorithm due to Linde, Buzo and Gray is used for clustering a set of L training vectors into a set of M codebook vectors. For comparison purpose, the distance between each test codeword and each codeword in the master codebook is computed. The difference is used to make recognition decision. The entire coding was done in MATLAB and the system was tested for its reliability

Keywords: Automatic Speaker Recognition System, Speech Processing, Speaker Verification

1. Introduction

Speech is one of the natural forms of communication. Advancements in scientific technology have made it possible to use this in security systems. Speaker recognition is a process that enables machines to understand and interpret the human speech by making use of certain algorithms and verifies the authenticity of a speaker with the help of a database. That is, speaker recognition or identification is essentially a method of automatically identifying a speaker from a recorded or a live speech signal by analyzing the speech signal parameters. First, the human speech is converted to machine readable format after which the machine processes the data. The data processing deals with feature extraction and feature matching. Then, based on the processed data, suitable action is taken by the machine. The action taken depends on the application. Every speaker is identified with the help of unique numerical values of certain signal parameters called 'template' or 'code book' pertaining to the speech produced by his or her vocal tract. Normally the speech parameters of a vocal tract that are considered for analysis are (i) formant frequencies, (ii) pitch, and (iii) loudness.

These are the characteristic parameters of the modulated sound energy produced by the vocal tract. 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, robust, accurate and most popular. The Mel frequency scale is linear frequency spacing below 1000Hz and logarithmic spacing above 1000Hz. In other words, frequency filters are spaced linearly at low frequencies and are logarithmically at high frequencies which have been used to capture the phonetically important characteristics of speech. This is an important property of a human ear. Hence the MFCC processor mimics the human ear of perception. This is the process of feature extraction. Pattern recognition does the job of feature extraction. 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. 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. This algorithm is formally implemented for various speakers and its robustness verified in this paper.

Speech Communication is one of the basic and most essential capabilities possessed by human beings. It is this amazing ability that sets him apart from other species living in this Earth. It is also the most natural form of communication between humans and thus as a subject, it has attracted attention over many years. The fundamental purpose of speech is communication, i.e., the transmission of messages. The speech wave itself conveys linguistic informati/n, the speaker's vocal characteristics and emotion. The speech production process involves three sub processes- source generation, articulation and radiation. MThe human vocal organ is complex. It consists of the lungs, trachea, larynx, pharynx and nasal and oral cavities. Together these form a connected tube. The upper portion beginning with the larynx is called as the vocal tract. Vocal tract is changeable into various shapes by moving the jaw, tongue, lips and other internal parts. Adjusting the vocal tract shape to produce various linguistic sounds is called articulation. In addition to controlling the type of excitation, the shape of the vocal tract is also adjusted by manipulating the tongue, lips, and lower jaw. The shape determines the frequency response of the vocal tract. The way that the human auditory system works plays an important role in speech coding systems design. By understanding how sounds are perceived, resources in the coding system can be allocated in the most efficient manner, leading to improved cost effectiveness. The human auditory system behaves much like a frequency analyzer; and system characterization is simpler if done in the frequency domain.

The human vocal organ is complex. It consists of the lungs, trachea, larynx, pharynx and nasal and oral cavities. Together these form a connected tube. The upper portion beginning with the larynx is called as the vocal tract. Vocal tract is changeable into various shapes by moving the jaw, tongue, lips and other internal parts. Adjusting the vocal tract shape to produce various linguistic sounds is called articulation. In addition to controlling the type of excitation, the shape of the vocal tract is also adjusted by manipulating the tongue, lips, and lower jaw. The shape determines the frequency response of the vocal tract. The way that the human auditory system works plays an important role in speech coding systems design. By understanding how sounds are perceived, resources in the coding system can be allocated in the most efficient manner, leading to improved cost effectiveness. The human auditory system behaves much like a frequency analyzer; and system characterization is simpler if done in the frequency domain.

2. Basic Concept Behind Speaker Recognition

The main principle behind the speaker recognition system is that of feature extraction and matching. Feature extraction deals with the extraction or reading of important characteristics or feature from a human speech signal. Those characteristics might be pitch, or frequency, which are unique to different persons. Speaker recognition methods can also be divided into text-independent and text-dependent methods. In a text-independent system, speaker models capture characteristics of somebody's speech which show up irrespective of what one is saying. In a text-dependent system, on the other hand, the recognition of the speaker's identity is based on his or her speaking one or more specific phrases, like passwords, card numbers, PIN codes, etc. All speaker recognition systems have to serve two phases. The first one is referred to the enrolment session or training phase while the second one is referred to as the operation session 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. It consists of two main parts. The first part consists of processing each persons input voice sample to condense and summarize the characteristics of their vocal tracts. The second part involves pulling each person's data together into a single, easily manipulated matrix. The testing system mirrors the training system architecture. First the input signal is analyzed, and then it is compared to the data stored in the codebook. The difference is used to make recognition decision.

