Implementation of Speech Recognition System for Mobile Devices

A B. Tech Project Report Submitted in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Technology

by

Abhinav Singh
(140101002)

under the guidance of

Prof. Pradip K. Das

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM

<u>CERTIFICATE</u>

*This is to certify that the work contained in this thesis titled "**Implementation of Speech Recognition System for Mobile Devices**" is a bonafide work of **Abhinav Singh** (**Roll No. 140101002**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Prof. Pradip K. Das**

Professor

November, 2017 — Department of Computer Science & Engineering

Guwahati — Indian Institute of Technology Guwahati, Assam

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Prof. Pradip K. Das, for his continuous support, enthusiasm and immense knowledge. He has always guided and motivated me towards the completion of my work despite his busy schedule.

I also take this opportunity to thank my mentor Parabattina Bhagath P for his consistent assistance and the Department of Computer Science and Engineering, IIT Guwahati for providing me the facilities that were essential to carry out my project.

Abstract

Automatic speech recognition systems are an integral part of every voice-based assistant. Voice centric interfaces have become widely available with the evolution of mobile phones. Advancements in speech recognition algorithms, increased computational power and networking capacities have enabled this growth. Its applications include but are not limited to dialing, web search, and dictation of text messages. In this work, we implement a lightweight embedded speech recognition system using phoneme segmentation, linear predictive coding and template matching. We have taken a dataset of digits spoken in moderately quiet environment. The results indicate that the proposed system is able to identify the spoken digits with over 75% accuracy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Speech recognition, which is able to recognize human speech and convert it to text, has emerged as the new trending technology in IT industry. Mobile users desire devices that they can control without physical interaction.

Applications like Siri and Cortana are steadily gaining popularity with their hands free features like taking reminders and notes, opening other applications, and saying witty replies. With the advancement in network capabilities, these applications can send data over to the server where sophisticated algorithms provide exceptionally accurate recognition results. In countries like India where over 75 percent of the population has no internet, these applications fail. Moreover, privacy is a big issue as these systems share the voice data with third party servers.

In this work, we aim to implement a light-weight embedded speech recognition system for mobile devices. In chapter 2, we discuss few related works and technologies of the proposed system. Chapter 3 presents the design and implementation of the proposed speech recognition system. Finally, we draw out conclusions and discuss the future work in chapter 5.

Chapter 2

Related Works and Technologies

2.1 Phoneme Segmentation

The accurate segmentation and labeling of speech into phoneme units is useful for ASRs and speech synthesis. It is essential to embedded speech recognition system due to the constraint on the lexicon size [LLNB04]. Development of an embedded ASR requires an accurately segmented speech database. Several approaches such as dynamic time warping [MD97] and viterbi forced alignment [BFO93] have been used for automatic segmentation.

For embedded devices, we look forward to less computationally intensive methods for phoneme segmentation. [AEEM01] presents fourier analysis to segment phonemes based on energy jumps. The algorithm achieves an accuracy of 74% comparable to more sophisticated HMM based methods, and with no over segmentation.

2.2 Native Development on Android

[FWSC10] discusses a new component development approach by using Java Native Interface technology. The Native Development Kit (NDK) is a set of tools that allows writing native code for Android in C++. Speech processing and recognition requires fine grain control over memory management and speed, hence NDK is preferred over Dalvik Virtual Machine.

2.3 Distance Measures

Cepstral coefficients are widely used in speech recognition. Various distortion methods have been applied to find the distance between two coefficients.

The Itakura Saito distance is a measure of the difference between a spectrum and the estimate of that spectrum. The cepstral distance between points p and q is the length of the line segment joining them. Also known as Euclidean distance, this distance causes uneven weighing of the different orders of coefficients, and hence is rarely used. [NSRK85] compares these distances and concludes that Itakura Distance (a.k.a log likelihood ratio) performs better than Euclidean distance.

