|  |
| --- |
|  |
| Touch Tone Recognition |
| Progress Report 1 |
|  |
| **EE301** |
| **3/29/2010** |

|  |
| --- |
|  |

Group 1:

Norman Chung

Rocky Mark Juan

Hieu Nguyen

Alexander Nobles

Bryce Toth

**Summary**

Our dual tone multiple frequency (DTMF) Touch Tone Recognition project has been moving along. As a group, we have reviewed and discussed the theoretical concepts from a signal and systems perspective. In order to facilitate our progress, we set some goals and split up some of the work to do individually over the spring break. Each team member was given responsibility for a part of the project. Having just gotten back from spring break last week, we still need to reconvene and discuss our findings, difficulties, and new ideas. We plan to meet sometime this week to figure out our next steps, which include implementing the touch-tone recognition in Matlab, preparing for our in-class presentation, and documenting our progress for the final report.

Before the spring break, our team found some existing literature and example code on the internet, from which we have divided into five parts for each member to focus on individually. The following is a snippet of what each member has been working on. Norman has been learning how to implement a DTMF encoder algorithm in Matlab. By setting row and column frequencies, he is able to output an audible DTMF signal and store it into a wav file. Because Alexander has the most experience related to the project, he looked through some of his previous Matlab projects to begin work on developing a filter to identify signals. He found a function that finds the Fourier series coefficients of the argument, an array with an even number of elements, which will be useful in designing filters in Matlab. Bryce worked on developing and implementing the DTMF dialer code, which takes the input on the keypad and then, based on the given frequencies for each number, generates a different tone for the different numbers and symbols. Hieu focused on a scoring function to indicate if a specific frequency was present in a DTMF signal segment, such as the one generated by a touch tone. This function is essential for determining the frequency components of a DTMF signal and mapping it to its corresponding touch-tone. Rocky has been…

Although we have each worked significantly on our individual assignments, we still face the challenging task of integrating each part together into a seamless touch-tone generator/analyzer/recognizer. The next step is to meet as a group to set new goals and assignments, and to begin putting our Matlab code together for testing.

**Norman Chung**

The DTMF signals are recognized by creating a function called dtmf\_encoder. In order to implement the algorithm on a filter, the plan was to go through these steps: First, we model the algorithm in a m-file called dtmf\_encoder. This file defines y to be dtmf\_encoder(r,c,T). R stands for row, c stands for column, T stands for period which is set to be 0.1 second. Let time to be a row vector of 800 points linearly spaced between 0 and 0.1. Then we define fs, T, fr, fc, and then let pause signal to be zero for the first 320 points. We set row frequency to be 697, 770, 852, 941Hz and column frequency to be 1209, 1336, 1477, 1633Hz. We use the digits\_entry to be enter input digits. We set the first digit duration to be 800 points. If we enter number 1, then the row frequency is the first item which is 697Hz, and the column frequency is the first item which is 1209Hz. If we enter number 8, then the row frequency is the third item which is 852Hz, and the column frequency is the second item which is 1336Hz. We define 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, A, B, C, D. If there is entry other than the above inputs, the output frequency is zero. In order to make sounds of each input, we use the function soundsc( ) which corresponding to the each recognized digit. We also stored the sound in a wave file by the function wavwrite( ) and dld function. If we input one digit, then the length of the signal is 1440 which is 320+800+320. If we input two digit, the length of the signal is 2560 which is 1440+800+320. If we input three digit, the length of the signal is 3680 which is 2560+800+320, and so on.

The next step is to create filters to exclude certain frequencies so that the system is able to recognize the input by stating out the row and column frequencies at each digit. We define the order of the filters to be twentieth order. The fundamental frequency is 8000 as before. We have eight filters which corresponding to four row frequencies and four column frequencies. The first column frequency is 697Hz, so we set the first filter to be greater than 0.17\*4000 which is 680, and less than 0.18\*4000 which is 720. The second column frequency is 770Hz, so we set the second filter to be greater than 0.1875\*4000 which is 750, and less than 0.1975\*4000 which is 790, and so on. In order to show all eight graphs on one picture, we use the function subplot 425, which stands for four row, two column, and the fifth sub-picture. We use function fir1(n,Wn) to design a 20th order FIR bandpass filter with passband from 680 to 720Hz.

**Rocky Mark Juan**

**Hieu Nguyen**

So far, I have not made very much progress on this project. The majority of the time I’ve spent working on the project has been devoted to learning the theoretical concepts and the Matlab implementation. Our team found some existing literature and example code on the internet, from which we have divided into five parts for each member to focus on individually.

