18-551, Fall 2006
Group 8: Final Report

# *Say That Again?* Interactive Accent Decoder

| | |
|---|---|
| Cherlisa Tarpeh | ctarpeh@cmu.edu |
| Anthony Robinson | aarobins@andrew.cmu.edu |
| Candice Lawrence | clawrenc@andrew.cmu.edu |
| Chantelle Humphreys | chumphreys@andrew.cmu.edu |

## *Table of Contents*

# 1. Project Overview

### Project Description

Is your computer out of warranty? Is there a problem with your bank account? Do you need to schedule a flight but you don't have the internet ready? The common solution to these problems is to call a customer service department by phone. In the current world, you will seldom be greeted by a live representative, but instead an automated system. ASR (automated speech recognition) systems provide a cost-effective solution to the problem of busy customer service lines, unnecessary manpower and most importantly efficiency and ease of assisting customers.

While these systems achieve their goals for the most part, with the growing rate of non-native citizens in the country and globalization of the world, these American English Based systems will decrease in efficiency. It is critical to develop speech recognition systems which take into account the accent of its user.

### Our Solution

Our solution is to increase the efficiency of a typical ASR system by implementing an accent recognition system.

We will build an ASR that will take accents into account:

- The ASR will first determine the accent of the user
- It will switch codebooks (dictionary) upon accent detection
- The ASR will have a small database of 15 words

Our Implementation will utilize:

- MFCC (Mel Frequency Cepstral Coefficients)
- DTW (Dynamic Time Warping)

### Prior Work

There are two previous projects that we found to be similar to what we were to implement. Group 15 of Spring 2002 was the most similar.

Group 15 Spring 2002 implemented a language translation system

- Utilized Linear Predictive Coding to model the frequency signal.
- Used a Dynamic Time Warping Algorithm to compare signals
- Did not switch codebooks
- Had a ternary output and results were fairly low.

Group 11 Spring 2000 created a "Voice Recognition and Identification System"

- Utilized Linear Predictive Coding
- Vector Quantization to compare signals

# *2. Background Information*

### *Where do accents come from?*

Every language is equipped with a set of sounds which combined create all of the words in the language. These are called phonemes. For instance, the English word ‘cat’ consists of three phonemes: the phoneme that makes the /c/ sound, the phoneme that makes the /ae/ sound, and the phoneme that makes that /t/ sound. The English language contains phonemes that are not present in other languages such as the /ae/ sound in cat which is rarely found in other languages [1].

When an individual is learning English as a non-native language, they will often replace unknown phonemes of the English language with the closest phoneme present in their native language. An example of this occurs with a class of sounds called diphthongs, which are a combination of vowel sounds in one syllable exemplified in the words “seat” and “rounding”. Some languages, such as Japanese are lacking or completely void of diphthongs which would cause a native Japanese individual to replace the diphthong with a single vowel sound closest to it. This would mean a word such as seat being pronounced “sit”. [6] This is one of the ways that accented speech can be distinguished from native speech. Accented speech may also differ in rhythm, stressing and intonation.

### *How can we distinguish accents through mathematics and signal processing?*

In speech, there is a classification which distinguishes voiced speech from unvoiced speech. Voiced speech is described as the sounds that have a pitch. The ‘a’ in cake and the ‘i’ in write are classified as voiced speech. Examples of unvoiced speech can often be found in consonant sounds such as the ‘p’ and ‘t’ in pet (here the e is classified as voiced speech). Since voiced parts of speech create a tone when they are pronounced, they resonate. Formant frequencies are defined as the center frequencies of the resonances produced from the sound [2]. It has been noted that the F2 and F3 formants are very key in accent classification. The table below [2] displays the average frequency value for the F1, F2 and F3 formants for men.

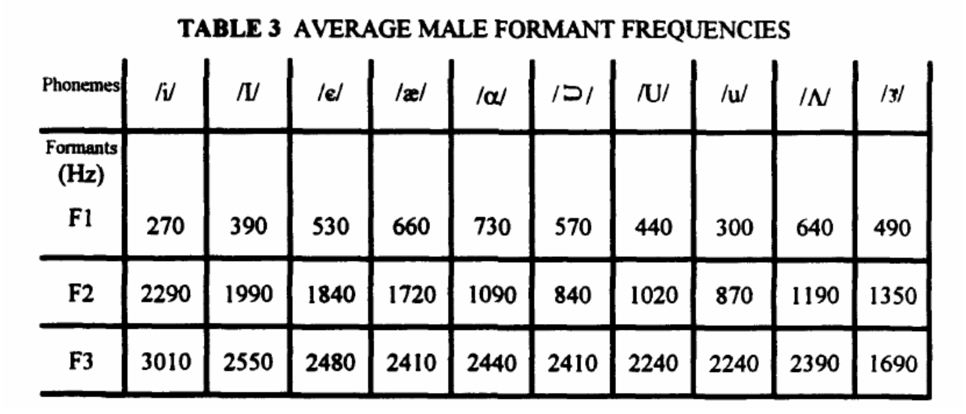


**TABLE 3 AVERAGE MALE FORMANT FREQUENCIES**

| Phonemes | /i/ | /I/ | /ɛ/ | /æ/ | /ɑ/ | /ɔ/ | /U/ | /u/ | /ʌ/ | /ɜ/ |
|---|---|---|---|---|---|---|---|---|---|---|
| Formants (Hz) F1 | 270 | 390 | 530 | 660 | 730 | 570 | 440 | 300 | 640 | 490 |
| F2 | 2290 | 1990 | 1840 | 1720 | 1090 | 840 | 1020 | 870 | 1190 | 1350 |
| F3 | 3010 | 2550 | 2480 | 2410 | 2440 | 2410 | 2240 | 2240 | 2390 | 1690 |

When the frequency spectrum of a speech signal is reviewed, generally it will not be smooth but rather a jagged signal. It is hard to determine from these signal where the formants actually occur. This is where signal processing becomes important. There are several methods that are used to smooth a signal in the frequency domain including Linear Predictive Coding, Auto-Regressive Modeling and the computation of Mel-Frequency Cepstral Coefficients. Once these signals are smoothened, it is fairly simple to recognize the peaks at which the formants occur. Combining this with the knowledge of phoneme replacement patterns concerning non-native English accents will allow a sound algorithm to be compiled that takes these factors into account.

