JAMES COOK UNIVERSITY

SCHOOL OF ENGINEERING AND PHYSICAL SCIENCES

EG4011/2

ROBUST SPEECH RECOGNITION VIA HUMAN AUDITORY MODELING

Anthony Bath

Thesis submitted to the School of Engineering and Physical Sciences in partial fulfilment of the requirements for the degree of

Bachelor of Engineering (Electrical)

Bachelor of Science (Computer Science)

September, 24th 2010

Statement of Access

I, the undersigned, the author of this thesis, understand that James Cook University will make it available for use within the University Library and, by microfilm or other means, allow access to users in other approved libraries. All users consulting with this thesis will have to sign the following statement:

> In consulting this thesis I agree not to copy or closely paraphrase it in whole or in part without written consent of the author; and to make proper public written acknowledgement for any assistance which I have obtained from it.

Beyond this, I do not wish to place any restriction on access to this thesis.

Anthony Bath Date

Sources Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

Anthony Bath Date

Abstract

The presence of convolutive distortion and additive background noise along with differing microphones and environmental conditions is a continuing problem for speech recognition systems and effectively dealing with these factors has proven to be complex. While there has been a lot of research in this field, currently there are no systems that provide a 100% recognition rate even in ideal conditions and as a result, when the conditions differ to that from which the system was trained in, the speech signal is degraded resulting in a much lower recognition rate. The degradation is a direct result of a mismatch between the training and testing data and robust speech recognition allows for dealing with this mismatch.

The aim of this thesis is to consider an auditory model front-end for robust speech recognition and determine how it performs in comparison to what is considered standard in human auditory-based speech recognition. A review on the current literature pertaining to the relevant considerations and techniques for robust speech recognition will first be conducted to gain insight into the field along with what research and experiments have already been conducted. A test bench will be implemented to verify the validity of the proposed methods.

Acknowledgements

First of all, my thanks goes to my supervisor Owen Kenny; fondly known to me as K-Dog and OPK. Without his support and knowledge along the way this thesis would never have been possible.

Secondly, thank-you to my World of Warcraft guild “Khazuals” for allowing me time off to do my thesis during progression through Icecrown Citadel hard modes. Without this extra time I wouldn’t have been able to meet the deadlines.

Thirdly, thanks to my close personal friend Matthew John Franklin Eleanor Grasso for his ongoing support and inspiration over the last five years. The late nights we spent in EL2xx and HX computer labs have made me the man I am today and I am forever grateful.

Finally, thanks to my family and the rest of my friends for being so supportive and understanding throughout the last challenging and stressful five years of my life. Apologies for any events I may have missed out on whilst working on this thesis.

Table of Contents

List of Figures

List of Tables

1. Introduction

Research into the field of speech recognition dates back to the 1940's when AT&T Bell Laboratories developed a basic speech recognition device [1]. Further development was made in 1952 with the advent of a device that was able to recognise single digits [2, 3]. By the 1960's researchers realised that speech recognition was going to be a large part of the future and thus more efforts were devoted to the development of a larger speech recognition system [1].

With the early success of speech recognition, there are currently many applications where speech recognition systems are employed including healthcare, military, telephony and for people with disabilities [3]. However, with the growing demand of speech recognition systems, one large problem presents itself.

1.1 Problem Definition

The large issue with speech recognition systems is that the conditions the systems are used in often differ from where they were trained in and thus as a result of this mismatch, the systems perform worse than expected. With this problem often limiting the capabilities of speech recognition systems, the increasing need for robustness has become evident [4, 5]. The proposed solution is to implement a front-end based on the human auditory model due to the robustness in distinguishing between speech and noise that the ear is capable of [6, 7].

1.2 Research Questions

After developing a potential solution to the problem, the next aim of this thesis was to evaluate the success of the proposed solution. This evaluation allowed the development of the following research questions:

1. How can we use the inherent features of the human auditory system to enhance robust speech recognition?

2. How do we interface this auditory front-end for robust speech recognition and how well does it work?

These research questions allow the scope of the thesis to be realised. In the development of the solution to the problem, the answers to these questions will become evident and thus contribute new knowledge to the field of robust speech recognition.

2. Literature Review

In order to design a more efficient front-end for robust speech recognition, a thorough review of relevant literature must first be conducted to determine areas within the field that have had little to no research. Several things that need to be considered when experimenting with methods of robust speech recognition include:

- What type of speakers will the system cater to and what type of speech will be used?
- What classifiers will be used for modeling?
- What methods are there for robust automatic speech recognition?
- What speech corpus will be used for testing and training?

This section of the thesis reviews literature pertaining to the aforementioned considerations.

2.1 System Specifications

In terms of speakers, automatic speech recognition (ASR) systems fall into one of two categories: Speaker Dependent and Speaker Independent. Depending on which category the system is in affects the training of the acoustic-phonetic models, discussed below. Additionally, the type of speech that can be recognized by ASR systems can be considered as isolated, discontinuous or continuous [8]. Furthermore, the type of recognition of ASR system is generally either word or phoneme recognition.

2.1.1. Speaker Dependence

By definition, a speaker dependent system is trained for the recognition of only a single speaker. The acoustic-phonetic models the system uses for recognition are trained based on the speaker's voice and thus if another speaker were to use the system the results would be very poor. Speaker independence on the other hand is when a system can be used by speakers who have not submitted any speech for training of the acoustic-phonetic models. Speaker independence can be difficult to obtain as the system is trained on a different speaker or speakers, and the acoustic-phonetic models that are generated are very speaker specific [8, 9]. Researchers have shown that the error rate in speaker independent systems is almost three times that of an equivalent speaker dependent system [10].

2.1.2. Isolated, Discontinuous or Continuous Speech

The performance of speech recognition systems vary greatly depending on what type of speech the system is presented with. Isolated speech is considered as single words and is thus better considered as word recognition. In discontinuous speech, each word of a sentence is spoken clearly however they are artificially separated by a distinct silence.

Continuous speech refers to naturally spoken sentences with no artificial silences inserted to make each word distinct. Isolated and discontinuous speech recognition is a much easier task as each word is pronounced clearly and the boundaries surrounding each word are easily distinguishable [8].

2.1.3. Recognition Type

The preferred recognition type for ASR systems is word recognition. The resulting transcription of the system, assuming it is correct, can be easily read and understood by the user. The inherent problem with this system is, when a large vocabulary is concerned, which is likely in almost any real world application utilising ASR, the system becomes very complex as it requires models for each word.

A simpler form of recognition is to consider the components that form the words including, but not limited to, phonemes, bi-phones and tri-phones [11]. In the English language there are approximately 40 phonemes that exist and are shown below in Table 2-1 [12]. Examples of how individual phonemes make up both words and phrases are shown in Table 2-2 [12, 13].

Table 2-1: List of English Phonemes and Manners of Articulation

Phoneme	Manner of Articulation	Phoneme	Manner of Articulation
iy	vowel	l	liquid
ih	vowel	r	liquid
ia	vowel	m	nasal
ey	vowel	n	nasal
eh	vowel	ng	nasal
ae	vowel	f	fricative
ea	vowel	v	fricative
aa	vowel	th	fricative
ao	vowel	dh	fricative
ow	vowel	s	fricative
uh	vowel	z	fricative
uw	vowel	sh	fricative
ua	vowel	zh	fricative
ah	vowel	hh	fricative
er	vowel	p	stop
ax	vowel	b	stop
ay	diphthong	t	stop
oy	diphthong	d	stop
oh	diphthong	k	stop
aw	diphthong	g	stop
y	glide	ch	affricate
w	glide	jh	affricate

Table 2-2: Examples of Phonetic Decomposition of Words/Phrases

Word/Phrase	Phonetic Decomposition
call	k ao l
dial	d ay ax l
seven	s eh v n
recognise speech	r eh k ao g n ay z s p iy ch
wreck a nice beach	r eh k ay n ay s b iy ch

Training time of the recognition models is also significantly reduced with phoneme recognition when compared to word recognition. This is due to phoneme recognition only requiring a recognition model for each of the phonemes shown above in Table 2-1. The inherent flaw with phoneme recognition is that it produces a significantly lower recognition rate when compared to that of word recognition. This can be attributed to a reduced level of confusion due to the tied states [12].

2.2 Feature Vectors

In speech recognition, feature vectors (or features) are the specific measurable parameters that represent the input speech waveform. The selection of both unique and discriminatory features is mandatory to the success of the speech recognition algorithm in classification, discussed below [3]. Due to the limited amount of time available, this literature review will only consider feature vectors comprised of Mel-frequency Cepstral Coefficients. However, many other feature vectors are available; see [14].

2.2.1. Mel-frequency Cepstral Coefficients (MFCC)

The Mel-frequency cepstrum (MFC) is used in sound processing to represent the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on the nonlinear mel scale of frequency [3].

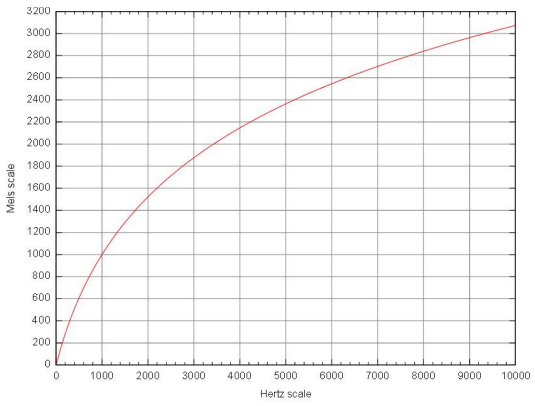


Figure 2-1: Mel scale vs Hertz scale [3]

The mel scale seen above in Figure 2-1 was proposed in 1937 and consists of a scale of pitches an equal distance apart as deemed by listeners. The reference point between the mel scale and the hertz scale is defined by equating a 1000 Hz tone which is 40 dB above the threshold of the listener, to a pitch of 1000 mels as can be seen in Figure 2-1. From 500Hz and higher, the intervals are estimated by listeners to result in an equivalent increment in the pitch. The result of this is four octaves greater than 500 Hz on the Hertz scale is estimated to consist of two octaves on the mel scale. [3] The formula to convert between *f* Hz and *m* mels is as follows [3]:

$$m = 2595 \log_{10}\left(\frac{f}{700} + 1\right) \tag{1}$$

Mel-frequency Cepstral Coefficients are what jointly make up an MFC. The difference between an MFC and a normal cepstrum is such that the MFC frequency bands are equally spaced using the mel scale which is a much better approximation of the human auditory system's response than the linearly-spaced frequency bands utilised in a normal cepstrum [3].

The process for extracting the features comprised of Mel-frequency Cepstral Coefficients from the input speech waveform is shown below in Figure 2-2. For the relevant equations at each processing block, see [15].

Figure 2-2: Mel-frequency Cepstral Coefficients Extraction Process [14]

2.3 Classifiers

Classification is a large application of Digital Signal Processing used in many fields including, not unexpectedly, speech recognition [4, 5, 16-18]. In terms of classification within speech recognition, the largest issue is not the choice of classifier used in the implementation but rather the quality of the feature vectors that are extracted for use as input to the classifier. Without a reliable input, the classifier will likely operate much poorer than expected [19]. Due to the limited amount of time available, this literature review will only consider the three main classifiers used in speech recognition: Dynamic Time Warping, Hidden Markov Models and Artificial Neural Networks.

2.3.1. Dynamic Time Warping

Since the advent of the Hidden Markov Model (HMM), Dynamic Time Warping (DTW) has become a less popular classifier for speech recognition [20]. It uses patterns considered as templates to represent each word in the vocabulary of the ASR system. The recognition process simply involves the comparison of the input speech to the stored templates to find a match. This process is shown below in Figure 2-3 [18].

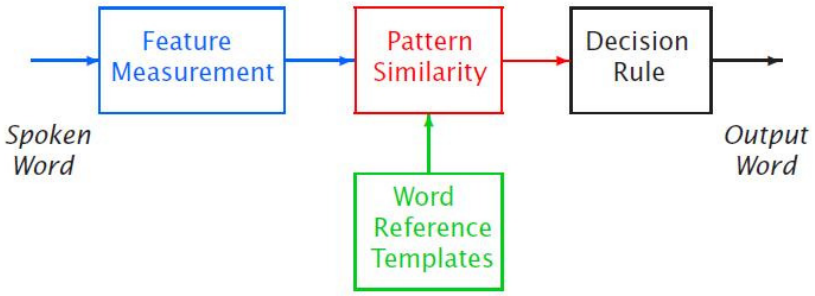


Figure 2-3: DTW Word Recognition Process [21]

Dynamic Time Warping is derived from the class of algorithms called dynamic programming. The time and space complexity of DTW is linear for the duration of the speech sample and the vocabulary size. If we consider an unknown speech sample, the DTW algorithm will make a single pass through a matrix of frame scores while simultaneously calculating optimized components of the global alignment path [8].

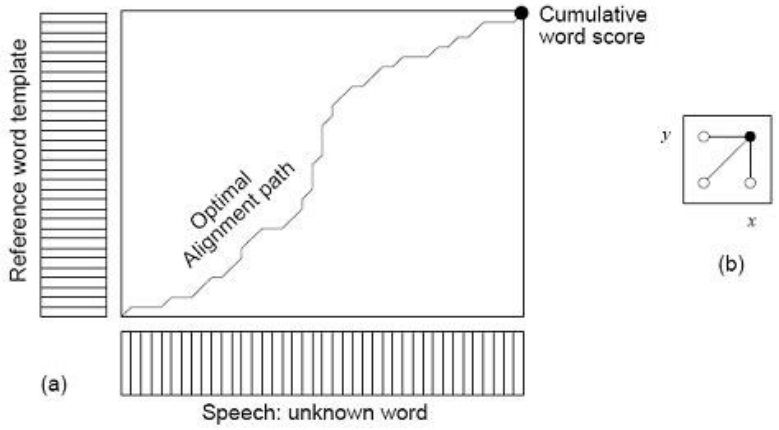


Figure 2-4: DTW Algorithm Sample [8]

If we define D(x,y) as the Euclidean distance separating frame x of the unknown speech sample and frame y of the reference word template, and C(x,y) is the cumulative score along an optimum alignment path leading to point (x,y) then:

$$C(x, y) = MIN(C(x - 1,y), C(x - 1,y - 1), C(x, y - 1)) + D(x, y)\ [8] \quad (2)$$

Going through each available reference word template, a optimal alignment path is computed and the best match to the unknown speech sample is the template with the lowest cumulative word score [8].

The inherent flaws of DTW include it being $O(N^2V)$ (where N is the length of the sequence and V is the number of reference word templates being considered) which is considerably slow, along with a distance metric (in the previous sample, Euclidean) having to be defined between frames which can often be problematic [20].

2.3.2. Hidden Markov Models

The Hidden Markov Model (HMM) is defined as a statistical model that is considered a Markov process however the states are unobserved (hidden). The transition between states has an associated probability and is defined in the transition probability matrix A; an *NxN* matrix for a model with *N* states where a_{ij} is the probability of transitioning from state *i* at time *t* to state *j* at time *(t+1)*. Additionally the model also defines an observation probability matrix B; an *NxM* matrix defining the probability of each of the *M* observations occurring in each of the *N* states of the model where B = [$b_j(k)$] and $b_j(k)$ is the probability of observing observation *k* at time *t* while being in state *j* at time *t* . Unlike a regular Markov model, the HMM doesn't have fixed probabilities for the observation probability matrix but instead has a probability density function (PDF) describing the likelihood of an observation being observed. The final component of the model is the initial probability matrix π; which is a *1xN* matrix where π_i defines the probability that the model initially starts in state *i* at time *t = 0*. Thus a HMM model, λ is defined $\lambda = (A,B,\pi)$ which is the compact notation for a HMM definition [3, 22-24].

When considering speech recognition, each phoneme/word is represented by an individual HMM. A left-right topology is implemented for speech recognition whereby transition between states is limited to the state itself or the state to the right. Before a HMM can be used to recognise input speech, the models must first be trained. Information contained in the feature vectors provides the characteristics needed to define the model along with Baum-Welch (or backwards-forwards) algorithm. The speech recognition is achieved via the Viterbi algorithm which determines the most likely sequence path which would have produced the observed state sequence [3, 12, 24, 25].