3. Block Diagram Of A Speaker Recognizer

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 3.1.1 shows the basic structures of speaker identification and verification systems.

3.1 Speaker Identification

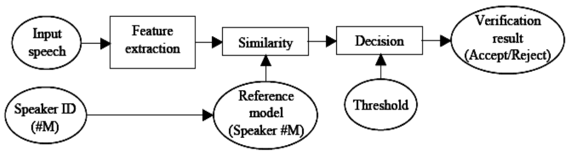


Figure 3.1.1: Basic structure of speaker recognition system

3.2 Speaker Verification

All technologies of speaker recognition, identification and verification, text-independent and text-dependent, each has its own advantages and disadvantages and may requires different treatments and techniques.

The choice of which technology to use is application-specific. The system that we will develop 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 3.1.1): 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.

All speaker recognition systems have to serve two 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 3.1.1), the input speech is matched with stored reference model(s) and recognition decision is made.

4. The Concept Of MFCC

The Mel-Frequency Cepstrum (MFC) is a representation of short-term power spectrum of a sound. The MFCCs are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum.

The cepstrum is a common transform used to gain information from a person's speech signal. It can be used to separate the excitation signal (which contains the words and the pitch) and the transfer function (which contains the voice quality). It is the result of taking Fourier transform of decibel spectrum as if it were a signal. We use cepstral analysis in speaker identification because the speech signal is of the particular form above, and the "cepstral transform" of it makes analysis incredibly simple.

Mathematically, cepstrum of signal = FT[log{FT(the windowed signal)}]

Algorithmically, the concept of cepstrum is presented here in the form of a block diagram. Figure 4.1 shows the flow chart that describes as to how to obtain cepstrum from a signal.

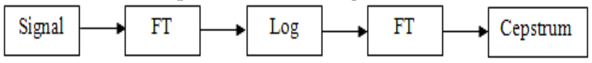
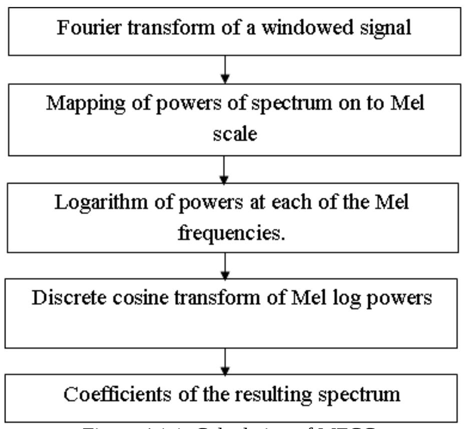


Figure 4.1: Flow chart of a 'Cepstrum'

The cepstrum can be seen as information about rate of change in the different spectrum bands. It is now used as an excellent feature vector for representing the human voice and musical signals.

4.1 Calculation of MFCCs:

MFCCs are commonly calculated by first taking the Fourier transform of a windowed excerpt of a signal and mapping the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows. Next the logs of the powers at each of the mel frequencies are taken, Direct Cosine Transform is applied to it (as if it were a signal). The MFCCs are the amplitudes of the resulting spectrum. This procedure is represented step-wise in the figure below.


Fourier transform of a windowed signal
Mapping of powers of spectrum on to Mel scale
Logarithm of powers at each of the Mel frequencies.
Discrete cosine transform of Mel log powers
Coefficients of the resulting spectrum
Figure 4.1.1: Calculation of MFCCs

5. Feature Extraction

5.1 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 to as the signal-processing front end. The speech signal is a slow time varying signal (it is called quasi-stationary). 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 characteristics change to reflect the different speech sounds being spoken. Therefore, short-time spectral analysis is the most common way to characterize the speech signal.

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 a linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz. The process of computing MFCCs is described in more detail next.

5.1 MFCC Processing

A block diagram of the structure of an MFCC processor is given in Figure 5.1.1. 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, MFFCs are shown to be less susceptible to mentioned variations.