# 3. Database Creation

### Accent Selection

To construct the optimal collection of words for our project, we conducted research on foreign accent studies. Speech instructors in the Carnegie Mellon Fine Arts department at Carnegie Mellon were particularly helpful. These professors (who have studied foreign accent speech) were able to provide us with information detailing the specific sounds that typically indicate non-native English speakers. From this research, we deduced that Singaporean and Turkish accents were the most distinct compared to the English language. Individuals native to Singapore do not have diphthongs in their accent, which prohibit them from correctly enunciating certain English words. For example, the word "seat" is pronounced "sit". Most Turkish natives are unable to pronounce the consonant "r" in their speech. This causes them to pronounce the word "corners" like "cah-nas". Thus, we concluded that these two accents possess characteristics that would facilitate the speech recognition process.

After identifying these accents, we constructed a list of the most mispronounced words for each accent based upon the availability of previously conducted research. Each word from the Singaporean list was then compared and contrasted with the pronunciation of that word by a Native English speaker and one with a Turkish accent, and vice versa. The list was then narrowed down based upon that evaluation. The final dictionary consisted of words where one of the three accents (Turkish, Singaporean, and English) could be identified.

In order to decrease pitch and formant variation in samples, data was recorded from males only. Samples of 15 male speakers from Turkey, Singapore and America (non-regional English speakers) were recorded for a total of 45 tested individuals. Each test subject was asked to articulate 29 words. Words were individually recorded using Windows Sound Recorder. Upon signal, each subject would read and pronounce a word from the list.

## *Wordlist by Accent of Significance*

| Turkish | Singaporean | American |
|---|---|---|
| Drilling | Seat | December-Thirtieth |
| Ben | Parliament | LaGuardia |
| Yam | Three | New York |
| Cup | Singapore | Istanbul |
| Thomas* | Thomas | Little |
| Pool* | January Third | |
| Pittsburgh | Economy | |
| Istanbul | Pool | |
| Bone | Catch | |
| Tricks | Target | |
| Pump | Yes | |
| Thirty* | No | |
| Orlando | Roundtrip | |

Successful words used in other studies: catch/target

## *Training Set*

In order for the codebooks to be created, we decided to use 30 samples for each word from our database (10 samples of each accent were included). Therefore, three codebooks would be created for each word (one for each language). They would be based upon an average of the ten samples taken.

## *Testing Set*

We utilized the remaining 15 samples (5 samples for each accent, for each word) for each word to test the classification and recognition of the system.

## *Total Word List*

| | | | |
|---|---|---|---|
| a) Ben | i) Catch | p) Orlando | w) Thirty |
| b) Seat | j) Three | q) No | x) Corners |
| c) Cup | k) Singapore | r) Roundtrip | y) Yam |
| d) Bone | l) Target | s) December-Thirtieth | z) Istanbul |

| | | | |
|---|---|---|---|
| e) Drilling | m) Yes | t) LaGuardia | aa) Parliament |
| f) Pittsburgh | n) Thomas | u) January-Third | bb) New York |
| g) Pump | o) Economy | v) Pool | cc) Little |
| h) Tricks | | | |

## 4. Algorithm Selection

Our original path in developing this system was to utilize an autoregressive algorithm based on work by John Dewey [2]. This work, implemented in MATLAB™, classified accents by using an autoregressive model in order to find the first three formant frequencies. The Itakura distance, cross-correlation coefficient, log spectral distance, bounds measures are computed and combined with the results from the AR model. This study was lacking in that it only compared Brazilian accents with native English accents. Also, while this study had a pool of 31 native English speakers, it had only 6 Brazilian individuals to use for training and testing. We hypothesized the measures in this study along with careful word choice and a larger testing pool would produce meaningful results.

In order to receive a professional technical opinion on our choice of algorithm, we looked to members of the SPHINX group (A team specializing in speech processing). While were unable to speak to Richard Stern, we spoke with another member of the SPHINX group, Mosur K Ravishankar to receive some insight and advice for the project. While he was not familiar with the autoregressive model he advised that we utilize Mel Frequency Cepstrum Coefficients for recognition and classification. He also provided us with contact information for Evandro Gouvea (Egouvea@andrew.cmu.edu) who wrote the code for the MFCC portion in the SPHINX project.

We sought a second opinion from Tanja Schultz, who has had some experience with accent recognition. She also was not familiar with the autoregressive model and suggested using Mel Frequency Cepstrum Coefficients (MFCCs) in order to extract features from the speech samples and a Gaussian Mixture Model as a classifier for the accent.

Due to the lack of reference related to the autoregressive model, and the strong suggestions to utilize Mel Frequency Cepstrum Coefficients, we chose to switch from using the AR Modeled Accent Recognizer to an algorithm based on using MFCCs.

### Algorithm

The final algorithm can be broken down into three main parts and calculations.

1. Pre-Processing: This consists of preparing the signal so that the desired features may be removed from it.

2. Feature Extraction: This is the step in which the Mel Frequency Cepstral Coefficients are extracted from the signal.
3. Classification: This is the final step in which the input signal is compared to another signal and classified accordingly.

## *Pre-processing*

### Silence Removal

When the speech samples were recorded, we allowed for a few seconds of time to pass before prompting the speaker to say the word and a few seconds afterwards. This guaranteed that the whole word would be captured in the recording. While silence removal can be performed manually, we wanted to be able to remove the silence automatically for the voice inputs to the actual system. In order to achieve this, we utilized the short-time energy calculation. Below is the equation used to compute the average energy of a signal. [3]

$$E_n = \sum_{k=1}^{Wn} |x[k]|^2 w[n-k]$$

$$E_{avg} = \frac{1}{N} \sum_{k=1}^{N} |x[k]|^2$$

Once the average energy is known, the signal is taken 80 samples at a time. The short-term energy of that block is calculated and if it is less than a certain percentage of the average energy (in this project we use 0.25*($E_{avg}$) as the threshold), the block is excluded from the final signal. This is done until you reach a block that satisfies the threshold condition. This is successful in removing the silence from the beginning of the signal. In order to remove the silence from the end, the same process is performed but from the tail end of the signal.