2.3.3. Artificial Neural Networks

Artificial Neural Networks (ANN) are a mathematical model implemented to simulate the function of Biological Neural Networks that exist in the human brain. An ANN is comprised of interconnecting artificial neurons which model the neurons in the human brain. Each artificial neuron receives a number of inputs and produces the resulting output (analogous with a synapse from the human nervous system) as a sum of the inputs. Generally the sums at each node are weighted with each sum passing through a non-linear transfer function which usually have a sigmoid shape but can also exist in the form of piecewise linear functions or step functions, both nonlinear. Consider the McCulloch-Pitts model of the artificial neuron below in Figure 2-5 [3]:

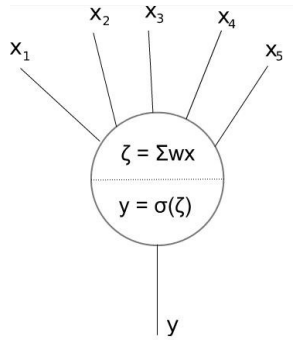


Figure 2-5: McCulloch-Pitts model of an artificial neuron [3]

In the above model, X_i are the inputs to the artificial neuron, ζ is the weighted sum of the inputs, w represents the relative weight of each input and the output y is a function of the weighted sum. The vector w is adjusted accordingly through the implementation of various training algorithms, which in the case of speech recognition is the back-propagation algorithm. Speech is provided to the input layer (see Figure 2-6) and compared to the transcriptions at the output layer. The weight vector within the hidden layer is then adjusted accordingly to produce the lowest possible error. Thus this presents an inherent weakness in ANN in that it requires a substantial amount of training data to best optimise the weight vector. The benefit of this enhanced training regime is that it makes ANNs proficient at dealing with noisy, incomplete or incorrect data making them very robust. The type of network in Figure 2-6 is a Feedforward Neural Network which only allows the data to move in a single direction making it ideal for the speech recognition process [3, 12, 26].

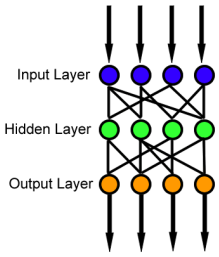


Figure 2-6: Feedfoward Neural Network Model [3]

Depending on the way the ANN learns, it can be considered either static or dynamic. The difference between the two is that a static ANN implementation has distinct and separate training and production stages whereas a dynamic implementation continues to learn during production [3, 8] .

2.4 Robust Speech Recognition Methods

The methods for robust speech recognition generally fall into one of four categories (or a combination of two or more):

- Extracting robust feature vectors in the front-end, see [27]
- Converting noisy speech features to clean speech features, see [28]
- Adapting references towards the current environment, see [17]
- Using noisy and/or distorted references during training, see [29]

Due to the limited amount of time available for this thesis only the auditory front-end method will be considered in this literature review.

2.4.1. Auditory Modelling

An auditory model based front-end aims to emulate the functionality of the human auditory system. The way the human ear operates (see [6, 7]) allows for much more robust distinction between speech and noise and would thus make an ideal model for enhancing current speech recognition systems in noisy environments.

Seneff's Auditory Model

The initial research into front-end auditory modelling came from physiological data and was published by Seneff. His design implemented an initial stage comprising of 40 linear filters, followed by a series of nonlinearities used to model the conversion between basilar membrane motion and auditory nerve stimulation. Among the nonlinearities were soft half-wave rectifiers, a model for short-term adaptation and a rapid Automatic Gain Control (AGC) [30-32].

This front end comprised of two outputs. One type of output, labelled "Mean rate", is produced via envelope detection of the nonlinear stage output. These outputs are a rough estimate of the spectral magnitude. The other type of output, labelled "Synchrony", is used to detect whether the nonlinear stage output of a given channel contains energy at the central frequency for that channel [30].

Ghitza's Auditory Model

Another auditory model front-end is the Ensemble Interval Histogram (EIH) which was developed by Ghitza. The first stage of the EIH model is similar to Seneff's however it contains 133 filters compared to 40 in Seneff's. During the second stage of the EIH model, outputs from each of the filters are taken to compute the intervals between the positive crossings of the filtered waveforms at various logarithmically spaced thresholds. A histogram based on the frequencies of these intervals can then be generated before the final stage where the histograms from each channel are combined to a single final output; the EIH. In this regard, the EIH model has similar functionality to Seneff's "Synchrony" output. Experiments have shown that the EIH model performs exceptionally well at isolated word recognition in low SNR [30, 33].

A block diagram of the auditory model front-end can be found in Appendix A. For information regarding the processing at each block, see [34].

2.5 Training and Testing Data

Given the short amount of time available for completion of this thesis, it is unfeasible to manually record the large amount of training data, testing data and noise along with producing the associated phoneme and word transcriptions of each recorded utterance.

This issue can be overcome through the use of one of the many available speech corpora. Speech corpora generally contain a large number of speakers of both genders whom have been recorded reciting sentences in a controlled environment. Two of the most common corpora, TIMIT and Wall Street Journal (WSJCAM0) are reviewed to determine their suitability.

2.5.1. TIMIT Acoustic-Phonetic Continuous Speech Corpus

The TIMIT Acoustic-Phonetic Continuous Speech Corpus was recorded by Texas Instruments (TI) before the Masschusetts Institute of Technology (MIT), SRI produced the associated transcriptions [35].

The corpus contains a total of 630 speakers each reciting 10 sentences and coming from one of the eight major dialect regions of the United States. The dialect region refers to the area in the United States where the speaker grew up as a child however one of the dialect regions indicates the speaker frequently moved during their childhood [35].

2.5.2. WSJCAM0 Cambridge Read News

The Wall Street Journal Cambridge Read News Corpus was recorded at, as the name suggests, the Cambridge University. It is the UK English equivalent of a subset of the US American English corpus, WSJ0 [36].

The corpus is made up of speaker-independent (SI) read material which is broken into three main areas: training, development testing and evaluation testing. In the training set, there are a total of 92 speakers of both genders each reciting 90 utterances. For the testing material, a total of 48 speakers were recorded reciting 40 sentence utterances of which consisted of only words from a fixed 5,000 word vocabulary and an additional 40 sentence utterances formed from the 64,000 word vocabulary. Furthermore, a common set of 18 sentences were recorded by each of the 140 total speakers. It's also important to note that with this corpus, the recording was actually done using two microphones: a far-field desk microphone along with a close-talking microphone [36].

2.6 Summary

After completing a thorough literature review on relevant literature pertaining to the theory of speech recognition, the work of this thesis has been put into perspective. A lot of research and experimentation has been conducted in the field of robust speech recognition; however as with all fields in science and technology there are missing components.

While there has been some testing with an auditory model based front-end for robust speech recognition, the discovered research which was covered in this literature review generally only consisted of the basic isolated word recognition. For its time, isolated word recognition was a big step forward for speech recognition. However in the 21st century as we are now, the demand for robust continuous speech recognition is forever growing. This is effectively why this thesis aims to determine the performance of an auditory model based front-end for robust speech recognition on continuous speech using phoneme recognition.

3. Methodology

This section of the thesis deals with the methodology used to implement the solution to the proposed problem. Justification of the decisions made leading to the solution of the problem is provided along with the system specifications that will be used for experimenting.

3.1 Justification of Methods

3.1.1. System Specifications

Due to the heavy reliability on training, it was decided that a speaker dependent system would not be implemented instead opting for the more robust speaker independent system. Previous research has already been conducted using a speaker independent, isolated word recognition system [34]. Continuous speech was chosen due to there being little to no research or experimentation on an auditory model front-end for robust speech recognition using continuous speech. The chosen recognition type is phoneme recognition for its comparative ease in training and testing and significantly reduced model training time. Although the recognition rate will be quite low, the main focus of this thesis is an improvement on the recognition rate rather than simple producing a high recognition rate.

3.1.2. Choice of Classifier

After a thorough literature review on the three most common classifiers used in speech recognition, it was decided that the Hidden Markov Model would be the most appropriate for the proposed problem. Because of its connective nature, HMM lends itself to being efficient for continuous speech recognition more so than the similarly connective ANN classifier. Additionally, from most of the reviewed literature it became evident that HMM was the most widely used classifier among the speech recognition community thus by implementing it for this thesis will allow a direct comparison of results to previous research. Furthermore, the software and documentation (HTK) that are available for use with HMMs far exceeds that available for DTW or ANN.

3.1.3. Choice of Corpus

The Wall Street Journal Cambridge Read News (WSJCAM0) corpus was chosen as the most appropriate to be used for both training and testing in this thesis. It provides a large number of training and testing sentences composed from a very large vocabulary along with providing a good mix of speakers in terms of gender and age. Additionally, it comprises of speaker independent read material (continuous) which is a direct match to the system specifications of this thesis. Furthermore, while the TIMIT corpus would also be a feasible

option, budget and availability were also considered making the WSJCAM0 the superior choice as it was less expensive and available almost instantaneously.

3.2 Work to Be Done

Due to the repetitive nature of the tests, the experimenting process will be largely automated through the use of Perl [37] scripts to run commands from the Hidden Markov Model Toolkit (HTK) [11]. The mathematics laboratory software MATLAB® will also be used for the insertion of random white noise and processing of the sound files for the auditory model front-end.

3.2.1. Benchmark

Initially, a benchmark or baseline result will be generated using a (MFCC) front-end which is recognized as the standard for feature vectors by the speech recognition community and are generally utilized in most speech recognition research experiments. *(eg.* [14, 16, 27, 38]*)* Characterisation of its performance will be carried out to gauge the accuracy in a number of speech recognition tests on noisy speech, clean speech and at different signal-to-noise (SNR) ratios.

3.2.2. Auditory Modelling

An auditory model front-end will then be implemented with the use of MATLAB® and be subjected to the same tests as the MFCC test bench to form a basis for comparison and determine whether the auditory model front-end outperforms the standard MFCC implementation.

4. Benchmark

This section of the thesis will discuss in detail the test bench that was implemented using the Hidden Markov Model Toolkit (HTK) and Mel-frequency Ceptral Coefficients (MFCC). As previously stated the purpose of the test bench is to obtain what can be considered an average recognition rate at declining SNR when considering auditory modelling. This average will allow a direct comparison to identical recognition tests that will be performed using an auditory model front end developed in MATLAB® outlined in section 5.

4.1 Hidden Markov Model Toolkit (HTK)

The Hidden Markov Model Toolkit is comprised of a number of independent modules that when implemented can be used for developing HMM-based speech processing tools. Fundamentally, there are two major stages involved in the development as shown below in Figure 4-1.

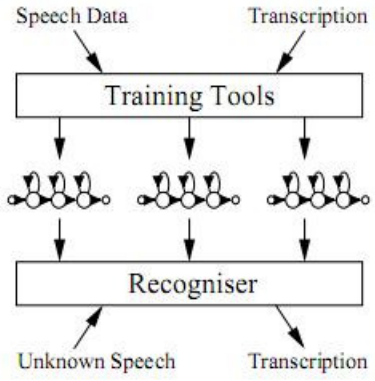


Figure 4-1: HTK Major Processing Stages [11]

Initially, both speech data and associated transcriptions are provided to HTK such that the HMM parameters can be trained. From there, unknown speech can then be parsed by the recogniser and transcriptions can be generated. The full process including data preparation, training, testing and analysis can be seen on the following page in Figure 4-2.

HTK is designed to perform best in a UNIX shell environment however as this was not possible for this thesis, the modules which are written in C++ were able to be compiled into executable form and run from the emulated DOS environment command prompt on a machine running Windows. Due to the large number of speech and transcription files that are required to develop adequate models for recognition, it would become very cumbersome, repetitive and time consuming to manually run the modules from the command line on each file. Through the use of the Perl scripts shown in Appendix C the entire process can be

automated by utilising an alternate command available on most HTK modules which is to run the module on each of the filenames listed in another file provided as an input parameter to the module. An example of such a file from the scripts in Appendix C is '*wav_file_list.scp'* which contains a list of all the .wav speech files contained in the *data* directory which are used for training.

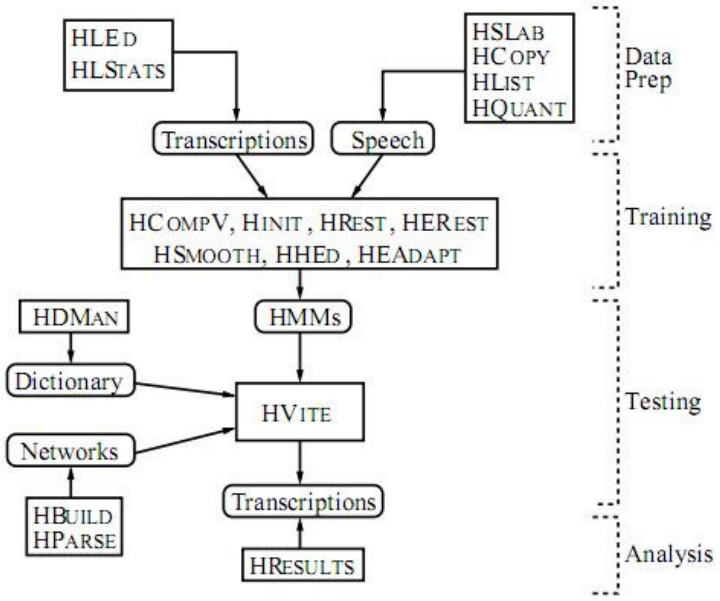


Figure 4-2: Full HTK Process Showing Modules [11]

With the WSJCAM0 corpus another initial step was required in the data preparation stage. The speech files provided by the corpus are in the SPHERE format with a *.wv1* extension which is unreadable by the modules of HTK. The *sph2pipe* conversion utility was used in conjunction with the Perl script *convert.pl* (see Appendix C) to convert all the *.wv1* files that would be used for both training and testing to *.wav* format before further data preparation steps could occur. Although all of the modules in the toolkit are able to parameterise source files into mel-frequency ceptral coefficients on the fly, it is computationally less expensive to perform this operation only once. The *HCopy* module was used to achieve this and as the name implies, it copies a source file and stores it in its parameterised form with a *.mfc* extension which refers to Mel-frequency Cepstral Coefficients. The *HList* module can then, if needed, be used to view the contents of the *.mfc* files.

The first step in training is creating flat start Hidden Markov Models which creates identical models initially for all the phonemes. The module *HCompV* was used for this. Once the initial models are defined, the module *HERest* was used several times to perform re-estimation of the model's parameters based on the supplied training data. After three re-

estimations, the *HHed* module was used to add extra transition states to the silence (sil) model and to tie the short pause (sp) state to the centre sil state as shown in Figure 4-3.

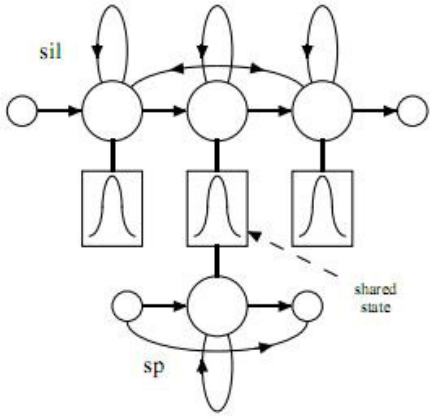


Figure 4-3: Short Pause State Tied to Centre Silence State [11]

Finally, *HER*est is used to re-estimate the model's parameters four more times to improve the recognition model for each phoneme. HTK has the ability to continue on from this stage and produce Tied-State Tri-Phones however this is beyond the scope of this thesis. Although better recognition models will improve the recognition rate, the main goal of this thesis is to improve the recognition rate and thus the basic monophone models will be sufficient and standard across all tests.

Once the complete set of HMM models for each phoneme had been trained, the testing process could begin. As with training, the first step was preparing the data by first using the *sph2pipe* conversion utility to convert all the SPHERE *.wv1* files in the evaluation directory to *.wav* before running *HCopy* to parameterise the input files. The module *HVite* could then be run on the evaluation data which used the Viterbi algorithm and produced the transcriptions of the input files. The final step was to run the *HResults* module on the output file from *HVite* which compared the produced transcriptions to the correct transcriptions. The module provided two scores - percent correct which is based on the ratio of the correctly recognised phonemes and the total number of phonemes in the testing set; and percent accuracy which penalises the score for any incorrectly inserted phonemes. Additionally, confusion matrices (See Appendix B) can also be produced which provide further information on the recognition rates of individual phonemes from the testing set. For the purpose of this thesis, only the overall percent correct and percent correct for individual phonemes were considered as it is a simple score and provides a clear indication if there was any improvement.

