Speech Recognition using a Dynamic Time Wrapping approach

G.S. Drenthen, *Member, IEEE*

***Abstract*—This report describes the development of a Dynamic Time Wrapping based speech recognition module. The recognition module roughly consists of four separate modules; preprocessing, feature extraction, clustering and classification. The preprocessing module prepares the speech signals for further analysis by removing pauses and increasing the signal-to-noise ratio. After preprocessing the features of the signal will be extracted using a Mel-frequency cepstrum approach, these features are then clustered using a k-means algorithm to produce relevant centroids. In the last module, classification, the Dynamic Time Wrapping method is used to recognize the speech signal. The recognizer's performance ranges from 35% to 100% heavily depending on the algorithm used and the training dataset. The module is developed in MATLAB and is easy to use due to the GUI (Graphical User Interface).**

***Index Terms*—Classification, clustering, dynamic time wrapping, feature extraction, mel-frequency cepstrum, speech recognizer.**

I. Introduction

Speech recognition has become more and more applied in the recent years. Phones, tablets, navigation systems, gaming platforms and cars are a few examples of devices which can include speech recognition. Since this is a hands free way to communicate with a device it is not only convenient, but it can also contribute to the safety of its user. For example, while driving a car, switching the radio or the navigator's destination will distract the driver. However, when a speech recognition module is implemented to deal with these actions the driver barely loses focus.

Besides the fact that speech recognition is hands free, it is also often a faster and easier way of communication with a device. For example, when calling a friend with your phone pronouncing his name is a lot faster than looking for his contact information or typing his number. When using speech recognition for the applications mentioned above it is very important that the recognizer is reliable; when the program fails to recognize the speech correctly too often the user will lose interest in the speech recognition feature. Current popular speech recognizers are for example Apple's SIRI for the iPhones and Google's Voice Search for the Google Chrome browser.

The goal of this research is to gain knowledge on speech recognition and to apply this knowledge in order to build a reliable isolated word[1] speech recognizer. This isolated word recognizer will be developed to serve as a remote controller for a device (e.g. a robot) using single worded commands like start and stop. During this project three scripts of the unofficial MATLAB toolbox VOICEBOX [1] were used; $melcepst.m$, $kmeans.m$ and $disteusq.m$ to build the speech recognizer.
To develop a speech recognizer for this purpose some unique design decisions had to be made. Since the recognizer only has to recognize a given amount of isolated words, training on these specific words will be the most effective approach. Besides that, the recognizer should be able to operate in real time; the calculation speed of the program shouldn't be too long. Keeping those requirements in mind, the following decisions were made during the development process, which led to the final result;

- The implemented preprocessing algorithm has been adjusted to the specific hardware and situation of the developer.
- Only the first, and most important, twelve MFCC, were used for recognizing the speech, to lower the calculation speed of the program.
- A weight vector has been applied to emphasize certain features (e.g. boost the important features of specific speech and lower the features containing a low frequency noise).
- Vector Quantization was applied (k-means) to reduce the amount of feature vectors and thus the calculation speed.
- Two different classification algorithms were implemented to perform the recognition.
- The recognizer is has been built with a GUI (Graphical User Interface), making it easier to recognize speech and create a training set consisting of your own speech.

A graphical overview of the speech recognizer is displayed in appendix A, and a guide on how to use the model is presented in appendix B.

The outline of this report is as follows;
Chapter 2: Preprocessing
A crucial step to building a proper functioning speech recognizer is preprocessing the speech signals. Without decent

[1] An isolated word speech recognizer is only capable of recognizing single, isolated, words.

preprocessed speech the recognizer is doomed to fail. Noise reduction and voice detection are the main topics discussed in this chapter.

Chapter 3: Feature Extraction

To recognize speech certain features must be compared. For the extraction of these features the Mel-frequency cepstrum algorithm is used. The extraction using this algorithm is discussed in this chapter.

Chapter 4: Vector Quantization

The amount of feature vectors increases drastically when using a larger speech dataset. Comparing all these vectors is too time-consuming, so the dataset has to be compressed. This chapter describes this compressing using a k-means clustering method.

Chapter 5: Classification

After building a dataset consisting of a reduced number of feature vectors, these vectors can be compared to the feature vectors of the input speech using the Dynamic Time Wrapping algorithm.