### Pre-Emphasis

A natural occurrence in speech causes a drop of six decibels for each increasing tonal octave in frequency. It is favorable to be able to review the frequency response of a signal at the same dynamic. For this reason, it is common to use a high pass FIR pre-processing filter to boost this drop and in turn flatten out the spectrum. This filter is simple to compute in the time domain as it takes the form

$$y[n] = x[n] - ax[n-1]$$

where $0.9<a<1$. In our project a will take the value of 0.95.

### Windowing

In order to obtain as many features as possible from the signal, it is typical in ASR systems to break the signal into small frames on the order of 20-30 ms.

To smooth the effects of the segmentation, windows are applied to each frame. Examples of windows are the rectangular window, Hamming window and the Hanning window. Most ASRs take advantage of the Hamming window's ease of computation as well as the results it produces. Below is the equation for the hamming window.

$$w[n] = 0.54 - 0.46\cos\left\{\frac{2\pi(n)}{N}\right\}$$

It is imperative to consider the situation in which the end of a frame contains important spectral information. Due to the weighting of the feature vectors and the hamming window, this important spectral information will not receive its rightful recognition. Therefore, instead of segmenting the signal into consecutive frames, there is usually a 50% overlap in the framing. This means that in a signal with frame size of 512 samples, a typical step size (or frame shift) would be 256 samples. The first frame would begin at the first sample (sample 1) and end at the 512th sample. The next frame would begin 256 samples after this frame, at sample 256 and end 512 sample later at sample 767. The total number of frames in a signal can be determined by computing R = (2L/M)-1 where L is the length of the sample and M is the frame size.

## Feature Extraction

The goal of feature extraction is to represent an input signal with a smaller yet just as informative vector. Extracting features also reduces the amount of unnecessary and irrelevant computation during classification. There are several methods for extracting features such as Autoregressive Modeling, Linear Predictive Coding (which utilizes an Auto-Regressive Model) and computing Mel Frequency Cepstral Coefficients. We chose to use Mel Frequency Coefficients due to their reputation to be more robust and reliable than Linear Predictive Coding which use a linear scale and more informational than regular Cepstral Coefficients which use a logarithmic scale.

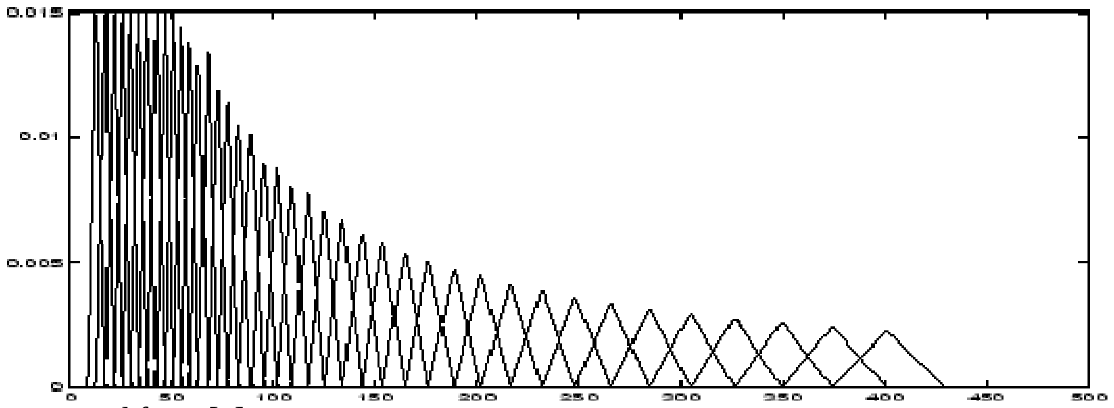
MFCCs utilize a Mel Scale which models the human auditory system. Humans have the ability to distinguish a difference of 50 Hz in low frequencies but not in high frequencies and this scale reflect this. [4] In order to produce the MFCCs of a frame in a signal the following steps are taken.

1. Compute the Discrete Fourier Transform and use this to compute the power spectrum.

$$S[k] = (\mathrm{real}(X[k]))^2 + (\mathrm{imag}(X[k]))^2$$ [5]

2. Convert to the Mel Spectrum (multiplication of power spectrum by each triangular filter.)

$$\tilde{S}[l] = \sum_{k=0}^{N/2} S[k]M_l[k] \qquad l = 0, 1, \ldots, L-1$$ [5]



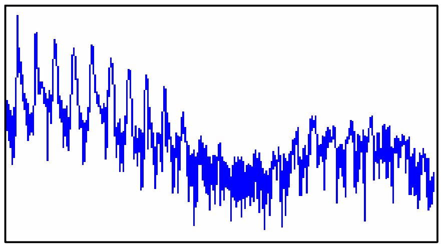
Extracted from [5]

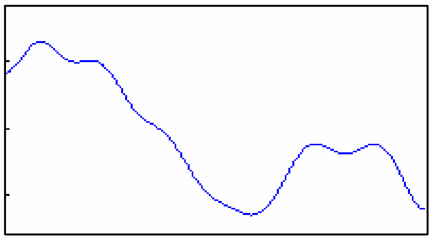
Mel Frequency Filter Bank – Series of Bandpass Filters

3. Compute the Discrete Cosine Transform of the natural log of the Mel Spectrum in order to determine the Mel Frequency Spectral Coefficients. .

$$c[n] = \sum_{i=0}^{L-1} \ln(\tilde{S}[i]) \cos\left(\frac{\pi n}{2L}(2i+1)\right) \qquad c = 0,1,\ldots,C\text{-}1$$ [5]

Once the coefficients for each frame have been computed, they may be combined to produce smoothened version of the original signal. We utilized the first 13 Mel Frequency Coefficients which is standard in ASR systems.


Original Spectrum Signal


Signal obtained from computing MFCC

## *Dynamic Time Warping*

Dynamic Time Warping utilizes a common computer science technique called Dynamic Programming to compute the minimum distance and shortest paths between two input vectors. It searches for the shortest path to get from one vector to the other, utilizing a matrix structure. How is this important to this project? DTW determines that the difference between a person saying “Booone” and “Bone” to be relatively close despite the difference in signal length. By searching for the shortest path as opposed to the Euclidean distance, word speed and pitch do not play as large of a role in computing similarities.