4.2 Training and Testing Data

As has been previously stated, the WSJCAM0 corpus will be used for training and testing purposes in this thesis. A sufficient amount of data is required to produce reasonably performing recognition models however an increase of data results in increased computational time for each process. The right balance needed to be struck between sufficiency and computational time and it was decided that with approximately 100 utterances each, that 20 speakers would be ideal for both training and testing. The WSJCAM0 corpus breaks the data down into distinct groupings for training and testing. Twenty speakers were chosen from Discs 1 and 2 of the corpus which were allocated for training and recorded with the primary microphone. A further 20 were then chosen from Disc 3 which was allocated to evaluation and recorded with the primary microphone. The chosen speakers are shown below in Table 4-1.

Table 4-1: List of Speaker IDs Chosen for Training and Evaluation

Training – Disc 1	Training – Disc 2	Evaluation – Disc 3	Evaluation – Disc 3
C0A	C1C	C3C	C3B
C0B	C1D	C3D	C3H
C0C	C1E	C3F	C3O
C0D	C1F	C3J	C3Q
C0E	C1G	C3K	C3R
C19	C29	C49	C48
C18	C28	C45	C46
C17	C27	C42	C39
C16	C26	C41	C37
C15	C25	C40	C33

4.3 Noise Addition

A method for the addition of noise to a large number of input speech files at a chosen SNR was also required. This was achieved through the use of a MATLAB® script (See Appendix D). Each input speech file was loaded in one at a time and the signal power in RMS for an input signal *s* was calculated through the use of equation (3) below.

$$S_{RMS} = \sqrt{\frac{1}{T}\int_0^T s^2(t)dt} \tag{3}$$

With the calculated signal power and the required SNR, the amount of noise power to obtain the required SNR was calculated via equation (4) below.

$$N_{RMS} = \frac{S_{RMS}}{10^{SNR/20}} \quad (4)$$

The MATLAB® function *randn* was then used to generate a random array *noise* of Gaussian distribution the length of the signal. By dividing the required noise power (N_{RMS}) by the RMS power of the generated array, a set of gains *k* were found which could then be used to multiply by the randomly generated noise array to adjust its power to that of the required noise power, N_{RMS}. The new signal at the desired SNR was then simply the addition of the original signal *s* and the adjusted array *noise*.

The reconstructed speech files were then written to an alternate directory that indicated the SNR in the directory name to avoid confusion as the files maintained the same name to allow for easier processing. The effects of the noise addition at varying SNR on the same file is shown below in the Figures 4-4 to 4-9. In each case, the blue plot represents the file with the added noise and the red plot shows the same unaltered file to allow for an easy distinction between the two.

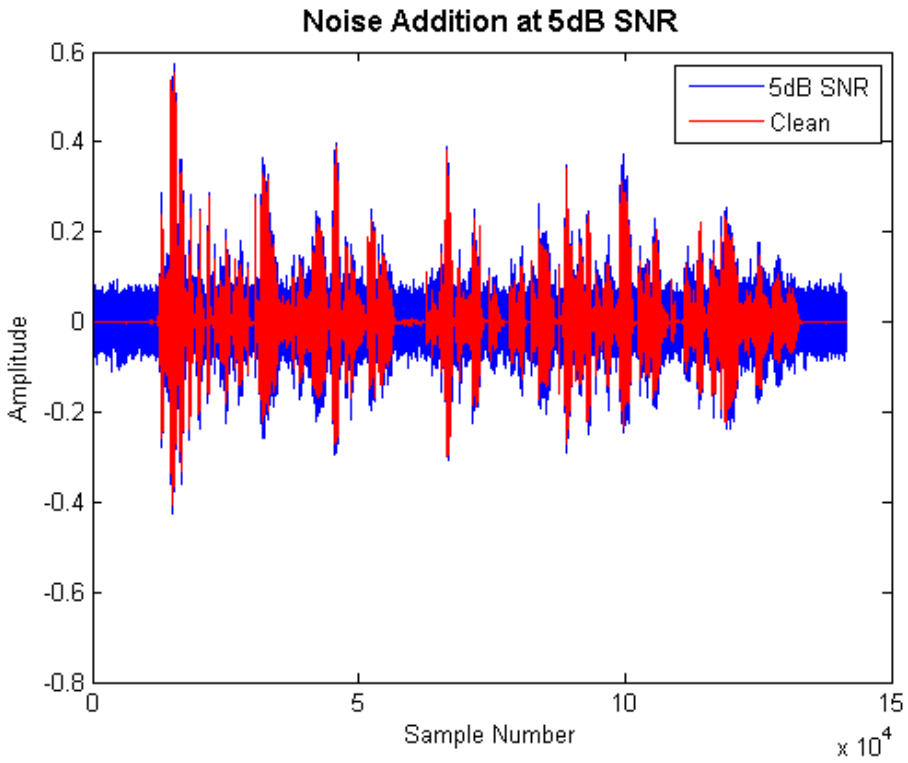


Figure 4-4: Noise Addition at 5dB SNR

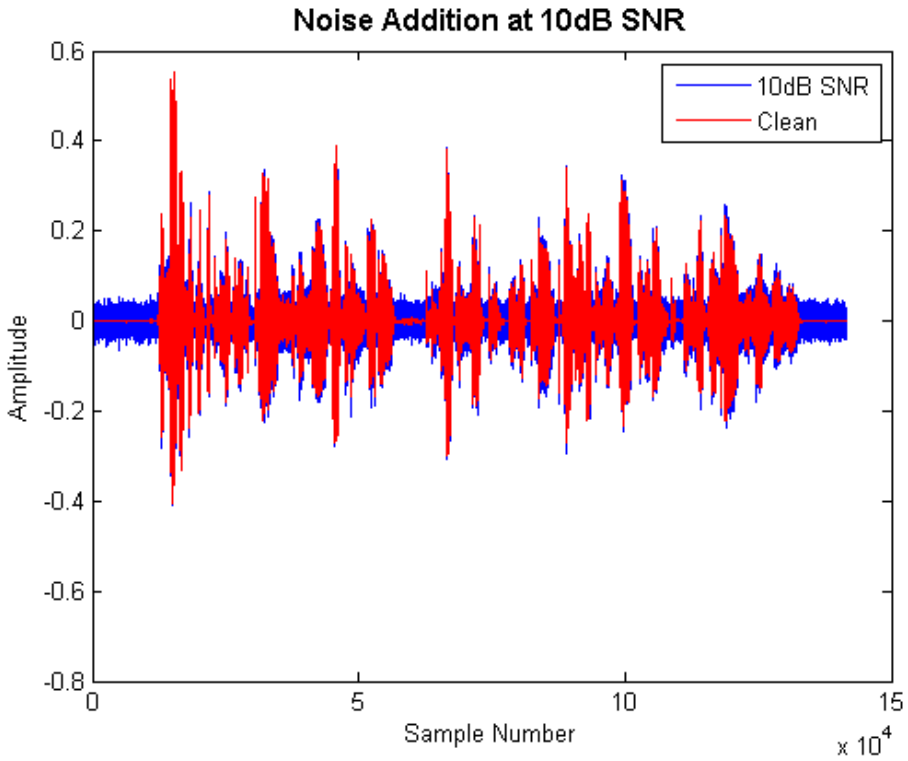


Figure 4-5: Noise Addition at 10dB SNR

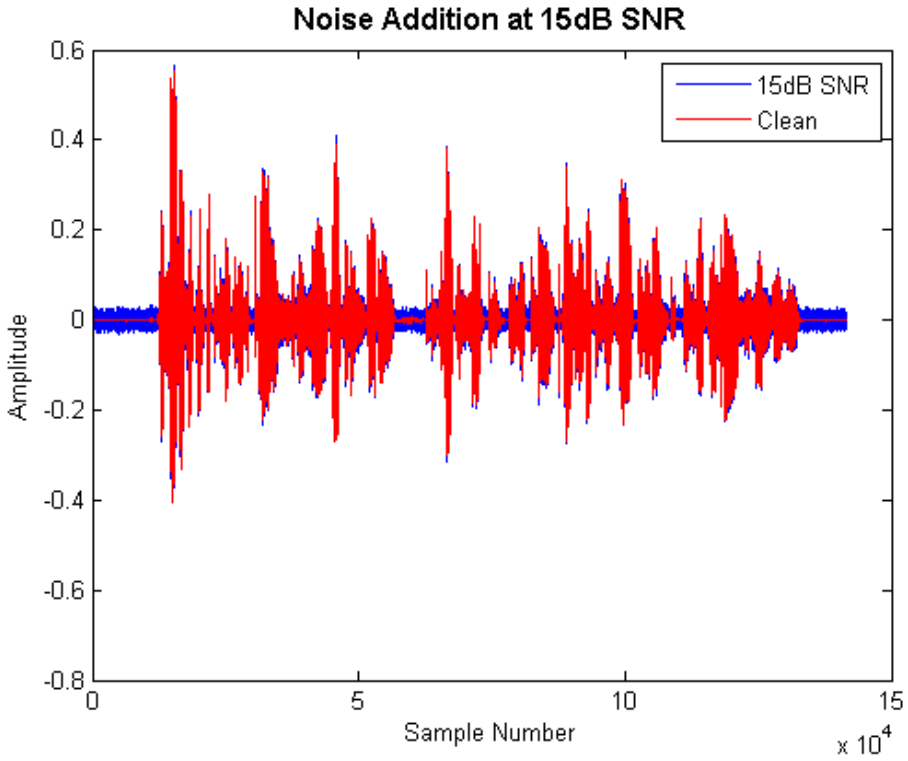


Figure 4-6: Noise Addition at 15dB SNR

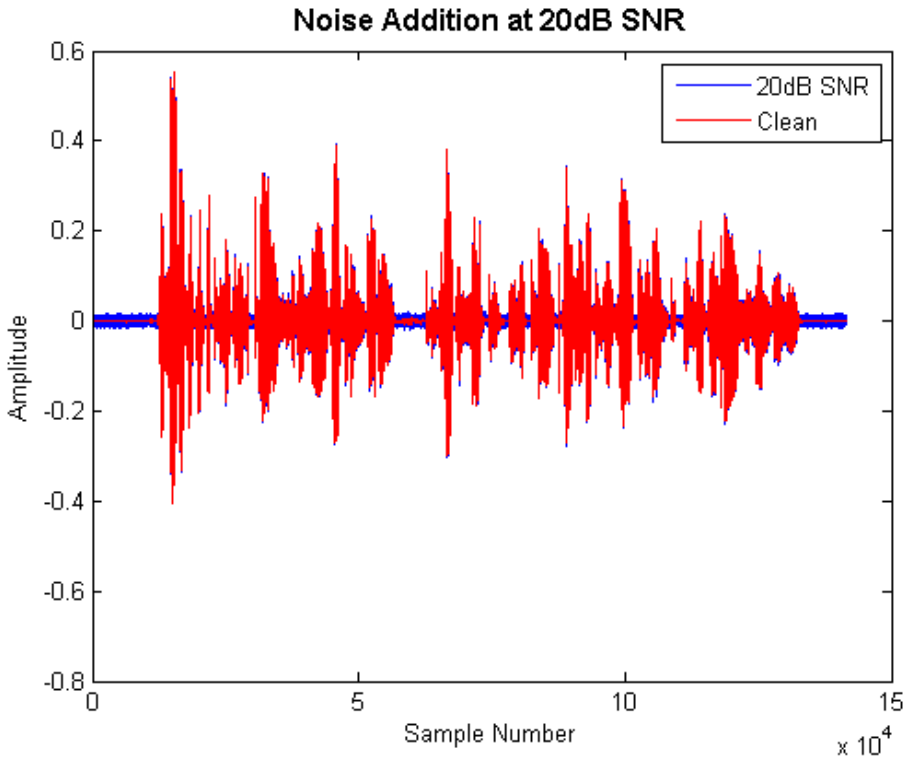


Figure 4-7: Noise Addition at 20dB SNR

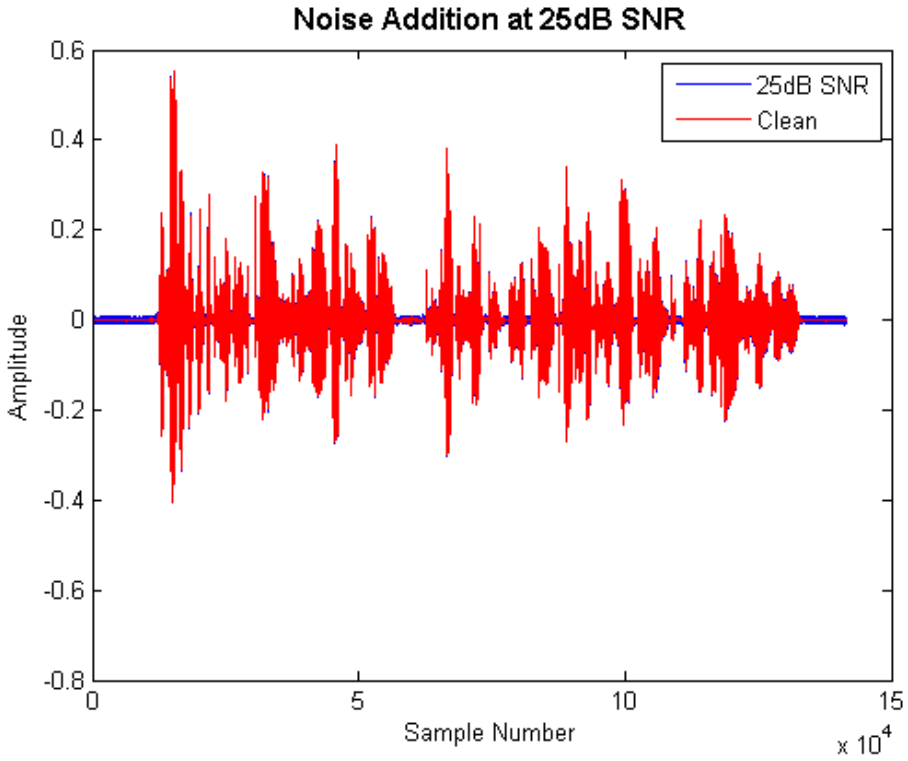


Figure 4-8: Noise Addition at 25dB SNR

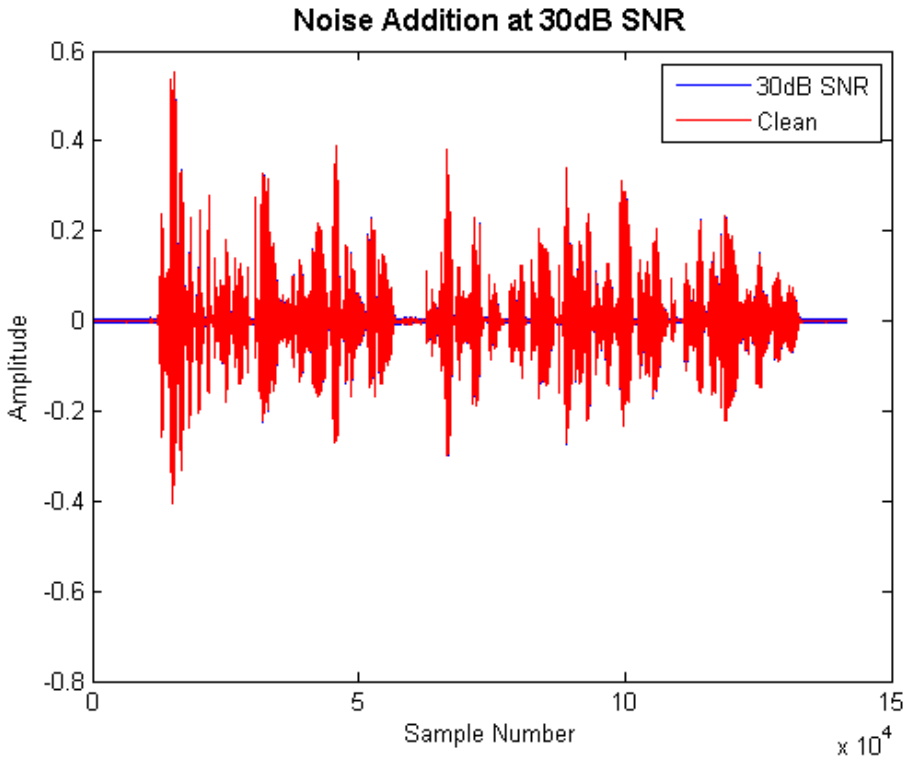


Figure 4-9: Noise Addition at 30dB SNR

The additional noise is most obvious in the regions that contain no speech where the amplitude of the red signal is approximately 0 however it can also be seen that there is additional noise on the speech itself. When listening to the files, the additional noise becomes very distinct audibly, particularly at a low SNR such as 5 dB..

