Registration number 100271864

2022

A Wearable Embedded Device for Learning Phonemes Through Vibrotactile Sensations on the Skin

Supervised by Edwin Ren

University of East Anglia
Faculty of Science
School of Computing Sciences

Abstract

Speech is a unique thing we as humans experience, it is an everyday occurrence that many of us do not even think about. However, too many of us have never, or no longer, experience this human experience though a loss of hearing or being born without it.
This project aims to provide a solution to those who cannot experience the sounds and voices of those around them. There are many devices that allow someone to understand what someone else is being said but these are often in the form of text-to-speech solutions which provide much in the way of understanding but nothing in the way of experiencing, providing no solution to the disconnect between two people who are communicating.
A solution to this project was explored in the form of a spoken phoneme to vibro-tactile sensation device which would allow the user to feel vibration signals mapped to phonemes, allowing them to learn speech though these vibrations. In this way it is thought that not only words could be understand but accent could be discerned and experienced, providing a new level of connection between the user and those around them.

Contents

List of Figures

List of Tables

Contents

1 Acknowledgements

The main inspiration for this project came from the NeoSensory Perrotta et al. (2021) which aims to provide the experience of sound to those who cannot hear, this project aims to take this idea and apply it directly to speech whilst keeping with the idea of experience not just understanding.

The use of open-source data sets has allowed for more development in any area as developing at data-set is extremely time-consuming and creating one with multiple speakers with high recording quality and boundary definitions is an even more challenging time. The TCD-TIMIT Harte and Gillen (2015) and NTCD-TIMIT Abdelaziz et al. (2017) data-sets have been used extensively throughout and made this project viable.

2 Introduction

Many advancements in technology have allowed those with disabilities and impairments to overcome these or at the very minimal become more comfortable in their day-to-day lives. The invention of Braille allowed millions to read whilst the invention of sign language allowed millions to communicate with similar ease as others however, adoption of these as mediums have been slow and a lack of teaching in public schools have hindered their use. This project has aimed to look at the many advancements in Vibro-tactile devices and progress in speech recognition to produce a low-cost way for those with hearing impairments to hear the people around them if they do not know sign language.

In recent years speech recognition has moved toward two primary methods, namely Hidden Markov Models and Neural Networks. These methods have become extremely accurate but at a cost of computation time, deployment size, and space requirements; three priorities when working with micro-controllers. This project will aim to explore and discuss alternative methods which meet all three of these requirements with defined size, speed, and memory limitations which would allow the ASR to be deployed on numerous low-cost microcontrollers.

During this project, it was paramount that the model produced would be suitable for a low cost, small form factor device whilst producing a result before the McGurk effect's range. The McGurk effect, "multisensory illusion occurring with audio-visual speech" Tiippana (2014), has been defined as 180-250ms Munhall et al. (1996) and as such the ideal response time was below this however, if significant results could be achieved past this upper bound then this model would also be considered.

2.1 Aims

The aims of this project were split in two, the aims of the proposed device, and the methods and process proposed for allowing others to experiments and produce their own model for use with their own hardware.

The device:

- Must have:
  - A small form factor to be suitable for everyday use.
  - Hardware which is easy to develop and a project which is easily produced.
  - A simple way to load a produced model.
  - A minimal amount of tactor sequences to produce at least one set.
- Should have:
  - A suitable amount of PWM pins to allow for a variety of tactor sequence permutations.
  - Expandable memory and storage options to allow for more extensive models.
  - An easy method to change the model parameters and tactor sequences without changing code.
- Could have:
  - An implemented rechargeable battery - though this would be needed for actual use its process is not necessary to this project.
  - A designed and produced wearable wrist strap which contains the device and tactors.

- Wont have:
  - A user interface or application to interface with the device.
  - The ability to change the number of tactors without re-soldering.

The program:

- Must have:
  - Complete methods of feature extraction and classification
  - Testing methods to display the accuracy to the user.
  - Methods to output the model for use on the device.
- Should have:
  - Output for displaying confusion matrices and other metrics.
  - A complete set of parameters to alter the MFCC and classification methods.
  - Methods for replicating the data-set format used during the project.
- Could have:
  - A number of methods of classification to test different models before settling on one.
  - A simple desktop application for a better ease of use.
- Wont have:
  - Fully tested versions for every operating system.

2.2 Motivation

The motivation behind the project was to explore an alternative to the text-to-speech options which provide services for those hard of hearing and language translation. Such devices have provided significant benefits to those indeed especially as it is estimated that only around 151,000 thousand people in the United Kingdom use sign language. Unfortunately, these devices create a block between the user, they are reading from a screen instead of listening to the person, and their voice and accent are not portrayed.

An accent is a key part of language, it not only describes where someone is from but also how formal, or informal, they consider their relationship to their recipient

is. Much like spoken languages, sign language has local dialects and so this is not something only a characteristic of spoken speech and when this is removed some of the connection and information is lost from the words being spoken.

The decision to use phonemes instead of words for recognition was not only to produce a smaller set of data for training the recogniser but to allow this accent to be experienced through the device and allow a new connection between the user and those around them.

As there has not been a lot of development in this area available to the public it was hoped that a program could be developed for those with knowledge in transferring sensor information through the skin could use this project to easily produce a model for their onset of sequences, eliminating the time required to develop their own system. Additionally, if a set of sequences was to be released then someone could the projects proposed device build to make their device or someone versed in hardware development could produce a more real world device perhaps with a custom board and wearable build that could be reproduced by others.

3 Literature Review

3.1 Similar Devices

Many vibro-tactile devices have been proposed and designed in the realm of research however, none have made it to the public as use-able devices the only one similar to these which has is the NeoSensory Perrotta et al. (2021). The NeoSensory differs from the aims of this project in one key aspect is the focus on general sound rather than speech and so the idea and premise of the device has been expanded upon to focus on and incorporate phoneme recognition.

Some devices found have certain properties that limit its use in a real-life setting such as the micro-controller used such as the Arduino Uno used in Novich and Eagleman (2015) or the programming language used such as MATLAB as used in Barbacena et al. (2009). Both of these properties are perfectly fine in most situations however, the Uno is far too large to be used in this project and using MATLAB would restrict the ability to tailor the methods but the ease of use and reduced development times makes them both desirable when experimenting with methods and testing theories.

3.2 Automatic Speech Recogniser

There exists a plethora of speech recognition methods available including libraries like Sklearn for Python which provides a number of feature extraction and classification methods, self-contained programs like HTK which has seen extensive use in research and education, and tailored languages such as MATLAB which provides an easy to use environment for quick development and complete methods.

Unfortunately, none of these are designed for micro-controllers, there exists MicroPython - an implementation of Python designed for micro-controllers - however, not all libraries are available for use. MATLAB has processes to produce embedded code however, without the knowledge of the implementation if the methods produced did not work or fit on the desired board then they could do not be altered and subsequently used.

It was decided to use a method that has been seen less use in recent years due to the progress of neural networks and hidden Mark models, Dynamic Time Warping. Though this method has seen less use this option seemed more promising for a quicker development time and smaller footprint than other more recent methods and with speedups and optimisations such as Early Abandon DTW Junkui and Yuanzhen (2009), Piecewise Aggregate Approximation Chu et al. (2002), and FastDTW Salvador and Chan (2004) which decrease computation time, and methods such as One-against-All Weighted Zhang et al. (2014) which greatly reduces the amount of training data needed it was thought that an efficient and accurate model could be produced.

4 Description of Figures

Throughout this project, numerous tests and experiments were run all whilst continually updating and changing the process and implementation, as such the figures, charts, and graphs displayed will be accuracies based on the final chosen parameters as defined below with only the stated parameter changed.

- PAA (Piece-wise aggregation) is described subsection 8.1

Table 1: Optimal parameters used during final experiments

Window Size	Filter Banks	Coefficients	Overlap	NFFT	DTW Window	PAA*	K
128	16	14	50%	512	20%	2	7

5 Requirements

Throughout the development of this project a number of requirements have guided and constrained the resultant model; producing a recognition system which is suitable for a micro-controller limits the scope of implementation. Many of the methods explored have many pre-existing libraries though many of these are either too large be used by the chosen device whilst others were written in a language not suitable for the intended device and so it was decided to focus on a method that could be implemented in the given time frame as this would allow the methods to be altered as space or accuracy requirements changed.

5.1 Form Factor

In this project the device is intended to be worn around the user's wrist - in the same fashion as a watch or bracelet - and the nature of the device means it could remain there for the entire day and so it is important to produce a device with a small form factor such that it would be comfortable for the user.

5.2 Cost

This project aims to not only provide methods and a final device, but a template for others to improve and refine. In this way cost is important for two reasons, firstly, by making it cheaper for the end user, it provides better access to the device to more people and considering the purpose of this device it is believed that access should be available to those that would stand to benefit from it. Secondly, an additional focus of the project was to not only create methods and a final product but to provide a system and template for others to use and produce their own variation of the device. Those familiar with vibrotactile stimulation and their applications in information transfer may not be familiar with embedded system design or implementing the feature extraction

and classifying methods but with the use of the provided system and user guide it is believed that one could produce a model, build the device and produce a product that suits their proposed tactor design.

5.3 McGurk Effect

The McGurk effect, first described in McGurk and MacDonald (1976), takes the form of an auditory illusion, when a sound is heard by someone whilst they are viewing video of someone saying a different, but similar sound that a person will often hear a third sound which is again similar to the original two. In McGurk and MacDonald (1976) the experiment saw the syllable [ba] being dubbed over a video of someone saying the syllable [ga], the participants reported hearing the syllable [da].

This is a well-documented illusion for audio and video however, it is believed that when a person is 'hearing' through their skin using a vibrotactile device this effect will still occur, perhaps more so as many deaf individuals rely on lip-reading a lot of the time whereas most hearing people do not intentionally focus on lip movement. It was decided that any model and method produced would need to be below the range of this effect (180-250ms) and so the entire process of boundary detection, feature extraction, and classification should be done on the given micron-controller within this time.

6 Phonemes

Phonemes are units of sound which, when changed, distinguish different words in their respective language such that the words 'bat' /bæt/ and /kæt/ are phonetically the same except for one phoneme, the begging /b/ and /k/ - denoted here using the International Phonetic Alphabet (IPA) transcript. These two words are different in English through the change in only one unit of sound and as such are called minimal pairs. When a unit of sound has a minimal pair they are described as a phoneme; interchanging these would result in different meanings when speaking. There are, however, some exceptions to minimal pairs, consider the phonemes /ŋ/ as in 'song' (/sɒŋ/) and /h/ as in 'high' (/haɪ̯/), the only minimal pairs as provided by Mairano and Calabrò (2016) have at least one loan word, commonly a surname, in the pair. Additionally, English speakers would not interchange /ŋ/ and /h/ in words such as 'high' (/haɪ̯/) - pronouncing it as (/ŋaɪ̯/) - because they may not be understood and so these are generally considered

distinct phonemes in English.

Results returned by Mairano and Calabrò (2016) for minimal pairs of vowels and consonants will return very few results, if any, with many of these results again being loan words or obscure words most would not use therefore, vowels and consonants are typically considered so distinct that there is no need to define minimal pairs.

The decision to use phonemes was based on multiple factors, firstly, the number of phonemes in English is significantly smaller than the number of words with an extensive set like the one found in the TIMIT data-set defining 52 and a core set like the one found in the TCD-TIMIT defining 38, this greatly reduces number of reference sequences needed to train the model, size of the model required and the number of buzzers required to produce unique sequences for each class. Secondly, 'Differences in pronunciation, in accent and intonation of speech in general, create one of the most common problems of speech recognition' Kardava et al. (2016), and so any variation in speech could cause mis-classifications and major dialect differences can be so different that native speakers can find it difficult to fully understand. Finally, the use of phonemes as opposed to words comes with the benefit of allowing the user to learn and experience accents which to many can be a key part of their culture and speech, the aim of the project was not only to allow someone to understand speech but for them to experience it in a sensory manner, to have a connection to voices of others.

7 Data-sets

Training and testing any form of recognition or classification system requires some form of data as an input in this case speech recordings were necessary. It was initially thought producing the recordings myself would be necessary however, this could result in the resultant model being speaker dependant and although many tests were performed on a single speaker due to time constraints, it is shown that the model still performs relatively well with many different speakers.