Chapter 6: Results

The results will be discussed in this chapter, where two situations are distinguished; All the speech is recorded by the same speaker and the training set is recorded by four different speakers while the input speech is recorded by a fifth.

Chapter 7: Conclusions

In this final chapter the conclusions are presented. Furthermore the recommendations on further use of the module are presented in this chapter.

II. Preprocessing

In this chapter the preprocessing of speech signals will be discussed. Preprocessing is a crucial part of the speech recognition module, it involves improving the signal-to-noise ratio and detection of voice. After preprocessing a raw speech signal, there should remain a signal with a fair signal-to-noise ratio and without any pauses. After preprocessing the speech features can be extracted. If the preprocessing is performed incorrect or insufficient the speech features will not correspond to the speech and the recognition will most likely fail.

A. *Pre-emphasis filtering*

The first step performed to preprocess a raw speech signal is applying a pre-emphasis filter. This filter will improve the signal-to-noise ratio by boosting the higher frequencies of the signal with respect to the lower frequencies [2], removing the adverse effects caused by recording the speech. A typical pre-emphasis filter is the high pass filter shown below:

$$H(z) = 1 - az^{-1} \tag{2.1}$$

Typical values for a are from 0.9 to 1.0, to find the best possible a different values have been used, and $a = 0.98$ proved to provide the best results. Giving the following pre-emphasis filter:

$$H(z) = 1 - 0.98z^{-1} \tag{2.2}$$

The magnitude and unwrapped phase of this filter are shown below, in Figure 2.1.

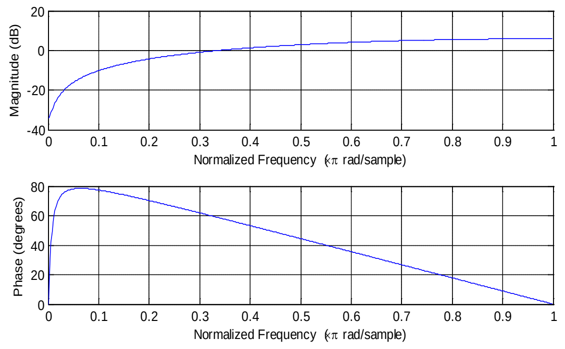

Fig. 2.1: Visual representation of the pre-emphasis filter

Applying this filter on a raw speech signal will provide a 'smoother' speech signal, listening to both signals one can clearly observe that the filtered signal includes less noise. In Figures 2.2 and 2.3 a raw speech signal and its filtered version are shown in the time domain.

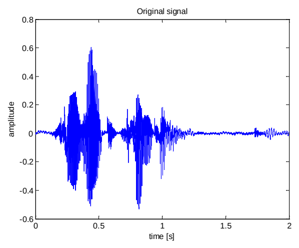

Fig. 2.2: Raw speech signal

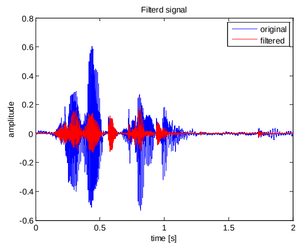

Fig. 2.3: Filtered speech signal

B. *Voice activation detection*

The next step in the preprocessing is to detect speech, and separate it from pauses. After distinguishing the speech from the pauses, a new signal without these pauses can be constructed. A speech signal roughly consist of three parts; voiced/unvoiced/silence where the voiced part is desired, while the unvoiced (noisy) and silence part must be removed. Unlike unvoiced and silence parts, the voiced part can be considered periodic over short-time periods due to the periodic vibration of vocal cords [3]. Dividing the speech into M blocks of length L ($20\ ms$) allows the speech to be stationary and periodic in each block (Figure 2.4).

The most important features for determining the voiced part are the short-term energy and the zero-crossing rate [4]. Voiced sound consists of a large amount of energy compared to unvoiced sound, and silence has no energy. Thus the short-term energy will increase when speech is present and is a useful tool to detect speech.

$$STE = \sum_{n=m-L+1}^{M} s(n)^2 \tag{2.3}$$

Since the only difference between the short-term energy and the short-term power is a scaling factor of $1/L$ the short-term power will also increase when speech is present.

$$STP = \frac{1}{L} \sum_{n=m-L+1}^{M} s(n)^2 \tag{2.4}$$

The zero-crossing rate is the rate at which the speech signal crosses the $x = 0$ axis. The short-term zero-crossing rate can be calculated for every block M, and this rate appears to be larger during the unvoiced part.