My part of the code included a function that would be used to indicate if a specific frequency was present in a dual tone multiple frequency (DTMF) signal segment, such as the one generated by a touch tone. Given a DTMF segment and the impulse responses for a specific band-pass filter, the function should return a binary decision indicating if there is a sinusoid, with frequency equal to the center frequency of the band-pass filter defined by the impulse response, present in the DTMF segment. At first, I was unsure where to begin, but after reading through some related documentation, I eventually figured out how the code would be implemented. The scoring function would scale and convolve the DTMF input with a band-pass filter, and look through the convolution output for the maximum value (the frequency of the pass-band).

I also looked through the example code pertaining to the other parts of our DTMF recognition project. For the most part, it was straightforward, except for parts dealing with band-pass filters. The example Matlab code defined the impulse response of a band-pass filter using an “L-point FIR filter”; essentially, it defined the amplitude and frequency coefficients of a cosine, to create a unique pass-band. After going through the documentation, I tested, debugged, and compiled the code on my personal computer. It worked spectacularly, and I figured out some of the limitations that arise when the sampling frequency is too small.

Some things I need to get started on include researching exactly how to take in an analog audio input from a computer microphone, formatting the useful data into a vector (to be used in our DTMF recognition code), and learning how to parse the DTMF sound file to recognize timing differences for other applications. I have already looked into the audiorecorder function in Matlab, but I will need to do more testing to figure out how to integrate it with our code. Once that is complete, we should be able to start analyzing DTMF signals in real-time.

**Alexander Nobles**

To begin the work on developing a filter to identify the signals, I looked through some previous Matlab projects I had done on the subject. Unfortunately, I seem to have misplaced all of my related notes, and all the projects are from several semesters back, so I have forgotten the context for which these code files apply.

To begin work on coding a bandpass filter for the project, I looked through a lab on the topic that I completed in EE 200. Among this code is a function that finds the Fourier series coefficients of the argument, an array with an even number of elements, N. The function returns two arrays of length N/2 + 1, the first as an array of the amplitudes of the sinusoidal components of the argument and the second as an array of phases. Both these arrays are discrete, but since we expect very specific frequency values for the argument, this should not be an issue.

Understanding that the Fourier series of the argument will be discrete, it should be trivial to filter out unwanted frequencies; in Matlab, all that needs to be coded is a for loop that runs through a spectrum of frequencies and zeroes out all frequencies except the one we are looking for in that particular iteration of the filter.

However, there is still the matter of capturing an analog signal and reading it into the filter. The signals used in that lab were simulated with an 8000 samples per second rate, rather than recorded and sampled from an analog source.

**Bryce Toth**

My task for the project is the initial step in the coding. I am working on developing and implementing the DTMF dialer code. It takes the input on the keypad and then, based on the given frequencies for each number, generates a different tone for the different numbers and symbols. This will all be done in MATLAB, which I am not completely proficient in as of yet, but am learning the steps necessary in implementing my section of the code. Depending on the input, the code will generate a tone that last for 0.5 sec with the associated frequencies for the input, followed by 0.2 sec of silence so that the system can recognize the separation in the text strings entered. All data entered will be stored in vectors in MATLAB and saved for later processing. Basically, my section of the code takes the raw input data and generates a string of tones depending on which numbers where entered, stores the data in vectors and passes the information on to the next step. The form that the information passed to the next step will be through DTMF signals, essentially a convolution of two distinct frequencies to be read later for processing.

A basic outline of my code will include the following: a user input function for the input of the numbers from the keyboard, then to run a for loop to determine the number of input digits, followed by an if loop to determine its value. From here, the code will generate the appropriate tone for each value and store them in a vector to be sent to the rest of the system.

From here, the DTMF signal will be indexed so that a start and end point are noted for each tone that is generated in step 1. This allows our system to easily distinguish each different input, which will be separated for later processing. After this, our system will filter the signal so that the frequencies of each input can be isolated. This step will use filtering and Fourier Transform to take the signal into the frequency domain. After this, our last step will be to recognize the specific tones that are generated and then to output a string of the original input. In short, the process takes a string input, generates DTMF tones corresponding to the input, the system analyzes and recognizes the tones, and produces back the original input.

As far as writing the code, I will need to look back on previous classes to brush up on my coding. My section is relatively simple, though, so I am comfortable in my ability to complete it proficiently.