Despite there existing some working systems, implementations are still seen as challenging and are generally expected to be imperfect and imprecise.

A1.4 Stance taken with justification

Various speeches emotional recognition systems reviewed and discussed based on different approaches. Here the performance is also compared in terms of classifier, features, recognition rate, and datasets. Well-design classifiers have obtain high classification accuracies between different types of emotions. In this study HMM with adopting short time LFPC as a feature proves a good accuracy on different levels in the chart. Basharirad(2017) The majority of the current datasets are not capable for evaluation of speech emotion recognition. In most of them, it is hard even for human to specify different emotion of certain collected utterances; e.g. the human recognition accuracy was 80% for Berlin

A1.5 Conclusion

**Introduction**

B**2.1 Discuss and Compare recent algorithms for audio surveillance.**

This section intents to provide a representative picture of what has been developed so far in the area of audio surveillance (see also Table 1). The emphasis of previous approaches is mainly placed on the classifier, the feature extraction process, the training data, and the number of classes.

**feature extraction techniques for Speech Recognition.**

**classification techniques for Speech Recognition.**

Classifier compares the obtained features with stored features. Based upon this comparison classifier recognizes the particular speaker.

|  |  |  |  |
| --- | --- | --- | --- |
| Technique | Characteristics | Advantages | Disadvantages |
| Guassian Mixture Model | Unsupervised | Needs less trainning and test data | Compromise between DTW and HMM |
| Dynamic Time Warping (DTW) | Unsupervised | Requires less storage space,beneficial for variable length | Cross-channel issue |
| Hidden Markov Model (HMM) | Unsupervised | Rail system outputs,efficient performance | Computationally more complex,more storage space |
| Vector Quantization(VQ) | Unsupervised | Computationally less complex | Real time encoding is complex |
| Support Vector Machine (SVM) | Supervised | Simple operation | binary SVM has limitations in speaker recognition |

B2.3 **Capture real time audio samples and design and simulate a model for audio**

**analysis using MATLAB.**

The speech surveillance system that is being designed here can also be called as a Speaker Authorization/Recogniser. This system can be implemented at places of restricted access and the need for the identity of the person present at a highly protected areas fig shows the pictorial representation of the application.

**Speech Signals used for the system**

The audio samples used for the identification of the speaker are as described in tabel. All the voices are computer generated and each of them vary from one another.

|  |  |  |
| --- | --- | --- |
| File name | Voice | Uttered word |
| S1.wav | Female  (Computer Generated) | Zero |
| S2.wav | Female  (Computer Generated) | Zero |
| S3.wav | Female  (Computer Generated) | Zero |
| S4.wav | Female  (Computer Generated) | Zero |
| S5.wav | Female  (Computer Generated) | Zero |
| S6.wav | Female  (Computer Generated) | Zero |
| S7.wav | Female  (Computer Generated) | Zero |
| S8.wav | Female  (Computer Generated) | Zero |

**Feature Extracion**

Choosing which features to extract from speech is the most significant part of speaker recognition. Some popular features are: MFCCs, LPCs, Zero-Crossing Rates etc. In this work, I have concentrated on MFCCs and LPCs. Here is a brief overview of these features.

Mel-Frequency Cepstral Coefficients

Human hearing is not linear but logarithmic in nature. This implies that our ear acts as a filter. 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. The relationship between frequency in Hz and frequency in Mel scale is given by:

To calculate MFCCs, the steps are as follows. A very good tutorial is available in [2]. A schematic of this process is given in Figure 2.

1. The speech signal is divided into frames of 25ms with an overlap of 10ms. Each frame is multiplied with a Hamming window.
2. The periodogram of each frame of speech is calculated by first doing an FFT of 512 samples on individual frames, then taking the power spectrum as:

Where P(k) refers to power spectral estimate and S(k) refers to Fourier coefficients for the kth frame of speech and N is the length of the analysis window. The last 257 samples of the periodogram are preserved since it is an even function.

