Ila Torfin

April 27, 2014

Computer Science Research

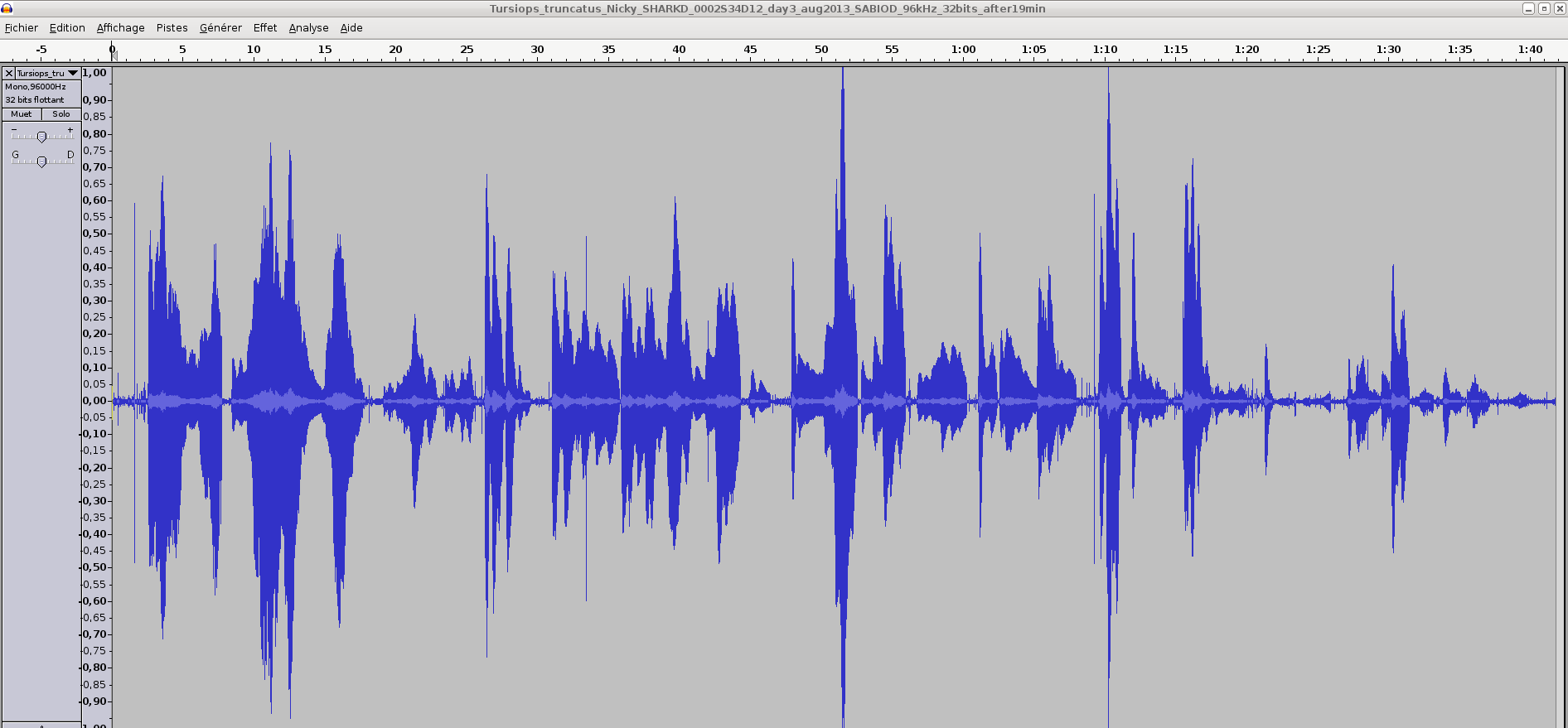
B. Koster

# Sonar

**1.1 Introduction**

In the beginning of January, 2014, my advisor, Barrett Koster, and I came together and decided to research making a sonar-like program which will take a sound recording, find the distance between the initial sound and the echoes that follow in order to find the size of a surrounding room. The speed of sound for above sea level is 340 meters per second.

Sonar is a system for the detection of objects under water and for measuring the water's depth by emitting sound pulses and detecting or measuring their return after being reflected. It is used to navigate, communicate with, or detect objects on or under the surface of water. Submarines use this system in order to know the topography of underwater and to know if there are other vessels nearby. There are two types of sonar, passive sonar, which listens for sound, and active sonar which emits a pulse sound and listens for the echo. In our research, we use passive sonar which listens for the initial sound and the echo afterwards. The speed of sound in water is about 3316 meters per second and about 1530 meters per second in salt water. Typically, the sound signal looks similar like a wave centered at a zero value, like the figure shown below.

figure 1.1

**1.2 Method**

In order to first begin this program, we needed to understand the coding process behind getting sound recorded and analyzed. We incorporated simple audio recording code from Matthias Pfisterer's that enabled us to get the sound recorded and the sound data recordable in order to graph the wave. This code starts a timer function which is set to half a second, plenty of time to record a sound, and enters the data into an array, known as the wave array here after, which we read and graph in order to see the wave form.