Upon further reading, however, there appeared to be many available data-sets designed for speech recognition which could be used in this project. Using such a data set would allow testing on different speakers to see how robust the model is to properties such as accent and gender but more importantly this would facilitate much earlier and more thorough testing.

7.1 TCD-TIMIT

The choice was made to use the TCD-TIMIT Harte and Gillen (2015), and later the NTCD-TIMIT Abdelaziz et al. (2017), data-sets as they contain speech from multiple different speakers with various accents and with the NTCD-TIMIT data-set being based on the TCD-TIMIT it would allow comparable testing on noise robustness.

The data set was first split into training and testing with a roughly 70/30 split with only the volunteer data being in the main set as the lip speaker data would be used for quick testing of parameters. The split was achieved by considering the number of the folders which were named 1-59(M/F) with any adjustments in split needing to be made being done by hand. This is perhaps not the best method as the split may not include an equal number of male and female speakers however, during this process the male and female folders were also separated into sets for testing the effect of gender and so in this way any effect that a disproportionate split could have on accuracy may be seen in these separate tests.

7.2 NTCD-TIMIT

The NoisyTCD-TIMIT, or NTCD-TIMIT, uses the TCD-TIMIT as a base and applies various noisy audio onto the clean speech at SNR ratios ranging from -5dB to 20dB. As the final device would be intended for everyday use it was important to test how much everyday noise would affect not only the classification accuracy but also the boundary detection.

8 Feature Extraction

One of the most common methods for extracting features from audio and speech, in particular, is the Mel Frequency Cepstral Coefficient, or MFCC. The MFCC has been used alongside many different classifier methods, including the one used in this project Mohan and N. (2014), Dekel et al. (2004). The main principle of an MFCC is to extract information that closely resembles how we as humans hear and understand speech; as speech and hearing have evolved together and adjusted for each other then it stands to reason that a method attempting to imitate human hearing should provide useful information.

An MFCC, therefore, sees the raw time-domain be transformed into the frequency domain as humans recognise the frequencies used in speech, this frequency domain is further mapped to the human hearing by converting the standard Hz values to their Mel scale equivalents - the Mel scale, first described in Pedersen (1965) more closely resembles the human ear's response to frequency.

A Discrete Fourier Transform (DCT), as described here Winograd (1976), is a lossy data compression method often known for its use in the image compression format JPEG. The decision to use the DCT for JPEG was due to the minimal loss of important information and the speed at which is data is compressed and uncompressed.

8.1 Piecewise Aggregation Approximation

Before producing an MFCC a method of reducing the time domain signal's length is applied, this is done to reduce the total space requirement of each MFCC and reduce the time taken for each test. If a PAA parameter can be found that provides a significant reduction in size whilst retaining accuracy this will prove very useful due to the limitations of size and processing power of the micro-controller any reduction in size that requires minimal processing time will allow this saved space and time to be spent on other methods which may provide additional accuracy.

Piece-wise aggregation approximation (PAA) is a simple method which reduces the length by moving a window of size X over the raw data and averaging all within this window, producing one value.

$$S_x = (\sum_{n=1}^{x} S_n)/x \tag{1}$$

8.1.1 Experiments

Piece-wise aggregation was tested at different division amounts to see how the loss of data would affect the accuracy, an optimal parameter would produce some reduction in data size whilst having minimal effect on the accuracy. As the Fast Fourier Transform was used these decimations, or interpolations, would need to be in powers two and so only minimal testing could be done.

In subsection 8.1 it is shown that reducing the data by a factor of 2 had a noticeable, but not drastic, effect on the accuracy with the higher the decimation values having a

significant effect on the accuracy. This was expected as less data will result in small but important differences being lost.

In addition to decimation, one interpolation test was tried using cubic interpolation, this method was used as it was felt that a cubic function more closely represents the appearance of a sound wave. The result shows that although this would have increased the amount of data for comparison the method used may have been inadequate, or interpolating a sound wave produces erroneous or unsubstantial information which causes more confusion when classifying.

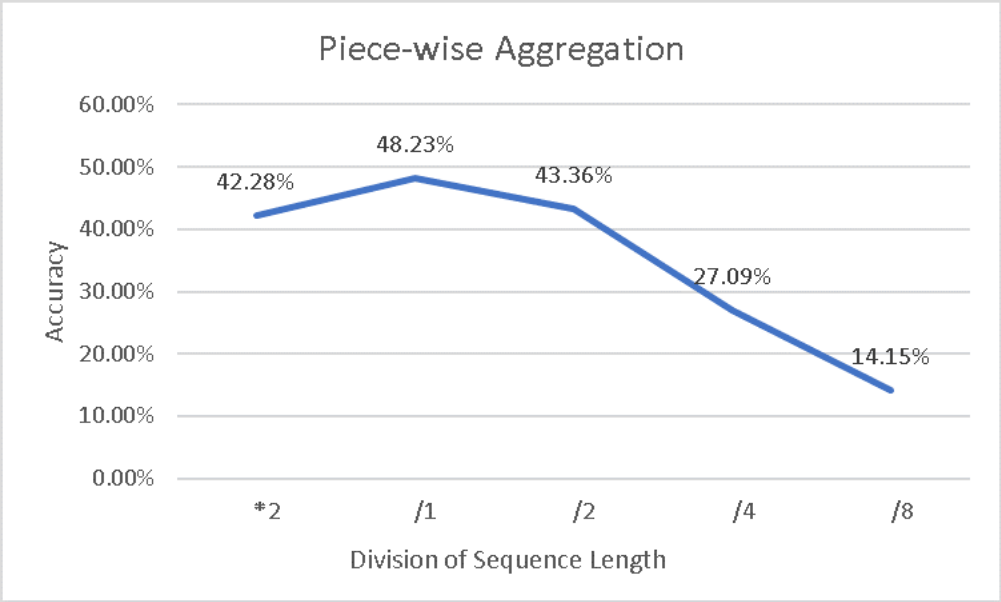

Figure 1: Piece-wise aggregation divisor used in MFCC creation

8.2 Windowing

The audio signal is taken in overlapping frames to which a window function is applied, in this case, a Hanning window. Multiple window types were tested in early and late testing, the results of the different windows can be found in Fig. 2 The reasoning behind each is the window function reduces spectral leakage of the Fourier Transform whilst the overlap ensures that no information is lost.

To find the number of frames per audio signal the below equation is used where Fc is the frame count, l is the length of the audio signal in samples, w is the window length

in samples, and d is the overlap division.

$$Fc = l - w/(w/d) \tag{2}$$

The below version has the parameters used in the final version of this project substituted in.

$$Fc = l - 128/(128/2) \tag{3}$$

It was found during testing that a window width of 64 and 128 samples gave no major improvement in accuracy with a window sizes 32 and 256 producing noticeably worse results. Ultimately 128 was chosen over 64 as the slight increase in performance was not preferable to a reduction in size for each MFCC created. Similarly, a division of 4 produced slightly better results however the increased time to test and size requirements did not make this option viable for this project.

The windowing function is given below where W is the window and S is the input signal.

$$X_i = S_i \times W_i \tag{4}$$

The equation which is applied to each value can be found is as follows

$$W_i = 0.5 \times (1 - cos(2\pi \times i/l)) \tag{5}$$

l is the length of the input signal in this case it is the same as the window with (128).

8.2.1 Experiments

The window width used partially determines the amount of data reduction achieved but also what type of data is represented in the resultant feature vector. It is shown that the window size affects phoneme types in different ways Kelly and Gobl (2011) and as such it was not only important to find a parameter that produced a high accuracy but to determine the one which did not favour one group too much.

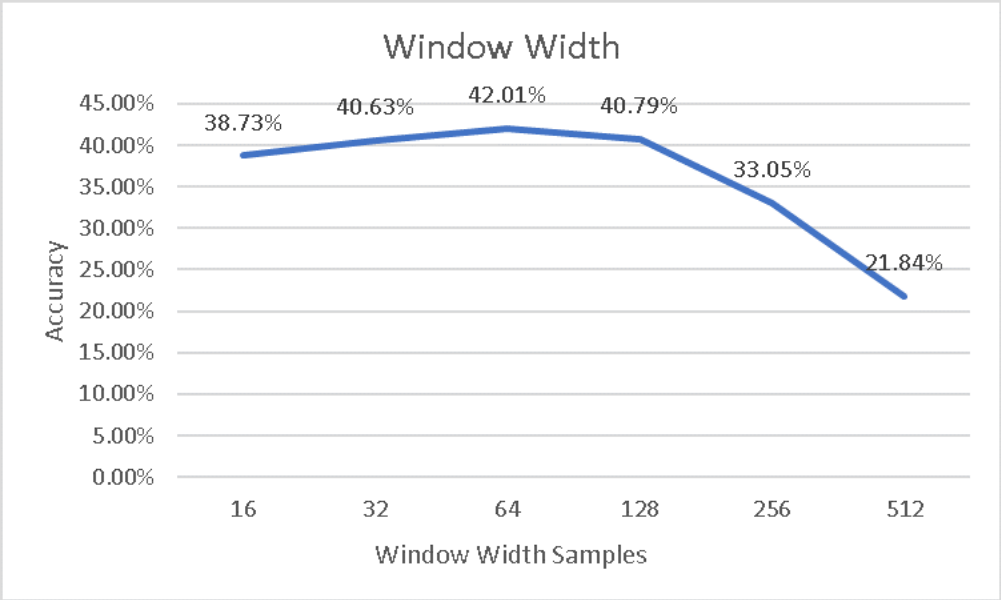

Figure 2: Window width used in MFCC creation

In, Fig. 2, it is shown that a window size that is too small or too large will have a negative effect on the accuracy, this is due to the amount of information being captured in each window. A window that is too small will capture too much change from frame to frame and therefore small differences found between different utterances of the same phoneme will be accentuated too much. Whereas a window that is too large will not capture enough change and any variation of the raw signal over time will become lost.

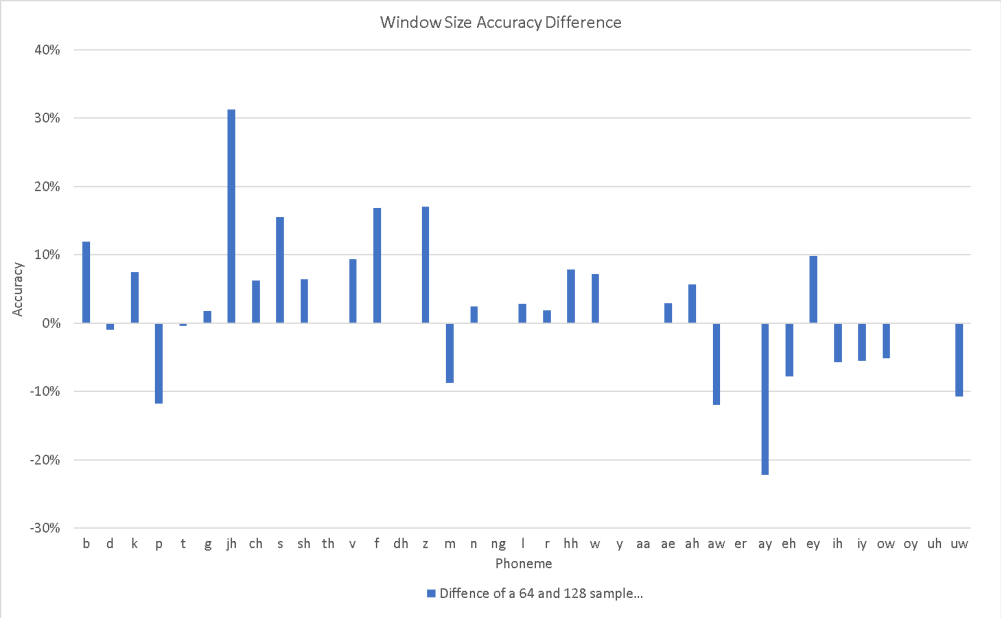

Figure 3: Voiced/unvoiced window preference

8.3 Fast Fourier Transform

After windowing the result is passed to a Fast Fourier Transform, or FFT, to transform the data into the frequency domain. The FFT is a variation on the DFT (Discrete Fourier Transform) which takes advantage of symmetry in the data to speed up processing time, however, this restricts the use of this function to powers of 2 but this is thought to be of little consequence due to the vast number of work that has used the FFT over the DFT [link papers]. Human hearing is focused on the frequency information rather than change over time and so this transforms the audio data into a more usable state.

