Lund University

Algorithms in Signal Processors

ETIN80

Speech Recognition using a DSP

Authors:
Mattias Sonnerup
Mingzhe Guo
Venkatesh Prasath Manoharan
Ripudaman Khattar
Gani Kumisbek

Supervisors:
Mikael Swartling
Nedelko Grbic

March 16, 2018

Contents

1 Introduction

Speech recognition is becoming more common in today's society. Appearing in phones as well as other technology to help carry out voice instructions. With voice recognition like Siri and Amazon's alexa being so easy to use for looking up information and controling other technology, it is of no surprise that the demand for it is increasing. When demand for a technology increases it also becomes important for engineers to learn more about the technology behind it, in order to make better implementations of it.

The purpose of this project is to learn the very basic of speech recognition by implementing a system on a DSP that can distinguish between a few spoken words. At the beginning of the project we started with distinguishing between two words in matlab. In later stages we expanded the system to a DSP and distinguish between three instead of two.

2 Overview of the implementation

The system can be divided into three phases the pre-processing, processing and matching/recording phase.

2.1 Pre-processing phase

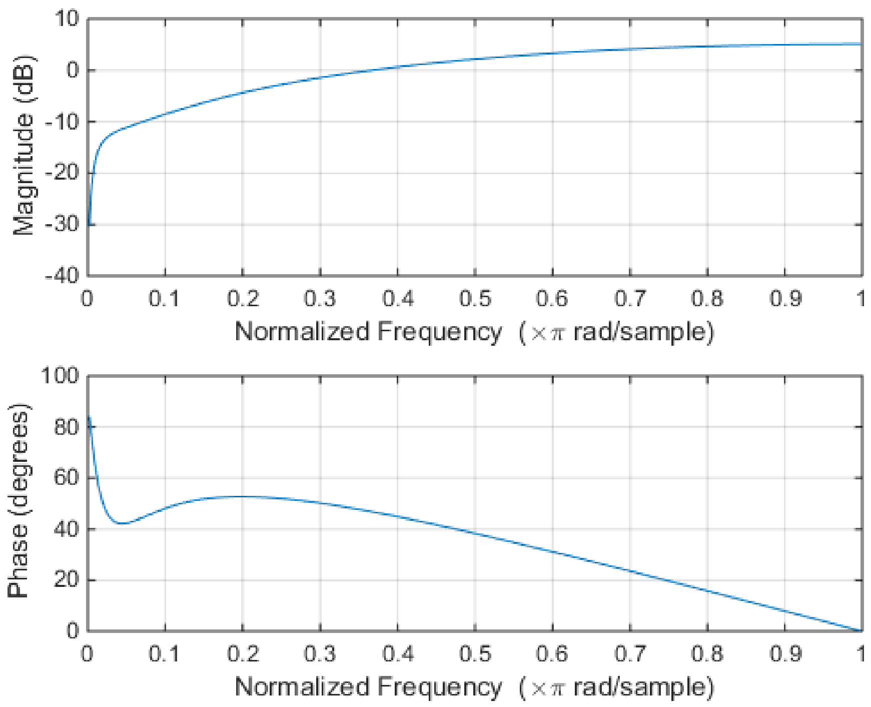
In the pre-processing state a signal from a microphone (DSP) or from a .wav file (matlab) is put through a filter as displayed in the figures below.

The filter is a convolution of a notch and pre-emphasis prefilter. The notch filter will filter out the noise in low frequency, while the pre-emphasis filter will raise the high frequency part in speech for distinguishing. We mix them up to get a combination of them by doing convolution for the filter coefficients, by doing so, our prefilter can have the function of both. In DSP, we use a function called 'biquad' to design that prefilter. The coefficients of A2,A1,B2,B1,B0 are -0.95,10.75,-1.75,1 respectively for the filter transfer function.

$$Y(z) = \frac{B(0) + B(1)z^{-1} + B(2)z^{-2}}{1 + A(1)z^{-1} + A(2)z^{-2}}$$

2.2 Processing phase

After filtering is done, the signal is processed in blocks. Each block consists of 320 samples that is equivalent to 20 miliseconds of speech. At the start of the phase the autocorrelation of a block is calculated. The autocorrelation is then used to generate coeffients of a wiener filter for the block, which is done through the levison and schur algoritmn. In this project 15 coeffients are generated for each block. When the coefficients are generated they are then saved into memory. Finally the phase is repeated for 100 blocks, which is equivalent to two seconds, before moving onto the recording phase or the matching phase.