The algorithm begins by storing the local distance (Euclidean distance) between frames of the two input vectors in a matrix called the local distance matrix. The input vector is placed on the x axis and the reference vector on the y axis. Each cell in the input and reference vectors represent the MFCCs in a frame. For example, the third cell in the input vector represents the computed MFCCs of the third frame of the input signal.

Cells in the local distance matrix are obtained by computing the distance between frames of the input and reference vectors corresponding to the particular cell. Cell (3,4) in the local distance matrix represents the distance between frame 3 of the input vector and frame 4 of the reference vector.

**Example of Local Distance Matrix**

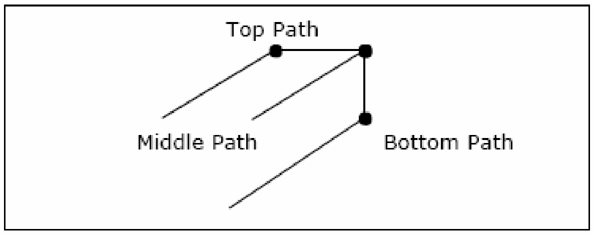
Reference Vector

| | | | | |
|---|---|---|---|---|
| Frame 4 | 5 | 6 | 7 | 7 |
| Frame 3 | 9 | 3 | 4 | 8 |
| Frame 2 | 5 | 1 | 2 | 1 |
| Frame 1 | 2 | 3 | 9 | 2 |
| | Frame 1 | Frame 2 | Frame 3 | Frame 4 |

Input Vector

A global distance matrix is then computed in order to determine the total minimum distance from the starting point (0,0) to the ending point (n,m) where n is the size of the input vector and m is the size of the reference vector. Each cell in the matrix is representative of the minimum distance it takes to get from the initial point to that particular cell. This matrix is initialized by setting (0,0) to the value of (0,0) in the local distance matrix and (1,1) equal to the local (0,0) + local (0,1) . It also initializes the rest of the first row and first column to a very large number (Here we use 1000). [6]

Group 14 from Spring 2002 utilized code written by Tony Robinson along with additions they added to create a complete DTW algorithm. Along with including restraints for obvious observations (i.e., the path cannot go backwards, the end is at the top right corner, etc.) their DTW algorithm took into account the fact that the paths from the input to the reference should not be too steep or shallow as that may cause inaccuracies in the added feature of time and pitch alignment. Based upon this, it was determined that there are three reasonable paths to reach any given cell in the global matrix. Therefore, each cell that has not been pre- initialized or set to the large number (1000), is determined by computing the bottom path, middle path, top path and taking the minimum.



A representation of the possible paths in the global matrix. [6]

**Example of Global Distance Matrix – The minimum distance between these vectors is 14 as found in cell (4,4)**

Reference Vector

| | | | | |
|---|---|---|---|---|
| Frame 4 | 1000 | 1000 | 17 | 14 |
| Frame 3 | 1000 | 1000 | 7 | 18 |
| Frame 2 | 1000 | 3 | 1002 | 1001 |
| Frame 1 | 2 | 1000 | 1000 | 1000 |
| | Frame 1 | Frame 2 | Frame 3 | Frame 4 |

Input Vector

This algorithm is obviously very useful in classification as it compares two input signals. It also is helpful in training a system with the use of backtracking. Backtracking produces the shortest path taken to receive the minimum distance between the inputs. If two speech signals of an American Male saying “Bone” are recorded and compared, the shortest path between these two will be. in effect. an average of their speech. If this is done multiple times, comparing each successive person to the average vector, a codebook is created.

# 5. Signal Flow

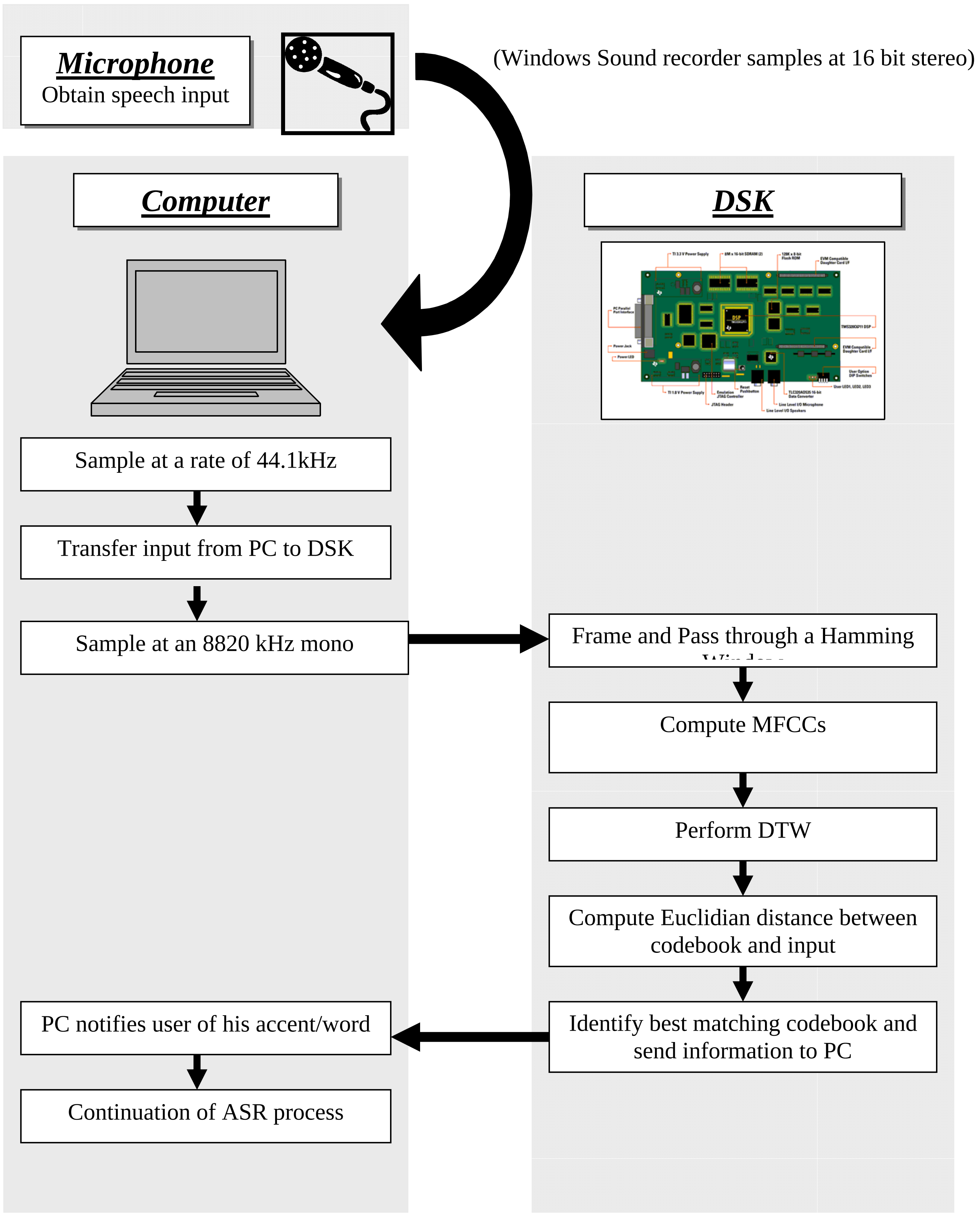
## Accent Detection and Word Recognition