$$ZCR = \frac{1}{L} \sum_{n=m-L+1}^{M} \left| \frac{sgn(s(n)) - sgn(s(n-1))}{2} \right| \tag{2.5}$$

Combining both the short-term power and the short-term zero-crossing rate will provide the following formula, which is calculated for every block M (C is a scaling constant to prevent small values in W, typical value for $C = 1000$) [4]:

$$W = STP(1 - ZCR)C \tag{2.6}$$

The next step is to calculate a threshold to distinguish the voiced part. This threshold is calculated using the mean and variance of the first 10 blocks, assuming there is no speech present in these blocks [4].

$$t = \mu_{10} + \alpha\delta_{10} \tag{2.7}$$

The constant α was to be fine-tuned to properly calculate the threshold. This constant is not defined and may vary using different hardware (e.g. microphone). The value presented below proved to be able to make a decent threshold.

$$\alpha = 0.3\delta_{10}^{-0.88} \tag{2.8}$$

Comparing the threshold to the W function will provide the VAD function, also shown in Figure 2.5:

$$VAD = \begin{cases} 1\ for\ t \leq W \\ 0\ for\ t > W \end{cases} \tag{2.9}$$

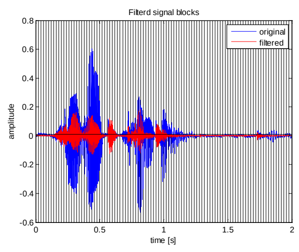

Fig. 2.4: The M blocks and threshold

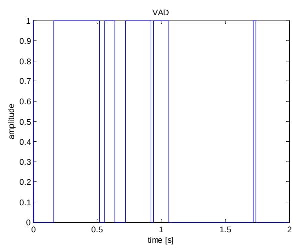

Fig. 2.5: The VAD function

C. Results

The above equations are implemented in MATLAB, after applying this routine on the raw speech signal (Figure 2.6) it is successfully transformed to a new speech signal without any pauses and with a fair signal-to-noise ratio (Figure 2.7).

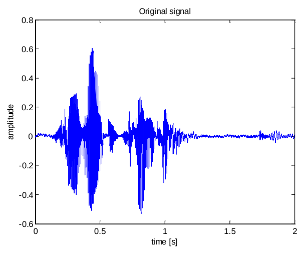

Figure 2.6: The raw speech signal

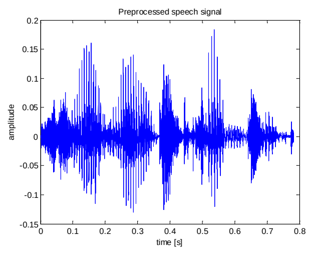

Figure 2.7: The preprocessed speech signal

III. Feature extraction

After preprocessing the raw speech signals the speech features must be extracted to allow further analysis. To gain the relevant features of the speech the Mel-frequency cepstrum algorithm [5] [6] is used.

A. Mel-frequency cepstrum

The VOICEBOX MATLAB routine $melcepst.m$ is used to calculate the Mel-frequency cepstrum coefficients (MFCC's). The MFCC's are calculated using the following steps:

1. The preprocessed speech is divided in overlapping frames.
2. A Hamming window to avoid spectral leakage is used.
3. A Discrete Fourier Transformation (DFT) is performed on the windowed signal.
4. Perform a Mel bank filtering, changing the scale of frequency from linear to mel scale.
5. Take the logarithm of the mel spectrum.
6. Take the Discrete Cosine Transformation (DCT) of the logarithm.

The results of the DCT are the MFCC's. The first coefficient represents the log energy followed by the delta coefficients and the delta-delta coefficients, these features are capable of distinguishing speech signals. The fourth coefficient is the 0'th order cepstral coefficient.

In Figure 3.8 the MFCC's of a speech signal are shown. This figure shows that not all of the coefficients hold relevant information, after the first twelve coefficients the MFCC's possess almost no new information. Therefore, to improve the calculation speed of the program only the first twelve MFCC's are calculated, as shown in Figure 3.9.