The result of the FFT is a sequence that contains the dominant frequencies in a signal and has these values mirrored, as such half of the FFT result can be discarded as this would add no information to the resultant MFCC.

8.4 Filter Bank

Once the magnitude of the FFT is obtained a number of triangular filter banks are applied to the result. These filter banks are linearly space in the Mel frequency range and when applied will multiply the segment by values between 0 and 1 such that any value

at the peak will be multiplied by 1 those outside of the triangle will be multiplied by 0. Applying filter banks in this way serves to reduce the data and select more relevant, specific data to process in further steps. Once each filter bank has been computed the results are summed to create a 1-dimensional segment which can be passed to the log and DCT functions described in section subsection 8.5.

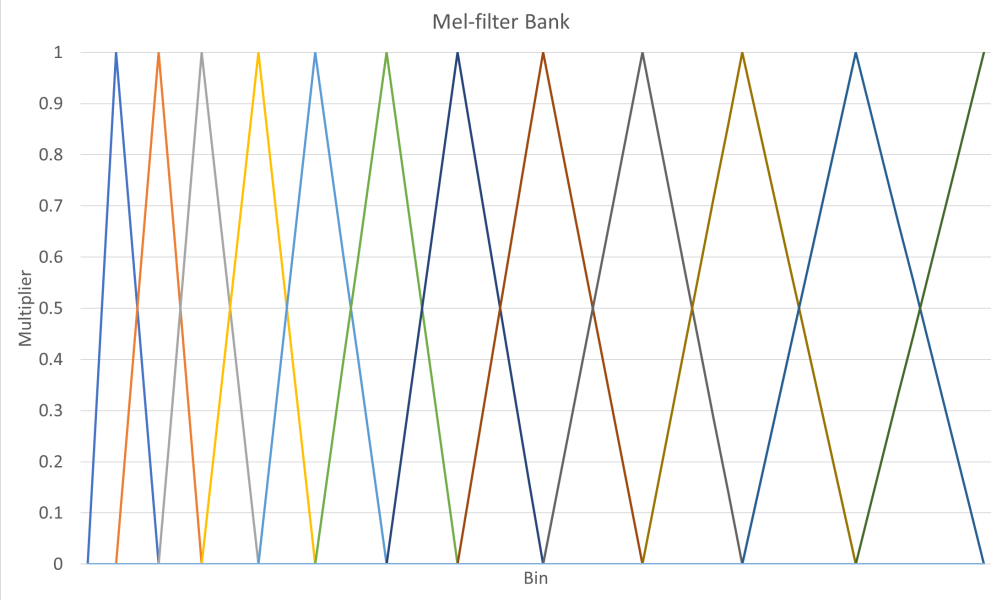

Figure 4: Mel-filter bank using an NFFT of 512

8.4.1 Experiments

A number of experiments were performed to determine an optimal number of filter banks and kept coefficients. As mentioned in subsection 8.4 the pitch information at the end of the coefficients does not typically provide useful information, this is most clearly seen in Fig. 5 where the accuracy decreases at 40 filters and above 26 kept coefficients.

		Banks			
		16	**24**	**32**	**40**
Coefficents	**8**	38.46%	38.14%	37.78%	38.25%
	14	40.61%	41.10%	40.72%	40.63%
	20		42.28%	41.04%	41.53%
	26			41.62%	42.03%
	32				41.76%
	36				41.49%

Figure 5: Accuracies for single speaker using varying filter banks and coefficients

The fewer filter banks and kept coefficients, the less time it will take to test and the less space required per reference MFCC and so it was necessary to balance accuracy with these two factors. This balance was decided through further experiments to be 16 filter banks and 14 coefficients which provided comparable accuracy to the best shown in 5 whilst keeping both mentioned requirements low.

8.5 Log & Discrete Cosine Transform

Once the filter bank coefficients have been computed a log is applied followed by a Discrete Fourier Transform (DCT), this transform brings the coefficients into the quefrency domain. After this transform two areas of the graph will be seen, the vocal tract signal and the pitch signal, this pitch signal will present itself as the trailing coefficients and as such, they will be truncated out as the pitch of the signal does not provide useful information in this case.

In this project, a DCT-2 was used although a DCT-1 has also seen use in MFCCs, through testing, however, it was found that the DCT-2 provided marginally better results.

$$D_i = \sum_{n=1}^{N} x_i \cos[\frac{\pi}{N}(n+0.5)i] \tag{6}$$

In the above equation N is the length of the summed Mel filter bank segment, with an iteration of n to N being summed for each value in the segment.

9 Boundary Detection

Many methods have been developed for determining the boundaries of phonemes including filters Ramteke and Koolagudi (2019), zero-crossing rate Manikandan et al. (2009), short-time energy Shete et al. (2014), and entropy with derivatives Salvi (2006). All methods mentioned of course have their benefits and situational uses however many do not fit the needs of this device. Common filters in use need a considerable length of audio before an accurate result can be obtained and therefore they are not appropriate for real-time applications as this amount of delay we be disorientating to the use. As previously discussed neural networks, though can be highly accurate, are not easily constrained to space and time requirements.

The method of boundary detection used for this device uses three main features, namely zero-crossing rate, short-time energy, and entropy. These methods only require one frame of lag in order to produce a result. The three methods mentioned produce a change in value frame by frame and will return a chosen frame when a value, or combination of values, go above a set threshold.

A non-overlapping rectangular window with a size of 16 samples was applied to the signal in order to find the change in values between them. Changing the window size requires finding new thresholds each time. This method of determining this does not account for guesses made within silence and the difference is calculated from the closest guess; some issues arise from this method namely comparing to the closest guess does apply much penalty to insertions however, comparing guesses in order can place a huge penalty for deletions, and discounting any guesses made within silence would be acceptable if the model was accurate at detecting silence as multiple guesses of silence is no different from one guess of silence, unfortunately, the model is not as accurate at detecting silence as first hoped.

Thresholds were found by recording all values in a selection of files then recording the values at each known boundary at each file and graphing each together in order to find a suitable threshold. Initial readings did not show a distinct set of thresholds to use as many of the values at boundaries were often lower than found at non-boundaries. After looking at the values found near boundaries as well it was found that more suitable thresholds could be found if the accepted accuracy of each guess was prioritised less than the number of boundaries found.

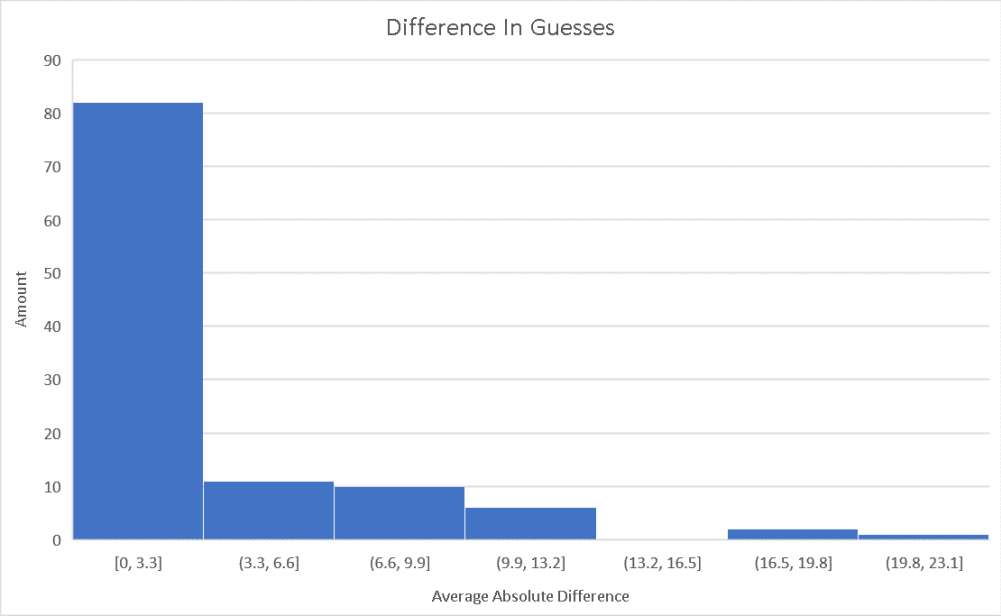

Figure 6: Number of boundaries found in file difference

The two presented graphs Fig. 6 and Fig. 7 show the accuracy of the boundary detection on its own. As mentioned the leading and trailing silence was removed during this testing as if the system finds multiple silences in only one then it produced the same output to the user; the model is not perfect at determining silence, however, after multiple experiments and tests these results produce the best results on its own and when combined with phoneme classification.

Overall both graphs show good competency in detecting the correct amount of phoneme boundaries and find them in a respectable range of the actual boundary however, there are some major outliers in both graphs. Some reference files produce a difference between 19.8 and 23.1 guess which shows the model has much room for improvement to reduce this. Finding threshold combinations began by recording and determining a baseline by hand and then using iterative testing to fine tune these findings, further testing could have been done on files which have these larger outliers to determine an additional case which can be used in conjuncture with the current, unfortunately, time restraints caused focus to be moved towards completing the hardware implementation and so this was not explored further but it is believed that with more testing, fine-tuning, and more specific cases the results could be improved significantly.

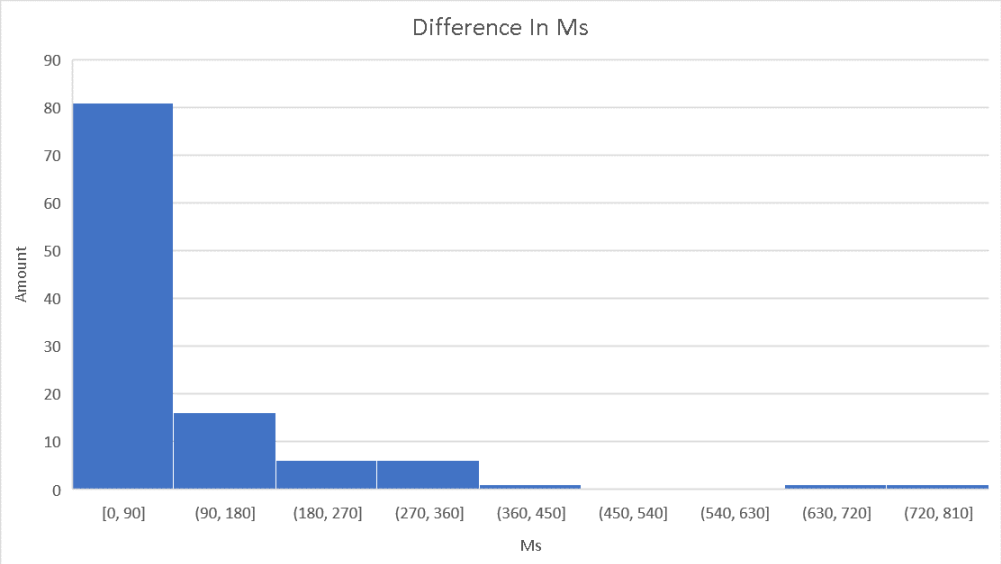

Figure 7: Difference of boundary position in milliseconds

9.1 Zero Crossing Rate

Zero crossing rate is a common feature for determining the phoneme voice type and can be used to find boundaries between them as voiced and unvoiced phonemes often follow each other and so a change in zero cross typically represents a change in phoneme. The graph found in Fig. 8 hows the minimum and maximum values of each phoneme, it is clearly shown that voiced phonemes such as vowels produce a low zero crossing rate whilst unvoiced phonemes produce a higher zero crossing rate.

It was thought that using these values could reduce the number of tests needed to be performed, by determining the test's zero crossing rate and comparing them to the maximum and minimum of the reference phonemes then it could conclude which phonemes it is likely to be. However, using this method provided $<1\%$ increase in accuracy whilst requiring additional testing of these features and so it was decided not to include this method. It was thought that outliers caused this lack of accuracy increase however, restricting the values using Z-scores did not show any significant improvement and so this pre-classification check was not implemented in the final device.