[Toh87] brings out a novel distance measure, the Tohkura distance - a weighted distance that outperforms even Itakura Distance. It also mentions the Mahalanobis distance, a multi-dimensional generalization of the idea of measuring how many standard deviations away P is from the mean of D. The sensitivity introduced by the matrix inversion is one of the main difficulties for applying this distance to speech recognition.

2.4 Network Speech Recognition

In this mode of speech recognition, the feature extraction and recognition are both delegated to the network server. The server is capable of doing complex computation and returns extremely accurate results. The system can also be upgraded without changes to the local distribution. One of the major cons of this mode is the dependence on telephone network and the performance degradation caused by low bit rate codecs, which becomes sever in presence of transmission errors [KTH$^+$].

Chapter 3

Design and Implementation

3.1 Preprocessing

First, the signal is fixed for DC offset by subtracting the mean sample value, and normalized to a prechosen threshold. Then we trim the signals using the algorithm described in [RS75].

The trimming algorithm uses short-time energy and zero crossing rates and it possesses a self-adapting nature towards the background acoustic environment. The energy is given by sum of magnitudes of samples near the window,

$$E(n) = \sum_{w} |s(n+i)| \tag{3.1}$$

The threshold variables can be calculated as given in following equations [RS75].

$$I1 = 0.03 \times (IMX - IMN) + IMN \tag{3.2}$$

$$ITL = MIN(I1, 4 \times IMN), ITU = 5 \times ITL \tag{3.3}$$

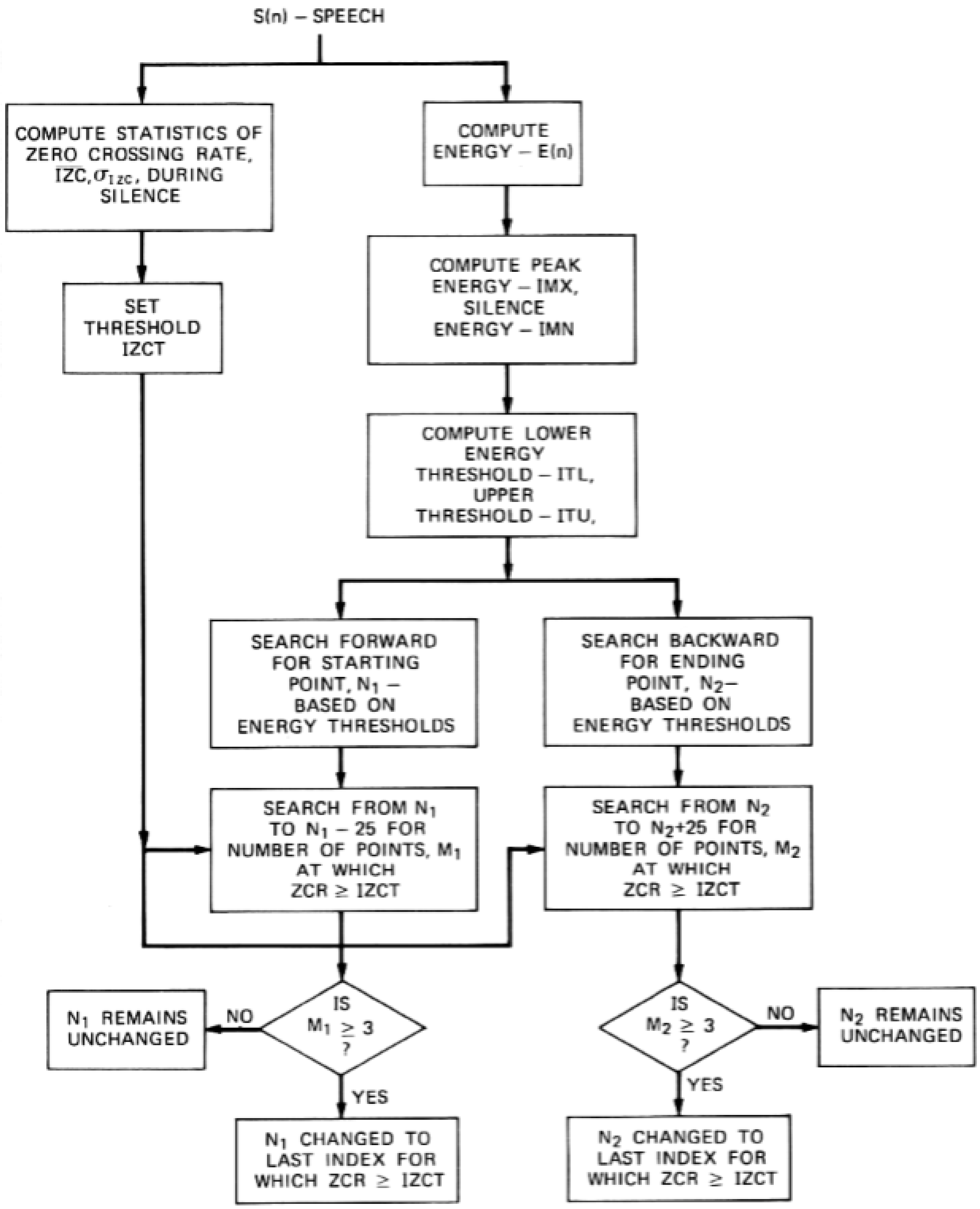

S(n) – SPEECH
COMPUTE STATISTICS OF ZERO CROSSING RATE, $\overline{IZC}, \sigma_{IZC}$, DURING SILENCE
COMPUTE ENERGY – E(n)
COMPUTE PEAK ENERGY – IMX, SILENCE ENERGY – IMN
SET THRESHOLD IZCT
COMPUTE LOWER ENERGY THRESHOLD – ITL, UPPER THRESHOLD – ITU,
SEARCH FORWARD FOR STARTING POINT, N_1 – BASED ON ENERGY THRESHOLDS