4.4 Testing Methodology

The first test was conducted using the unaltered evaluation data which can be considered as clean speech with a very high SNR. The Perl script *evaluation.pl* shown in Appendix C was used to prepare the testing data, run the recognition test and store the results in an output text file as outlined in section 4.1. The result of this recognition test set the upper limit for the recognition rate of the thesis as it was hypothesised that a decreasing SNR would lower the recognition rate due to the increase in the noise power and further deviation from the original speech statistics that the recognition HMMs were trained on.

The next step involved adding noise at decreasing SNR starting at 30dB and decreasing in 5dB intervals down to 5dB SNR. Ideally, a larger sample of SNR would have been examined however due to time constraints only 6 different SNR's could be included for testing. Noise was then added to all evaluation files at all specified SNR by running the MATLAB® script file outlined in section 4.3 and shown in Appendix D.

The *evaluation.pl* Perl script shown in Appendix C was modified to use the speech files from each different directory containing the evaluation files at varying SNR. To avoid including mostly similar scripts, only the original *evaluation.pl* was included in the Appendix. Using the same trained HMMs, identical recognition tests as for clean speech were performed for all SNR. The following two sections will show the obtained results from all recognition tests and discuss what can be taken from them.

4.5 Benchmark Results

Figure 4-10 below shows the overall percent correct recognition rate for each of the recognition tests on clean speech and at declining SNR.

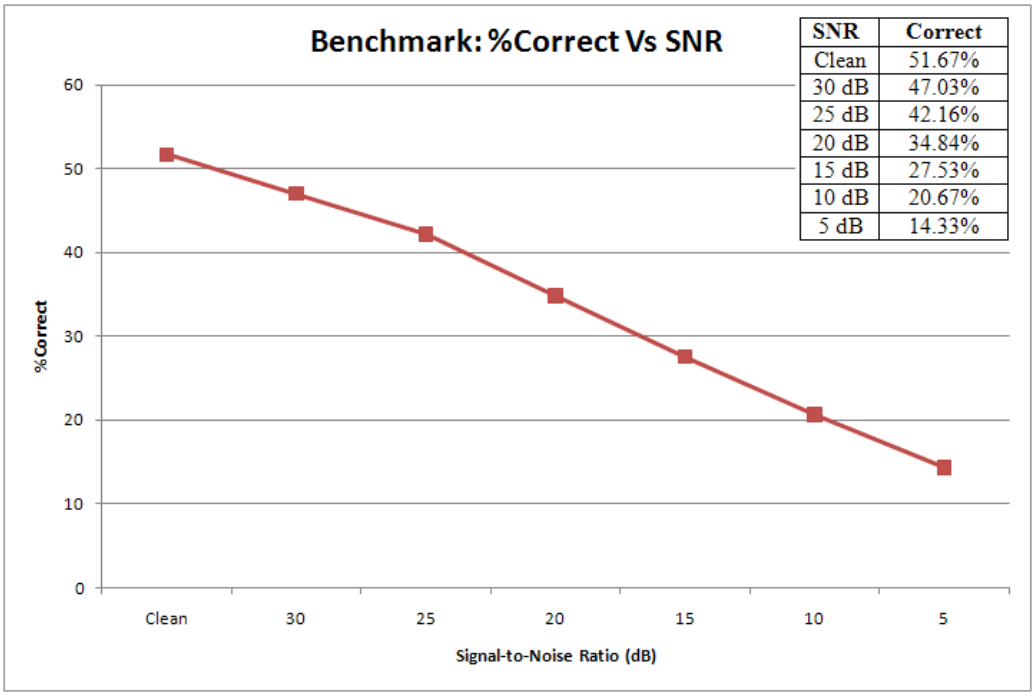


Figure 4-10: Recognition Rate (%Correct) Vs SNR

While overall results are useful at providing a quick insight into the performance of the speech recognition system, considering the recognition rate of individual phonemes can provide further information into which group of phonemes have the greatest reduction in recognition rate at declining SNR. The results provided in the following tables are extracted from the full confusion matrices in Appendix B which provide detailed information on the recognition rate and errors for each included phoneme.

Table 4-2: Recognition Rate of "Vowel" Phonemes

Phoneme	SNR						
	Clean	30dB	25dB	20dB	15dB	10dB	5dB
iy	66.6%	67.6%	65.8%	62.0%	49.9%	33.7%	15.8%
ih	46.4%	45.6%	43.8%	40.8%	37.0%	33.8%	31.7%
ia	58.2%	65.9%	67.1%	73.6%	77.3%	72.7%	56.3%
ey	65.1%	66.5%	67.8%	68.2%	66.4%	62.0%	53.5%
eh	52.1%	52.1%	52.2%	52.0%	51.3%	49.8%	52.0%
ae	33.1%	31.2%	31.4%	30.2%	29.2%	30.2%	33.0%
ea	57.7%	63.4%	67.2%	73.2%	77.7%	81.6%	90.1%
aa	63.4%	67.4%	68.8%	68.8%	64.1%	45.0%	18.8%
ao	69.8%	71.1%	67.1%	49.9%	19.9%	3.5%	0.1%
ow	30.5%	25.6%	22.8%	18.8%	16.9%	15.4%	16.3%
uh	48.0%	34.0%	26.3%	22.1%	18.7%	16.4%	25.4%
uw	41.3%	36.4%	34.3%	30.2%	27.8%	19.6%	15.9%
ua	60.3%	56.7%	56.3%	55.2%	44.5%	34.4%	28.2%
ah	43.6%	40.3%	37.9%	34.8%	29.9%	22.4%	14.6%
er	35.4%	38.5%	40.0%	41.5%	42.9%	48.4%	60.1%
ax	38.5%	36.0%	34.4%	32.7%	30.7%	28.8%	30.2%

Table 4-3: Recognition Rate of "Diphthong" Phonemes

Phoneme	SNR						
	Clean	30dB	25dB	20dB	15dB	10dB	5dB
ay	68.2%	70.8%	70.7%	68.1%	62.3%	51.3%	37.6%
oy	81.7%	84.7%	82.1%	74.6%	55.1%	30.2%	4.7%
oh	50.0%	46.0%	42.4%	37.9%	29.5%	19.5%	8.9%
aw	58.1%	60.9%	61.7%	63.2%	61.9%	60.1%	54.3%

Table 4-4: Recognition Rate of "Glide" Phonemes

Phoneme	SNR						
	Clean	30dB	25dB	20dB	15dB	10dB	5dB
y	60.0%	57.5%	55.8%	51.9%	43.5%	31.2%	20.9%
w	66.6%	53.7%	43.5%	27.1%	11.4%	2.4%	0.3%

Table 4-5: Recognition Rate of "Liquid" Phonemes

Phoneme	SNR						
	Clean	30dB	25dB	20dB	15dB	10dB	5dB
l	38.8%	33.3%	31.5%	30.6%	25.9%	18.2%	10.9%
r	55.6%	45.0%	36.4%	25.7%	15.6%	8.7%	3.8%

Table 4-6: Recognition Rate of "Nasal" Phonemes

Phoneme	SNR						
	Clean	30dB	25dB	20dB	15dB	10dB	5dB
m	73.0%	72.2%	68.1%	55.8%	33.2%	11.0%	1.9%
n	59.8%	54.6%	45.4%	26.4%	8.4%	1.1%	0.1%
ng	66.7%	66.2%	57.4%	34.6%	12.6%	0.9%	0.0%

Table 4-7: Recognition Rate of "Fricative" Phonemes

Phoneme	SNR						
	Clean	30dB	25dB	20dB	15dB	10dB	5dB
f	84.3%	89.7%	94.7%	97.9%	98.9%	99.5%	99.8%
v	70.8%	81.7%	84.8%	86.1%	83.9%	75.3%	57.3%
th	47.2%	51.7%	47.4%	33.9%	23.1%	16.6%	20.7%
dh	45.3%	32.5%	20.4%	9.4%	5.3%	4.9%	6.5%
s	76.7%	77.6%	78.6%	79.4%	78.0%	69.2%	48.1%
z	75.4%	72.8%	68.8%	60.7%	48.0%	30.4%	11.9%
sh	80.8%	82.1%	81.7%	79.5%	76.0%	63.7%	35.2%
zh	88.0%	83.2%	87.2%	84.8%	80.2%	66.3%	40.8%
hh	80.0%	79.3%	76.0%	73.0%	66.7%	54.5%	50.6%

Table 4-8: Recognition Rate of "Stop" Phonemes

Phoneme	SNR						
	Clean	30dB	25dB	20dB	15dB	10dB	5dB
p	68.0%	34.1%	7.7%	1.5%	0.3%	0.0%	0.0%
b	69.7%	10.8%	0.6%	0.0%	0.0%	0.0%	0.0%
t	45.2%	37.9%	32.3%	22.7%	11.3%	3.6%	0.9%
d	54.3%	34.9%	22.1%	10.1%	3.5%	0.9%	0.3%
k	62.0%	46.8%	26.5%	9.0%	1.4%	0.1%	0.0%
g	67.6%	42.4%	24.1%	7.1%	1.6%	0.0%	0.0%

Table 4-9: Recognition Rate of "Affricate" Phonemes

Phoneme	SNR						
	Clean	30dB	25dB	20dB	15dB	10dB	5dB
ch	54.7%	60.8%	58.8%	58.7%	47.9%	33.9%	17.3%
jh	58.2%	53.9%	51.7%	45.7%	36.4%	17.3%	5.9%

4.6 Analysis

The results of the benchmark tests shown above in section 4.5 clearly show the negative effect that the addition of noise has on the recognition rate of a speech recognition system when considering the overall recognition rate of the system. It is important to note that the recognition rate across all SNR's is low as only basic monophone HMMs have been trained and no additional language or grammar models have been applied. The primary goal of this thesis is to provide an improvement on the recognition rate with the inclusion of noise and not just to provide a high recognition rate.

As was expected, the recognition rate is significantly reduced as the level of noise present is increased. At 5dB SNR, the recognition rate has reduced to just 27.7% of the recognition of clean speech rendering the system practically unusable for any real world application.

When looking at the tables for the groupings of individual phoneme recognition, a number of observations can be made. While most of the phonemes follow the notion that a decrease in the SNR results in a reduced recognition rate, some anomalies have appeared. For example, consider the 'eh' vowel phoneme in Table 4-3 above. The recognition rate for this phoneme across all SNR including clean speech varies only slightly between 49.8% and 52.2% which suggests that it is somewhat impervious to the effects of additive white noise. Based on a thorough analysis of all tables, this effect only occurs amongst vowels. The vowel phoneme 'ae' displays this particular characteristic with its recognition rate only varying between 29.2% and 33.1% across all SNR including clean speech.

Another phenomenon can be observed when considering the 'ea' vowel phoneme. On the clean speech test the recognition rate of this phoneme was 57.7% however as the SNR decreased, the recognition rate increased each time up to 90.1% at 5dB SNR. This goes against the preconceived idea that the addition of noise will reduce the recognition rate. This effect can likely be attributed to the recogniser inserting the recognition of this phoneme. This is likely to be caused by the noise altering the statistics of the speech that the recogniser uses to perform recognitions. This effect can also be seen on the 'er' vowel phoneme and the 'f' fricative phoneme.

A similar effect to that mentioned above can be seen with the 'aw' diphthong phoneme. Initially as the SNR is decreased, the recognition rate steadily increases from 58.1% on clean speech up to 63.2% at 20dB SNR. From this point however the recognition rate begins to decrease to 54.3% at 5dB SNR which is lower than the recognition rate on clean speech. Between clean and 20dB SNR, the insertion effect mentioned above is likely to be the cause of the increased recognition however beyond this it would appear that the noise is too severely altering the speech statistics used by the recogniser and the normal trend of an adverse effect on the recognition occurs.

When considering each of the groups of phonemes as a whole, it can be seen the Stop phonemes in Table 4-9 are most affected by the reduction in SNR. Incidentally, it is this group of phonemes that have the highest initial recognition rate with clean speech getting up to 88.0% recognition. However, at 5dB SNR the recognition rate of essentially all the phonemes in this group have decreased to 0.0%. Even at higher SNR the recognition rate was quickly diminishing whereas all other groups of phonemes retain some recognition at the lower SNRs. This is due to the fact that this group of phonemes are usually at a higher frequency and lower energy which is similar to the noise.

Many other observations could be made but this could be considered as a thesis in itself. Due to the randomness of the additive noise, a static effect of the recognition rate across all phonemes and all SNRs is very unlikely. With that being said, the majority of the phonemes display the expected characteristics of having a reduced recognition rate as the level of noise is increased.

Overall, the benchmark tests have provided useful insight into the recognition rate under varying levels of noise. Additionally, baseline recognition rates for each tested SNR have been established which will allow any improvements to the recognition rate by the auditory model to be easily comparable. Furthermore, when looking at the lower level of individual phoneme recognition, several exceptions have been discovered which will be closely monitored during the recognition tests performed using the auditory model front-end. Also, the grouping of phonemes into their manner of articulation has allowed easy identification of those groups which are least and most affected by the decreasing SNR. These results can be compared to the results from the auditory model front-end recognition tests to confirm whether these characteristics are altered by the auditory model.

5. Auditory Modelling

This section of the thesis will discuss in detail the front-end auditory model that was implemented using MATLAB®.

5.1 Auditory Model

Initially the proposed model was intended to extract its own set of features to be used for classification. However due to an issue with file formats in HTK this approach was not possible. The second approach was to enhance the speech files themselves and then pass them to HTK for MFCC parameterisation in the same manner as the original clean and noisy files in the benchmark tests. While this approach was computationally less efficient and cannot be implemented in real-time, it provides insight into the feasibility of such an auditory model for robust speech recognition.

The proposed model will follow the block diagram shown below with each step outlined in detail in the following subsections.

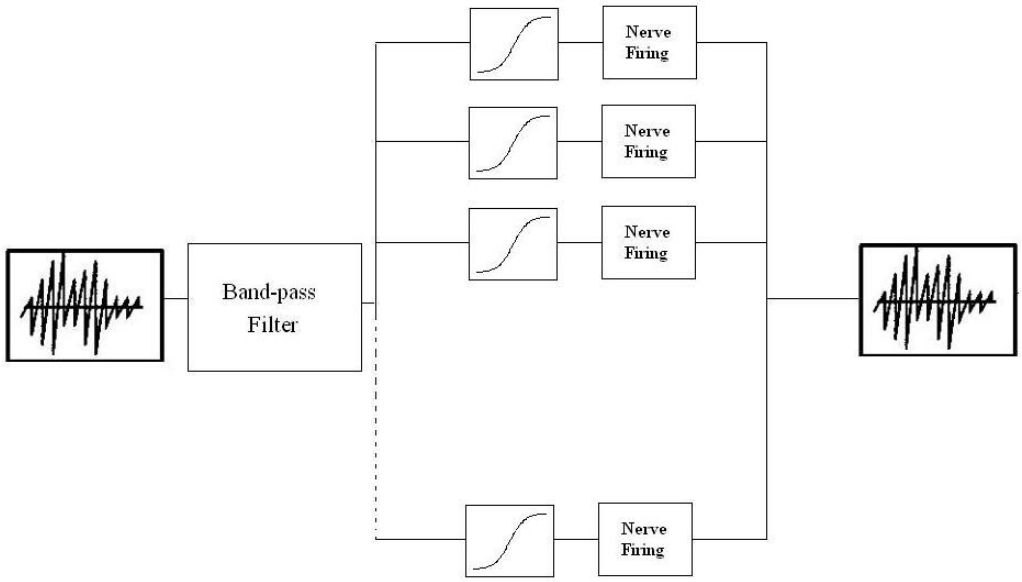


Figure 5-1: Proposed Front-end Auditory Model

The noisy signal is applied and passes through a linear filter bank of band-pass filters to split the signal into channels. Each channel then goes through the application of threshold via a sigmoid function and nerve firing. The channels are then recombined to form the enhanced signal which can then go through the same evaluation process as in the benchmark tests to establish the recognition rate after enhancement.