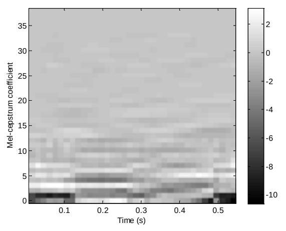

Figure 3.8: All the MFCC's

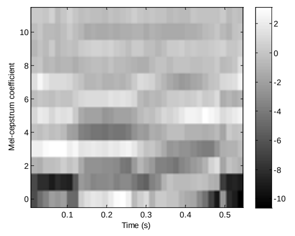

Figure 3.9: The twelve first MFCC's

After extracting the speech features a weighting vector can be defined to emphasize certain features which maybe more important than others. This allows the speech recognizer to be fine-tuned for given tasks. The, trivial, weight vector shown in equation 3.1 is assumed, but might be altered to provide better results.

$$\overline{w} = [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1] \tag{3.1}$$

B. Results

After calculating the coefficients with the Mel-frequency cepstrum approach, a set of feature vectors can be constructed. In Figure 3.10 a set of feature vectors is displayed, this set was created with a total of fifteen speech signals and with the weight vector shown in (3.1).

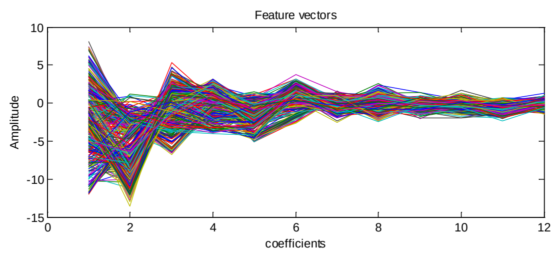

Figure 3.10: Complete set of feature vectors

IV. Vector Quantization

When a large data set is provided it is very time consuming to considering all the feature vectors. In order to lessen the amount of feature vectors, and thus lower the calculation time of the program, a form of vector quantization is performed. In this project the k-means clustering approach is used.

A. *K-means clustering*

The k-means clustering method is a good and fast unsupervised learning algorithm. The algorithm can, using only inputs, automatically discover representations and structure allowing clustering [7]. The VOICEBOX routine $kmeans.m$ is used during the clustering step.

The routine starts with choosing k random means and associating every data point to the closest mean. The centroids of these k clusters become the new means for the next calculation. When the means equal the centroids the algorithm has finished, and k centroids remain.

B. *Results*

After applying the k-means algorithm on all the feature vectors, only k centroids per word remain. In Figure 4.11 the whole set of feature vectors calculated from fifteen speech signals (three different words, each spoken five times) is shown. After applying the k-means algorithm with k set to 5 a dataset of only fifteen vectors remain, this dataset is shown in Figure 4.12.

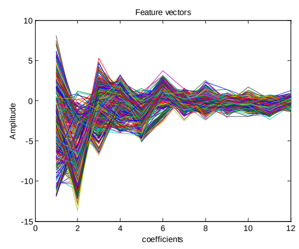

Fig. 4.11: Feature vectors

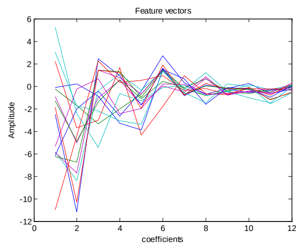

Fig. 4.12: Clustered feature vectors

V. Classification

The next, and last step, is to recognize a given speech signal using the created feature vector dataset. There are a lot of methods to do this; recently the most common one being a classification using Hidden Markov Models. This method has a drawback, the difficulty to implement it in MATLAB. Attempts to implement this approach failed, so a different classification method had to be found. This other method is Dynamic Time Wrapping [8], which was used in speech recognizers before the Hidden Markov Models approach became popular. A simple Euclidean distance calculation is also performed to classify the speech.

A. *Euclidean distance*

Using the VOICEBOX MATLAB routine $disteusq.m$ the distance between two matrices with different size can be calculated. Define two matrices with the same number of columns but different number of rows X and Y (5.1). Using the $disteusq.m$ function the distance matrix Z (5.2) is computed. Each of this matrix now contains the distance between one row of X and the whole matrix Y.

$$X = \begin{pmatrix} \overline{x}_1 \\ \overline{x}_2 \end{pmatrix} \quad Y = \begin{pmatrix} \overline{y}_1 \\ \overline{y}_2 \\ \overline{y}_3 \end{pmatrix} \tag{5.1}$$

$$Z = \begin{pmatrix} dist(\overline{y}_1, \overline{x}_1) & dist(\overline{y}_2, \overline{x}_1) & dist(\overline{y}_3, \overline{x}_1) \\ dist(\overline{y}_1, \overline{x}_2) & dist(\overline{y}_2, \overline{x}_2) & dist(\overline{y}_3, \overline{x}_2) \end{pmatrix} \tag{5.2}$$