To determine the accent of the user and the word he is saying the input signal must go through several stages. First the GUI that we implemented must call MATLAB™ and prompt it to begin receiving input from the microphone. The users speech is sampled 8820 kHz and the samples are written to a file and saved. The PC side (Microsoft Visual C++ 6.0) then reads the file and sends the samples to the DSK. Once the DSK receives the users speech it performs two pre-processing functions. First silence removal and pre-emphasis filter is applied. After the signal has been prepared appropriately it is divided into frames that 256 samples long and passed through a hamming window. For each frame 13 Mel frequency cepstral coefficients are calculated and then stored into a two dimensional array (Nx13 where n is the number of frames the signal has). Once the MFCCs of the input signal are calculated they are compared to the previously calculated MFCCs of each existing code book through use of dynamic time warping. DTW calculates the distance between the input and existing codebook to signify which stored word or accent the input signal is closest too.

## Training the System / Codebook creation

To train the system and create a codebook for each word (in each accent-base) several steps were followed. First we had 15 people from each accent base speak into a microphone and we sampled their speech at a 44.1 kHz sampling frequency using Windows Sound Recorder. Ten of the fifteen people are used to create a codebook for a given word in that accent base. We then used MATLAB™ to resample the speech at a sampling frequency of 8820 kHz and remove silence from the beginning and the end of the speech. The resampled speech was written to a file to be stored. The PC side (Microsoft Visual C++ 6.0) then reads in the samples, pre-emphasizes and passes them through the Hamming window for framing. For each frame 13 Mel frequency cepstral coefficients are calculated and stored into a two dimensional array. If there is existing information in the codebook, dynamic time warping is carried out with the newly input speech and existing information. An average of the existing information and the new information then replaces the existing codebook. In the initial case where there is no information in the book the DTW is carried out against an array of zeros. Once all ten speakers have been used to create this ideal average of how the word should be represented, the codebook is written to a text file to be used later on.

## Data Flow Graph


Microphone
Obtain speech input
(Windows Sound recorder samples at 16 bit stereo)
Computer
DSK
Sample at a rate of 44.1kHz
Transfer input from PC to DSK
Sample at an 8820 kHz mono
Frame and Pass through a Hamming
Compute MFCCs
Perform DTW
Compute Euclidian distance between codebook and input
Identify best matching codebook and send information to PC
PC notifies user of his accent/word
Continuation of ASR process

# 6. Comparison, Data and Results

## DSK Speed Results: Latency, Speed, and Profiling

### Overhead

There are 30 codebooks ranging in size from 4kb – 12kb
Total data sent from PC to DSK = 229kb
Transfer rate of 10 MB/s totals a time of (229kb / 10 MB) seconds = 22.9ms

User input speech on average is 100kb
Transfer rate of 10 MB/s totals a time of (100kb / 10 MB) seconds = 10ms

Sending back information back to the pc is negligible because only 3 ints are sent back at a transfer rate of 2.5 MB/s from the DSK to the PC.

### Profiling the Algorithm

We obtained the following results from profiling functions

| *Function* | *Code Size* | *Average* |
|---|---|---|
| Sdtw() | 1048 | 167814 |
| mfccs() | 912 | 120546 |
| SHammwindow() | 76 | 10146 |
| Smatrix | 296 | 970 |

Sdtw()- performs Dynamic Time Warping
mfccs()- computes Mel frequency cepstral coefficients
SHammwindow() – smoothes out signal using hamming window
Smatrix() – produces a matrix to put info in 2-D form

### Attempts to reduce time

In an attempt to help speed things up arrays that were used to store the Hamming windows, framing windows and cepstral weights were stored on internal memory.