5.1.1. Filter Bank – Basilar Membrane

As with both Ghitza's and Seneff's models, the first stage of the proposed model shown above in Figure 5-1 is a filter bank. The filter bank used is a set of 16 linear band pass filters which cover 0 Hz up to the sampling frequency of the speech files, 16 kHz. This filter bank was used to emulate the mechanical displacement that is created by the basilar membrane. High frequencies are detected at the base of the basilar membrane at the opening of the cochlea and lower frequencies are detected at the apex at the top of the cochlea. A distortion is created in the basilar membrane when a frequency is found to be a match to the basilar frequency sensitivity at that point. This causes the inner hairs at this point to be displaced which results in the generation of chemicals and an electrical current to move to the auditory nerve endings.

5.1.2. Sigmoid – Threshold

A sigmoid function is used to control the gain in a similar manner to the threshold in the human ear. The shape of the sigmoid is influenced by the noise which shifts the threshold limit altering the sigmoid. The envelope of the input signal maps the auditory spike amplitudes which are then multiplied by the sigmoid. Smaller amplitudes correspond to noise and are attenuated the most. Larger amplitudes correspond to speech and are attenuated the least. A basic sigmoid is shown in Figure 5-2.

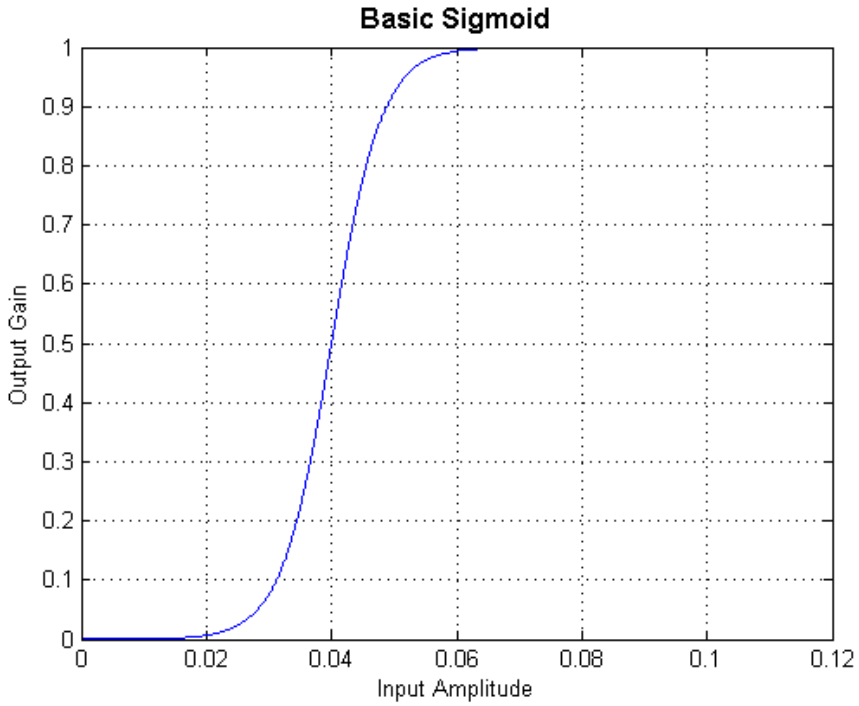


Figure 5-2: Example of a Basic Sigmoid Function

5.2 MATLAB® Implementation of Auditory Model

The Perl script *enhanceall.pl* shown in Appendix E provides the code that was used to implement the auditory model in MATLAB®. The first step was to specify to the script what SNR was being enhanced for that particular execution. In preparation for later stages, the linear filter bank was created. The clean version of *C39C030N.wav* and the noisy version at the specified SNR of the same file were then loaded. Before the curve fitting could take place however, both the clean and noisy files first were filtered through the filter bank to split the signal into the different channels. The envelope of each channel can then be obtained by taking the natural logarithm of the Hilbert transform of the channel. Under Bedrosian conditions, If $z(t)$ represents the Hilbert transform of the k^{th} channel of an input signal s than

$$\ln(z(t)) = \ln[A(t)] + j2\pi f_i(t) \tag{5}$$

where the real component corresponds to the amplitudes and the imaginary component corresponds to the phase information. By taking the exponential of the real component, $A(t)$ can be obtained which provides information on the amplitudes of the channel which can then be used as a reasonable estimate of the envelope. This is known as homomorphic processing and is used commonly in speech processing.

Once the envelope was obtained, the curve fitting could take place to estimate the shape of the resulting sigmoid. The first step was to sort the envelope of both the clean and noisy channel. It becomes evident, as shown in the Figure 5-3, where the noise is mostly concentrated. The clean channel will have the first section of the curve at approximately zero whereas the noisy channel will have a constant value greater than zero in this region depending on the SNR of the signal.

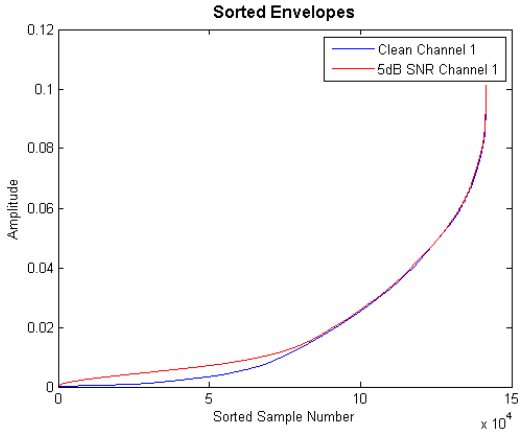


Figure 5-3: Sorted Envelope of Channel 1 for Clean and 5dB SNR

It was then noted that if the ratio of the sorted envelopes were plotted, the resulting plot would produce a sigmoid curve. Due to the large range of values towards the end of the curve that are similar, the sigmoid would plateau at 1 when the ratio of the two curves was taken. Figure 5-4 shows the resulting sigmoid normalised (in the x-axis) to the amplitude of the 5dB SNR envelope.

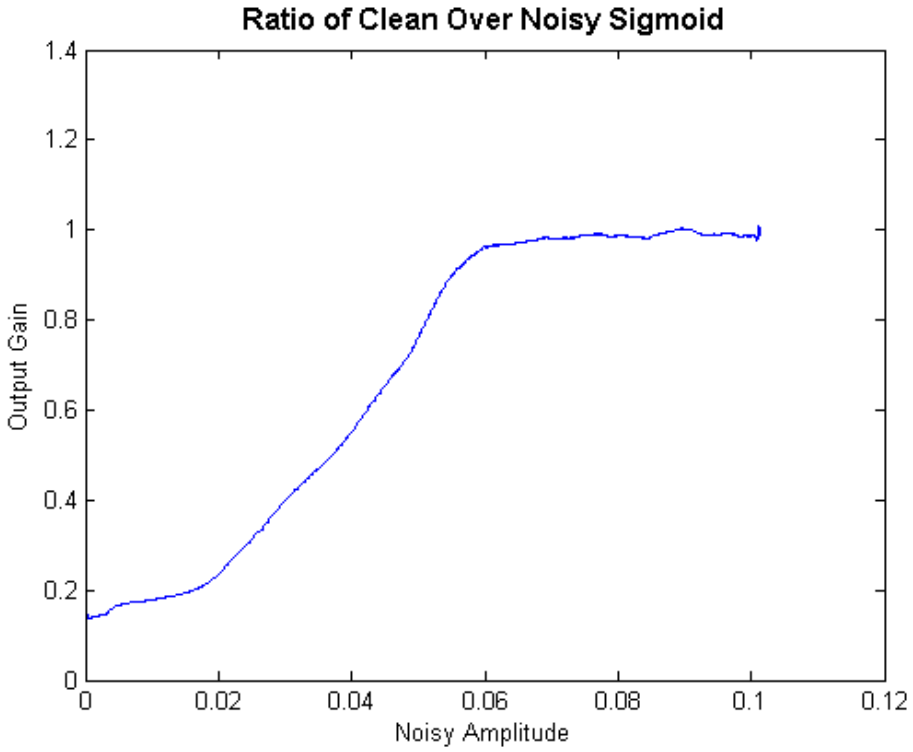


Figure 5-4: Resulting Sigmoid from Ratio of Clean Over Noisy

The general form of the sigmoid used for the curve fit is shown below in equation (6).

$$f(x) = A + \frac{B}{1+e^{\frac{-(x-C)}{D}}}, \tag{6}$$

where, *B* corresponds to the maximum value of the fraction part of the function and *A* and B affect the y-axis intersection. The *C* value affects the point where the curve begins rising and the *D* value affects the gradient of that section of the curve. Using the MATLAB® function *nlinfit*, the following curve fit was obtained for the sigmoid generated from speech data shown in Figure 5-4. As can be seen, it provides a reasonably accurate estimate of the speech data.

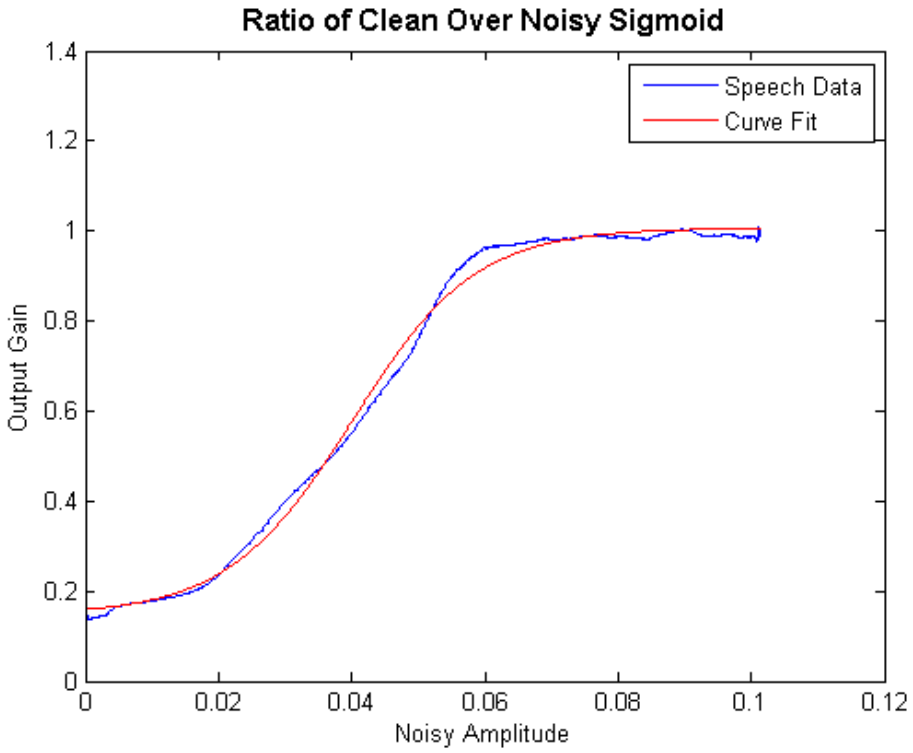


Figure 5-5: Curve Fit of Sigmoid

Due to the poor quality of curve fits at higher channels because of the high percentage of noise in these channels, it was opted to use the sigmoid generated on channel 1 for all channels. As only the channel 1 sigmoid was going to be utilised, it was necessary to normalise each input envelope to the max value of the channel 1 envelope used in the curve fit. When reconstructing the channel which is explained below, it was then necessary to de-normalise back to the original values.

Each input file was then read in one at a time and was then filtered through the filter bank to be split it into its different channels before getting the envelope of each channel in the same manner as described earlier. The output of the sigmoid with each channels envelope as the x-data was then obtained. This gave the ratio of clean speech over noisy speech for each channel. To complete the mapping to a clean envelope from a noisy envelope input, this ratio was then multiplied by the original noisy envelope.

The phase information than needed to be restored by multiplying by the exponential of the product of *j* and the imaginary component of the Hilbert transform of the respective channel. The total equation for recombining the amplitude and phase information of each channel into an output channel is shown below in equation (7).

$$chan_k^{OUT} = \frac{M}{K}\left[f\left(chan_k^{ENVN}\right).chan_k^{ENVN}.e^{(j.L)}\right] \quad (7)$$

In the above equation:

- $chan_k^{OUT}$, k[th] output channel
- *M*, max value of current channel envelope
- *K*, max value of channel 1 envelope (M and K used for de-normalising)
- *f(…),* sigmoid function from curve fit
- $chan_k^{ENVN}$, normalised k[th] input channel envelope
- *L*, imaginary component of *z (Equation (5))*

The enhanced output signal can then be obtained by simply summing all 16 output channels. As this will still contain imaginary components, it was thus necessary to only include the real components when writing to the new file. An example of an enhanced file is shown in Figure 5-6. The example is of a file that was originally 5dB SNR. The original file is shown in blue with the enhanced file plotted on top in red. As can be seen, the amplitude of the noise in the unvoiced regions is severely reduced indicating the sigmoid has been relatively effective. As a perfect curve fit is unobtainable due to the random inconsistencies within the speech data, some of the speech will also be attenuated by the sigmoid which is unavoidable with this approach.

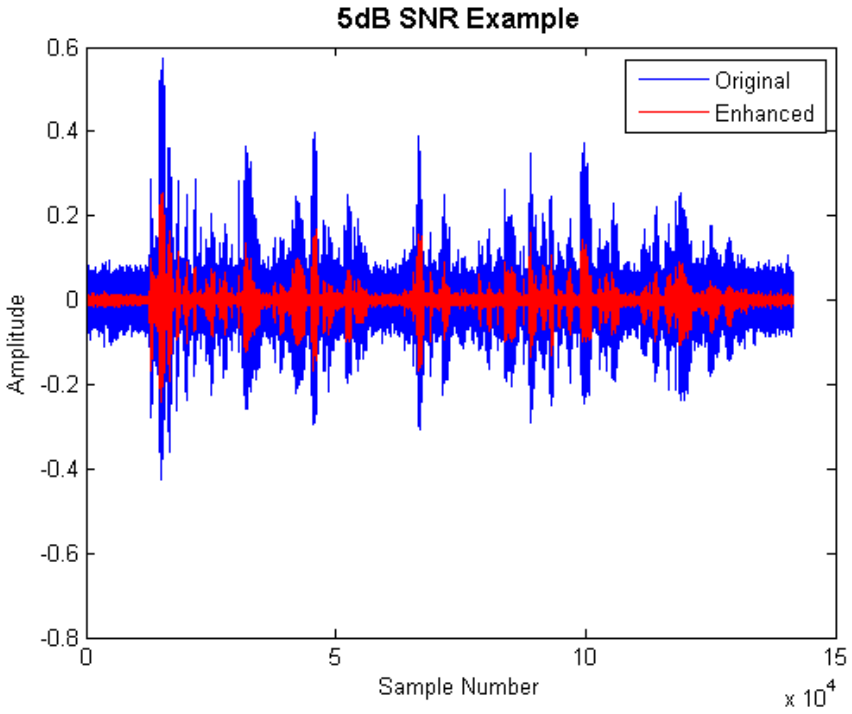


Figure 5-6: Example of Enhancement at 5dB SNR

With this approach, a different sigmoid was required for each SNR. This is because as the SNR is decreased, the amount of attenuation required to reduce the level of noise is increased. The resulting effect on the sigmoid is as the SNR decreases, the y-axis intersection (*A*) decreases and the beginning of the function is flatter for a longer period before beginning to rise. Additionally, a lower SNR has a much smaller gradient in the rise. The justification of this is, signals with a lower SNR have a larger range of amplitudes in the lower region that require attention. For example, a file at 30dB SNR has very minimal noise and thus trying to attenuate it too much will result in the destruction of the speech itself. Figure 5-7 on the following page provides insight into this idea. The 30dB SNR sigmoid is almost the straight line y = 1 as signals at 30dB SNR require very little attenuation compared to those at 5dB. The following equations show the sigmoid functions that were generated for each SNR.

$$f_{5dB}(x) = 0.1477 + \frac{0.8589}{1 + e^{\frac{-(x-0.0401)}{0.0093}}} \tag{8}$$

$$f_{10dB}(x) = 0.2164 + \frac{0.7853}{1 + e^{\frac{-(x-0.0163)}{0.0107}}} \tag{9}$$

$$f_{15dB}(x) = 0.9953 + \frac{-0.6410}{1 + e^{\frac{-(x-0.0265)}{-0.0064}}} \tag{10}$$

$$f_{20dB}(x) = 0.5422 + \frac{0.4546}{1 + e^{\frac{-(x-0.0225)}{0.0044}}} \tag{11}$$

$$f_{25dB}(x) = -44.0055 + \frac{45.0036}{1 + e^{\frac{-(x+0.0333)}{0.0059}}} \tag{12}$$

$$f_{30dB}(x) = -1.999 + \frac{2.998}{1 + e^{\frac{-(x+0.0241)}{0.0057}}} \tag{13}$$

A visual representation of equations 8 to 13 is shown in Figure 5-7.