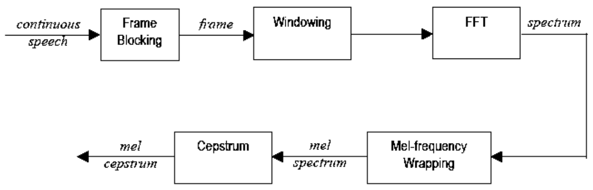


Figure 5.1.1: Block diagram of a MFCC processor

5.1(a) Frame Blocking

The concept of 'short time analysis' is fundamental to most speech analysis techniques. The assumption made is that, over a long interval of time, speech waveform is not stationary but that, over a sufficiently short time interval say about 10-30msec, it can be considered stationary. This is due to that fact that the rate at which the spectrum of the speech signal changes is directly dependant on the rate of movement of the speech articulators. Since this is limited by physiological constraints, most speech analysis systems operate at uniformly spaced time intervals or frames of typical duration 10-30 msec. In frame blocking, 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 msec windowing and facilitate the fast radix-2 FFT) and M = 100.

5.1(b) 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. In other words, when we perform Fourier Transform, it assumes that the signal repeats, and the end of one frame does not connect smoothly with the beginning of the next one. This introduces some glitches at regular intervals. So we have to make the ends of each frame smooth enough to connect with each other. This is possible by a processing called Windowing. In this process, we multiply the given signal (frame in this case) by a so called Window Function. The following figures illustrate the concept of windowing:

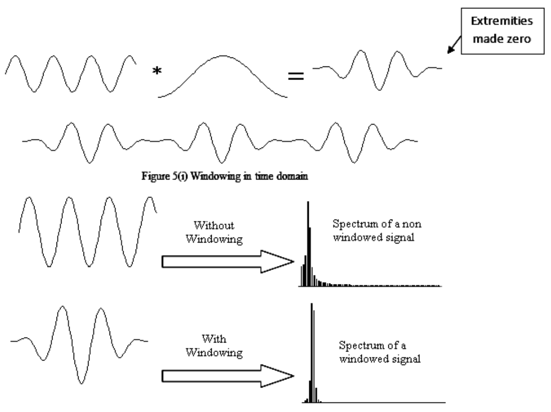

Extremities made zero
*
=
Figure 5(i) Windowing in time domain
Without Windowing
Spectrum of a non windowed signal
With Windowing
Spectrum of a windowed signal

Figure 5.1.2: Windowing in time and frequency domain

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

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

It is preferred to use a 'soft windowing' technique, which smoothly tapers the ends of the speech segment to zero. There are many 'soft windows' which can be used, but usually Hamming window is used, which has the form

$$w(n) = 0.53836 - 0.46164\ \cos\left(\frac{2\pi n}{N-1}\right)$$

where,

$$0 \le n \le N-1$$

The graphical representations of the hamming window are shown in figure 5.1.3.

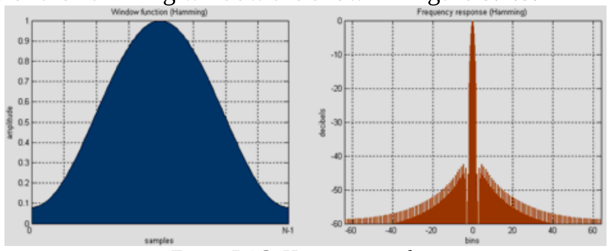


Figure 5.1.3: Hamming window

5.1(c) 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:

$$X_n = \sum_{k=0}^{N-1} x_k e^{-2\pi jkn/N}, \qquad n = 0,1,2,\dots,N-1$$

Note that we use j here to denote the imaginary unit, i.e.

$$j = \sqrt{-1}\ .$$

In general X_n's are complex numbers. The resulting sequence $\{X_n\}$ is interpreted as follow: the zero frequency corresponds to n = 0, positive frequencies

$$0 < f < F_s / 2$$

Correspond to values

$$1 \le n \le N/2 - 1$$

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 periodogram.

5.1(d) 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. As a reference point, the pitch of a 1 kHz tone, 40 dB above the perceptual hearing threshold, is defined as 1000 mels. Therefore we can use the following approximate formula to compute the mels for a given frequency f in Hz:

$$mel(f) = 2595 * \log_{10}(1 + f / 700)$$

One approach to simulating the subjective spectrum is to use a filter bank, spaced uniformly on the mel scale.