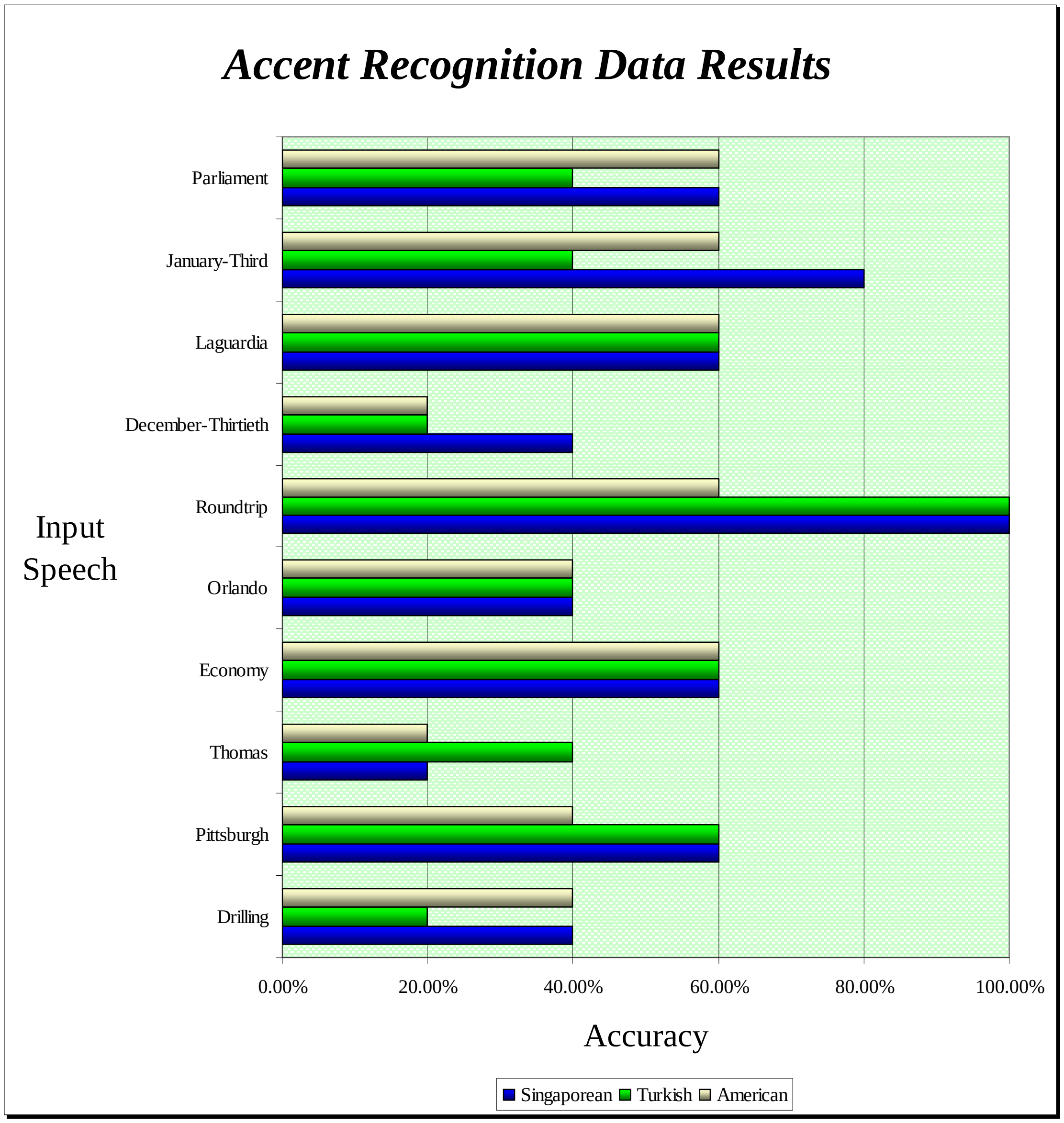
The dynamic time warping function was unrolled as well in order to save time.

However, despite our attempts we weren't able to get our ASR to work in real time, but A user wouldn't really notice a 1 second delay that much.

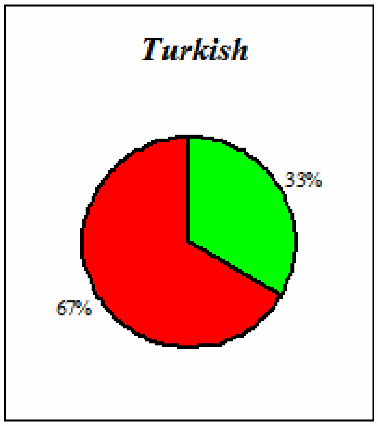
Our data results describe two different areas we tested. Our word recognition results show the accuracy in which we detected the word that the user said in a particular accent base. For example the accuracy percentage for 'Drilling' in the Singapore Accent base is 1/5 which means for the five times we loaded a testing file with a different user saying 'Drilling' only once did the ASR correctly detect that the user said Drilling. In this case, the ASR knew the accent, but not the word that was being said.

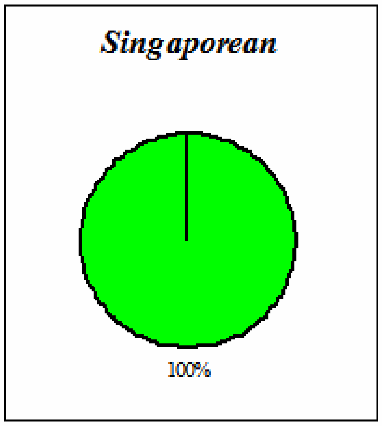
When detecting the accent we tested a speaker saying the three appropriate words and we did this with three different speakers per accent base. In this case, the system knew which word was going to be said. For example after saying the words the ASR requested three different Singaporean speakers all were able to have their accent classified correctly thus resulting in a statistic of 3/3.

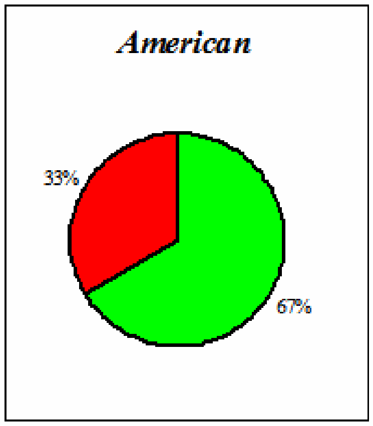
Graphical Depictions of Data



***Note:*** *the accuracy recorded is the result of testing the five speakers sampled that were excluded from the codebook for each accent.*







## Available Code

This site contained MATLAB™ code for removing silence from the beginning and end of signals. It came from a similar class project where students were attempting to implement a speaker recognition system.
http://web.mit.edu/~sharat/www/
Chikkerur, S et al. “Speaker Recognition” State University of New York at Buffalo. 2003