SEARCH BACKWARD FOR ENDING POINT, N_2– BASED ON ENERGY THRESHOLDS
SEARCH FROM N_1 TO $N_1 - 25$ FOR NUMBER OF POINTS, M_1 AT WHICH ZCR $\geq$ IZCT
SEARCH FROM N_2 TO N_2+25 FOR NUMBER OF POINTS, M_2 AT WHICH ZCR $\geq$ IZCT
IS $M_1 \geq 3$?
IS $M_2 \geq 3$?
NO
NO
N_1 REMAINS UNCHANGED
N_2 REMAINS UNCHANGED
YES
YES
N_1 CHANGED TO LAST INDEX FOR WHICH ZCR $\geq$ IZCT
N_2 CHANGED TO LAST INDEX FOR WHICH ZCR $\geq$ IZCT

Fig. 3.1 Trimming algorithm flowchart [RS75]

3.2 Phoneme Segmentation using Pitch Period Analysis

A speech signal can be classified into voiced, unvoiced and silence regions. There is no excitation during the silence region and majority of speech regions including vowels and semi vowels are voiced. We use [DSR76] to label the segments as voiced or unvoiced and then find their boundaries.

- Find peak signal threshold from the beginning background noise.
- Select segments of length 300 samples with a moving window of 100 samples.
- The segment is silence if peak signal level is below threshold.
- Find clipping level as a fixed percentage of the minimum of the maximum absolute values in the first and last 100 samples of segment.
- Clip the signal and then compute the autocorrelation function.
- Compare peak of autocorrelation function with the fixed threshold and classify the segment as unvoiced if the peak falls below, and as voiced if above.

3.3 Linear Predictive Coding Feature Extraction

Linear Predictive Coding is very useful for encoding high quality speech at a low bit rate and provides extremely accurate estimates of speech parameters. Its main advantage comes from the reference to a simplified vocal tract model and the analogy of a source-filter model with the speech production system.

After cutting the phonetic units, we find out the linear predictive cepstral coefficients(LPCC) using the autocorrelation method from [LRR78]. First, the phoneme is divided into overlapping segments, and then we apply a tapering window to each segment to hide the discontinuities.