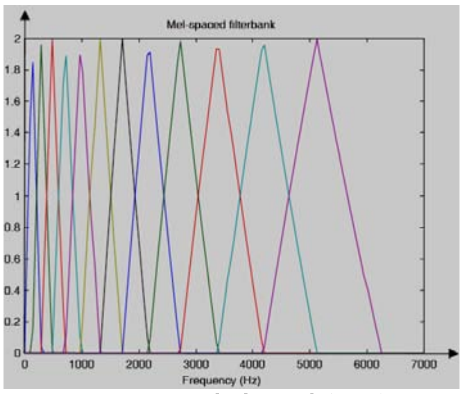


Figure 5.1.4: Mel Filter Bank (K=12)

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 modified spectrum of S(ω) thus consists of the output power of these filters when S(ω) is the input. The number of mel spectrum coefficients, K, is typically chosen as 12 or 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 5.1.4 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.

5.1(e) 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

$$\tilde{S}_k , \; k = 1,2,\ldots, K ,$$

we can calculate the MFCC's, as

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

Note that we exclude the first component, C_o , 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.

5.1(f) Modifications For Improving Robustness

The purpose of modification in the MFCC based algorithms generally being used was to improve its performance by making it more robust and making it faster and computationally efficient. Thus, an effort to reduce the MIPs counts so that the algorithm can be considered for real time applications. The failure of the speaker recognition systems is that they are not robust enough to be considered for commercial applications, the intend was also to take a step forward in making them robust.

Modification of Window

Instead of using the hamming window, a more efficient Kaiser window is used that is based on the concept of minimizing the mean square error rather than maximum error. The explanation of the Kaiser window is given below. The Kaiser window has an adjustable parameter, α, which controls how quickly it approaches zero at the edges. It is defined by

$$w_n = \begin{cases} \frac{I_0\left(\alpha\sqrt{1-\left(\frac{2n}{N}-1\right)^2}\right)}{I_0(\alpha)} & \text{if } 0 \le n \le N \\ 0 & \text{otherwise} \end{cases}$$

where $I_0(x)$ is the zeroth order modified Bessel function. The higher α, the narrower gets the window and therefore, due to the not so severe truncation then, the less severe are the bumps above.

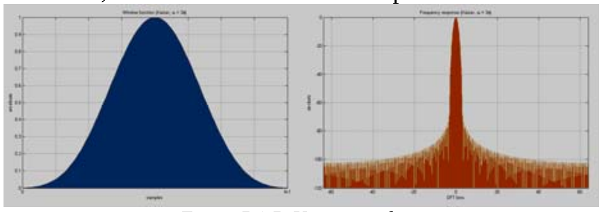


Figure 5.1.5: Kaiser window

In this work, the Kaiser windowed cosc function will be used to reconstruct gradients at arbitrary positions which are, of course, more costly.

Modification of DFT

Before applying to the mel filter banks the DFT of the absolute of the DFT of the frame was taken rather than taking the square. This not only reduces the cost of computing the square but also is an attempt of making the algorithm more robust.

6. Feature Matching

6.1 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 speech with the ID of the speaker. 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 paper, the VQ approach is 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 6.1.1 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, 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 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.

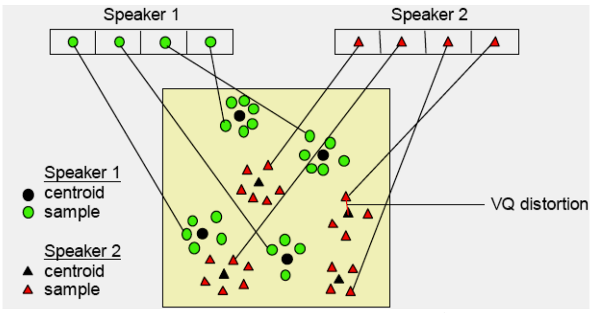


Figure 6.1.1: Vector quantization codebook formation

6.2 Clustering of the Training Vectors

After the enrolment session, the acoustic vectors extracted from input speech of a speaker provide a set of training vectors. As described above, the next important step is to build a speaker-specific VQ codebook for this 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

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

   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 6.2.1 shows 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.

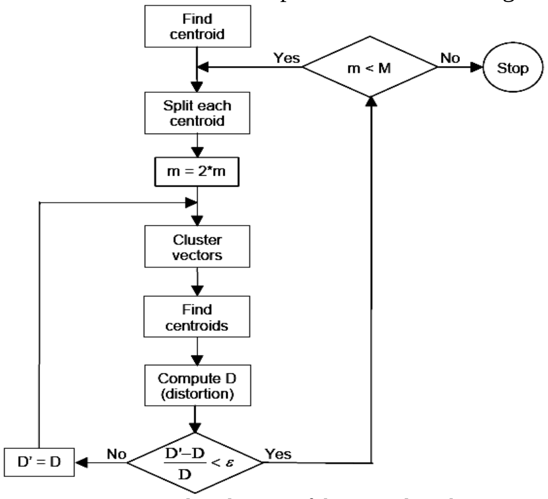


Figure 6.2.1 Flow diagram of the LBG algorithm

7. Case Studies And Results

Step 1: Technical data of samples

Read a sound file into MATLAB. Check it by playing the sound file in MATLAB using the function: sound.