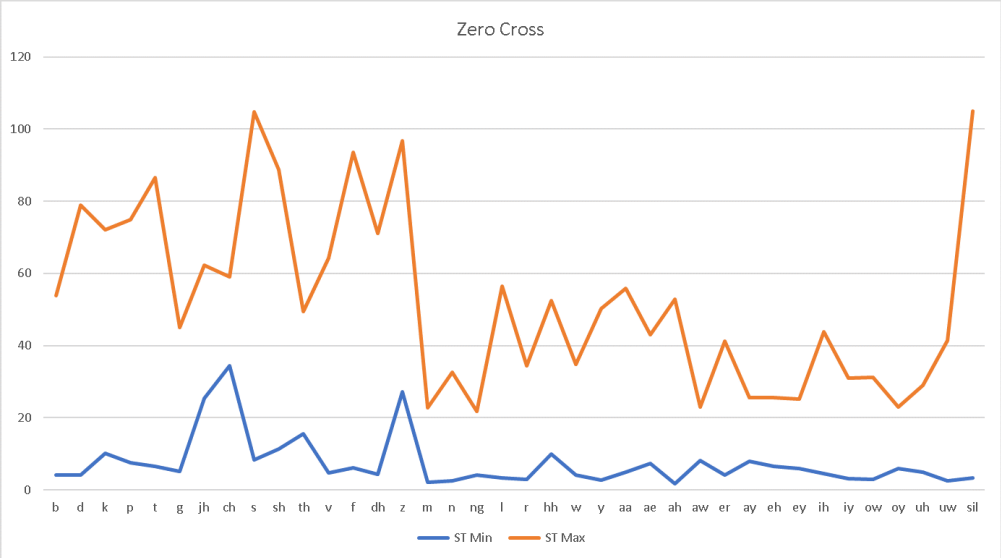

Figure 8: Zero cross minimum and maximum values

9.2 Short Time Energy

Short Time Energy is the energy of a segment of a given input signal and is commonly used to determine whether a signal is voiced or unvoiced. Computing the short time energy is simple but effective at classifying voiceness. In this project the window size used for MFCC creation is the same for the short time energy when classifying the voice with KNN as this creates equal-length sequences of feature vectors which is much easier to work with however, a much smaller window size of 32 samples is used for boundary detection as this proved to be sufficiently accurate with the maximum offset of a boundary being 32 samples.

$$S = (\sum_{n=1}^{N} x^2) \tag{7}$$

In the above equation x is a sample from the given signal segment and N is the length of the segment.

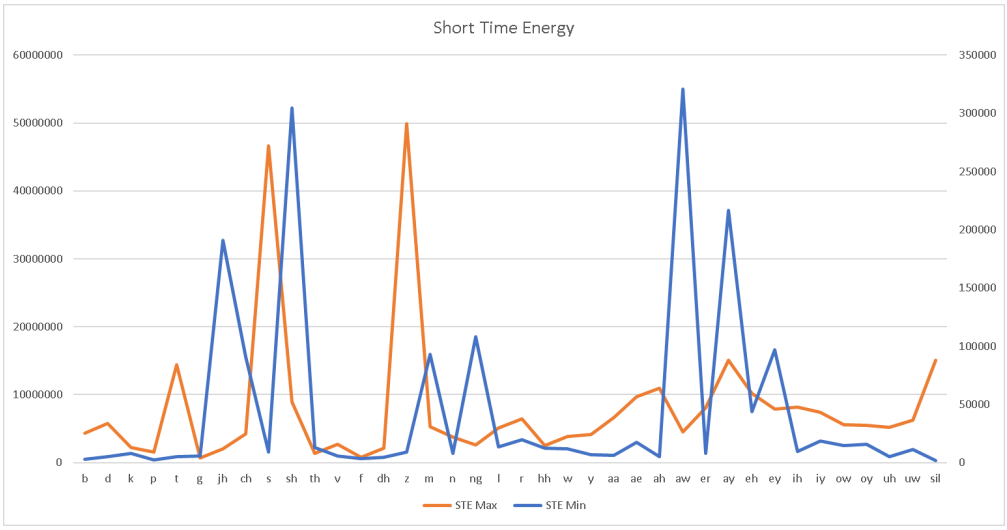

Figure 9: Short time energy minimum and maximum values

Unlike the zero-crossing rate graph the groups of voiced and unvoiced phonemes are not as clear, however, when changing between phonemes of a different voice type the change is significant. Therefore, the weak boundaries of maximum and minimum do not exclude this property from being useful as these extrema would be considered uncommon and so having two adjacent phonemes, both with these extrema would be unlikely.

9.3 Entropy

Entropy can be described as a measure of disorder in a signal and so unlike zero-crossing rate and short time energy where the change is due to the nature of different phonemes, entropy is used as this value should be high around the transition between phonemes - especially if the window is partially over both. Combining this metric with the previous metrics provided a significant increase in accuracy, this is believed, at least in part, to be the information provided when the window is overlapping the boundary as with the other two metrics the values will get average and a significant change may not be seen until further into the sequence.

$$E = -(\sum_{n=1}^{N} x_i \times log_2(x_i)) \tag{8}$$

In the above equation, x is the sample of the given segment and N is the number of samples in the segment.

10 Phoneme Recognition

10.1 Dynamic Time Warping

Dynamic Time Warping (DTW), first described in Bellman and Kalaba (1959), is a distance measure between data sequences, most commonly in the time domain. This method has been extensively explored for use in speech recognition with the most notable version being described in Sakoe and Chiba (1978). The main advantage of DTW over metrics such as Euclidean is the ability to compare series of different lengths and the reduced penalty for similar time series that are out of phase. The visualisations found in Fig. 2 compare how Euclidean and DTW would compare out of phase sequences, it is shown that DTW more closely matches the notable features of the time series than Euclidean.

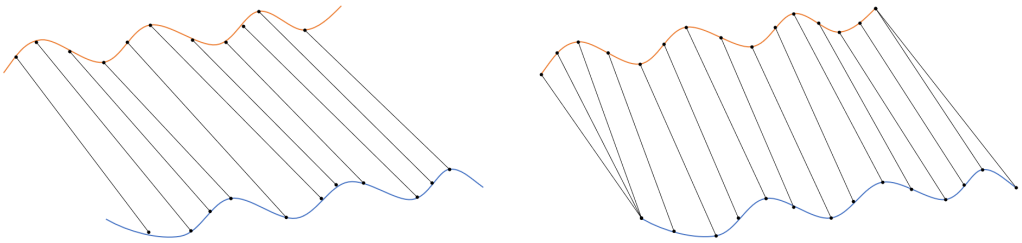

Table 2: Visual representation of time series comparison using Euclidean distance (left) vs dynamic time warping (right) Buckingham (2021)

A description of dynamic time warping can be seen in Alg. 1 and sees an N x M matrix being created. A basic version of DTW would see the entire matrix being traversed however, this makes the method time-intensive whilst also potentially allowing too much warping of the series resulting in an over-fitted comparison. The large time requirement can be reduced by applying a window around the ideal path - the diagonal path - and as such reducing the number of comparisons made to traverse the matrix, Fig. 3 demonstrates how a window is applied to DTW path.

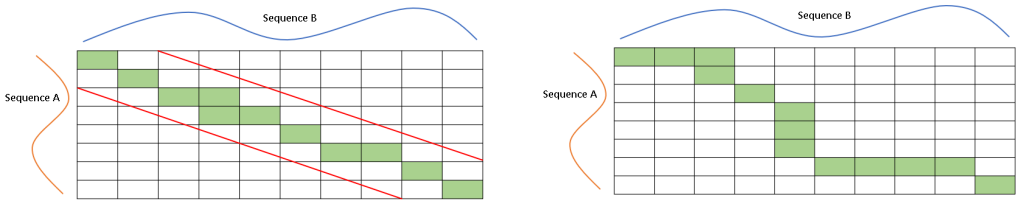

Table 3: Dynamic time warping matrix with windowing (left) and without (right) Buckingham (2021)

”The most commonly used constraint is the Sakoe-Chiba Band, which expresses w as a percentage of the time series length”, Dau et al. (2017), this constraint is demonstrated in Fig. 3 and is expressed as a uniform band around the optimal path. It was known during the implementation of these methods that ”its warping window size is crucial for the final accuracy of the model”, Buza and Peška (2019), therefore considerable reading was undertaken at the beginning to determine the most efficient parameter. Many methods use an extensive empirical method or statistical approaches typically using a leave-one-out approach as described in Chen et al. (2012), Tan et al. (2018) and Dau et al. (2018).

Through the use of numerous other methods, however, the size of the DTW matrix is at most 15 x 15 for the final chosen parameters, this results in a search that is not only quick but is not significantly affected by changes in window size. A window size of 20% was determined through empirical testing to be a fairly optimal window size for smaller window sizes and not frame limit but this optimal window now only provides $<1\%$ accuracy increase.

Algorithm 1 DTW of MFCCs

```
function MFCC-DTW(test, train)
    tr_l := lengthof(test)
    te_l := lengthof(train)
    w := max(test_length, train_length) × limit
    m := matrix(test_length, train_length)
    for te_frame := 1 do te_l
        for tr_frame := max(1, i-w) do min(tr_l, i+w)
            for coeff := 1 do number_of_coefficients
                cost := cost + | test[te_frame][coeff] - train[tr_frame][coeff] |
            end for
            last_min_cost := min(matrix[te_frame - 1][tr_frame], ...
                matrix[te_frame][tr_frame - 1], ...
                matrix[te_frame - 1][tr_frame - 1])
            matrix[te_frame][tr_frame] := last_min_cost cost
        end for
    end for
    score := matrix[te_l][tr_l]
end function
```

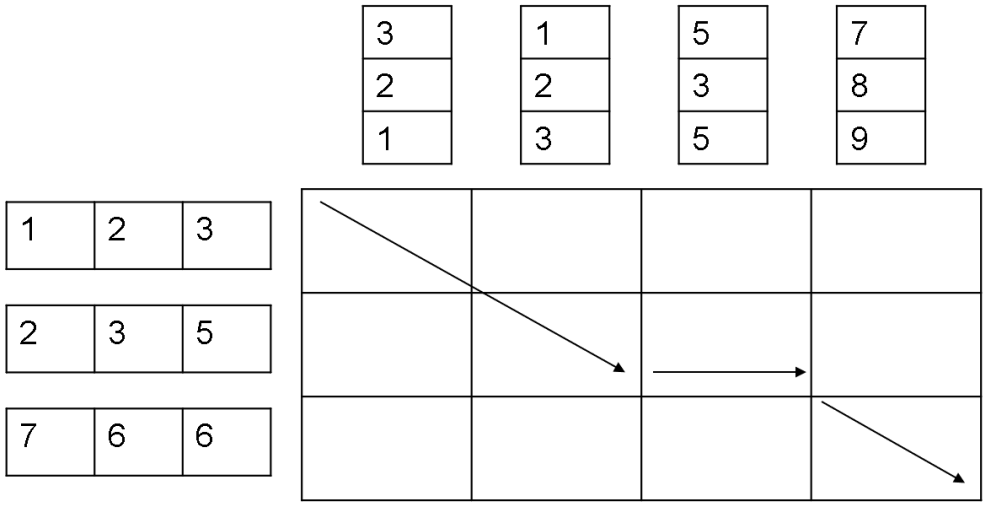

Figure 10: MFCC DTW

In Fig. 10 the method described in Fig. 1 is shown visually where each feature vector is compared as a whole, with a element-wise comparison of the features being summed to produce the total difference for that frame.

10.1.1 Experiments

1. Distance Metric

   The classical Dynamic Time Warping method commonly used in speech recognition, described in Sakoe and Chiba (1978), describes the difference of two points as the absolute difference.

$$D_i = |A_i - B_i| \tag{9}$$

   Additional DTW distance metrics were used to test the effect of the distance metric. Firstly, a standard square function was applied to distance to penalise greater distances more than closer distances.

$$D_i = |A_i - B_i|^2 \tag{10}$$

   Secondly, a weighted metric as defined in Jeong et al. (2011) was used; in the

weighting of the distance is described as the phase of the current point comparison resulting in a higher penalty for phase warping.