1. The entire frequency range is divided into ‘n’ Mel filter banks, which is also the number of coefficients we want. ‘For ‘n’ = 12, the filter bank is shown in Figure 3 - a number of overlapping triangular filters with increasing bandwidth as the frequency increases.
2. To calculate filter bank energies we multiply each filter bank with the power spectrum, and add up the coefficients. Once this is performed we are left with ‘n’ numbers that give us an indication of how much energy was in each filter bank.
3. We take the logarithm of these ‘n’ energies and compute its Discrete Cosine Transform to get the final MFCCs.

Linear Prediction Coefficients

LPCs are another popular feature for speaker recognition. To understand LPCs, we must first understand the Autoregressive model of speech. Speech can be modelled as a pth order AR process, where each sample is given by:

Each sample at the nth instant depends on ‘p’ previous samples, added with a Gaussian noise u(n). This model comes from the assumption that a speech signal is produced by a buzzer at the end of the tube (voiced sounds), with occasional added hissing and popping sounds.

LPC coefficients are given by α. To estimate the coefficients, we use the Yule-Walker equations. It uses the autocorrelation function Rx. Autocorrelation at lag l is given by:

While calculating ACF in Python, the Box-Jenkins method is used which scales the correlation at each lag by the sample variance so that the autocorrelation at lag 0 is unity.

The final form of the Yule-Walker equations is:

The solution for **a is given by:**

In this case, I have normalised the LPC coefficients estimated so that they lie between [-1,1]. This was seen to give more accurate results. We first divide speech into frames of 25ms with 10ms overlap, then calculate ‘p’ LPCs for each frame.

**Feature Matching**

The most popular feature matching algorithms for speaker recognition are Dynamic Time Warping (DTW), Hidden Markov Model (HMM) and Vector Quantization (VQ). Here, I have used Vector Quantization.

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.

LBG Algorithm

The LBG algorithm [Linde, Buzo and Gray], is used 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 yn according to the

where n varies from 1 to the current size of the codebook, and is a splitting parameter (we choose =0.01).

1. 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).
2. Centroid Update: update the codeword in each cell using the centroid of the training vectors assigned to that cell.
3. Iteration 1: repeat steps 3 and 4 until vector distortion for current iteration falls below a fraction of the pervious iteration’s distortion. This is to ensure that the process has converged.
4. 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.

**B2.4 Design an audio surveillance system using MATLAB.**

Feature Training

The main algorithms needed for speaker recognition have been implemented. Now, everything needs to be brought together to train our dataset and derive codebooks for each speaker using VQ. The number of speakers is nSpeaker = 8. As mentioned before, speech recordings of 8 female speakers uttering the word ‘zero’ has been taken for training and testing. Each codebook should have 16 codewords, hence nCentroid = 16 (it is highly recommended to keep this number a power of 2).

Codebooks for both MFCC features and LPC features are plotted for all 8 speakers. One of them is shown in Figure.

**B2.5 Test and validate the developed algorithm on benchmark audios.**

It is finally time to test our speaker recognition algorithm. the speaker recognition is done by comparing their feature vector to the codebooks of all trained speakers and computing the minimum distance between them. The results are yielding an accuracy of 37.5% with MFCC and 50% with LPC. Reasons for this low accuracy can be the fact that there wasn’t enough data to train on. Other complex classification algorithms such as ANNs and SVMs should yield better results. I observed that training with12 MFCC features and LPC coefficients of order 15 gives the best results. There are other parameters that can be varied, such as number of codewords in a codebook and FFT size while computing MFCCs. It is possible that a different combination of these will give higher accuracy.