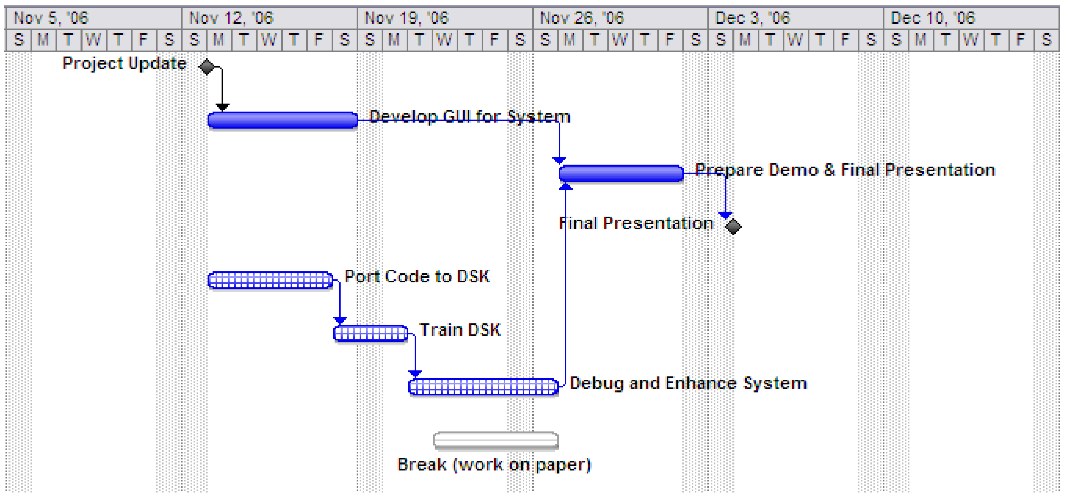
C code for DTW needed to be modified – this came from group 14 2002.

MFCC C-code
provided by Evandro Gouvea from the Sphinx Group, http://cmusphinx.sourceforge.net/html/download.php#sphinxbase but fairly hard to adapt to this implementation.

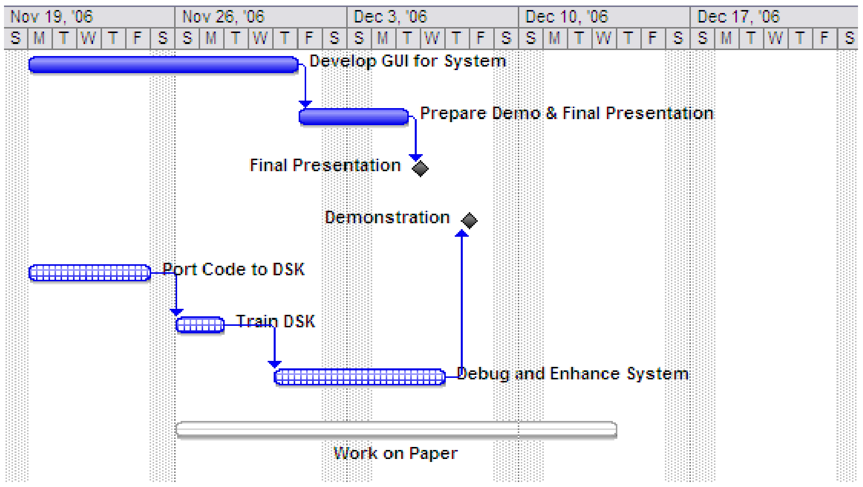
MFCC Matlab Code
http://www.ofai.at/~elias.pampalk/ma/documentation.html

# 7. Group Information

## Desired Schedule



## Actual Schedule



## Main Task Division

**Chantelle**
GUI Layout
Accent Research

**Cherlisa**
MATLAB™ Implementation
Algorithm Research

Training and Testing

**Anthony**
Modified and Wrote C code
DSK Implementation and Optimization
Training and Testing

**Candice**

GUI Impletmenation
Data Preparation

# 7. Demonstration

## Ideal Implementation

Ideally, the demonstration of our project would present an ordinary ASR system and a "super" ASR system that would take into account the accent of the user, if deemed necessary. The graphical user interface would allow the user to select between the ordinary and enhanced ASR system at the commencement of the program. Both systems would take the user through a simulated customer service hotline for booking flight reservations.

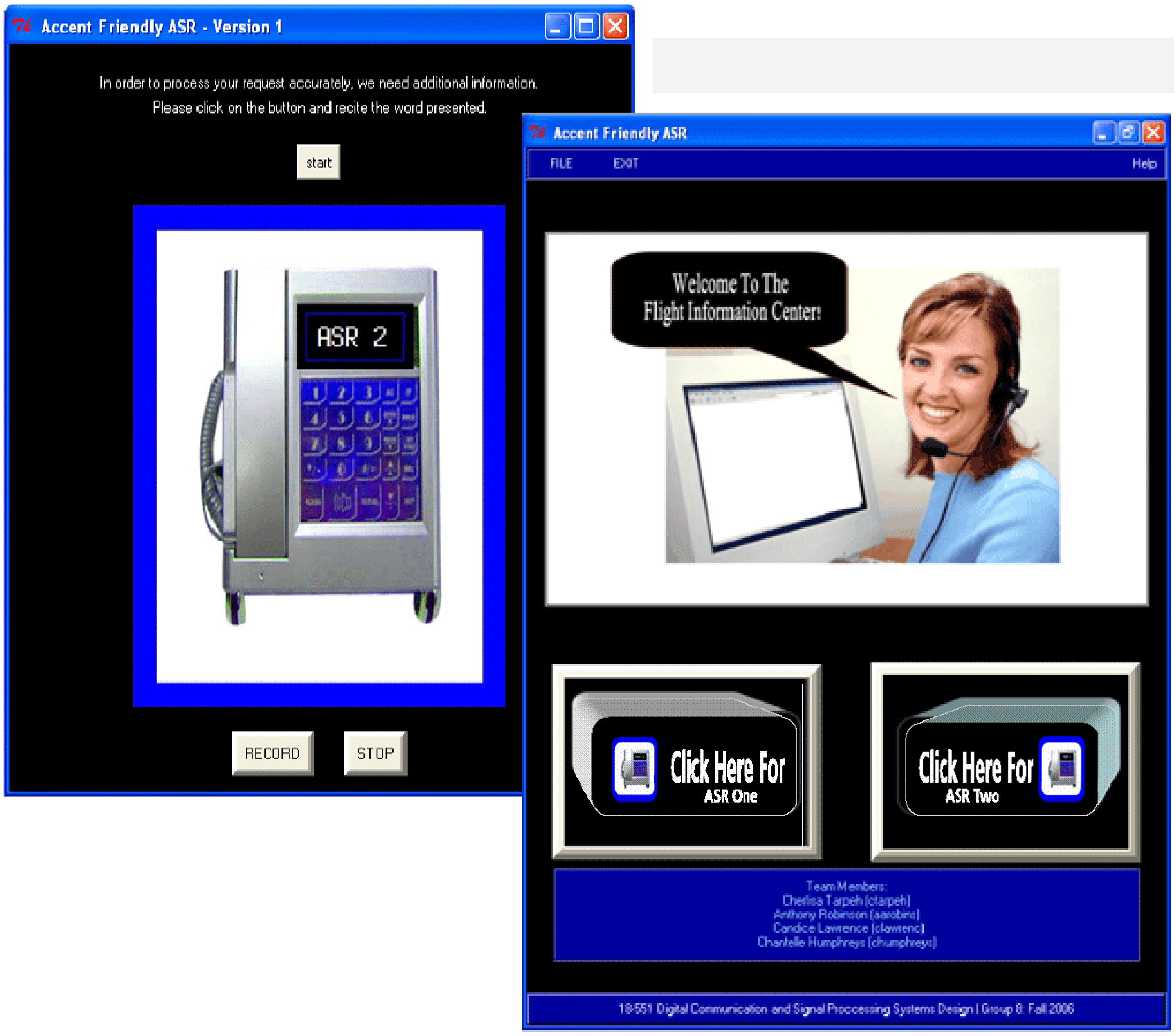
The first system would ask the user questions, prompting one word answers. After each question, the program would record the sound clip of the user's speech, and clip the recording. The recording would then be sent to the DSK for processing, and the DSK would indicate which word was said. At the end of this process, the GUI would display the word said (as calculated by the program) as confirmation. The user would then be able to indicate if the program is correct. At the conclusion of the questioning, the GUI would display statistical information detailing the general efficiency of the system.