We analyzed the sample "train\s1.wave" with the following command:

```
[s1, fs1] = wavread('train\s1.wav');
```

This command puts the signal data into the vector s1. Plot the signal to view it in the time domain. The raw data in the time domain has a great amount of data and for this reason it is difficult to analyze the voice characteristic. So the purpose of sampling in speech feature extraction is clear now.

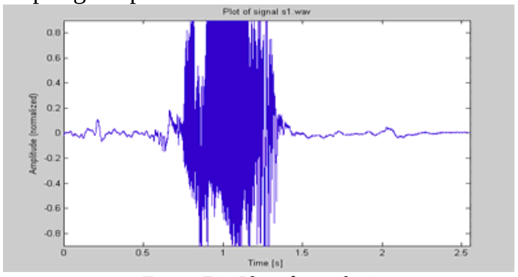


Figure 7.1: Plot of signal s1

Step 2: Power Spectrum

The next step is to compute the power spectrum. It is better to view the power spectrum on the log scale. Here, we have also plotted power spectrum in both linear and logarithmic. The result obtained is the following plot:

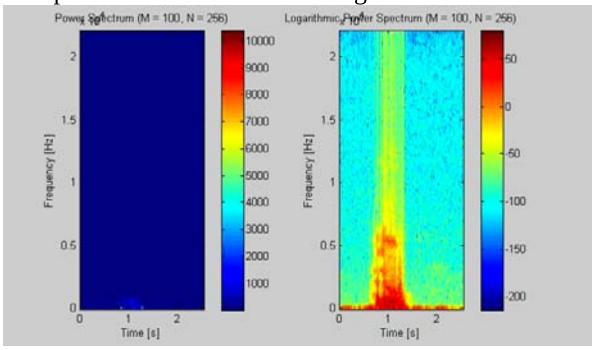


Figure 7.2: Power spectrum

Step 3: Power Spectrum with different M and N

Compute and plot the power spectrum of a speech file using different frames sizes: for example N = 128, 256 and 512. For N = 128, we have a high resolution of time. Furthermore each frame lasts for a very short period of time. This result shows that the signal for a frame doesn't change its nature (i.e. it will be for the same vowel or consonant). We have a poor frequency resolution. For N = 256 we have a compromise between the resolution in time and the frequency resolution.

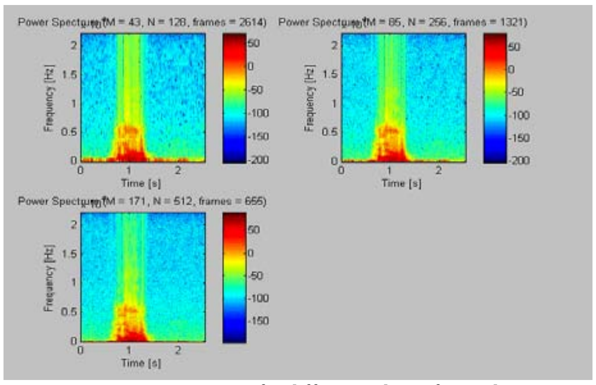


Figure 7.3: Spectrum for different values of M and N

For N = 512 we have an excellent frequency resolution but there are lesser frames, meaning that the resolution in time is strongly reduced. The value of 256 for N is an acceptable compromise. Furthermore the number of frames is relatively small, which will reduce computing time.

Step 4: Mel-Spaced Filter Bank

The Mel filter bank behaves like a succession of histograms on the spectrum. Each filter of the filter bank has a triangular frequency response. It quantifies the zone of the frequency spectrum. The filter bank is used to transform the spectrum of a signal into a representation which reflects more closely the behavior of the human ear. As the human ear (or the associated neurons) favors low frequencies for analyzing speech, the filters are denser for the lower frequencies. To mimic the human ear, the filters are linearly distributed for low frequencies (below 1kHz). For higher frequencies (above 1 kHz) the distribution of the filters is logarithmic. We calculated 20 filters here.