Finding the smallest values in each column of the distance matrix Z will leave a row vector representing the smallest distance from $\overline{y}_1$ to any vector of X, $\overline{y}_2$ to any vector of X and $\overline{y}_3$ to any vector of X. Assuming the k rows of X contain the k centroids of a single word and the rows of Y contain the feature vectors of the input speech a distance can be defined (5.3) (5.4). Calculating this distance for every word in the dataset, and selecting the smallest distance will give the best matching word.

$$dist\_matrix(n) = disteusq(dataset(n), features\_speech) \tag{5.3}$$

$$result(n) = sum(\min(dist\_matrix(n)))/size(dist\_matrix, 2) \tag{5.4}$$

B. *Dynamic time wrapping*

A similar classification method is Dynamic Time Wrapping. This is a method which also allows comparing vector sequences of different lengths. Consider the same two vector sequences, X and Y (5.1), in the grid displayed in Figure 5.13.

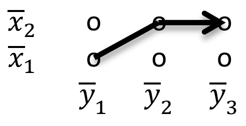

Fig. 5.13: Grid with optimal path

The DTW algorithm is initiated with start value $DTW(0{,}0) = 0$, and using the equation below (5.5) a value is assigned to all the grid points.

$$DTW(n,m) = dist(\overline{x}_n, \overline{y}_m) + min \begin{cases} DTW(n, m-1) \\ DTW(n-1, m-1) \\ DTW(n-1, m) \end{cases} \tag{5.5}$$

From each point in the grid an optimal path to the next point can be assigned, ending with an optimal path through the whole grid.

The end value of the algorithm $DTW(N, M)$ is calculated for all the feature vectors. The minimum of these values is most likely the spoken word.

C. *Results*

Even though the two methods mentioned above show similarities they don't always provide the same results. This is examined in more detail in the next chapter.

VI. Results

To determine the functioning of the recognizer it was tested thoroughly using different types of speech. During the testing several problems occurred using speech recorded on different systems with different hardware. The preprocessing algorithm proved to be unable to adjust to these different environments.
Due to these problems this chapter is split into two sections, the first section discussing the results using one speaker for both the dataset as the input speech. While the second section presents the results using four different speakers to create the dataset and a fifth speaker to provide the input speech.

A. Single speaker

The recognizer was tested using a dataset consisting of five words; each word recorded five times (total of twenty-five words). The words are shown below;

$$words = \{'aan', 'gerald', 'start', 'stop', 'uit'\} \quad (6.1)$$

The weight vector from equation (3.1) and a k-means clustering with 5 centroids were used, and each word was tested eight times. Using the Euclidean distance algorithm this provided a success rate of 85% (six out of forty words were not recognized), and using the DTW algorithm the success rate drastically decreased to below 40%.

Adjusting the weight vector (6.2) such that the log energy, the delta coefficients and the delta-delta coefficients are emphasized (these three features vary the most) compared to the other coefficients the success rate using the Euclidean distance increases to 100% (all forty words were recognized correct), and using the DTW algorithm the success rate is a mere 40%.

$$\overline{w} = [1.2 \quad 1.2 \quad 1.2 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1] \quad (6.2)$$

B. Multiple speakers

The database for the multiple speakers case consists of a total of one hundred words; five words, each recorded five times per speaker. The same words (6.1) were used for this case. Among the speakers was one female.

The first test was performed using the same settings as the single speaker case (weight vector (3.1) and 5 centroids). The results were, as expected, worse compared to the single speaker case with a success rate of 48% (nineteen out of forty words were recognized) using Euclidean distance and 45% using DTW.

Adjusting the weight vector as in the previous section provides a better performance, the success rate increased to 55% (twenty-two out of forty words were recognized) using Euclidean distance and 50% using DTW. Applying a k-means with 8 centroids instead of 5 further improves the success rate to 65% (twenty-six out of forty words were recognized) using Euclidean distance and 60% using DTW.

It is remarkable that the word $'aan'$ was only recognized once during all tests with multiple speakers. After examining the results more thoroughly, some of the speech signals were not preprocessed properly. The pauses of six signals (all of these signals recorded by the female speaker) were not removed. Removing those corrupt speech signals from the database unfortunately did not significantly affect the results.