Autocorrelation Analysis

The next step is to find the autocorrelation for each frame of windowed signal with

$$\sum_{n=0}^{n-1-m} \tilde{x}_l(n)\tilde{x}_l(n+m), \quad m = 0, 1, ..., p. \tag{3.4}$$

LPC Analysis

The p + 1 autocorrelations are mutated to LPC coefficients by using Durbins method:

$$E^{(0)} = R(0) \tag{3.5}$$

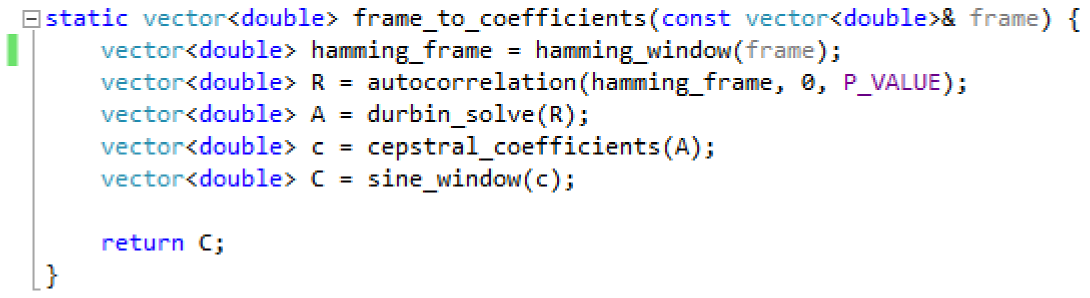
$$k_i = \frac{R(i) - \sum_{j=1}^{i-1} \alpha_j^{i-1} R(|i-j|)}{E^{i-1}}, \quad 1 \leq i \leq p. \tag{3.6}$$

$$\alpha_i^{(i)} = k_i \tag{3.7}$$

$$\alpha_j^{(i)} = \alpha_i^{(i-1)} - k_i \alpha_{i-1}^{(i-1)}, \quad 1 \leq j \leq i-1. \tag{3.8}$$

$$E^i = (1 - k_i^2)E^{i-1} \tag{3.9}$$

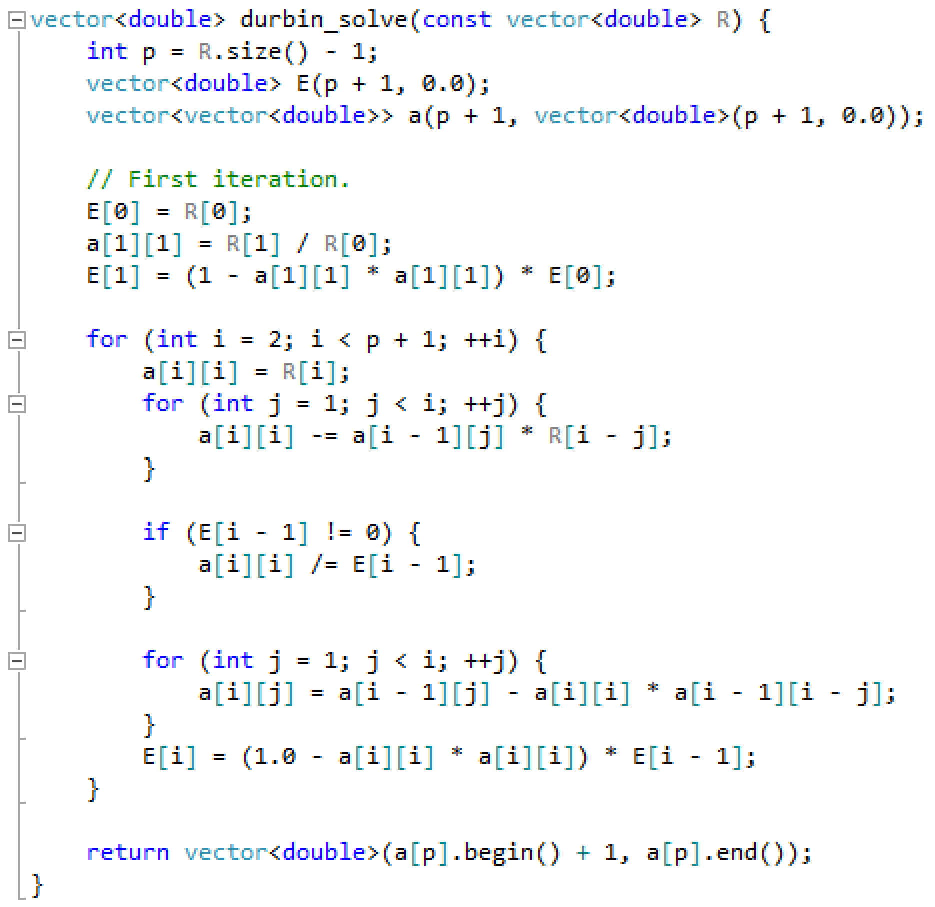
In the last step, the LPC cepstral coefficients are derived from the LPC coefficients. A sine window is applied over the cepstral coefficients to desensitize them towards noise and noise like sounds.



```
static vector<double> frame_to_coefficients(const vector<double>& frame) {
    vector<double> hamming_frame = hamming_window(frame);
    vector<double> R = autocorrelation(hamming_frame, 0, P_VALUE);
    vector<double> A = durbin_solve(R);
    vector<double> c = cepstral_coefficients(A);
    vector<double> C = sine_window(c);

    return C;
}
```