The following table gives the identification results for each of the 8 speakers with MFCC and LPC coefficients and Vector Quantization with LBG algorithm for classification.

|  |  |  |
| --- | --- | --- |
| True Speaker | Identified as (MFCC) | Identified as(LPC) |
| S1 | S1 | S5 |
| S2 | S8 | S2 |
| S3 | S5 | S1 |
| S4 | S6 | S4 |
| S5 | S5 | S5 |
| S6 | S3 | S1 |
| S7 | S8 | S8 |
| S8 | S8 | S8 |
|  | Accuracy=37.5% | Accuracy = 50% |

**C3.1 Analysis and comparison of existing algorithms for emotion detection**

**C3.3 Development of a software reference model of the chosen algorithm using PYTHON**

* Firstly, the train and test data are split and are placed in different folders named train\_sounds and new\_test\_sounds.

X, sample\_rate = librosa.load(file\_name)

* All the features of the input files present in train\_sounds is extracted using a function.

def extract\_feature(X):

stft = np.abs(librosa.stft(X))

mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample\_rate, n\_mfcc=40).T, axis=0)

chroma = np.mean(librosa.feature.chroma\_stft(S=stft, sr=sample\_rate).T, axis=0)

mel = np.mean(librosa.feature.melspectrogram(X, sr=sample\_rate).T, axis=0)

contrast = np.mean(librosa.feature.spectral\_contrast(S=stft, sr=sample\_rate).T, axis=0)

tonnetz = np.mean(librosa.feature.tonnetz(y=librosa.effects.harmonic(X),

sr=sample\_rate).T, axis=0)

return mfccs, chroma, mel, contrast, tonnetz

* The extracted features are made into vectors and are stacked and labelled.

def parse\_audio\_files(path):

features = np.empty((0, 193))

for fn in glob.glob(path):

try:

mfccs, chroma, mel, contrast, tonnetz = extract\_feature(fn)

except Exception as e:

print("Error encountered while parsing file: ", fn)

continue

ext\_features = np.hstack([mfccs, chroma, mel, contrast, tonnetz])

features = np.vstack([features, ext\_features])

target\_files.append(fn)

return np.array(features)

* The SVM classifier is loaded and is given the features to fit itself(Training process).

model = pickle.load(open(filename, 'rb'))

prediction = model.predict(ts\_features)

* Finally the each of the speech file from the floder new\_test\_sounds are read one by one and the features of them are extracted and the classifier is made to predict the output.

for i, val in enumerate(prediction):

print("Input File: ", target\_files[i], "|", " Predicted Emotion Is:", classes[int(val)])

**C3.4 Testing and validation of the reference model with the given speech files**

Emotion recognition by means of SVM is implemented and allowed to test the accuracy of the system.

The provided Data set is been tested over the model and the result of the software programme are as follows.

Input File: ./new\_test\_sounds/YAF\_young\_angry.wav | Predicted Emotion Is: angry

Input File: ./new\_test\_sounds/YAF\_young\_disgust.wav | Predicted Emotion Is: disgust

Input File: ./new\_test\_sounds/YAF\_young\_fear.wav | Predicted Emotion Is: fear

Input File: ./new\_test\_sounds/YAF\_young\_happy.wav | Predicted Emotion Is: happy

Input File: ./new\_test\_sounds/YAF\_young\_neutral.wav | Predicted Emotion Is: neutral

Input File: ./new\_test\_sounds/YAF\_young\_ps(Pleasant Surprise).wav | Predicted Emotion Is: surprise

Input File: ./new\_test\_sounds/YAF\_young\_sad.wav | Predicted Emotion Is: disgust

|  |  |  |
| --- | --- | --- |
| **Audio file** | **Emotion** | **Predicted Emotion** |
| YAF\_young\_angry.wav | Angry | Angry |
| YAF\_young\_disgust.wav | Disgust | Disgust |
| YAF\_young\_fear.wav | Fear | Fear |
| YAF\_young\_happy.wav | Happy | Happy |
| YAF\_young\_neutral.wav | Neutral | Neutral |
| YAF\_young\_ps(Pleasant Surprise).wav | Surprise | Surprise |
| YAF\_young\_sad.wav | Sad | disgust |
|  |  | Accuracy=85.71% |

**C3.5 Conclusion and justification.**