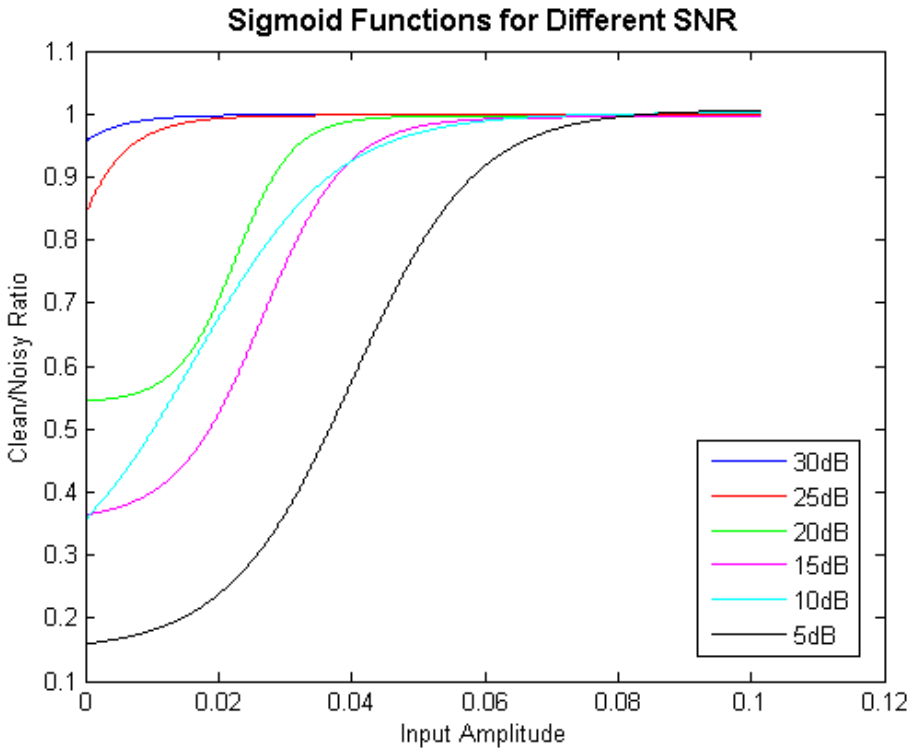


Figure 5-7: Sigmoid Functions for Different SNR

5.3 Testing Methodology

Before any recognition tests could be conducted, the speech files first needed to be enhanced by the auditory model which was achieved through the MATLAB® implementation of the auditory model outlined above in section 5.2. Identical recognition tests to those conducted in the benchmark were then conducted on each set of enhanced files of different SNR. The Perl script *evaluation.pl* shown in Appendix C was adjusted to perform recognition using the directories allocated to the enhanced files. To avoid including mostly similar scripts, only the original *evaluation.pl* was included in the Appendix. The recognition tests performed were done using the same set of HMMs that were trained for the benchmark. The following two sections will show the obtained results from all recognition tests and discuss what can be taken from them.

5.4 Results

Figure 5-8 shows the overall percent correct recognition rate for each of the recognition tests at all SNR using the auditory model front-end to enhance the speech files. Additionally, the benchmark results were also included to allow for easy comparison.

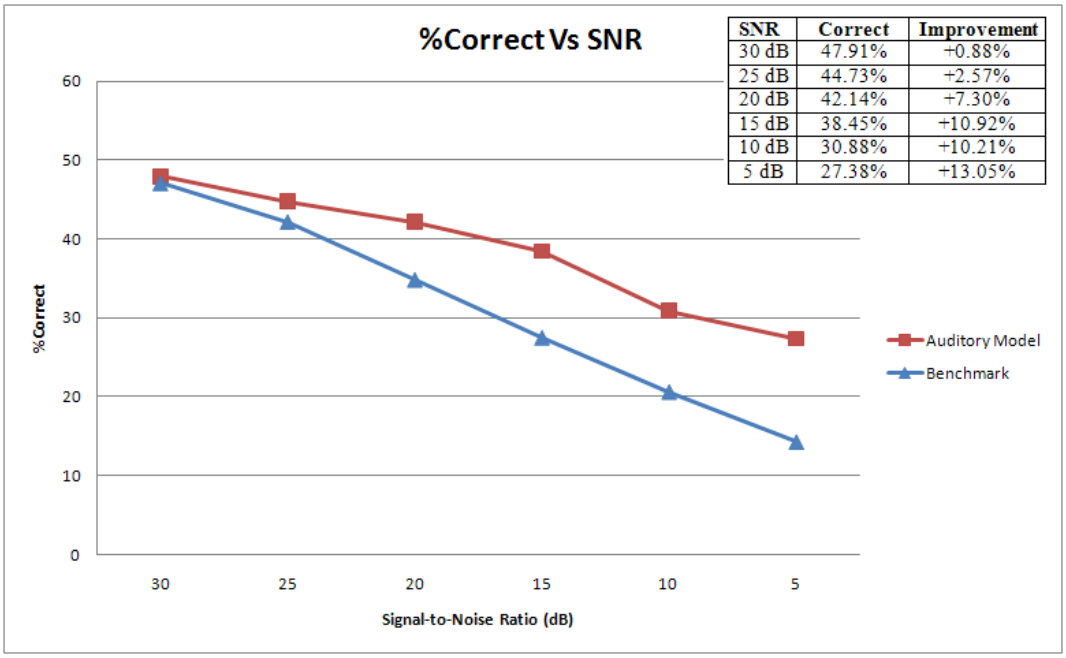


Figure 5-8: % Correct Vs SNR for Auditory Model and Benchmark

As with the benchmark, overall results are useful at providing a quick insight into the performance of the speech recognition system, however considering the recognition rate of individual phonemes can provide further information into which group of phonemes have had the greatest effect from the auditory model when compared to the recognition rate of phonemes from the benchmark. The results provided in the following tables are extracted from the full confusion matrices in Appendix F which provide detailed information on the recognition rate and errors for each included phoneme.

Table 5-1: Recognition Rate of "Vowel" Phonemes after Enhancement

Phoneme	SNR					
	30dB	25dB	20dB	15dB	10dB	5dB
iy	68.7%	67.9%	63.5%	57.7%	52.4%	38.5%
ih	46.8%	45.9%	48.0%	48.7%	43.7%	46.5%
ia	61.5%	63.1%	60.6%	60.8%	73.7%	61.6%
ey	57.0%	56.5%	52.6%	47.7%	53.1%	41.1%
eh	41.1%	40.7%	39.8%	37.0%	39.0%	34.1%
ae	28.5%	28.5%	28.8%	30.0%	29.5%	31.2%
ea	59.9%	63.5%	64.0%	60.8%	70.2%	54.2%
aa	63.4%	65.0%	61.5%	62.2%	63.0%	53.0%
ao	74.0%	73.9%	73.1%	69.4%	46.6%	40.9%
ow	27.0%	25.0%	21.4%	18.8%	19.4%	14.7%
uh	29.1%	23.8%	16.6%	14.5%	9.9%	7.0%
uw	44.1%	39.1%	33.7%	27.8%	26.6%	20.6%
ua	64.8%	58.7%	57.7%	51.7%	50.7%	40.8%
ah	36.0%	35.2%	34.9%	32.8%	27.8%	24.8%
er	36.5%	38.0%	33.2%	29.8%	32.5%	22.9%
ax	37.5%	36.8%	37.4%	37.7%	34.9%	37.9%

Table 5-2: Recognition Rate of "Diphthong" Phonemes after Enhancement

Phoneme	SNR					
	30dB	25dB	20dB	15dB	10dB	5dB
ay	65.6%	65.9%	62.8%	59.5%	59.0%	46.8%
oy	83.4%	81.1%	72.2%	61.3%	54.5%	31.3%
oh	41.5%	40.2%	41.6%	41.3%	34.6%	36.3%
aw	54.3%	56.1%	51.8%	50.2%	52.4%	42.1%

Table 5-3: Recognition Rate of "Glide" Phonemes after Enhancement

Phoneme	SNR					
	30dB	25dB	20dB	15dB	10dB	5dB
y	63.7%	64.3%	63.8	59.2%	54.7%	48.0%
w	63.5%	57.5%	55.7%	48.6%	24.2%	21.9%

Table 5-4: Recognition Rate of "Liquid" Phonemes after Enhancement

Phoneme	SNR					
	30dB	25dB	20dB	15dB	10dB	5dB
l	30.2%	26.8%	21.9%	19.0%	20.0%	17.5%
r	45.6%	39.7%	35.5%	28.7%	18.5%	18.6%

Table 5-5: Recognition Rate of "Nasal" Phonemes after Enhancement

Phoneme	SNR					
	30dB	25dB	20dB	15dB	10dB	5dB
m	74.1%	72.4%	69.1%	59.1%	38.7%	24.6%
n	61.6%	58.1%	54.1%	46.9%	23.2%	19.5%
ng	67.7%	62.6%	53.7%	43.0%	21.3%	14.4%

Table 5-6: Recognition Rate of "Fricative" Phonemes after Enhancement

Phoneme	SNR					
	30dB	25dB	20dB	15dB	10dB	5dB
f	80.4%	85.7%	86.5%	89.7%	96.9%	93.6%
v	78.8%	83.7%	86.8%	89.1%	90.2%	85.9%
th	57.4%	56.6%	54.7%	44.7%	28.4%	34.4%
dh	48.7%	37.5%	31.5%	25.6%	12.4%	21.5%
s	70.4%	71.0%	69.5%	66.2%	67.9%	57.8%
z	75.6%	72.8%	66.1%	56.7%	46.3%	33.1%
sh	75.1%	74.4%	66.3%	58.6%	57.9%	37.9%
zh	81.5%	80.9%	64.5%	53.3%	52.7%	26.9%
hh	78.8%	77.2%	70.8%	63.8%	60.7%	50.6%

Table 5-7: Recognition Rate of "Stop" Phonemes after Enhancement

Phoneme	SNR					
	30dB	25dB	20dB	15dB	10dB	5dB
p	59.1%	28.6%	11.3%	3.6%	0.7%	0.3%
b	39.5%	9.7%	2.1%	0.3%	0.0%	0.0%
t	35.1%	31.5%	30.4%	26.7%	14.6%	16.6%
d	41.4%	30.6%	24.8%	16.9%	4.2%	4.9%
k	41.0%	30.7%	24.8%	18.3%	5.0%	8.0%
g	45.4%	33.9%	28.6%	19.8%	4.7%	4.1%

Table 5-8: Recognition Rate of "Affricate" Phonemes after Enhancement

Phoneme	SNR					
	30dB	25dB	20dB	15dB	10dB	5dB
ch	51.8%	53.9%	55.5%	52.3%	47.3%	36.8%
jh	56.3%	55.1%	57.8%	52.8%	43.8%	38.8%

5.5 Analysis

The results of the auditory modelling tests shown in section 5.4 clearly show the improvement that the auditory model has had on speech recognition using MFCCs. Despite the relatively low recognition rates which are attributed to only using basic monophone HMMs for recognition, the auditory model has increased the recognition rate at all the SNR that was tested. Of particular note is the recognition rate at 5dB SNR which increased by

13% up to 27% which is almost double of the original recognition rate at 5dB SNR. When considering the initial benchmark tests, this is an improvement of 10dB which is very significant. As was expected, higher SNRs have a much less significant improvement due to the already low noise level and relatively good performance compared to that of the lower SNRs. An exception to this rule is 10dB SNR which improved by smaller margin than 5dB SNR as expected however it also improved by less than 15dB SNR. This can be attributed to a poor quality curve fit despite best efforts. When considering Figure 5-7, it can be seen that the 10dB SNR curve fit didn't match up with the other functions and thus has created some erratic results which can be seen in the above tables of phoneme recognition.

As was the case with the benchmark tests, the majority of phonemes followed the notion that a reduction in the SNR would result in a reduced recognition rate. With the inclusion of the pre-processing of the auditory model before MFCC extraction, some inconsistencies have occurred.

From the benchmark tests, two of the vowel phonemes 'eh' and 'ae' displayed characteristics of noise immunity as the recognition rate of these phonemes varies only slightly across all SNRs. While this phenomenon has mostly occurred after enhancement with the auditory model, particularly in the 'ae' phoneme, the 'eh' phoneme has a much larger variance after enhancement particularly at 5dB SNR. This change can likely be attributed to the attenuation by the sigmoid. Certain statistics contained in the speech data will have been affected and thus result in an increased vulnerability to noise.

Also comparing to the benchmark tests, it can be seen that the reverse effect of increasing recognition rate with reducing SNR on several phonemes has diminished. In the benchmark tests, the phonemes 'ea', 'er' and 'f' all showed higher recognition rates at 5dB SNR than on clean speech. In the auditory model testing however, while there was an initial increase in recognition rate for the 'ea' and 'er' phonemes, the recognition rate at 5dB SNR had dropped below that of clean. The fricative 'f' phoneme however increased in recognition rate at all SNRs. The increase at 10dB was more significant and can likely be attributed to the curve fitting issue outlined above. This change compared to the benchmark tests suggests that the auditory model better aligns phonemes towards the traditional effect of having a reduced recognition rate as the SNR is decreased.

When considering the recognition rate of phoneme groups as a whole, as was the case with the benchmark tests the Stop phonemes have been the most impacted by the noise. When comparing the recognition rate of these phonemes after enhancement however it is quickly evident that a significant improvement has occurred due to the auditory model. In the

benchmark tests, 4 of the 6 stop phonemes had a 0% recognition rate at SNR whereas with the auditory model only one of the phonemes is at 0% recognition as shown in Table 5-8. The 't' phoneme saw the greatest improvement increasing to 16.6% recognition after enhancement compared to 0.9% originally. The 'd' phoneme increased to 4.9% from 0.3%, the 'k' phoneme to 8.0% from 0% and the 'g' phoneme to 4.1% from 0%. These results suggest that the auditory model has been effective in reducing the noise but maintaining the majority of the speech to allow for an improved recognition rate.

When looking at the recognition rate of all 44 phonemes, after enhancement by the auditory model 34 phonemes showed an improvement in recognition rate when compared to the benchmark tests. Of the 34, 11 showed an increase of 20% or higher with one phoneme ('ao') actually increasing by 40.8%. Of the other 23, 10 increased by a range of 10 to 20% and the other 10 increased up to 10%.

Again, in-depth analysis of each individual phoneme recognition rate including pattern recognition can be considered a thesis in itself. The randomness of the additive noise coupled with the manipulation of the data by the auditory model means a static effect on the recognition rate across all phonemes and all SNRs is very unlikely as was the case with the benchmark tests.

As only 10 phonemes showed a reduction in recognition and the overall recognition rate has improved at all SNR, it can be stated with confidence that the front-end auditory model has been successful.

6. Conclusion

6.1 Summary

A front-end auditory model that utilised the inherent features of the human auditory system was successfully interfaced to a speech recognition system. The result of this was an improvement in recognition rate at all SNR that were tested. Of noteworthy mention, the recognition rate at 5dB SNR almost doubled when enhanced by the front-end auditory model which is a significant improvement that correlates to an approximate increase of 10dB in SNR.

6.2 Limitations

6.2.1. Computational Requirements

A minor limitation of the model is the heavy computational overhead required for the pre-processing of the speech files. With the filtering, sigmoid application and large multiplication when reconstructing the required processing power quickly escalates. While the computation is relatively intensive, it is still not beyond the scope of what modern DSP processors are capable of.

6.2.2. Static Implementation

As shown, speech files with different SNR require enhancement by a different sigmoid. The investigation into this method in this thesis only tested on a limited set of SNR and sigmoid functions. Furthermore, the enhancement relied on user intervention to specify the SNR which is impractical for any real world application. Additionally, as the speech is pre-processed before going through the recognition phase, real-time recognition is not obtained.

6.3 Continuations and Extensions

6.3.1. Dynamic Implementation

To improve on a limitation of the current proposed model, a dynamic approach could be implemented. A set of sigmoid functions could be generated over a larger range of SNR and at a higher frequency interval than 5dB as was done in this thesis. Using the information obtained from the sorted envelope plots, a method for noise power estimation could be implemented to obtain an estimate of the SNR of the speech file and based on this make a decision on which of the sigmoid functions to use for enhancement.