Fig. 3.2 Linear predictive coding implementation



```
vector<double> durbin_solve(const vector<double> R) {
    int p = R.size() - 1;
    vector<double> E(p + 1, 0.0);
    vector<vector<double>> a(p + 1, vector<double>(p + 1, 0.0));

    // First iteration.
    E[0] = R[0];
    a[1][1] = R[1] / R[0];
    E[1] = (1 - a[1][1] * a[1][1]) * E[0];

    for (int i = 2; i < p + 1; ++i) {
        a[i][i] = R[i];
        for (int j = 1; j < i; ++j) {
            a[i][i] -= a[i - 1][j] * R[i - j];
        }

        if (E[i - 1] != 0) {
            a[i][i] /= E[i - 1];
        }

        for (int j = 1; j < i; ++j) {
            a[i][j] = a[i - 1][j] - a[i][i] * a[i - 1][i - j];
        }
        E[i] = (1.0 - a[i][i] * a[i][i]) * E[i - 1];
    }

    return vector<double>(a[p].begin() + 1, a[p].end());
}
```

Fig. 3.3 Durbin solve implementation

3.4 Template Matching

We use Tohkura Distance for finding the distance between coefficients. The measure is a statistically weighted distance measure with weights equal to the inverse variance of the cepstral coefficients [Toh87].

The coefficients are the templates for recognising digits. We generate the training coefficients for each phoneme in each utterance. When we get the test signal, we apply the same preprocessing and feature extraction to get the test coefficients for each phoneme of the test signal. Then for each phoneme unit from test signal, we find its distance with respect to each phoneme unit in our training data using their coefficients. We output the digit, which gives the least average distance for all its phonemes.

Chapter 4

Experiment

We obtained the dataset of spoken digits, 15 utterances for each person per digit, and 3 persons in total. The recordings were done in moderately quiet environments using a general-purpose microphone.

For the obtained dataset, the trimming algorithm presented exceptionally accurate results. The results were close to 88% for the error of speech signal's endpoints missing by 100ms.