C. Overview of the results

In the two tables below the results of the speech recognizer are displayed, where k is the number of centroids used in the k-means algorithm and w is the weight vector from either equation (3.1) or (6.2).

Single speaker case

Method	*Euclidean distance*		*DTW*	
Settings	$k = 5$ $w = (3.1)$	$k = 5$ $w = (6.2)$	$k = 5$ $w = (3.1)$	$k = 5$ $w = (6.2)$
Success rate	85%	100%	35%	40%

Multiple speakers case

Method	*Euclidean distance*		
Settings	$k = 5\ w = (3.1)$	$k = 5\ w = (6.2)$	$k = 8\ w = (6.2)$
Success rate	48%	55%	65%

Method	*DTW*		
Settings	$k = 5\ w = (3.1)$	$k = 5\ w = (6.2)$	$k = 8\ w = (6.2)$
Success rate	45%	50%	60%

VII. Conclusion

The speech recognizer with the Euclidean distance approach proved to be reliable for the single speaker case, with a success rate from 85% up to 100% after adjusting the weight vector. Since the recognizer is fine-tuned for a single speaker, these results are not surprising. However, the DTW algorithm produced poor results, with a success rate of no higher than 40%.

The performance drops dramatically when the database is constructed using different speakers. Four speakers were used in this project, three male and one female, and the recognizer proved to be unable to handle the higher frequencies of the female voice. Moreover the recognizer is also adjusted to specific recording hardware, while the four speakers each used their own computers and microphones. The success rate using different speakers and the Euclidean distance approach varied from 48% to 65% after increasing the k-means centroids and adjusting the weight vector.

The DTW approach performed better for the second, multiple speakers, case compared to the single speaker (45%~60%). However it never exceeds the results of the Euclidean distance method. It can be concluded that the DTW algorithm performs better with significant larger datasets.

Looking merely to the results of both the DTW and Euclidean distance approach two possible conclusions can be made; The Euclidean distance is a far better approach compared to the DTW, or the DTW algorithm was somehow implemented incorrectly.

A. Recommendations

When improving or expanding the speech recognizer the following suggestions might be helpful:

- Providing a way to change some variables (e.g. the centroids in the k-means algorithm, or the weight vector) using the GUI makes it easier to adjust the program for specific situations.
- Using an adaptive approach for the preprocessing algorithm to further reduce the noise, and to be able

to set a proper threshold regardless of the input speech (male/female and different hardware).

- Implementing a Hidden Markov Model approach instead of the Dynamic Time Wrapping classification would most likely improve the results of the speech recognizer.
- Programming the speech recognizer in another environment (e.g. C code) will not only allow a standalone executable, but will also improve the calculation speed of the recognizer.
- Recording a large database, with a high amount of different speakers will assure a better performance of the speech recognizer.

References

[1] M. BROOKS, "Imperial College London Department of Electrical and Electronic Engineering," [Online]. Available: http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html. [Accessed 21 July 2012].

[2] H. BEIGI, Fundamentals of Speaker Recognition, New York: Springer, 2011.

[3] L. RABINER and B.-H. JUANG, Fundamentals of speech recognition, Englewood Cliffs: PTR Prentice Hall, 1993.

[4] M. E. M. NILSSON, "Speech Recognition Using Hidden Markov Model," Blekinge Institute of Technology, Ronneby, 2002.

[5] M. OH and H.-M. PARK, "Preprocessing of Independant Vector Analysis Using Feed-Forward Network for Robust Speech Recognition," Berlin, 2011.

[6] S. RAVINDRAN, C. DEMIROGULU and D. ANDERSON, "Speech recognition using filter-bank features," Atlanta, Nov. 2003.

[7] B. d. VRIES, "Technical University Eindhoven Adaptive Information Processing," [Online]. Available: http://www.sps.ele.tue.nl/members/B.Vries/teaching/5mb20/index.html. [Accessed 21 July 2012].

[8] P. N. M. LAMA, "Speech Recognition with Dynamc Time Wrapping using MATLAB," SPRING, 2010.

[9] J. MCAULEY, J. MING, D. STEWART and P. HANNA, "Subband Correlation and Robust Speech Recognition," Washington, Sept. 2005.