Once the sound was recorded and the data was collected, we wanted to be able to see a closer up of the exact moment the sound was created. To do that, we initialized another array to hold 512 samples from the wave array, which was named wave1. In order to see the initial sound, we had to be able to set the beginning of wave1 to the array position of wave which held that data, thus we needed a grapher function which would graph that exact point.

For the function, we first created a background on which the points would be most visible. In the sonar program, there are three distinct arrays which needed to be displayed. Thus, we created another program, Grapher, which does all the graphing calculations separately, draws a “ground zero” line which will show the positive and negative points. It then takes the data from the array it was sent from the sonar program, known to be SWave here after, and plots a polynomial line. SWave then takes the sample from which the user clicked on the wave graph and sets the first position of wave1 to that sample and does this for 512 samples. The code is shown below:

**public** **void** alignWave()

{

**int** offset = graph.sample;

System.*out*.println("alignWave: offset="+offset);

**for**( **int** i=0; i<8000; i++ )

{

wave1[i] = wave[i+offset];

}

doSum( wave1 );

validate(); repaint();

}

The next step was to smooth out the wave1 array so that we could calculate the distance from the initial sound and the echo which was recorded soon after. Thus, we summed up the values in wave1, in the doSum() method, took the absolute value of the values and graphed that array so that we could visually see how the data seemed. Then, we needed to calculate the distance between the initial sound and the echo by using the sum array. We found the absolute max of the sum array, which is the initial sound, then found a minimum to find the next maximum, which is the immediate echo from the sound. Below is the code for finding the sums:

**public** **void** doSum( **double** w[] )

{

**int** var = 300;

**for**( **int** i = 0; i < 8000 - 2\*var; i++ )

{

sum[i] = 0;

**for**( **int** n=0; n < var\*2 ; n++ )

{

**double** a = Math.*abs*( w[i+n] );

**double** d = Math.*exp*(-((n - var)\*(n - var))/(2\*var\*var));

sum[i] += a\*d;

}

}

doDistance(sum);

}

Similarly, here is the code for finding the distance:

**public** **void** doDistance( **double**[] s )

{

**int** var = 200;

**for**( **int** i = 0; i < 8000 - 2\*var; i++ )

{

absoluteMax = s[0];

**if**( s[i] > absoluteMax )

{

absoluteMax = s[i];

k = i;

}

}

**for**( **int** i = k+50; i < 8000 - 2\*var; i++ )

{

**double** minimum = s[k+50];

**if**( s[i] < minimum )

{

minimum = s[i];

j = i;

**if**( s[i+1] > minimum )

{

**break**;

}

}

}

**for**( **int** n=j; n < 8000 - 2\*var; n++ )

{

relativeMax = s[j];

**if**( s[n] > relativeMax )

{

relativeMax = s[n];

m = n;

}

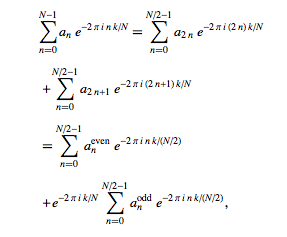
}

System.*out*.println("K: " + k + ", J: " + j + ", M: " + m );

**double** distance = m - k;

System.*out*.println( "Distance: "+distance );

}

Also used in order to analyze the sound data was Fast Fourier Transform. Fast Fourier Transform is, according to WolframAlpha, a [discrete Fourier transform](http://mathworld.wolfram.com/DiscreteFourierTransform.html) [algorithm](http://mathworld.wolfram.com/Algorithm.html) which reduces the number of computations needed for  points from   to , where [log](http://mathworld.wolfram.com/Lg.html) is the base-2 [logarithm](http://mathworld.wolfram.com/Logarithm.html). This method uses [aliasing](http://mathworld.wolfram.com/Aliasing.html) (also known as leakage) can be reduced by “[apodization](http://mathworld.wolfram.com/Apodization.html) using an[apodization function](http://mathworld.wolfram.com/ApodizationFunction.html)”. However, [aliasing](http://mathworld.wolfram.com/Aliasing.html) reduction has the expense of broadening the spectral response. The FFT formula is

In order to use this formula and analyze the sound wave, we needed to create an array of complex numbers of the variables from the wave array. Thus, we created another program which did all the calculations and all the mathematical arithmetic of a complex number. Once figuring the complex array, we then proceeded to find the auto correlation of the complex numbers, which told us the amount of times that sound sample rotated around the unit circle. In the end, we did not use it to calculate the distance; however, we plan to use it the information given by FFT and the auto correlation in order to reduce damping of the sound samples and apply it into the Laplace Transform which will alter damping and make the data liable, ideally.