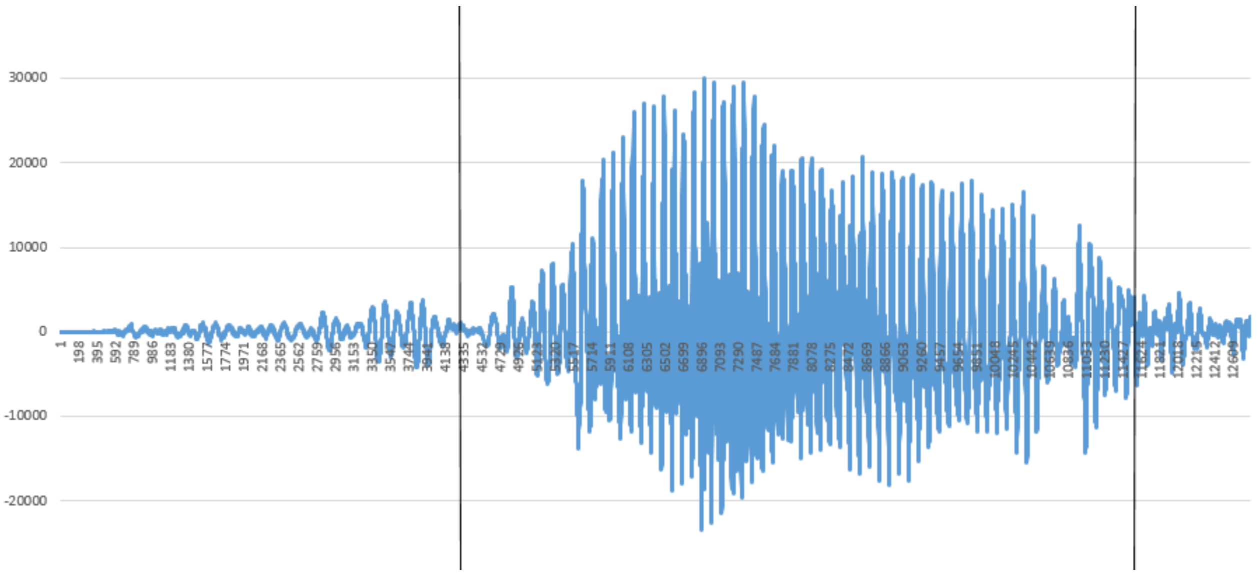


Fig. 4.1 Automatic trimming of Nine

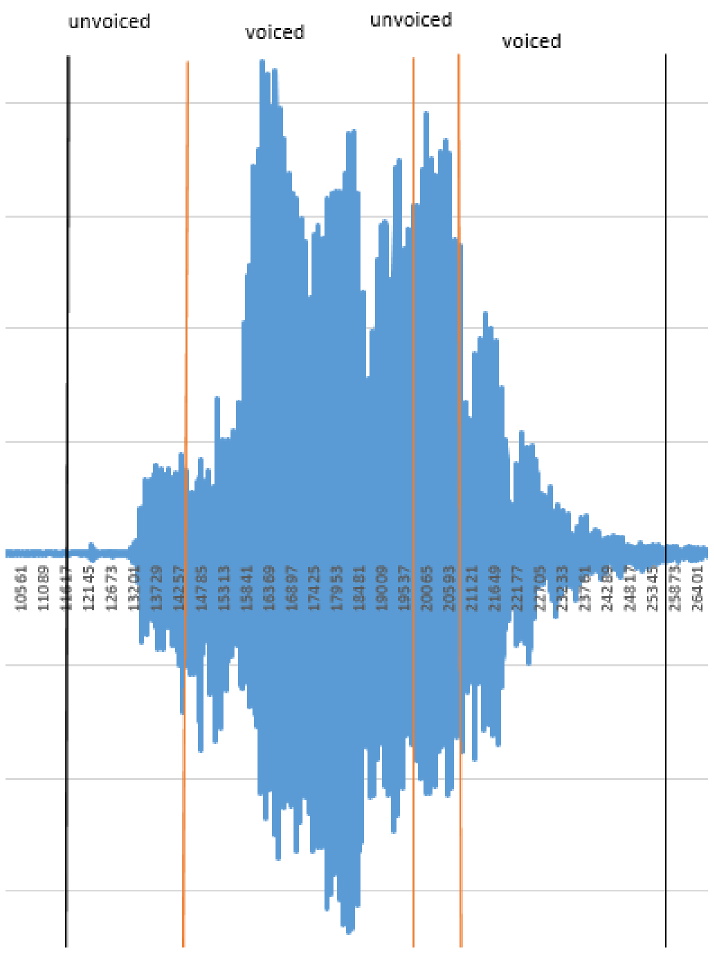


Fig. 4.2 Phoneme segmentation of Zero

The figure 4.2 shows the utterance of zero in which phoneme segmentation gives good result. For many other utterances, our implementation gave erratic results. The problem was seen to be an improper choice of voiced/unvoiced threshold, hence there were few small unvoiced segments between long stretches of voiced segments. This behaviour caused a major decrease (from 77% down to 69%) in the accuracy of the system as can be seen by the tables below. We also notice that due to the calculation of autocorrelation function, phoneme segmentation was approximately a 5 second operation for each utterance. This would make it unsatisfactory for real-time use.

Digit	0	1	2	3	4	5	6	7	8	9
Accuracy	62	80	86	64	95	77	73	73	86	82

Table 4.1 Accuracy without using phoneme segmentation

Digit	0	1	2	3	4	5	6	7	8	9
Accuracy	73	46	66	73	66	80	53	80	86	66

Table 4.2 Accuracy with using phoneme segmentation

Chapter 5

Conclusion and Future Work

It was noticed that the autocorrelation based phoneme segmentation is a slow algorithm for a real-time speech recognition system. We would like to improve phoneme segmentation by looking at other techniques. Our system did not use any sophisticated algorithms for template matching. We noticed that the system performed unsatisfactorily for changes in accents and environments. The main cause of this is the direct use of the cepstral coefficients; we would like to improve on that by using aligning techniques such as Dynamic Time Warping. Another improvement we can make in the system is by profiling the mobile user. It is generally seen that a there is a one to one correspondence between users and mobiles. Hence, we can save user's old recordings and improve our system continuously.

References