References

[1] LumenVox Resource Center. (2003, 7 April 2010). *History of Speech Recognition*. Available: http://www.lumenvox.com/resources/tips/historyOfSpeechRecognition.aspx

[2] K. Davis*, et al.*, "Automatic recognition of spoken digits," *Clearing House,* vol. 10, pp. 24-06, 2002.

[3] Wikipedia Foundation. (2010, *Wikipedia: The Free Encyclopedia*. Available: http://en.wikipedia.org/

[4] B. H. Juang, "Speech recognition in adverse environments," *Computer Speech & Language,* vol. 5, pp. 275-294, 1991.

[5] R. Stern*, et al.*, "Multiple approaches to robust speech recognition," 1992.

[6] W. Brownell, "How the Ear Works: Nature," *Volta Review,* vol. 99, pp. 9-28, 1997.

[7] A. Hudspeth, "How the ear's works work," *Nature,* vol. 341, pp. 397-404, 1989.

[8] learnartificialneuralnetworks.com. (2007, 6 April 2010). *Speech Recognition*. Available: http://www.learnartificialneuralnetworks.com/speechrecognition.html

[9] VoxForge. (2010, 6 April 2010). *What is a Speaker Dependent or Independent Acoustic Model?* Available: http://voxforge.org/home/docs/faq/faq/what-is-a-speaker-dependent-or-independent-acoustic-model

[10] C. Leggetter and P. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," *Computer speech and language,* vol. 9, p. 171, 1995.

[11] Young S et al, *The HTK Book*, 3.2 ed., 2002.

[12] B. Hayes, "Adaptive Speaker Phoneme Recognition System," Computer Systems Engineering, Electrical and Computer Engineering, JCU, Townsville, 2005.

[13] E. Grabianowski. (2006, August 2010). *How Speech Recognition Works*. Available: http://electronics.howstuffworks.com/gadgets/high-tech-gadgets/speech-recognition.htm/printable

[14] I. Mporas*, et al.*, "Comparison of Speech Features on the Speech Recognition Task," *Journal of Computer Science,* vol. 3, pp. 608-616, 2007.

[15] ETSI, "ES 201 108 v1.1.3," in *Front-end feature extraction algorithm*, ed, 2003, pp. 8-12.

[16] D. Dimitriadis*, et al.*, "Advanced Front-end for Robust Speech Recognition in Extremely Adverse Environments," *Proceedings of ICSLP'07,* 2007.

[17] H. Hirsch, "HMM adaptation for applications in telecommunication," *Speech Communication,* vol. 34, pp. 127-139, 2001.

[18] S. N. Wrigley. (1998, 6 April 2010). *Speech Recognition by Dynamic Time Warping*. Available: http://www.dcs.shef.ac.uk/~stu/com326/index.html

[19] E. Avci, "A new optimum feature extraction and classification method for speaker recognition: GWPNN," *Expert Systems with Applications,* vol. 32, pp. 485-498, 2007.

[20] University of New South Wales. (2002, 10 April 2010). *Dynamic Time Warping*. Available: http://www.cse.unsw.edu.au/~waleed/phd/html/node38.html

[21] Massachusetts Institute of Technology. (2003, 10 April 2010). *Dynamic Time Warping & Search*.

[22] L. Rabiner and B. Juang, "An introduction to hidden Markov models," *IEEE ASSp Magazine,* vol. 3, pp. 4-16, 1986.

[23] M. Stamp, "A revealing introduction to hidden Markov models," *Department of Computer Science San Jose State University,* 2004.

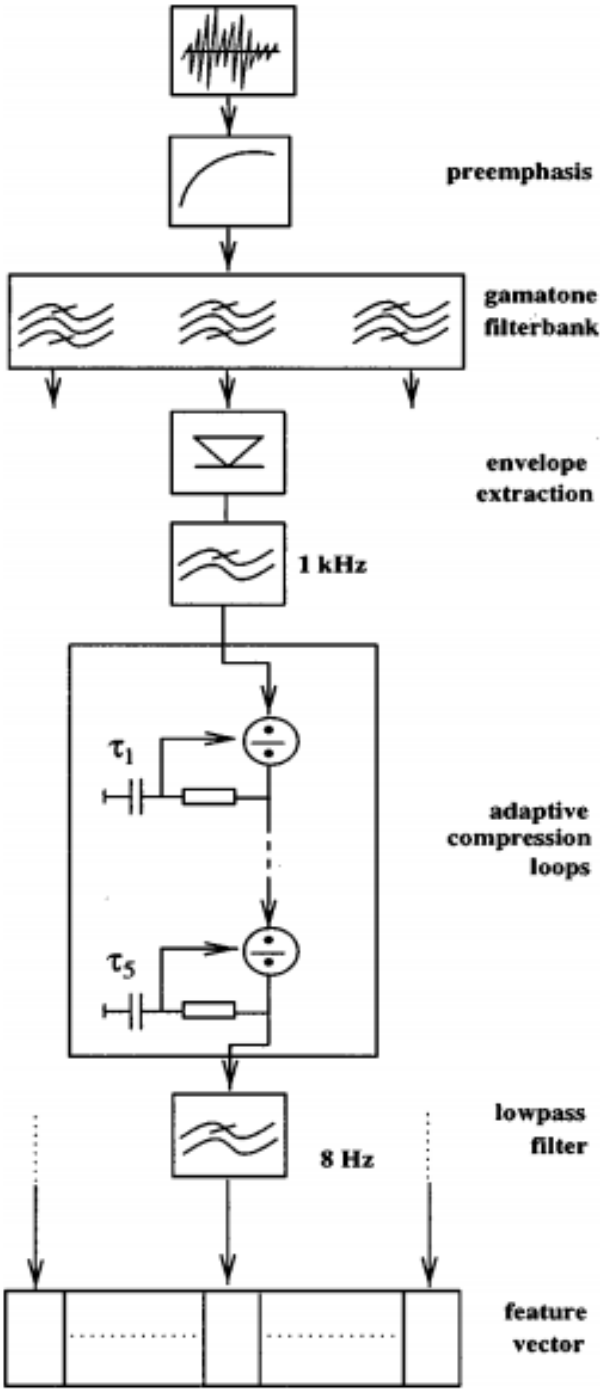
[24] L. Rabiner, "A tutorial on hidden Markov models and selected applications inspeech recognition," *Proceedings of the IEEE,* vol. 77, pp. 257-286, 1989.

[25] V. Petrushin, "Hidden markov models: Fundamentals and applications," 2000.

[26] D. Rumelhart*, et al.*, "Learning representations by back-propagating errors," *Cognitive modeling,* p. 213, 2002.

[27] B. Kotnik, *et al.*, "Efficient Noise Robust Feature Extraction Algorithms for Distributed Speech Recognition (DSR) Systems," *International Journal of Speech Technology,* vol. 6, pp. 205-219, 2003.
[28] N. Morales, *et al.*, "Statistical class-based MFCC enhancement of filtered and band-limited speech for robust ASR," 2005.
[29] L. S. Michael and A. Alex, "Training Wideband Acoustic Models Using Mixed-Bandwidth Training Data for Speech Recognition," *Audio, Speech, and Language Processing, IEEE Transactions on,* vol. 15, pp. 235-245, 2007.
[30] C. Jankowski and R. Lippmann, "Comparison of auditory models for robust speech recognition," 1992, p. 454.
[31] S. Seneff, "A Computational Model for the Peripheral Auditory System: Application to Speech Recognition," in *Proceedings of ICASSP-86*, 1986.
[32] S. Seneff, "Pitch and Spectral Analysis of Speech Based on An Auditory Model," PhD Thesis, Massachusetts Institute of Technology, 1985.
[33] O. Ghitza, "Auditory nerve representation as a front-end for speech recognition in a noisy environment," *Computer Speech & Language,* vol. 1, pp. 109-130, 1986.
[34] J. Tchorz and B. Kollmeier, "A model of auditory perception as front end for automatic speech recognition," *The Journal of the Acoustical Society of America,* vol. 106, p. 2040, 1999.
[35] Linguistic Data Consortium University of Pennsylvania. (1993, 5 April 2010). *The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus (TIMIT)*. Available: http://www.ldc.upenn.edu/Catalog/readme_files/timit.readme.html
[36] L. D. C. U. o. Pennsylvania. (1995, 5 April 2010). *WSJCAM0 Corpus and Recording Description*. Available: http://www.ldc.upenn.edu/Catalog/readme_files/wsjcam0/wsjcam0.html
[37] The Perl Foundation. (2010, *The Perl Directory*. Available: http://www.perl.org
[38] I. Magrin-Chagnolleau, *et al.*, "A further investigation on speech features for speaker characterization," 2000.

Appendix A: Auditory Model



Appendix B: Benchmark Confusion Matrices

Clean

30dB SNR

25 dB SNR

20 dB SNR

15 dB SNR

10 dB SNR

5 dB SNR

Appendix C: Perl Scripts for Training and Testing

training.pl

```
#This is the main file for training the phoneme recognition HMMs

#============================================================================
# 1. Convert all the .wv1 files in the data directory to .wav files
system("cls");
print "1) Convert all wv1 files to wav files\n";
system("PAUSE");
system("perl convert.pl");
#============================================================================
# 2. Create a list of all the .wav files and their "to be" .mlf names in a
#    file wav_file_list.scp"
system("cls");
print "2) Create a list of all .wav and to be .mfc names\n";
system("PAUSE");
system("perl wav_mfc_list.pl");
#============================================================================
# 3. Run HCopy to convert wav files into mfc files
system("cls");
print "3) Run HCopy to convert the wav files into mfc files\n";
system("PAUSE");
system("C:/Thesis/Tools/HCopy -T 1 -C C:/Thesis/Tools/HCopy_config.ini -S
C:/Thesis/data/wav_file_list.scp");
#============================================================================
# 4. Create a list of all the .mfc files in a file train_mfc_file_list.scp
system("cls");
print "4) Create a list of all the .mfc files\n";
system("PAUSE");
system("perl train_list_mfc.pl");
#============================================================================
# 5. Run HCompV to convert the .mfc files into hmm's.  This is the first
#    pass and will be placed in the direcotry hmm0
system("cls");
print "5) Run HCompV to convert .mfc files to first pass hmm's\n";
system("PAUSE");
system("C:/Thesis/Tools/HCompV -C C:/Thesis/Tools/HCompV_config.ini -f 0.01
-m -S C:/Thesis/data/train_mlf_file_list.scp -M C:/Thesis/out/hmm0
C:/Thesis/common/proto.ini");
#============================================================================
# 6. Create the master label file.
system("cls");
print "6) Create the master label file\n";
system("PAUSE");
system("perl create_mlf_phones.pl");
#============================================================================
# 7. Copy all .phn files in the data directory and rename with a .lab
#    extension
system("cls");
print "7) Copy all .phn files in the data directory; rename with .lab
extension\n";
system("PAUSE");
system("perl rename_phn_lab.pl");
#============================================================================
# 8. Create the hmmdef file and macros file
system("cls");
print "8) Create the hmmdefs file and macros file\n";
system("PAUSE");
system("perl create_hmmdefs.pl");
system("perl create_macros.pl");
#============================================================================
# 9. 1st Re-estimation of the monophone HMMs
system("cls");
print "9) 1st Re-estimation of the monophone HMMs\n";
```

```
system("PAUSE");
system("C:/Thesis/Tools/HERest -C C:/Thesis/Tools/HERest_config.ini -L
C:/Thesis/data/ -t 250.0 150.0 1000.0 -S
C:/Thesis/data/train_mlf_file_list.scp -H C:/Thesis/out/hmm0/macros -H
C:/Thesis/out/hmm0/hmmdefs -M C:/Thesis/out/hmm1
C:/Thesis/common/monophones0");
system("PAUSE");
#================================================================================
# 10. 2nd Re-estimation of the monophone HMMs
system("cls");
print "10) 2nd Re-estimation of the monophone HMMs\n";
system("PAUSE");
system("C:/Thesis/Tools/HERest -C C:/Thesis/Tools/HERest_config.ini -L
C:/Thesis/data/ -t 250.0 150.0 1000.0 -S
C:/Thesis/data/train_mlf_file_list.scp -H C:/Thesis/out/hmm1/macros -H
C:/Thesis/out/hmm1/hmmdefs -M
C:/Thesis/out/hmm2 C:/Thesis/common/monophones0");
#================================================================================
# 11. 3rd Re-estimation of the monophone HMMs
system("cls");
print "11) 3rd Re-estimation of the monophone HMMs\n";
system("PAUSE");
system("C:/Thesis/Tools/HERest -C C:/Thesis/Tools/HERest_config.ini -L
C:/Thesis/data/ -t 250.0 150.0 1000.0 -S
C:/Thesis/data/train_mlf_file_list.scp -H C:/Thesis/out/hmm2/macros -H
C:/Thesis/out/hmm2/hmmdefs -M
C:/Thesis/out/hmm3 C:/Thesis/common/monophones0");
#================================================================================
# 12. Copy hmmdefs and macros to hmm4 directory, manually add sp model.
system("cls");
print "12) MANUALLY COPY HMMDEFS AND MACROS FROM HMM3 TO HMM4 AND ADD SP
MODEL\n";
print "-----------------DO NOT CONTINUE UNTIL THIS IS DONE-----------------
--\n";
system("PAUSE");
#================================================================================
# 13. Run HHEd to add extra transition states and tie the "sp" state to the
#     centre "sil" state, stores in hmm5
system("cls");
print "13) Run HHed to add extra transition states and tie the sp state to
the centre sil state\n";
system("PAUSE");
system("C:/Thesis/Tools/HHEd -H C:/Thesis/out/hmm4/macros -H
C:/Thesis/out/hmm4/hmmdefs -M C:/Thesis/out/hmm5/ C:/Thesis/common/sil.hed
C:/Thesis/common/monophones1");
#================================================================================
# 14. 6th Re-estimation of the monophone HMMs, including the "sp" model
system("cls");
print "14) 6th Re-restimation of the monohome HMMs, including the sp
model\n";
system("PAUSE");
system("C:/Thesis/Tools/HERest -C C:/Thesis/Tools/HERest_config.ini -L
C:/Thesis/data/ -t 250.0 150.0 1000.0 -S
C:/Thesis/data/train_mlf_file_list.scp -H C:/Thesis/out/hmm5/macros -H
C:/Thesis/out/hmm5/hmmdefs -M C:/Thesis/out/hmm6
C:/Thesis/common/monophones1");
#================================================================================
# 15. 7th Re-estimation of the monophone HMMs, including the "sp" model
system("cls");
print "15) 7th Re-restimation of the monophone HMMs, including the sp
model\n";
system("PAUSE");
system("C:/Thesis/Tools/HERest -C C:/Thesis/Tools/HERest_config.ini -L
C:/Thesis/data/ -t 250.0 150.0 1000.0 -S
C:/Thesis/data/train_mlf_file_list.scp -H C:/Thesis/out/hmm6/macros -H
C:/Thesis/out/hmm6/hmmdefs -M C:/Thesis/out/hmm7
C:/Thesis/common/monophones1");
#================================================================================
```

```
# 16. Realign the training data and create new transcriptions
system("cls");
print "16) Realign the training data and create new transcriptions\n";
system("PAUSE");
system("C:/Thesis/Tools/HVite -l '*' -o SWT -b SIL -C
C:/Thesis/Tools/HERest_config.ini -a -H
C:/Thesis/out/hmm7/macros -H C:/Thesis/out/hmm7/hmmdefs -i
C:/Thesis/out/aligned.mlf -m -t 250.0 -y lab -L C:/Thesis/data/ -S
C:/Thesis/data/train_mlf_file_list.scp C:/Thesis/common/phonedict
C:/Thesis/common/monophones1");
#============================================================================
#17. 8th Re-estimation of the monophone HMMs, including the "sp" model
system("cls");
print "17) 8th Re-estimation of the monophone HMMs, including the sp
model\n";
system("PAUSE");
system("C:/Thesis/Tools/HERest -C C:/Thesis/Tools/HERest_config.ini -L
C:/Thesis/data/ -t 250.0 150.0 1000.0 -S
C:/Thesis/data/train_mlf_file_list.scp -H C:/Thesis/out/hmm7/macros -H
C:/Thesis/out/hmm7/hmmdefs -M C:/Thesis/out/hmm8
C:/Thesis/common/monophones1");
#============================================================================
#18. 9th Re-estimation of the monophone HMMs, including the "sp" model
system("cls");
print "18) 9th Re-estimation of the monophone HMMs, including the sp
model\n";
system("PAUSE");
system("C:/Thesis/Tools/HERest -C C:/Thesis/Tools/HERest_config.ini -L
C:/Thesis/data/ -t 250.0 150.0 1000.0 -S
C:/Thesis/data/train_mlf_file_list.scp -H C:/Thesis/out/hmm8/macros -H
C:/Thesis/out/hmm7/hmmdefs -M C:/Thesis/out/hmm9
C:/Thesis/common/monophones1");
```