Finally, one of the penalty metrics as described in Clifford et al. (2009), specifically, the second proposed metric defined as

$$P = 3 \times 10^5 \tag{11}$$

where P is the penalty applied if the comparison is not along the diagonal.

Unfortunately, none of these changes produced any significant differences in accuracy using the parameters as described [here] and this is because these metrics are affected heavily by the length of the signals being compared. Comparisons of small signals will only produce a small weighting or the weighting will only be applied a very small number of times and consequently using these metrics within this model do not provide any benefits.

10.2 K-Nearest Neighbour

The K-Nearest Neighbour (KNN) algorithm is an intuitive, and relatively simplistic classifier that has been used in various classification scenarios including those pertaining to speech recognition. The method of determining a classification sees a test sample be compared to some or all training samples using a distance metric. Different metrics are more suitable for certain classifications with the Hamming distance being commonly used in string comparison and Euclidean being one of the most common for data in 2- or 3-dimensional spaces. Once all necessary comparisons are made the distance metric s are sorted and the K smallest are selected, the most common class found within these selected K is the decided classification.

In this project, Dynamic Time Warping has been used as the distance measure for the KNN classifier as this allows the use of DTW's phase alignment with the robustness of KNN. The use of DTW on its own cannot produce significant results as outliers and low difference classifications are common but the use of KNN reduces the penalty of these as there would have to be multiple instances of these in order to create a misclassification. This robustness does come at a cost of computation time, DTW with no speedups is $O(n^2)$ and the addition of KNN results in $O(n^2 * m)$ where m is the number reference train sequences.

Methods have been described to account for this as although DTW does seek to compare signals of differing lengths there is an inherent bias when comparing those of similar lengths, these methods take the form of interpolating the smaller signal to be the same length or normalising the scores. These methods did produce improve overall accuracy, however, the improvement was only in accuracy and understandably did not improve the time to test. Through a number of experiments it was shown that comparing sequences of the same length produced the best improvement whilst reducing the computation time required; the reasoning for this improvement is utterances of similar accent and stress are more likely to be of similar length.

Algorithm 2 KNN of equal length sequences

```
function L-KNN(test, k)
    test_length := lengthof(test)
    distances := [number_of_train]
    for i := 1 do number_of_train
        distances[i] := dynamic_time_warp(train[i], test)
    end for
    closest := minimum(distances, k)
    classification := mode(closest)
end function
```

10.3 Hierarchical K-Nearest Neighbour

Table 4: Voice classification accuracy using different features

Feature	Group	Voice
Kurtosis	40%	69%
Flatness	45%	72%
Log Entropy	47%	73%
Short Time Energy	47%	73%
Zero Cross	47%	73%

English phonemes can be grouped into their voice type and the manner of articulation when the sound is produced. Firstly, the phoneme is categorised into voiced

	Group & Voice	Group	Base KNN	DTW Only
One-to-one	38.89% (62.94%, 82.52%)	41.94% (67.04%)	43.36%	12.82%

Figure 11: Testing hierarchical k-nearest neighbour on one speaker

or unvoiced speech, the distinction between these two are the use of the vocal chords when producing voiced speech. The two most common, and useful, features used when classifying speech into voice/unvoiced is zero cross rate and short time energy, these can be distinguished because ”zero crossing rates are low for voiced part and high for unvoiced part where as the energy is high for voiced part and low for unvoiced part” Shete et al. (2014). A hierarchical KNN as described in Dekel et al. (2004) was implemented and tested to improve the accuracy of the model.

Once the voice has been determined a finer group can be classified to further reduce the number of comparisons and reduce the risk of a misclassification. A number of features were tested to determine the group and voice, seen in Fig. 4, and the MFCCs themselves. Additionally, an alternative method, described in Hamooni and Mueen (2014) which uses the raw time-domain signals for classifying certain groups were attempted though none of these features proved to be successful enough to increase the accuracy of the system.

This testing also saw the use of multiple features to determine the group however, as no increase in accuracy was found these methods were simply adding more time and space requirements to the project with no benefit. It was decided to not continue implementing this method even though there is believed to be potential with this method. The results of these tests can be found in Tab. 11.

10.4 Reducing Model Size

In Tab. 5 it is shown that the results over five iterations with a threshold of 10% accuracy. Each iteration was performed on an equally sized subset of the test data to prevent overfitting the model to the test data, this did, however, result in less test data per test and so a typical 70/30 split could be used but it is believed that these results are consistent enough to consider this method an improvement.

In Fig. 12 we can we see much broader classifications when testing multiple speakers, in this case 50, when compared to only testing one speaker. This is of course, at

Table 5: Test iteration of all test sets

Iteration	Accuracy	Size
1	30.93%	32.70MB
2	32.31%	21.46MB
3	31.95%	16.28MB
4	32.88%	13.74MB
5	33.26%	12.52MB

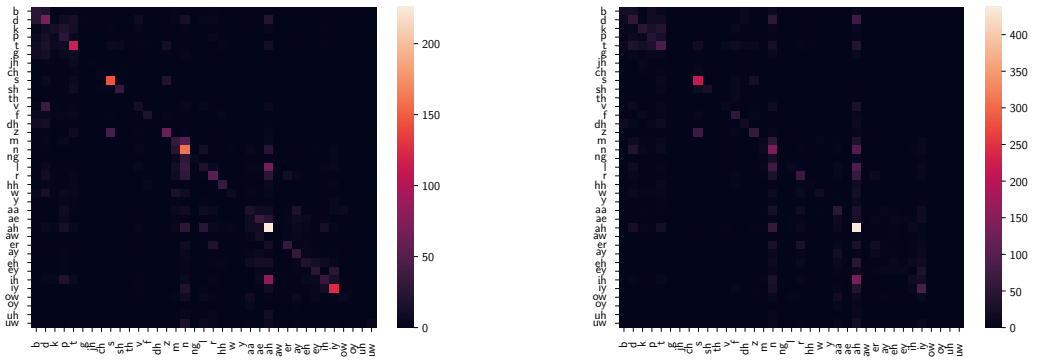

Figure 12: Single speaker heatmap (left) compared to all speakers heatmap (right) after five iterations

least in part, due to the variation in accent, tone, and articulation of the spoken speech but it is interesting to note that many misclassifications are common between both tests suggesting that there is another reason. Additionally, these common misclassifications are not always amongst phonemes of the same group and so the reason is not necessarily as simple as they are similar.

One reason for these common misclassifications could be the number of training instances for each phoneme, as KNN classifies based on reference instances and so more references are more likely to produce a higher accuracy. This is supported by Fig. 13 where the number of training instances and the accuracy follow very similar patterns.

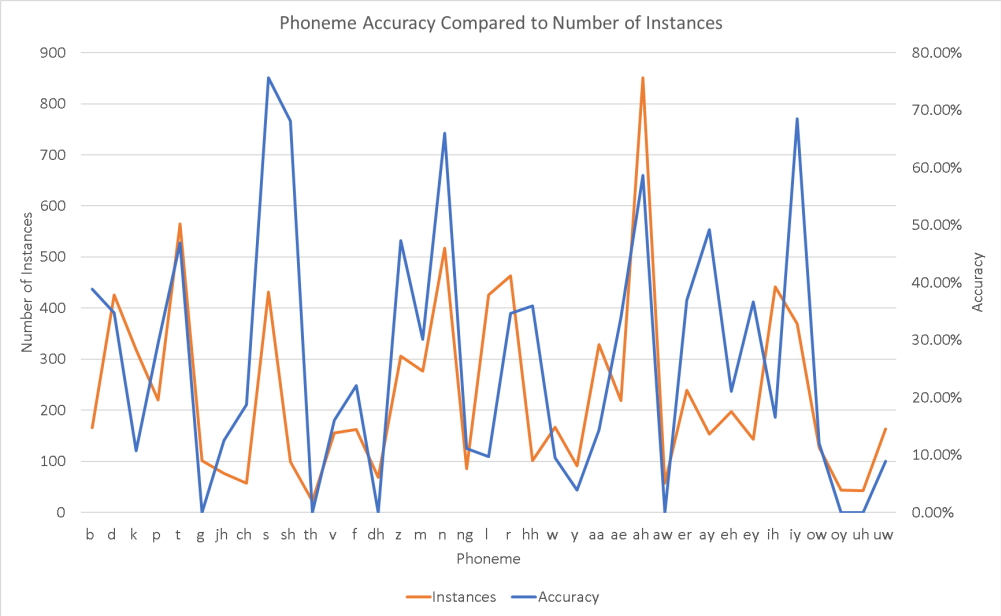

Figure 13: Accuracy of single speakers compared to number of tests instances

10.5 Frame Limit

During testing, it became clear that the longest sequences we that of silence and with these sequences varying little due to the lack of sound it was thought that the size of the model could be reduced by removing unnecessarily long silence sequences. To determine an appropriate point at which to remove truncate frames the variance between each phoneme for each frame was produced. As seen in Fig. 14 the variance has been normalised to accentuate the findings, it is clear that there is a reduction in variance after 40 frames and so this was determined to be where most phonemes stop and only silence reaches this length.

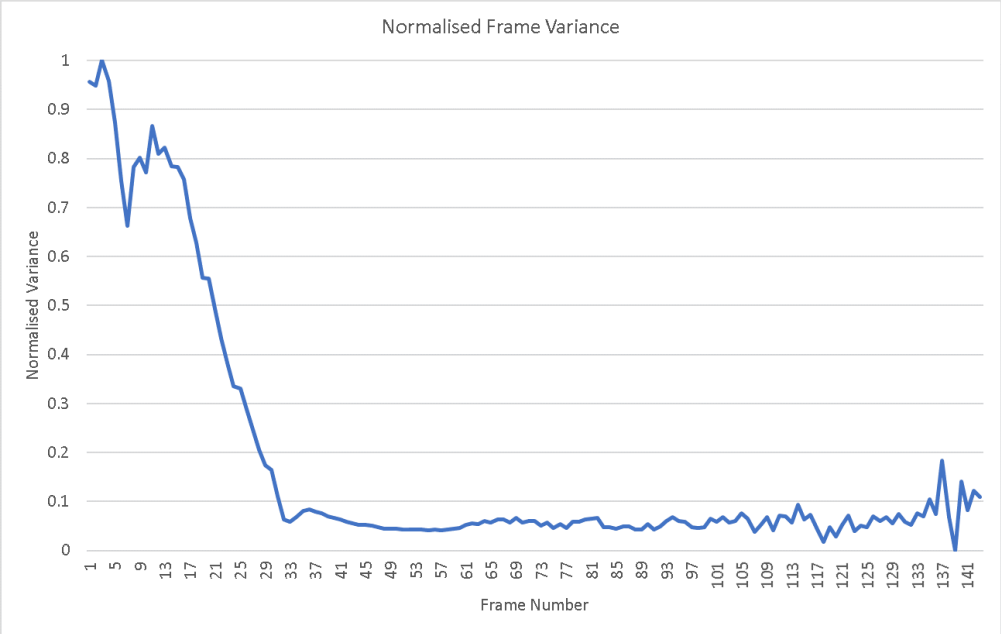

Figure 14: MFCC frame variance

Truncating the silence sequences only provided a noticeable increase in accuracy however, further experimenting showed that truncating phonemes produced further improvements in accuracy. The testing range was decided by the above graph, where the variance levels off around frame 40. This method not only increased accuracy but by limiting the frames the model size was reduced and the test time was also reduced.

Table 6: MFCC frame limit accuracies

5	10	15	20	25	30	35	40
34.30%	41.51%	43.36%	42.87%	42.35%	41.69%	41.56%	41.04%

This increase in accuracy is believed to be related to the slight bias relating to comparing signals of differing lengths and sequences of similar lengths are likely to be of a similar accent.

11 Recognition with boundary detection

Once a suitable set of parameters were determined for both the boundary detection and classification methods they were used in conjunction to produce a continuous speech

recognition model. As the boundaries were found to be imperfect it was expected that the accuracy would diminish however, combining these two methods will produce a much more concise accuracy of the boundary detection as insertions and deletions can be determined whilst providing an accuracy similar to a real-world application.