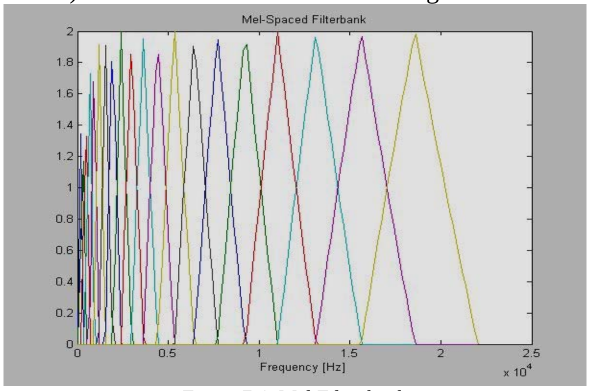


Figure 7.4: Mel Filter bank

Step 5: Spectrum before and after
Mel-Frequency wrapping

Compute and plot the spectrum of a speech file before and after the mel-frequency wrapping step. As we can see in the first plot, most of the information is contained in the lower frequencies. This information has been extracted and amplified in the second plot. The second plot therefore shows the main characteristics of the speech signal.

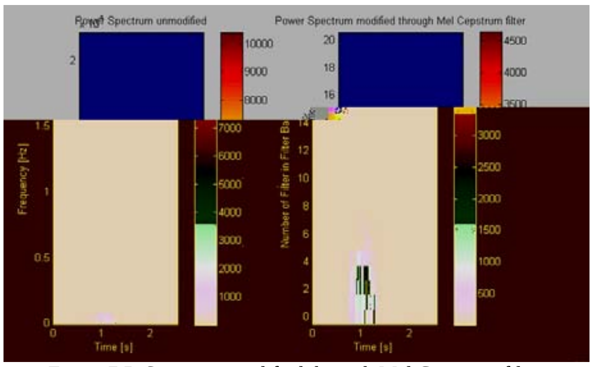


Figure 7.5: Spectrum modified through Mel Cepstrum filter

Step 6: 2D plot of acoustic vectors

We inspect the acoustic space (MFCC vectors) and plot the data points in any two dimensions (say the 5th and the 6th) in a 2D plane. The acoustic vectors of two different speakers are considered and the data points are plotted in two different colors.

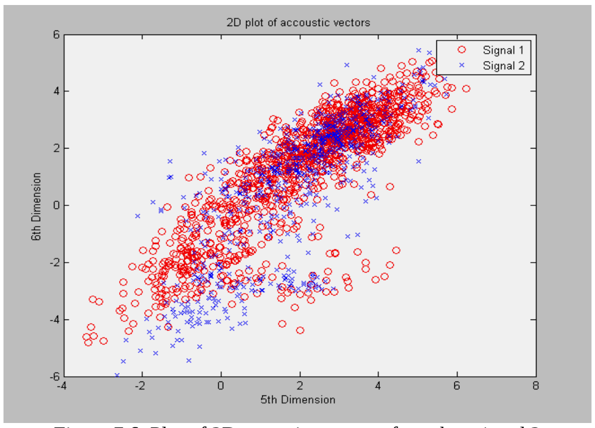


Figure 7.6: Plot of 2D acoustic vectors of speakers 1 and 2

Mostly the two areas overlap. But certain regions seem to be used exclusively by one or the other speaker. This is what will allow us to distinguish the different speakers.

Step 7: Plot of VQ codewords

We plot the data points of the trained VQ codeword in two dimensions. Fig.7.7(a) is the plot of the codebook of two speakers s1 and s2.

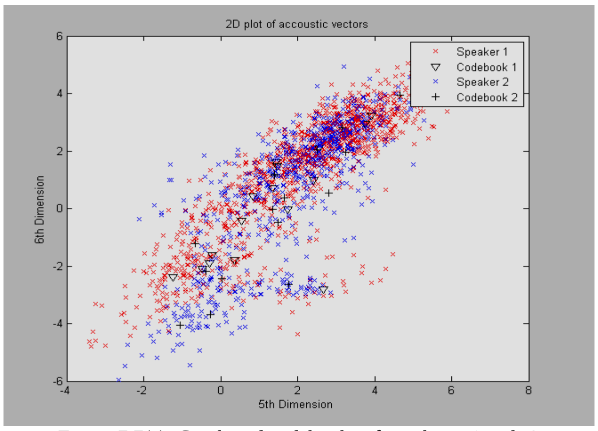


Figure 7.7(a): Combined codebooks of speakers s1 and s2

Figures 7.7(b) and 16(c) respectively show the individual plots of the vectors and code book of speakers s1 and s2 respectively.

Step 8: Recognition rate of the computer

Voice samples are again collected from speakers who had recorded training samples initially. The same microphone was used and the samples were collected. Mel-cepstral coefficients are calculated. These values are stored for the comparison phase. In the comparison phase, the distance between each test codeword and each codeword in the master codebook is computed. The difference is used to make recognition decision.