convert.pl

```
#This routine will convert a directory of .wv1 files to .wav files making
#use of the sph2pipe conversion utility

use Cwd;
$current_dir = getcwd(); # Get the current directory
$data_dir = "$current_dir/data";

opendir(DIR, $data_dir) || die "Unable to open dir $data_dir: $!";
@nFiles = grep(!/^\./, readdir(DIR));

foreach $nFiles (@nFiles)
{

       $file_ext = substr($nFiles,-3,3);

       if(($file_ext eq "wv1") || ($file_ext eq "WV1"))
       {
              $filename = substr($nFiles,0,-4);
              @list = ($filename,".wav");
              $new_filename = join "",@list;
              print "$nFiles converted to $new_filename";
              print "\n";
              system("$current_dir/Tools/sph2pipe -p -f wav
$current_dir/data/$nFiles $current_dir/data/$new_filename");
       }
}

closedir DIR;
```

wav_mfc_list.pl

```
# This file will create a list of all .wav files in the data direcotry along
# with their proposed mfc file name listing the full path for both.

use Cwd;
$current_dir = getcwd(); # Get the current directory
$data_dir = "$current_dir/data";

opendir(DIR, $data_dir) || die "Unable to open dir $data_dir: $!";
@nFiles = grep(!/^\./, readdir(DIR));

open(TXT,">$data_dir/wav_file_list.scp") || die("Cannot open file");

foreach $nFiles (@nFiles)
{
       $file_ext = substr($nFiles,-3,3);

       if($file_ext eq "wav")
       {
              print "$nFiles"; print "\n";
              $filename = substr($nFiles,0,-4);
              $new_filename = join "",$filename,".mfc";
              print TXT "$data_dir/$nFiles $data_dir/$new_filename";
              print TXT "\n";
       }
}

close(TXT);

closedir DIR;
```

train_list_mfc.pl

```
# This will create a text file listing the direct paths of all the .mfc
# files in the data directory.

use Cwd;
$current_dir = getcwd(); # Get the current directory
$data_dir = "$current_dir/data";

opendir(DIR, $data_dir) || die "Unable to open dir $data_dir: $!";
@nFiles = grep(!/^\./, readdir(DIR));

open(TXT,">$data_dir/train_mlf_file_list.scp") || die("Cannot open file");

foreach $nFiles (@nFiles)
{
       $file_ext = substr($nFiles,-3,3);

       if($file_ext eq "mfc")
       {
              print TXT "$data_dir/$nFiles";
              print TXT "\n";
       }
}

close(TXT);

closedir DIR;
```

create_mlf_phones.pl

```
# This will create a .mlf or Master Label File

use Cwd;
$current_dir = getcwd(); # Get the current directory
$out_dir = "$current_dir/out";
$data_dir = "$current_dir/data";

opendir(DIR, $data_dir) || die "Unable to open dir $data_dir: $!";
@nFiles = grep(!/^\./, readdir(DIR));

open(TXT,">$out_dir/phones0.mlf") || die("Cannot open File");

print TXT "#!MLF!#\n";

foreach $nFiles (@nFiles)
{
	$file_ext = substr($nFiles,-3,3);

	if(($file_ext eq "phn") || ($file_ext eq "PHN"))
	{
		$filename = substr($nFiles,0,-3);
		$filename = join "",'"*/', $filename,'lab"';
		$filename = join "",'"*/', $nFiles,'"';
		open(CURRENT_FILE,"$data_dir/$nFiles") || die "Cannot open
file";
		print TXT "$filename\n";
		#print TXT "$filename\n";
		while($record = <CURRENT_FILE>)
		{
			print TXT "$record";
		}
		close(CURRENT_FILE);
		print TXT ".\n";
	}
}
close(TXT);
closedir DIR;
```

rename_phn_lab.pl

```
# This will take a copy of each .phn file in the data directory and copy it
# using the .lab extension.

use Cwd;
use File::Copy;

$current_dir = getcwd(); # Get the current directory
$data_dir = "$current_dir/data";

opendir(DIR, $data_dir) || die "Unable to open dir $data_dir: $!";
@nFiles = grep(!/^\./, readdir(DIR));

foreach $nFiles (@nFiles)
{
	$file_ext = substr($nFiles,-3,3);

	if(($file_ext eq "phn") || ($file_ext eq "PHN"))
	{
		$filename = substr($nFiles,0,-3);
		$filename = join "", $filename, "lab";
		copy("$data_dir/$nFiles","$data_dir/$filename");
	}
}

closedir DIR;
```

create_hmmdefs.pl

```
# This creates the hmmdefs file by parsing through the list of monophones
# and placing each in the file along with the contents of the "proto" file

open(PHONELIST,"C:/Thesis/common/monophones0") || die("Cannot open file
PHONELIST");
{
	open(TXT,">C:/Thesis/out/hmm0/hmmdefs") || die("Cannot open file
hmmdefs");

	while($record = <PHONELIST>)
	{
		print TXT join "","~h ", "$record";
		if($record eq "sil")
		{
			print TXT "\n";
		}
		open(PROT,"C:/Thesis/out/hmm0/proto") || die("Cannot open file
PROTO");
		$I = 0;
		while($this = <PROT>)
		{
			if($I > 3)
			{
				print TXT "$this";
			}
			$I = $I + 1;
		}

		print TXT "\n";
		#close(PROT);
	}

	close(TXT);
}

close(PHONELIST);
```

create_macros.pl

```
# This creates the macros file for use with HERest.

open(TXT, ">C:/Thesis/out/hmm0/macros") || die ("Cannot open file macros");

	open(MACRO_HEADER, "C:/Thesis/common/macro_header") || die("Cannot
open file macro_header");

		while($record = <MACRO_HEADER>)
		{
			print TXT "$record";
		}

	close (MACRO_HEADER);

	print TXT "\n";

	open(VFLOOR, "C:/Thesis/out/hmm0/vFloors") || die("Cannot open file
vFloors");

		while($record = <VFLOOR>)
		{
			print TXT "$record";
		}

	close(VFLOOR);
close(TXT);
```

evaluation.pl

```
# This will perform the recognition tests

#========================================================================
# 1. Convert all the .wv1 files in the evaluation directory to .wav files
system("cls");
print "1) Convert all wv1 files to wav files\n";
system("PAUSE");
system("perl convert_eval.pl");
#========================================================================
# 2. Create a list of all the .wav files and their "to be" .mlf names in a
#    file wav_eval_file_list.scp"
system("cls");
print "2) Create a list of all .wav and to be .mfc names\n";
system("PAUSE");
system("perl wav_eval_mfc_list.pl");
#========================================================================
# 3. Run HCopy to convert wav files into mfc files
system("cls");
print "3) Run HCopy to convert the wav files into mfc files\n";
system("PAUSE");
system("C:/Thesis/Tools/HCopy -T 1 -C C:/Thesis/Tools/HCopy_config.ini -S
C:/Thesis/evaluation/wav_eval_file_list.scp");
#========================================================================
# 4. Create a list of all the .mfc files in a file eval_mlf_file_list.scp
system("cls");
print "4) Create a list of all the .mfc files\n";
system("PAUSE");
system("perl eval_list_mfc.pl");
#========================================================================
# 5. Copy all .phn files in the evaluation directory and rename with a .lab
#    extension
system("cls");
print "5) Copy all .phn files in the evaluation directory; rename with .lab
extension\n";
system("PAUSE");
system("perl eval_rename_phn_lab.pl");
#========================================================================
# 6. Recognition test
system("cls");
print "6) Recognition test\n";
system("PAUSE");
system("C:/Thesis/Tools/HVite -C C:/Thesis/Tools/HERest_config.ini -H
C:/Thesis/out/hmm9/hmmdefs -S C:/Thesis/evaluation/eval_mlf_file_list.scp -l
'*' -i C:/Thesis/out/recout.mlf -w
C:/Thesis/common/wdnet -p 0.0 -s 5.0 C:/Thesis/common/phonedict
C:/Thesis/common/monophones1");
#========================================================================
# 7. Results
system("PAUSE");
system("cls");
print "7) Results\n";
system("PAUSE");
system("C:/Thesis/Tools/HResults -f -p -L C:/Thesis/evaluation
C:/Thesis/common/monophones1 C:/Thesis/out/recout.mlf > OUTPUT.txt");
```

Appendix D: MATLAB® Code for Benchmark

addnoise.m

```
%------------------------------------------------------------------------
%This script file will read in all the evaluation speech files in the
%specified directory, request the required SNR before inserting random
%white Gaussian noise into the files to obtain the required SNR and
%re-writing the file to a directory based on the SNR.
%NB. Directory is of the form 'evaluation_<SNR>db' and must exist prior
%to the execution of this script
%------------------------------------------------------------------------

%Get the list of all the original clean speech files used for evaluation
nFiles = dir(fullfile('C:','Thesis','evaluation','*.wav'));

%Request the required Signal-Noise-Ratio from the user
SNR = input('Required SNR: ');

%Loop through all the files one at a time.
for i = 1:length(nFiles)

    %Load the i'th file in
    [signal,F,N] = wavread(strcat('C:\Thesis\evaluation\',nFiles(i).name));

    %Calculate the RMS power of the
    signal_power = sqrt(mean(signal.^2));

    %Based on the input SNR, calculate the required noise power
    noise_power = signal_power/(10^(SNR/20));

    %Generate some random noise as long as the signal
    noise = randn(length(signal),1);

    %Obtain gains k required to adjust the power of noise to the required
    %noise power
    k = noise_power/sqrt(mean(noise.^2));

    %Multiply the noisy array by the gains k to adjust the power
    noise = k.*noise;

    %Combine the signal and the noise
    output_signal = signal + noise;

    %Write to new directory
wavwrite(output_signal,F,N,strcat('C:\Thesis\evaluation_',num2str(SNR),'dB\'
,nFiles(i).name));

end
```

Appendix E: MATLAB® Code for Auditory Model

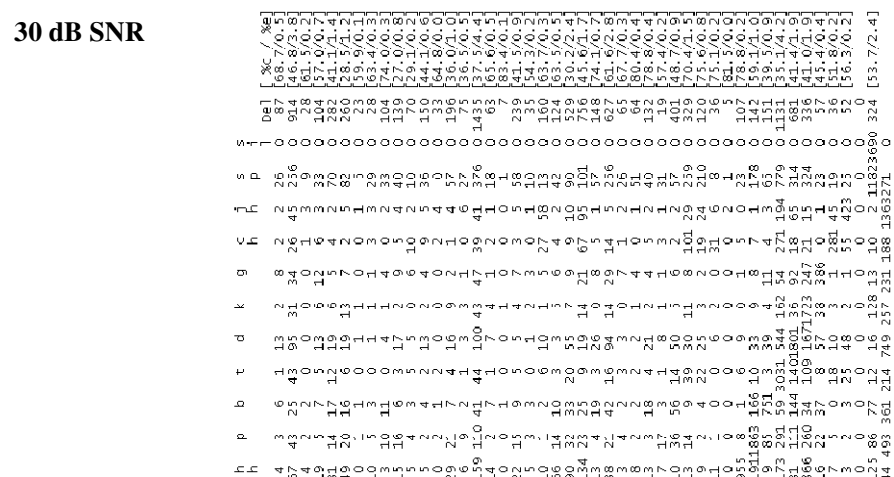
enhanceall.m

```
%-------------------------------------------------------------------------
%This script will request the SNR from the user before loading the clean
%version of C39C030N.wav and the noisy version of it at the specified SNR
%which will be used to fit the sigmoid. (Any file will produce mostly
%similar coefficients for the fit due to the similarities in magnitudes)
%Files in the respective SNR directory are then loaded one by one. %The
%sigmoid is fit based on channel 1 and applied to all channels before
%the channels are recombined to form the enhanced output file which is then
%written to a new directory.
%NB. Directory is of the form 'enhanced_<SNR>db' and must exist prior
%to the execution of this script.
%NB. In some cases due to values at the start of either file being orders
%of magnitude smaller than the rest, the curve fit will fail and a manual
%curve fit which ignores these values is required. Simply comment out the
%p = nlinfit(...) line and replace with p = [a b c d] after running a curve
%fit outside of this script and obtaining the p coefficients.
%NB.
%-------------------------------------------------------------------------

%Clear the workspace and console
clc
clear all;

%Get SNR from user (only in prototype)
SNR = input('Required SNR: ');

%Create the filter bank of 16 filters from 0 to 8000 Hz (16 kHz sampling)
[filters centre_freqs] = linfiltbank(8000,0,16,16000);

%-------------------------------------------------------------------------
%CURVE FITTING
%-------------------------------------------------------------------------
%Read the clean and noisy file in for curve fitting.
[sclean fs nbits] = wavread('C:\Thesis\evaluation\C39C030N.wav');
snoise =
wavread(strcat('C:\Thesis\evaluation_',num2str(SNR),'dB\','C39C030N.wav'));

%Filter each signal through the filter bank
for i = 1:16
    clean_channels(i,:) = filter(filters(i,:),1,sclean);
    snoise_channels(i,:) = filter(filters(i,:),1,snoise);
end

%Envelope of each channel for each signal
for i = 1:16
    clean_env(i,:) = exp(real(log(hilbert(clean_channels(i,:)))));
    snoise_env(i,:) = exp(real(log(hilbert(snoise_channels(i,:)))));
end

%Fit the curve
size = length(clean_env(1,:));
f = @(p,x) p(1) + p(2)./(1 + exp(-(x - p(3))/p(4)));
maxValue = max(snoise_env(1,:));
x = [0:maxValue/(size-1):maxValue];
p = nlinfit(x,sort(clean_env(1,:))./sort(snoise_env(1,:)),f,[0 1 1 1]);
%-------------------------------------------------------------------------

%Get the list of filenames
nFiles = dir(fullfile('C:','Thesis','evaluation','*.wav'));
```

```
%Loop through all the files one at a time
for t=1:length(nFiles)

    %Load the t'th input file from the respective SNR directory
    sinput =
wavread(strcat('C:\Thesis\evaluation_',num2str(SNR),'dB\',nFiles(t).name));

    %Obtain the 16 channel outputs by filtering through the filter bank
    for k = 1:16
        sinput_channels(k,:) = filter(filters(k,:),1,sinput);
    end

    %Get the envelope of each of the channels
    for k = 1:16
        sinput_env(k,:) = exp(real(log(hilbert(sinput_channels(k,:)))));
    end

    %Apply the sigmoid to channel 1 to obtain output channel 1
    sout(1,:) =
(f(p,sinput_env(1,:))).*(sinput_env(1,:)).*(exp(j*imag(log(hilbert(sinput_ch
annels(1,:))))));
    maxCh1 = max(sinput_env(1,:));

    %Apply the sigmoid to the rest of the channels
    for k = 2:16
        maxValue = max(sinput_env(k,:));
        sinput_env_norm = maxCh1.*sinput_env(k,:)./maxValue;
        sout(k,:) =
maxValue.*(f(p,sinput_env_norm)).*(sinput_env_norm).*(exp(j.*imag(log(hilber
t(sinput_channels(k,:)))))./maxCh1;
    end

    %Write the new file as the real part of the sum of all output channels
    %to enhanced_<SNR>db directory

wavwrite(real(sum(sout,1)),fs,nbits,strcat('C:\Thesis\enhanced_',num2str(SNR
),'db\',nFiles(t).name));

    %Clear variables and console ready for next iteration
    clc
    clear sinput;
    clear sinput_channels;
    clear sinput_env;
    clear sout;
    clear sinput_env_norm;

    %Status string to output to console as each file is completed
    str = sprintf('Completed file %d of %d',t,length(nFiles))

end
```

Appendix F: Auditory Model Confusion Matrices

30 dB SNR



25 dB SNR

20 dB SNR

15 dB SNR

10 dB SNR

5 dB SNR