The accuracy measure is the Word Error Distance (WER), as denoted in the below equation, which is derived from the Levenshtein distance with back-tracking which determines a string's similarity by calculating the minimum number of edits requires to make them the same.

$$WER = \frac{S+D+I}{S+D+C} \tag{12}$$

As shown above insertions, deletions and substitutions equally incur a penalty of 1 however, some methods apply a penalty of 2 for substitutions, such as the methods seen in Lilley et al. (2012), and so it was decided to test both methods in order to see how insertions, deletions and substitutions would affect the correctness score.

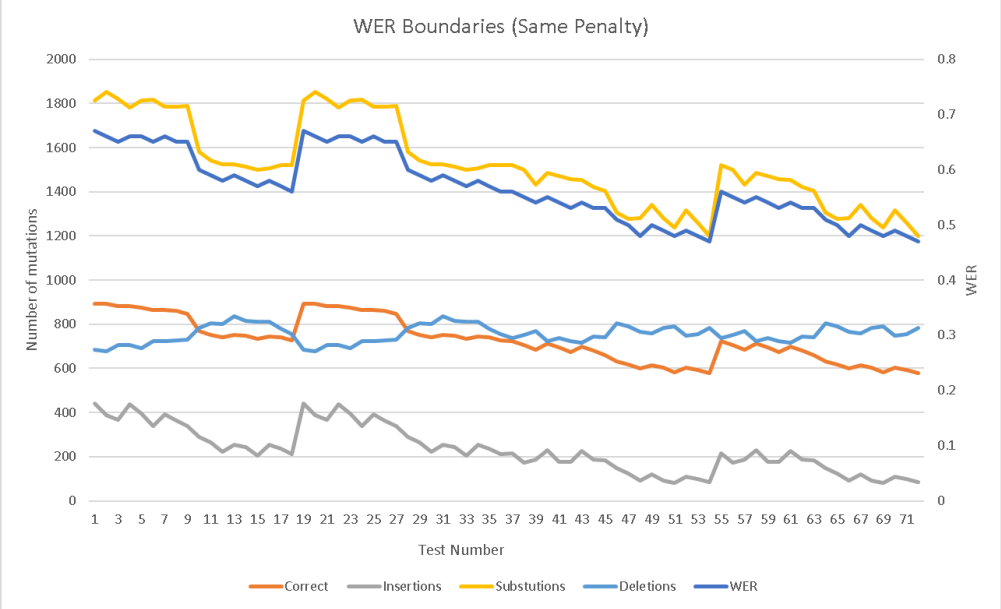

Figure 15: Mutations and word error rate with equal penalty

It is shown in Fig. 15 that when an equal penalty is applied to all mutation types the number of correct guesses and substitutions is much higher and more variadic throughout different testing parameters whilst in Fig. 16 it is shown that correct and

substitutions are much lower and flatter, and now it is deletions and insertions that have more influence over the WER.

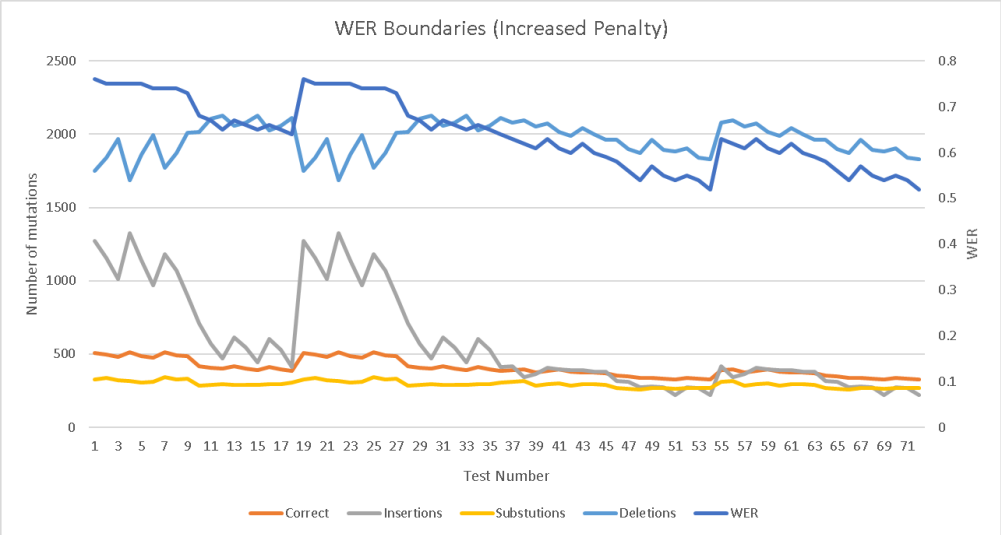

Figure 16: Mutations and word error rate with increased substitution penalty

Although both charts are significantly different there is one clear connection between the two, the lower the number of mutations the lower the WER, which seems reasonable as fewer mutations is a sign of a more accurate boundary detection method. However, both charts there is no consistent increase in correct classifications that happens alongside a consistent decrease in mutations.

Unfortunately, this leaves one with questions about the actual competency of these methods. As the boundary detection methods are based on thresholds we know that it is too high when deletions begin to increase and too low when insertions begin to increase, in both charts we can see insertions have a downwards trend whilst deletions stay within a range with no upwards or downwards trend suggesting the thresholds have yet to reach an upper limit.

It is believed that more robust testing is needed to determine a reliable accuracy measure however, I believe this approach was sufficient enough to find trends relating to changes in parameters and to show competency of methods that could be improved in future work.

Fig. 7 shows the WER rate when using the different tested KNN hierarchies, as we can see they follow the same trend as seen in Tab. 11, supporting that the one-to-one

Table 7: Hierarchical KNN accuracies

	Group & Voice	Group	Base KNN
Boundary (WER)	0.66	0.56	0.52

accuracies are transferable to a method which does not produce perfect boundaries. An example of these found boundaries can be seen in Fig. 17, it is shown that a lot of boundaries are found within silence due to the values being calculated through percentage change, this means noise can greatly affect the detection of a boundary. It was hoped however that these boundaries found within silence would still be classified as silence as so it would produce no output to the user, unfortunately, that is clearly shown not to be the case.

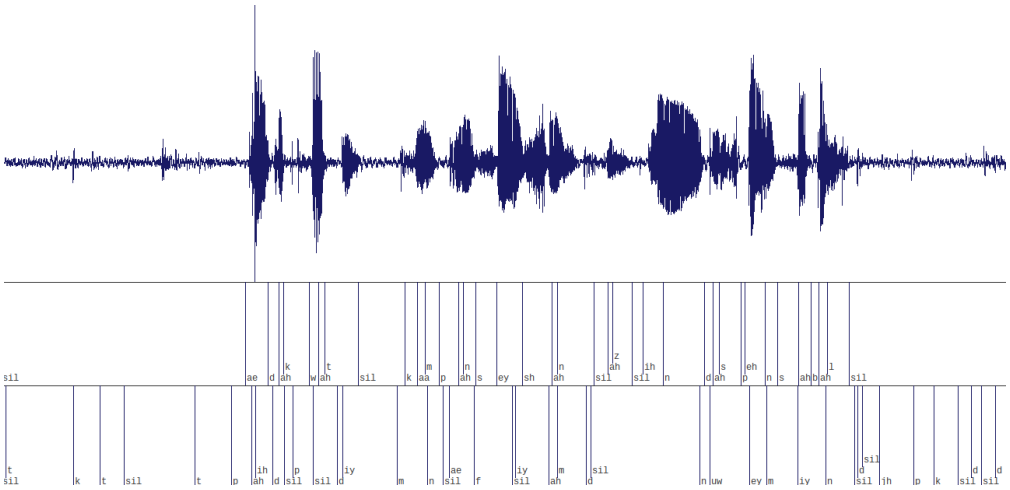

Figure 17: Reference (top) and result (bottom) labels of a test file

Another common property of the boundary detection method demonstrated in Fig. 17 is the finding of multiple boundaries when there is only one, sometimes this is two close together or one before a boundary and one after. This is thought to be caused by the same reason entropy was used as a parameter; the area around phoneme transitions produces more disorderly data. There are some boundaries close enough to be considered competent guesses however, these are often followed by a deletion or insertion which then results in a misclassification no matter how accurate the correct boundary guess was.

This method shows potential however, a more refined model with a more sophisti-

cated set of thresholds would need to be produced before this method shows any results that would allow this project to be viable in any real application.

12 Noise Robustness

The results of this testing have shown that the model cannot be said to be robust to noise as significant decreases in accuracy were seen in both one-to-one tests and boundary detection. As progress on other areas of the project proved to take longer than anticipated methods were not explored thoroughly enough to determine if this model could be made robust to noise. Unfortunately, this restricts the commenting of the device in everyday use.

Table 8: Robustness to different noise types

Noise Type	Cafe	Car	Street	White
Accuracy	14.99%	20.80%	13.77%	8.32%

13 Hardware

Many vibro-tactile devices produced in research have served to demonstrate the viability and competency of such devices however, they are prevented from being used in real-life situations, either by the device being used or by the language being used which provides an undesirable delay in response.

The main aim of this project was to produce a model which could easily be loaded onto a micro-controller which would be suitable for everyday use; such a device would be small enough to fit on the user's wrist without discomfort and be able to produce results outside of the McGurk effect's range.

13.1 Micro-controller

Many similar and successful devices have been produced but there were none found that focused on speech and provided a device that would be small enough for use in an everyday setting. The use of these devices allows for easier and quicker development

though with the correct processes implemented it wouldn't be impossible to move these devices onto a smaller device.

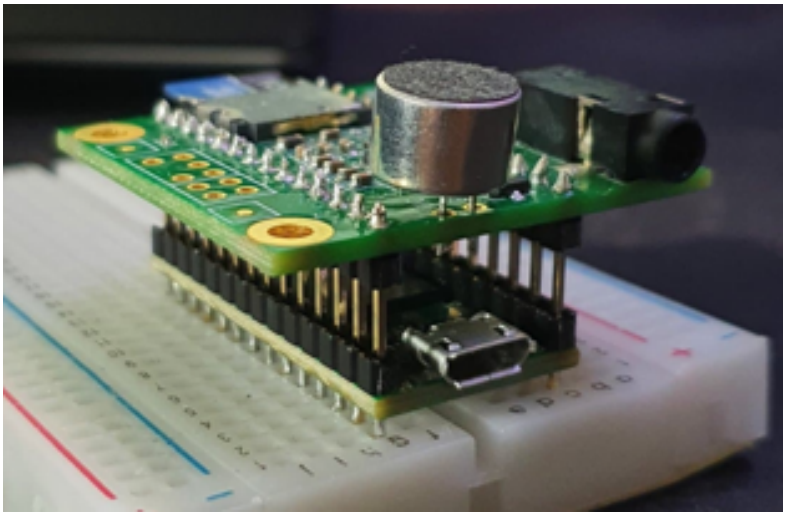

Figure 18: Teensy 4.0 with audio shield

As the aim of this project was to design such a device small enough for everyday use it was decided to develop solely on a suitable micro-controller. This was decided to be the Teensy 4.0 for its small form factor of just 35.56 x 17.78 $\pm$ 0.06 x 4.64 mm. This form factor was deemed small enough to be placed on the user's wrist comfortably and without looking too conspicuous.

This small form factor does come with the limitation of just 1k in memory split between the stack and heap, this meant the model could not be kept in memory and had only the currently needed MFCCs needed to be loaded in but even then this produced problems. Unfortunately, this process still proved to be too much and so the Audio Adaptor from Teensy was purchased to allow for a 1Mbit RAM chip to be added to the board, increasing the form factor to roughly 37.5 x 37.5 x 20 mm. Though this did reduce the usability it did provide a native SD card holder and microphone socket allowing for a much faster development.