The second system would first prompt the user to articulate 5 words for the sole purpose of detecting the accent of the user. Hypothetically, the customer service line tells the user that they will receive a discount for saying these words. After saying each word, the recording would be sent to the DSK for comparison with our generated codebooks. After all five words have been said, the program would employ a voting system, where the accent with the most "votes" would be chosen. The ASR would then switch to the corresponding codebook, and then proceed in the same manner as the first ASR system.

## Actual Implementation

Due to complications, we were not able to implement our ideal program in its entirety. Our graphical user interface was used solely for recording samples. MATLAB™ instructions sent via Perl were executed, creating a file in the format of the configure samples obtained during data collection. Two options were presented: the first option sends the user to a screen to record a word. The second section sends the user to a screen to record the five words for accent comparison. In our demonstration, we utilized the second option, and recorded three words to detect the accent of the user. These files were passed into the C code that analyzed each recorded sample. The accent chosen was the one that received the best vote out of three.

## *Graphical User Interface (GUI)*



In order to process your request accurately, we need additional information.
Please click on the button and recite the word presented.

Team Members:
Cherlisa Tarpeh (ctarpeh)
Anthony Robinson (aarobins)
Candice Lawrence (clawrenc)
Chantelle Humphreys (chumphreys)

18-551 Digital Communication and Signal Processing Systems Design | Group 8: Fall 2006

## 9. Further Work

Looking back, there are a few things that we think might have improved the success of this project. Taking into consideration that this was an accent classification system, we may have wanted to put more of a stress on extracting the formants as either an extra classifier or the main classifier. In a post-project MATLAB testing, it was found that there was a noticeable pattern in computing the F2 and F3 formants of the one syllable test words.

In reviewing the work with the MFCCs, we don't' believe that using a smaller amount of coefficients would have made a noticeable difference. When the MFCCs were viewed in a MATLAB graph, they all looked to be important to distinguishing the accents. It may have been useful to isolate certain parts of a word and only compute the MFCCs of part of the word (voiced or unvoiced) that we were hoping would be the classifier.

We also think that trying to compile more test samples and implementing a Gaussian Mixture Model along with the DTW may have increased the classification and recognition significantly. Ideally, it would be optimal to produce a system that does not rely on codebooks to identify accents, especially considering how outliers effect an average.

## *10. References*

[1] L. Arslan, J.H.L Hansen, "A study of Frequency Characteristics in Foreign Accent," IEEE ICASSP-97 : Inter. Conf. on Acoustics, speech Signal Processing, vol. 2, pp. 1123-1126, Munich, Germany, April 1997
*Provided information about accents and how they can be classified as well as one approach for accent recognition using an HMM

[2] Dewey,John K. Speech Recognition of Foreign Accent. NAVAL POSTGRADUATE SCHOOL MONTEREY CA. June 1994
*Provided initial algorithm as well as information about formant frequencies in their relation to accents.

[3]Jun Xu; Ariyaeeinia, A; Sotudeh, R; Zaki Ahmad; "Pre-processing speech signals in FPGAs," ASIC, 2005. ASICION 2005. 6$^{TH}$ International Conference On, Volume 2, 24-27 Oct. 2005
*Information and equations about pre-processing and silence removal

[4] Martens, John Pierre Continuous Speech Recognition Over the Telephone. Final Report – May 2000
* Gave some information about typical ASRs and Feature Extraction.

[5]  Seltzer, Michael. Sphinx III Signal Processing Front End Specifications. CMU Speech Group, 31 August 1999
*Provided detailed process information about computing MFCCs

[6] Group 14. "I didn't know you could speak my language" 18-551, 2002

[7] Deshpande, S.; Chikkerur, S.; Govindaraju, V.; "Accent classification in speech" Automatic Identification Advanced Technologies, 2005.  Fourth IEEE Workshop on"
17-18 Oct. 2005
*Provided information about how formant frequencies are important to accent recognition as well as a formant and GMM based approach to attacking the problem.

[9] Levent M. Arslan and John H.L.Hansen.; “Language Accent Classification in American English” Speech Communication Journal vol 18, No 4, pp 35-367. August 1996,

[9] Sigurdur Sigurdsson, Kaare Brandt Petersen and Tue Lehn-Schioler ; “Mel Frequency Cepstral Coefficients: An Evaluation of Robustness of MP3 Encoded Music” Proceedings of the Seventh International Conference on Music Information Retrieval (ISMIR) 2006.
*Provided helpful information about MFCCs usefulness and implementation.