Screen shot of the result of our automatic speaker recognition system is shown in figure 7.8. Our system was able to recognize 8 out of 10 speakers. However, this test does not really represent the computer’s efficiency to recognize voices because we only tested on 10 persons, with only one training session and with only one word.

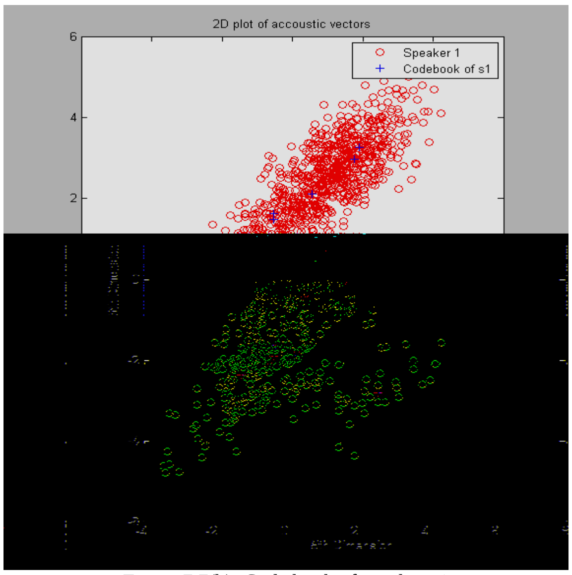


Figure 7.7(b): Code book of speaker s1

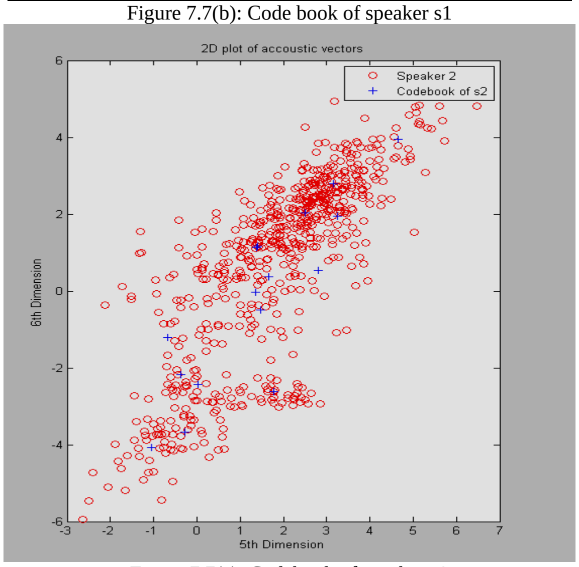


Figure 7.7(c): Codebook of speaker s2

8. Conclusion

In this work, we have developed a text-independent speaker identification system that is a system that identifies a person who speaks regardless of what is saying. Our speaker verification system consists of two sections (i) Enrolment section to build a database of known speakers and (ii) Unknown speaker identification system. Enrollment session is also referred to as training phase while the unknown speaker identification system is also referred to as the operation session 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. It consists of two main parts. The first part consists of processing each persons input voice sample to condense and summarize the characteristics of their vocal tracts. The second part involves pulling each person's data together into a single, easily manipulated matrix.

The speaker recognition system contains two main modules (i) feature extraction and (ii) 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.

The entire coding was done in MATLAB version 7.6. The system was tested many times with various databases and found to be very reliable.

9. 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] José Ramón Calvo de Lara, "A Method of Automatic Speaker Recognition Using Cepstral Features and Vectorial Quantization"10th Iberoamerican Conference on Pattern Recognition, CIARP 2005 Proceedings

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

[5] 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.