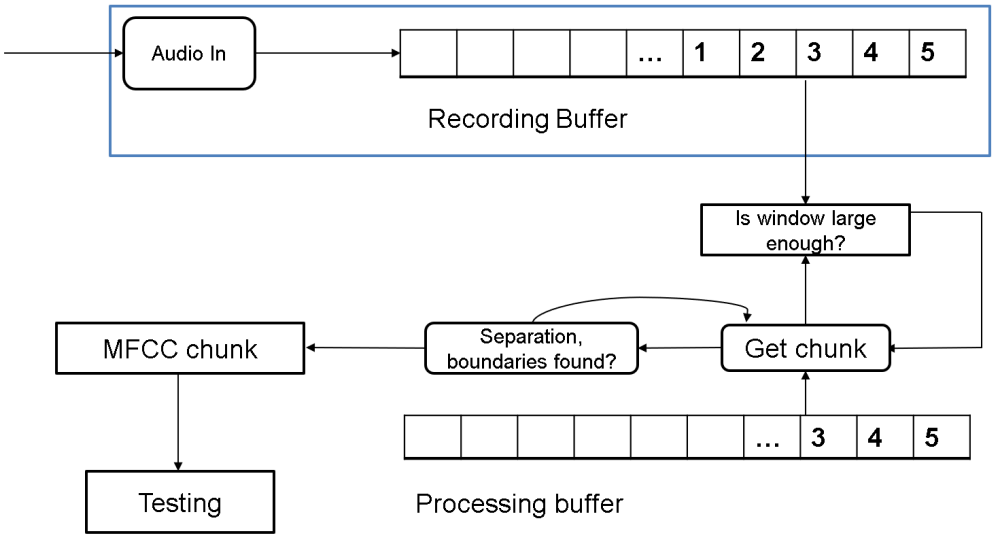

Figure 19: Process of feature extraction and classifying

The process, shown in Fig. 19 was first tested using a desktop application which had one thread reading in audio files at a rate of 16kHz with another thread reading this recording buffer, looking for boundaries then moving it to another buffer for processing before classifying. Producing a desktop method first allowed for much easier debugging and minimal changing of methods once moved on the micro-controller.

A UML, Fig. 20, and IPO, Fig. 21 diagram have been produced to describe the complete process from the device recording of audio to the output of vibration sequences to the user.

13.2 Microphone and Training Data

Throughout this project the TCD-TIMIT and NCTD-TIMIT data sets were used because recording and labelling my own data would not only take up a lot of this projects time but it would not match the quality provided by these data sets. Unfortunately, once the methods were moved onto the device it appeared that the boundary detection system was not as accurate, this is believed to be because the threshold values are unique to each the microphone which records the audio and without sufficient testing methods implemented on the device it was not possible to determine threshold values specific for the microphone used.

14 Tactor Sequences

It was this project's intention to discuss, develop and test tactor sequences specifically for this device, however, although exploration of these ideas was done setbacks in the development of this project resulted in little time to begin developing on the micro-controller and this pushed back any development of the tactor sequences.

A minimal example of the tactor sequences was produced however and it is believed that the simplicity of the required methods means this minimal example can be easily adjusted and expanded in order to produce a suitable set of sequences.

The initial design would see the spatio-temporal sequences split into their respective groups, similar to the methods described in Reed et al. (2019), with defining characteristics for each to help distinguish them when played to the user, for example, vowels may be played left to right with consonants being right to left. Though with the limited number of tactors in use it is thought that more subtle differences like the amplitude or frequency would be necessary. In both Reed et al. (2019) and Tan et al. (2020) a range of 60 to 300 Hz was described as optimal frequencies for transferring information through the skin and so this should be a sufficient range to distinguish groups by frequency alone.

15 Evaluation

The aims of this project were to create a wearable device that would allow the user to 'hear' phonemes through vibration sequences, and to allow the methods used to be easily applied by someone else with minimal knowledge of the implementation details.

I believe this project has managed to produce a program and the processes necessary to produce a model and deploy this model onto a micro-controller, that can be used by most without much prior technical knowledge. The process of changing parameters, outputting a model and deploying it onto a programmed device is an a fairly simple method requiring minimal input from the user beyond the necessary model properties. The program itself has a simple but effective GUI for those who are not familiar with command prompts and the output of the program is either in readable text or more detailed output can more easily be placed into excel or a Python script.

The main aim of this project has seen a lot of problems and setbacks, the decision to use KNN and DTW did limit the accuracy however, it is believed that developing a

neural network or hidden Markov model would have not only required more time but also produced more issues and set backs. The current methods for creating MFCCs, classifying and producing sufficient testing output prevented much focus from being placed on the tactor sequences themselves resulting in minimal exploration and testing of the sequences themselves. Though this intention was to produce a model that someone with knowledge of this area could apply to their own sequences, it would have been preferable to test the effectiveness of the device properly, but with such a low accuracy it may not have produced any valuable results anyway.

The Gantt chart, seen in Fig. 22, produced for this project early on became to be seen as too ambitious and many of the elements fell behind schedule with many of them having to overlap in order to complete the majority of the desired project on time. This overlap of development left less time for more in-depth formal testing, optimisations, and user experience designs however, in order to produce a working model to demonstrate the viability of the proposed product it was necessary to reduce the any additional features found described in the could and should-haves of the MoSCoW to focus on the must-haves.

15.1 Progress

Many of the tasks set in the final Gantt chart were not started on time due to other tasks taking much longer than expected. The device specific tasks were pushed back to allow for more time on the feature extraction and classification models as it was believed that the functions could be easily moved onto the device and as there were no debugging methods for the device it would be easier to debug on a desktop application.

Unfortunately one of the main tasks of this project was to produce a device with tactor sequences and demonstrate how the device would be used in a real-life situation however, delays in other tasks meant a change in project scope was needed to get at least the majority of the project completed to a satisfactory standard. It was decided that producing a model for the device would be more important than creating an unusable model with working tactor sequences as implementing the sequences would not take much time; only deciding how to design the sequences would. In this way, only one aim of the project failed to implement the tactors, the complete hardware device, but the other aim, allowing others to the use designs and model to develop their own would be fully complete.

15.2 Limitations

Many of the limitations have been mentioned or discussed in their respective sections however, this section aims to formally describe and discuss the main limitations of the final state of the project:

 The accuracy of the model, both with and without the boundary detection will make this device unusable for anyone in a real-life situation. Though inaccuracies were expected from the start it was hoped that the majority of incorrect classifications would at least be similar phonemes however, a considerable amount was completely wrong and would produce nothing useful to the user. The boundary detection method appears to be very sensitive to not only noise but to the microphone in use. White noise affected the accuracy the most for one-to-one accuracies and so a microphone with a constant low noise would prove to be less than useful for the device. Though the device was in part intended to provide a base for others to provide their own MFCC models and tactor sequences a lack of development on my own tactor sequences and subsequent testing of these results in a device that provides no confirmation of its viability. The device was not tested in a usable manner, no wrist mounting was made and no formal user testing was presented and so it can only be discussed as to the usability for everyday use by accessing the form factor of the micro-controller.

15.3 Further Work

Some of the limitations were intended to be implemented and fixed, and some simply did not achieve the aims of this project. A longer time frame to continue development and allow more uninterrupted focus on this project could have seen a more viable product by working on the limitations:

 Dynamic Time Warping and K-Nearest Neighbour would be used due to the development time and simplicity however, an Support Vector Machine or Neural Network would almost certainly have provided better accuracy. Enough development time could have seen these produced and reduced down enough to fit onto the desired micro-controller. Exploring different designs and tactor sequences could have seen different designs discussed, designed, and possibly implemented with the best performing chosen and developed more. User testing would have been an ideal addition to this project to determine the usability and learning potential of such a device with sufficient time and resources to develop a number of devices the device could have been tailored

to user feedback and testing results.

16 Conclusion

The aims of this project were perhaps too large for the scope and as such some of what was intended was not implemented however, a working proof of concept was produced and it is believed that by continuing with this project and implementing what is set out in subsection 15.3 this project could fulfill what it set out to do.

The methods explored in this project show potential to be useful in a speech recognition even when compared to newer methods, however, further work would see these other methods explored as a key reason for not exploring these routes was the time constraint of this project. Knowledge and experience in these other methods was acquired nearer the end of this project, if this was acquired before the project it may have been viable to implement them as well.

It was unfortunate that a complete device with a complete set of tactor sequences and wearable aspect was not fully developed and tested, it would have added much to the project to explore the results of someone learning through such a device. Even if the tactor sequences were simply played along with a method of learning the sequences it would have at least shown that if the accuracy of the device was sufficient then combining these sequences would have provided a suitable method for learning language through the skin, unfortunately, the insufficient accuracy and time restrictions did not make this option viable either.

References

Abdelaziz, A. H. et al. (2017). Ntcd-timit: A new database and baseline for noise-robust audio-visual speech recognition. In Interspeech, pages 3752–3756.

Barbacena, I. L., Freire, R. C. S., Barros, A. T., Aguiar Neto, B. G., Carvalho, E. A. N., and Tavares de Macedo, E. C. (2009). Voice codification evaluation based on a real-time training system with tactile feedback applied to deaf people. In 2009 IEEE Instrumentation and Measurement Technology Conference, pages 697–700.

Bellman, R. and Kalaba, R. (1959). On adaptive control processes. IRE Transactions on Automatic Control, 4(2):1–9.

Buckingham, T. (2021). Progress report.

Buza, K. and Peška, L. (2019). Individualized warping window size for dynamic time warping.

Chen, Q., Hu, G., Gu, F., and Xiang, P. (2012). Learning optimal warping window size of dtw for time series classification. In 2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA), pages 1272–1277.

Chu, S., Keogh, E., Hart, D., and Pazzani, M. (2002). Iterative Deepening Dynamic Time Warping for Time Series, pages 195–212.

Clifford, D., Stone, G., Montoliu, I., Rezzi, S., Martin, F.-P., Guy, P., Bruce, S., and Kochhar, S. (2009). Alignment using variable penalty dynamic time warping. Analytical chemistry, 81(3):1000–1007.

Dau, H. A., Silva, D. F., Petitjean, F., Forestier, G., Bagnall, A., and Keogh, E. (2017). Judicious setting of dynamic time warping's window width allows more accurate classification of time series. In 2017 IEEE international conference on big data (big data), pages 917–922. IEEE.

Dau, H. A., Silva, D. F., Petitjean, F., Forestier, G., Bagnall, A., Mueen, A., and Keogh, E. (2018). Optimizing dynamic time warping's window width for time series data mining applications. Data mining and knowledge discovery, 32(4):1074–1120.

Dekel, O., Keshet, J., and Singer, Y. (2004). An online algorithm for hierarchical phoneme classification. volume 3361, pages 146–158.

Hamooni, H. and Mueen, A. (2014). Dual-domain hierarchical classification of phonetic time series. In 2014 IEEE International Conference on Data Mining, pages 160–169.

Harte, N. and Gillen, E. (2015). Tcd-timit: An audio-visual corpus of continuous speech. IEEE Transactions on Multimedia, 17(5):603–615.

Jeong, Y.-S., Jeong, M. K., and Omitaomu, O. A. (2011). Weighted dynamic time warping for time series classification. Pattern recognition, 44(9):2231–2240.

Junkui, L. and Yuanzhen, W. (2009). Early abandon to accelerate exact dynamic time warping. Int. Arab J. Inf. Technol., 6:144–152.

Kardava, I., Antidze, J., and Gulua, N. (2016). Solving the problem of the accents for speech recognition systems. International Journal of Signal Processing Systems, 4(3):235–238.

Kelly, A. and Gobl, C. (2011). The effects of windowing on the calculation of mfccs for different types of speech sounds. volume 7015, pages 111–118.

Lilley, M. S., Garland, E. C., Rekdahl, M. L., Noad, M. J., Goldizen, A. W., and Garrigue, C. (2012). Improved versions of the levenshtein distance method for comparing sequence information in animals' vocalisations: tests using humpback whale song. Behaviour, 149(13-14):1413–1441.

Mairano, P. and Calabrò, L. (2016). Are minimal pairs too few to be used in L2 pronunciation classes?

Manikandan, J., Venkataramani, B., Preeti, P., Sananda, G., and Sadhana, K. V. (2009). Implementation of a phoneme recognition system using zero-crossing and magnitude sum function. In TENCON 2009 - 2009 IEEE Region 10 Conference, pages 1–5.

McGurk, H. and MacDonald, J. (1976). Hearing lips and seeing voices. Nature, 264(5588):746–748.