[AEEM01] G. Aversano, A. Esposito, A. Esposito, and M. Marinaro. A new text-independent method for phoneme segmentation. In *Proceedings of the 44th IEEE 2001 Midwest Symposium on Circuits and Systems. MWSCAS 2001 (Cat. No.01CH37257)*, volume 2, pages 516–519 vol.2, 2001.

[BFO93] Fabio Brugnara, Daniele Falavigna, and Maurizio Omologo. Automatic segmentation and labeling of speech based on hidden markov models. *Speech Communication*, 12(4):357–370, 1993.

[DSR76] J. Dubnowski, R. Schafer, and L. Rabiner. Real-time digital hardware pitch detector. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(1):2–8, Feb 1976.

[FWSC10] X. Fu, X. Wu, M. Song, and M. Chen. Research on audio/video codec based on android. In *2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*, pages 1–4, Sept 2010.

[KTH$^+$] Anuj Kumar, Anuj Tewari, Seth Horrigan, Matthew Kam, Florian Metze, and John Canny. Rethinking speech recognition on mobile devices.

[LLNB04] C. Levy, G. Linares, P. Nocera, and J. F. Bonastre. Reducing computational and memory cost for cellular phone embedded speech recognition system. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages V–309–12 vol.5, May 2004.

[LRR78] Ronald W. Schafer Lawrence R. Rabiner. *Digital Processing of Speech Signals*. Prentice-Hall, 1978.

[MD97] Fabrice Malfrre and Thierry Dutoit. High-quality speech synthesis for phonetic speech segmentation. volume 3329, 01 1997.

[NSRK85] N. Nocerino, F. Soong, L. Rabiner, and D. Klatt. Comparative study of several distortion measures for speech recognition. In *ICASSP '85. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 10, pages 25–28, April 1985.

[RS75] L. R. Rabiner and M. R. Sambur. An algorithm for determining the endpoints of isolated utterances. *The Bell System Technical Journal*, 54(2):297–315, Feb 1975.

[Toh87] Y. Tohkura. A weighted cepstral distance measure for speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(10):1414–1422, Oct 1987.