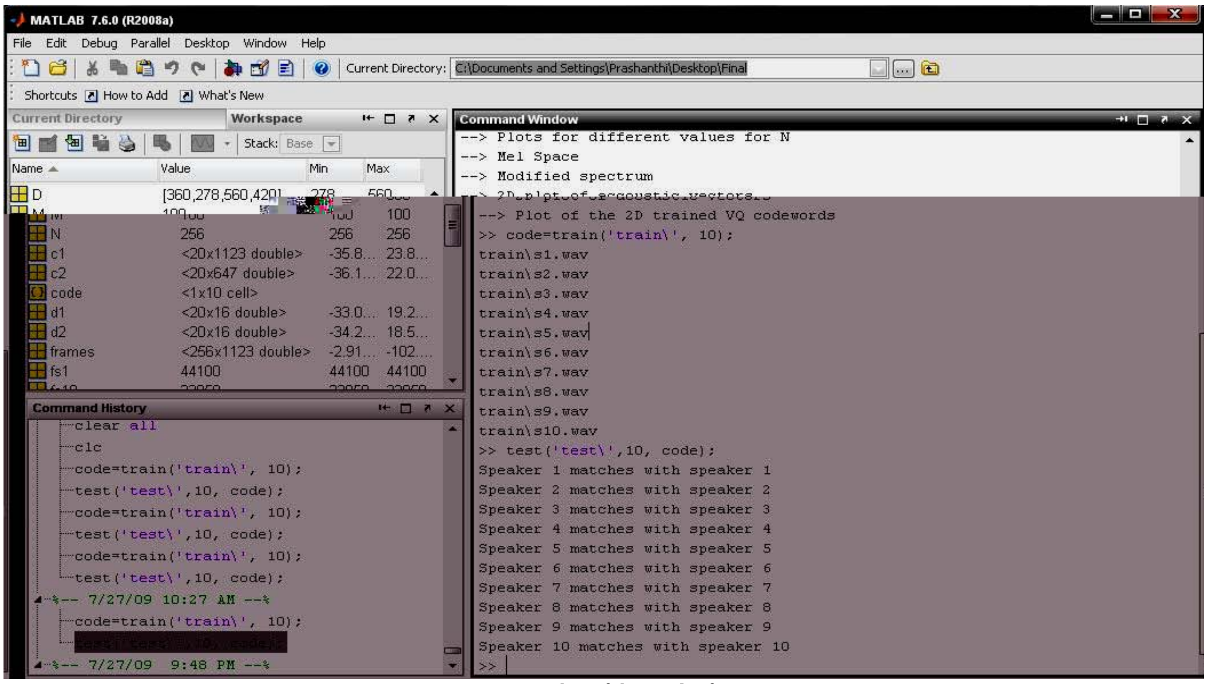
[6] Vijay K. Madisetti and Douglas B. Williams, Digital Signal Processing Handbook, Capman & Hall CRCnetBase.

[7] "ECE341 So You Want to Try and do Speech Recognition." A Simple Speech Recognition Algorithm. http://www.eecg.toronto.edu/~aamodt/ece341/speech-recognition

[8] The Mathworks – MATLAB and SIMULINK for Technical Computing. 10 June 2005. http://www.mathworks.com

[9] http://www.vision.caltech.edu/CNS248/Speech/speech95_1.ps.gz.

[10] Spoken Language Input http://cslu.cse.ogi.edu/HLTsurvey/ch1node1.html



MATLAB 7.6.0 (R2008a)
File Edit Debug Parallel Desktop Window Help
Current Directory: C:\Documents and Settings\Prashanthi\Desktop\Final
Shortcuts How to Add What's New
Current Directory
Workspace
Stack: Base
Command History
Command Window

Name	Value	Min	Max
D	[360,278,560,420]	278	560
M	100	100	100
N	256	256	256
c1	<20x1123 double>	-35.8...	23.8...
c2	<20x647 double>	-36.1...	22.0...
code	<1x10 cell>		
d1	<20x16 double>	-33.0...	19.2...
d2	<20x16 double>	-34.2...	18.5...
frames	<256x1123 double>	-2.91...	-102....
fs1	44100	44100	44100

```
clear all
clc
code=train('train\', 10);
test('test\',10, code);
code=train('train\', 10);
test('test\',10, code);
code=train('train\', 10);
test('test\',10, code);
%-- 7/27/09 10:27 AM --%
code=train('train\', 10);
test('test\',10, code);
%-- 7/27/09  9:48 PM --%
```

```
--> Plots for different values for N
--> Mel Space
--> Modified spectrum
--> 2D plot of accoustic vectors
--> Plot of the 2D trained VQ codewords
>> code=train('train\', 10);
train\s1.wav
train\s2.wav
train\s3.wav
train\s4.wav
train\s5.wav
train\s6.wav
train\s7.wav
train\s8.wav
train\s9.wav
train\s10.wav
>> test('test\',10, code);
Speaker 1 matches with speaker 1
Speaker 2 matches with speaker 2
Speaker 3 matches with speaker 3
Speaker 4 matches with speaker 4
Speaker 5 matches with speaker 5
Speaker 6 matches with speaker 6
Speaker 7 matches with speaker 7
Speaker 8 matches with speaker 8
Speaker 9 matches with speaker 9
Speaker 10 matches with speaker 10
>>
```

Figure 7.8: Screen shot of the result of our System