Mohan, B. J. and N., R. B. (2014). Speech recognition using mfcc and dtw. In 2014 International Conference on Advances in Electrical Engineering (ICAEE), pages 1–4.

Munhall, K. G., Gribble, P. L., Sacco, L., and Ward, M. (1996). Temporal constraints on the mcgurk effect. Perception & Psychophysics, 58:351–362.

Novich, S. D. and Eagleman, D. M. (2015). Using space and time to encode vibrotactile information: toward an estimate of the skin's achievable throughput. Experimental Brain Research, 233(10):2777–2788.

Pedersen, P. (1965). The mel scale. Journal of Music Theory, 9(2):295–308.

Perrotta, M. V., Asgeirsdottir, T., and Eagleman, D. M. (2021). Deciphering sounds through patterns of vibration on the skin. Neuroscience, 458:77–86.

Ramteke, P. B. and Koolagudi, S. G. (2019). Phoneme boundary detection from speech: A rule based approach. Speech Communication, 107:1–17.

Reed, C. M., Tan, H. Z., Perez, Z. D., Wilson, E. C., Severgnini, F. M., Jung, J., Martinez, J. S., Jiao, Y., Israr, A., Lau, F., Klumb, K., Turcott, R., and Abnousi, F. (2019). A phonemic-based tactile display for speech communication. IEEE Transactions on Haptics, 12(1):2–17.

Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. IEEE transactions on acoustics, speech, and signal processing, 26(1):43–49.

Salvador, S. and Chan, P. (2004). Toward accurate dynamic time warping in linear time and space. volume 11, pages 70–80.

Salvi, G. (2006). Segment boundary detection via class entropy measurements in connectionist phoneme recognition. Speech Communication, 48(12):1666–1676. NOLISP 2005.

Shete, D., Patil, S., and Patil, S. (2014). Zero crossing rate and energy of the speech signal of devanagari script. IOSR-JVSP, 4(1):1–5.

Tan, C. W., Herrmann, M., Forestier, G., Webb, G. I., and Petitjean, F. (2018). Efficient search of the best warping window for dynamic time warping. In Proceedings of the 2018 SIAM International Conference on Data Mining, pages 225–233. SIAM.

Tan, H. Z., Reed, C. M., Jiao, Y., Perez, Z. D., Wilson, E. C., Jung, J., Martinez, J. S., and Severgnini, F. M. (2020). Acquisition of 500 english words through a tactile phonemic sleeve (taps). IEEE Transactions on Haptics, 13(4):745–760.

Tiippana, K. (2014). What is the mcgurk effect? Frontiers in Psychology, 5.

Winograd, S. (1976). On computing the discrete fourier transform. Proceedings of the National Academy of Sciences, 73(4):1005–1006.

Zhang, X., Sun, J., and Luo, Z. (2014). One-against-all weighted dynamic time warping for language-independent and speaker-dependent speech recognition in adverse conditions. PloS one, 9:e85458.

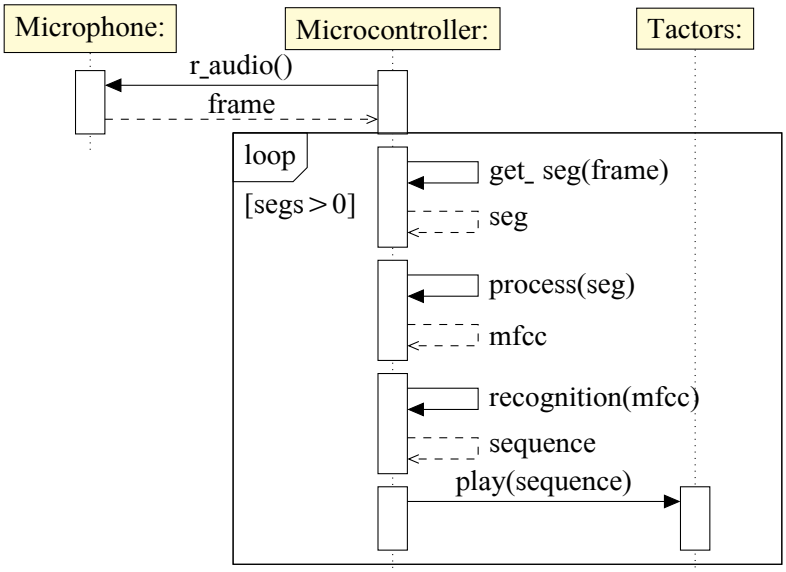

Figure 20: UML sequence diagram Buckingham (2021)

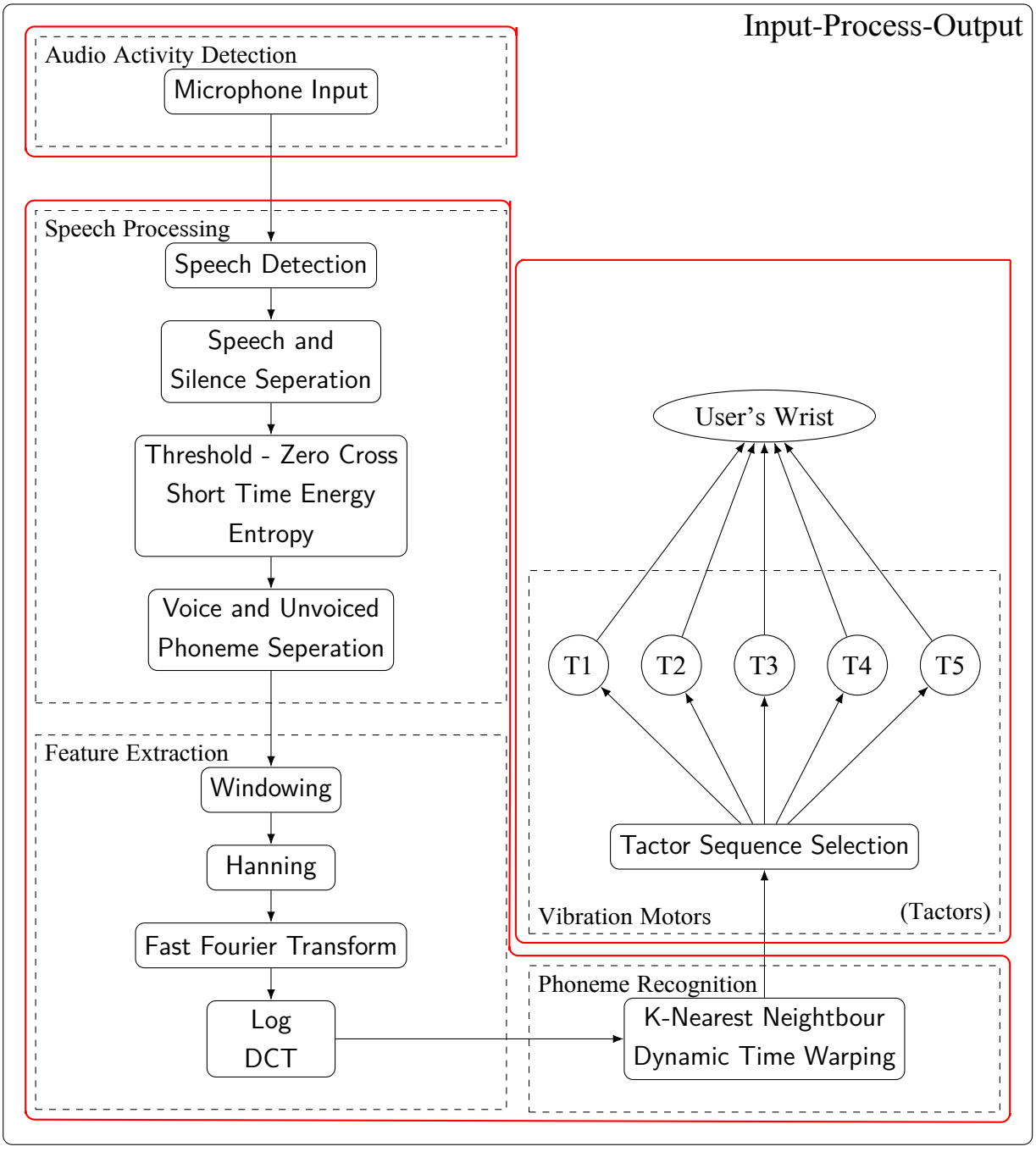

Figure 21: IPO diagram describing the processes performed for on the microcontroller
Updated from Buckingham (2021)

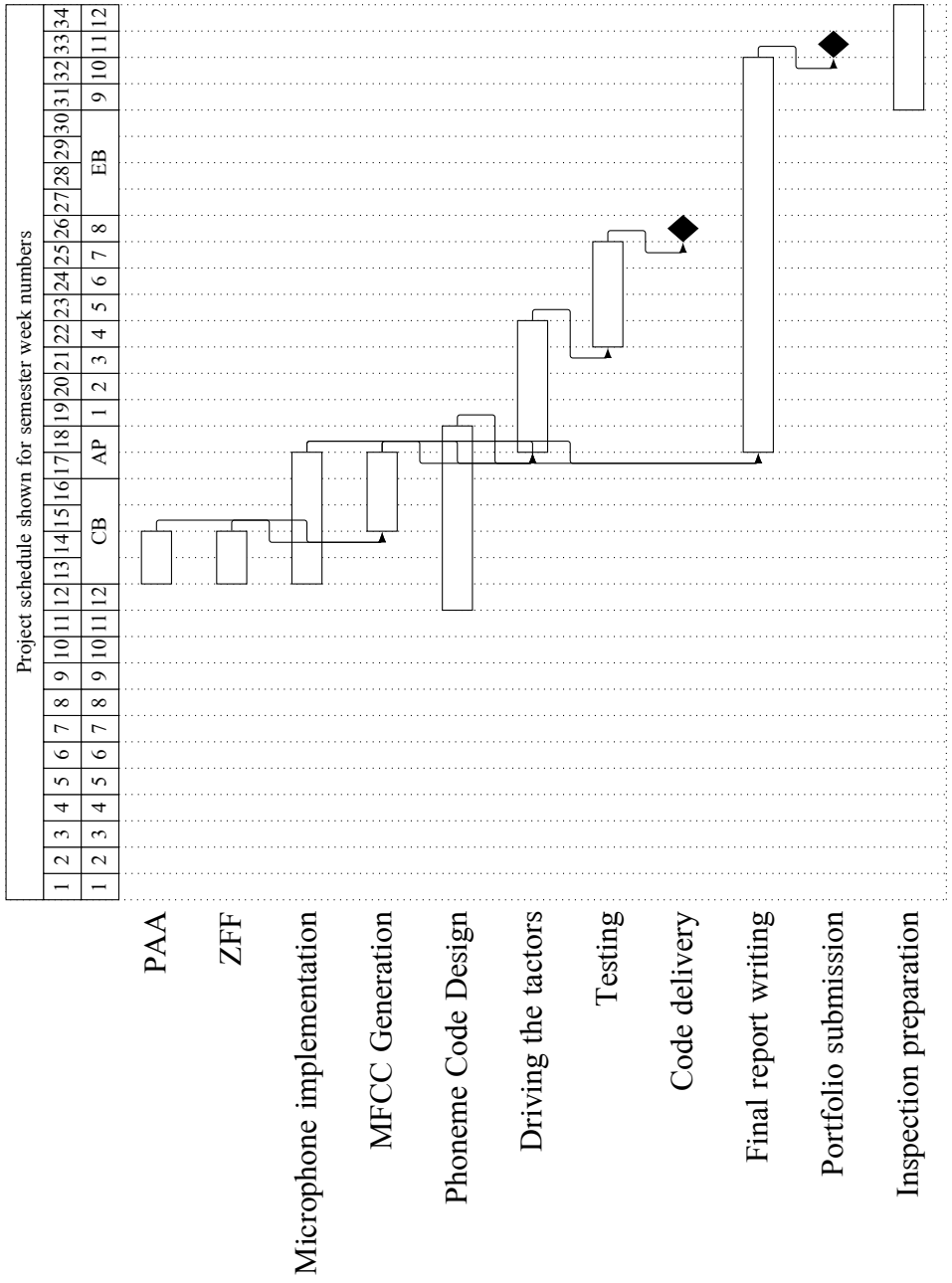

Figure 22: New project Gantt chart from Buckingham (2021).