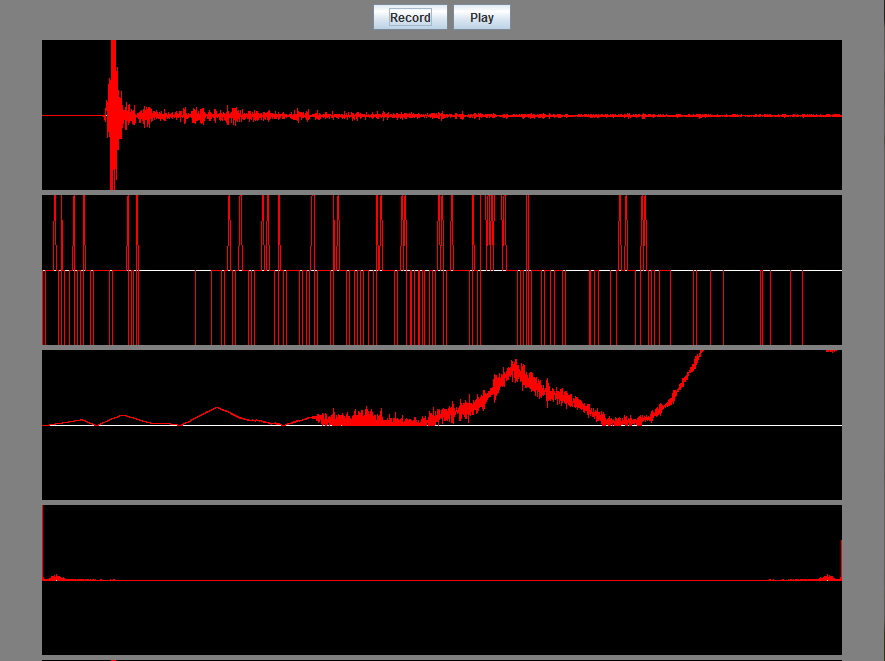
**1.3 Results**

After working out all the bugs and errors, we were able to collect data and compare it to other data which sets up the platform to calculate distance in meters of how big the room is.

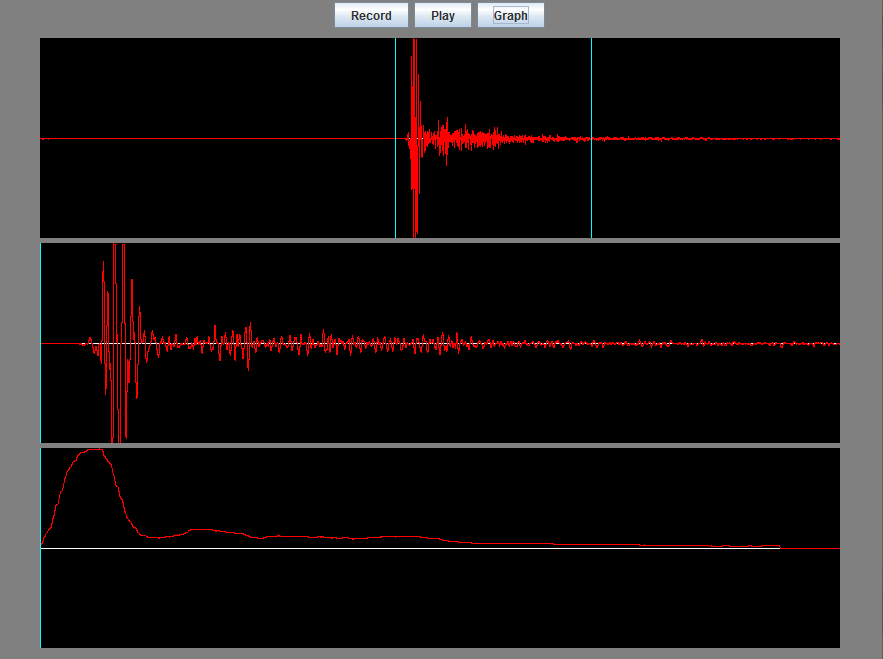
The sound we recorded was a typical hand clap done by me, Barrett Koster, or another colleague. Our results of distances from initial sound and echo are as follows:

* Mr. Koster’s office:
  + 61
  + 77
  + 76
* Meredith College Science and Math Building Atrium:
  + 82
  + 89
  + 84

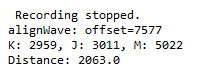
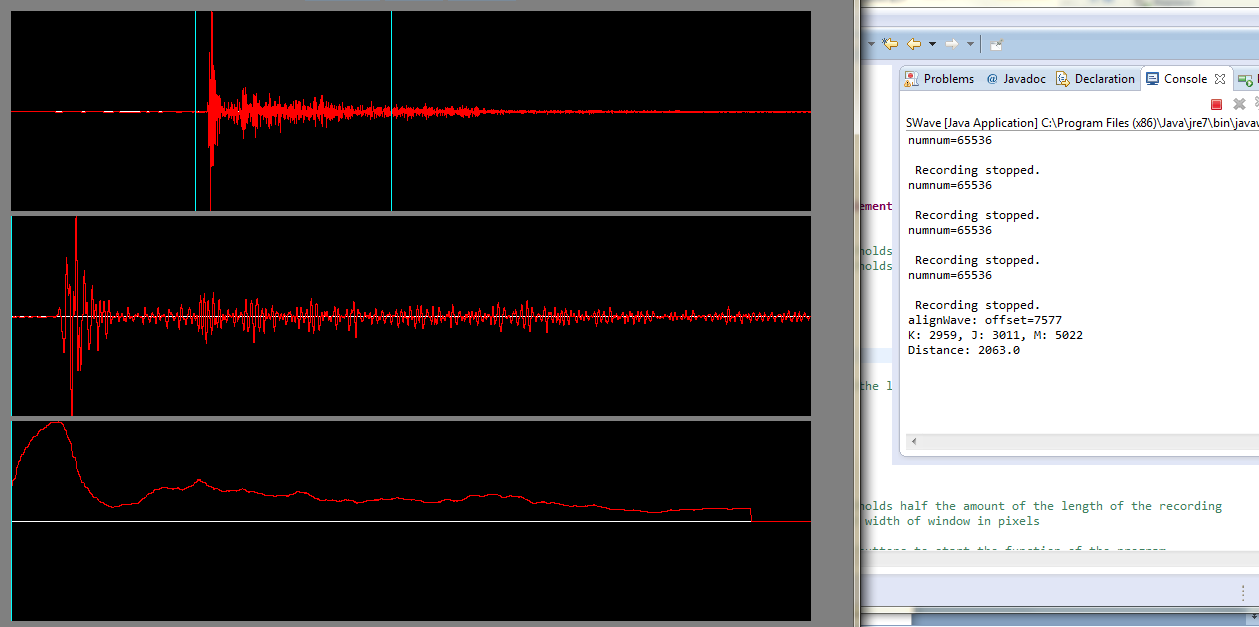
Here are some examples of the data being graphed:



*This diagram shows a clap in the Science and Math Building of Meredith College in the Atrium, which is the main entrance to the building. The first graph is the whole sound data, the second is the first 500 samples, the third shows the rotations around the unit circle each sound, and the last graph shows the Fast Fourier Transform before we stopped using it. This is what the beginning of our process shows*



*This figure shows sound data from a clap in the SMB Computer BitLab room. The whole sound wave, the 512 samples from where I clicked on the first graph to show the main sound data, and, finally, the smoothed out data which distance is calculated from.*

*This figure depicts the sound data taken from a clap in the hallway outside of the Science and Math Building Computer BitLab. This figure includes the distance calculation which is closer shown below this picture. This depicts the end product of the research we have done so far.*

**1.4 Analysis**

With sonar program is working thus far, it is a good start to future work. We have data to see and be able to analyze to tell the distance of a room. However, due to sound damping, the echo not taking into account the absorption of sound between an open area and a closed one with a lot of surfaces to absorb and rebound sound. As mentioned earlier, we will need to use the Laplace Transform in order to reduce damping effects. Also, we can use the Fast Fourier Transform of reverberations to help reduce the damping and use another sound that has less variations of sound than a hand clap.

**1.5 Conclusion**

In conclusion, working on this and getting it to work as been one of the best experience I have had while in college. In the future, I hope to be able to take the distance and map out the room. To do that, I must find all the distances of the echoes, find how many meters away and then set up a scaling method which will scale the distances into a model which will be 3D and show the user a model of the room which they are in. After having the program do that, I will work on being able to model a whole building.