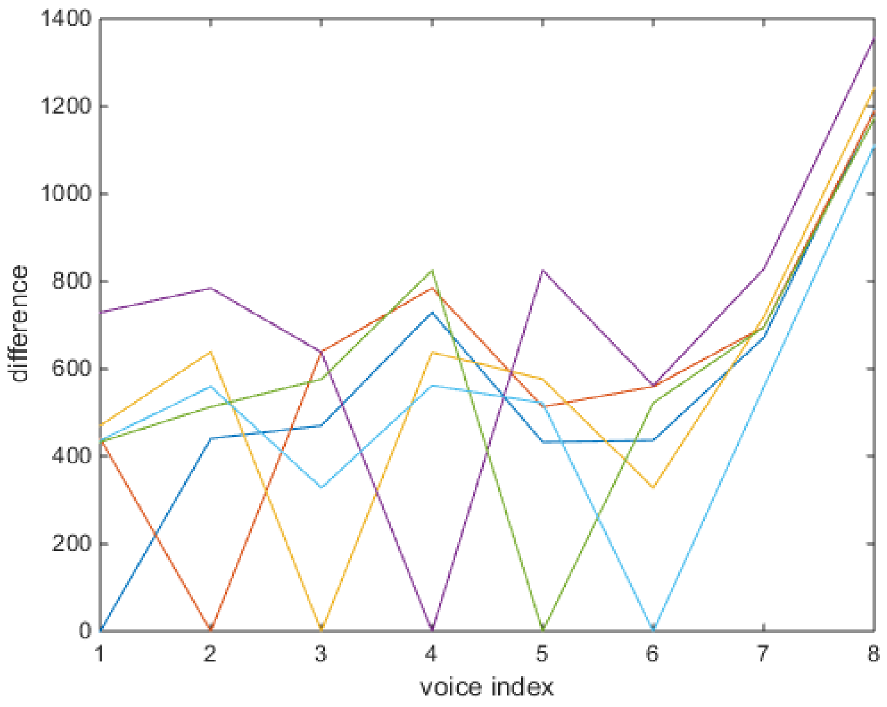


2.3 Matching Phase

In the matching phase the blocks are divided into 10 parts, where each part consists of 10 blocks. After that the average of each part is calculated to produce 15 coefficients for each part. When the 10 by 15 coefficients have been produced the actual matching begins. The coefficients are compared to three databases that each store the coefficients of a word. If the coefficients match a database, the spoken word will be detected as that of the word in the database. Feedback from the system is communicated through leds. If led 1 lights up the first word is detected, if led 2 lights up the second word is detected and if both leds light up the third word is detected.

2.4 Recording Phase

In order to make the matching work, there needs to be a recording phase where the coefficients for the words of our choosing is stored to then be used for matching. For this project the words ”you”, ”chair” and ”Amend” were the words that we eventually selected to distinguish between. If the system is set to record it will enter recording phase and print out the 10x15 coefficients that can then be put into the database.



3 Matlab

Before transferring to DSP, We tried the algorithm on Matlab to check the result first. We recorded 8 voice files in a sample rate of 16000, 6 of them are 'hello', one is 'table', and another one is 'chair'. These data are put into a prefilter at first to filter out noise and emphasis high frequency part. Then, we calculate the autocorrelation of them, and apply the correlation result on levison algorithm, and use the coefficients from levinson as the key part for recognition. Then we calculate the mean value for these coefficients, and calculated the difference between them, which is a sum of the absolute value of the difference between each element in all the voice coefficients. The result is shown below. In that figure,x axis is voice index, y axis is difference. index 1-6 are 6 'hello's, 7 is 'table' and 8 is 'chair'. It is obvious that the difference between different 'hello's are always smaller than the differences between 'hello' and 'table' and 'chair'. By doing so, we can judge which voices are the same one, and which are different from them. That means our method is correct in speech recognition.

4 Wiener Filter

The incoming signal is often corrupted by an ambient noise, such as background chatter, shuffles, steps and other undesirable sounds. In order to overcome this issue, we have used the Wiener filter. The Wiener filter is a statistical filter which compares the received signal with a desired noiseless signal. The output of such filtering should have a minimum mean square error.

5 Levinson Algorithm

There are two algorithms used in this project: Levinson and Schur algorithms. Both of these algorithms use feature coefficients from a stationary signal. In order to analyze the shape of incoming (stationary) signal of a short segment of speech with a limited number of parameters for computationally efficient program, Levinson - Durbin algorithm is often used. By definition, Levinson - Durbin algorithm is a recursive algorithm that utilizes a Toeplitz symmetry property of autocorrelation matrix, a property of a matrix in which each descending diagonal from left to right is constant. This algorithm helps us to determine the coefficients of the filter that predicts the next value in a sequence from current and the previous inputs as well as reflection coefficients.

6 Schur Algorithm

Compared to the Levinson algorithm, the Schur algorithms only uses the reflection coefficients to extract feature vectors through matrix multiplication.In Matlab, we can use function 'schurrc' to apply schur function, or use 'tf2latc' function to transfer Levinson redults into schur results. Schur is considerably faster than Levinson, but much more difficult to implement on C code. In the C code we use the coefficients in Levinson algorithm as the Schur coeffients instead.

7 Conclusion

The goal of the project was pretty much achieved; we managed to distinguish more than two words of the basic requirements. These words were to able match for different speakers, our speech recognition program can distinguish between three words for multiple speakers with satisfying results. we got incorrect matches sometimes which we assume may be because of the length of the word or noise or through by choosing incorrect threshold. Due to length of different words it was difficult for us to match the threshold for all the words which made the task hard to implement. Definitely there will be some way to make this work perfectly which requires good filtering techniques which is beyond this course.

References

[1] Empty Loop: Linear Prediction and Levinson-Durbin Algorithm , `http://www.emptyloop.com/technotes`

[2] LPC:Implementation of Linear Predictive Coding (LPC) of Speech,
`http://www.seas.ucla.edu/~ingrid/ee213a/speech/vlad_present.pdf`

[3] MatLab:Developing an Isolated Word Recognition System in MATLAB
`https://se.mathworks.com/company/newsletters/articles/developing-an-isolated-word-rec`

[4] MatLab :Speech recognition using MFCC and LPC
`https://se.mathworks.com/matlabcentral/fileexchange/36398-speech-recognition-using-mf`