Neural Networks

EL E 462 - Senior Design II

Formal Report

Started January 28, 2018

Honor Code Statement - In keeping with the Honor Code of the School of Engineering, I have neither copied laboratory report material, given any help to, nor received any help from any of my fellow students in the preparation of this laboratory report.

Chris Carpenter
William Jared Ufferman
Nawaf Aljezani

May 7, 2018

Contents

1 Abstract

Neural networks can be used to analyze an individual's positional data in order to determine who the person in question is. Our Senior Design team worked on developing a system in which one's identity can be guessed based on one's gait, sway, or general movement traits. Based on our findings, the system was modified as the resolution of initial measurement device was not high enough to properly train a neural network, and a new method was used to record positional data. The goal of this report was to review our design process and reflect on the methods we used to implement the design, as well as review the changed made in the final implementation.

2 Introduction

Our senior design project's goal was to use neural networks in order to 'guess' who an individual is based on their positional data, or data that was extracted or derived from said positional data. Neural networks have a variety of applications for military and civilian use, and can be used in order to observe patterns in human behavior. Initially, our design utilized a LiDAR infrared camera in order to measure distance data. Later, a HTC Vive remote was used to measure position in 3 dimensions. The data collected throughout the project was analyzed by MATLAB and used to train a neural network, which would be presented a test file of an individual who the network would attempt to identify. Throughout the course of the project,

recording techniques were updated, as the use of a tripod allowed us to re-record needed data and the HTC Vive remote allowed us to obtain higher resolution positional data.

3 Overview of The Design

Initially, our team began researching how to collect data from the LiDAR sensor, as well as study how neural networks work and what neural networks can achieve. We eventually managed to properly wire a USB-UART connector and an Arduino to a PC to collect data from the LiDAR sensor. However, we found that the Arduino's software had difficulty running, and we managed to find and edit code in C that would allow us to collect data using the USB-UART device connected to the LiDAR sensor. We recorded data by walking towards the camera, roughly from 4 meters away, and stopping roughly 30 cm in front of the camera. Most recordings were intended to train the neural network, and a few recordings were reserved for testing purposes. This data was recorded containing a sample number, distance, and signal strength data. The distance data was extracted and used in MATLAB for analysis. Using discrete fourier transforms, a graph of acceleration vs. frequency was able to be found. Additionally, using MATLAB we were able to plot acceleration vs. time and velocity vs. time. Dr. Goggans provided us with a tripod mount for the LiDAR sensor, which we used to re-record some data that we could not use due to the subject not being directly in the

line of sight of the camera. After all data was recorded, we initially intended to use the Google Tensorflow neural network toolbox to train and test the network to identify us based on our movement. However, we decided to use Python and the neural.py package that Dr. Jones suggested. Before the data could be fed into the neural network, it was preprocessed such that the values were normalized and it skipped the first 50 or so recorded values such that it only analyzed movement data.The neural network was trained by using our position as the inputs, and each target was a corresponding decimal value associated with one of us. The acceleration was also used as an input in another trial. It was found that the network could not identify an individual, despite adjustments made to the number of repetitions or if the network was trained on data sets of 2 or 3 people at once. After meeting with Dr. Jones once more, we were certain that the LiDAR camera simply did not have the proper resolution to train the network, and we set about to prove this by using an HTC Vive remote to record and process movement data through the same neural network in Python. The data was obtained by wearing the remote in a drawstring bag on our backs and walking 4 meters down a hall. Positional data in X, Y, and Z coordinates was recorded. The Y coordinates corresponded with sway, with the Y axis representing the subject's left and right motions. It was found that observable differences in sway could be seen, with each subject having a distinct plot of distance vs. time. Data was recorded for training and testing the neural network, and it was found that out the neural network was able to accurately predict the subject's identity

97 % of the time.

4 Setting up LiDAR

In order to record data, we had an Arduino Mega (Figure 1) and a USB-UART device that we could connect to LiDAR infrared camera for data collection. We initially attempted to use the Arduino, yet had difficulty in installing the drivers, and could not consistently record data with it. At the same time, we also attempted to use C code meant for use with the USB-UART connector, which was wired to the LiDAR sensor as according to Figure 3.

4.1 C Code Debugging

After finding C code that could read position data through the UART device, the code was modified to properly parse and collect the proper bytes to in order to record position data. Two bytes contained distance information, with one by containing high value bits and another containing low value bits, which was added to the bitshifted high value byte to obtain the position value, as can be seen in Figure 4. The same method was used to extract signal strength information. These values each were recorded with their corresponding sample number. The LiDar sensor operated at approximately 100 Hz and recorded roughly 100 sample per second. The C code was optimized and a glitch that would cause outlier values was found and fixed.

LiDAR position data could now be recorded reliably.

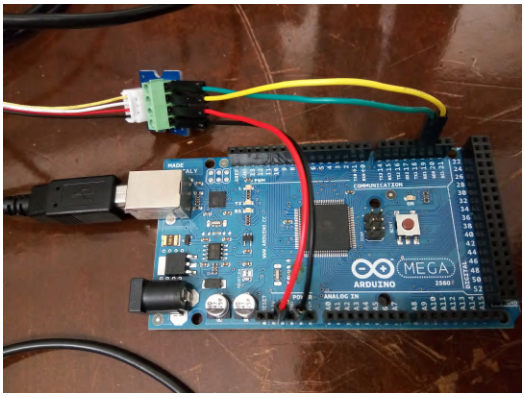

Figure 1: Arduino Mega

Figure 2: LiDAR infrared camera

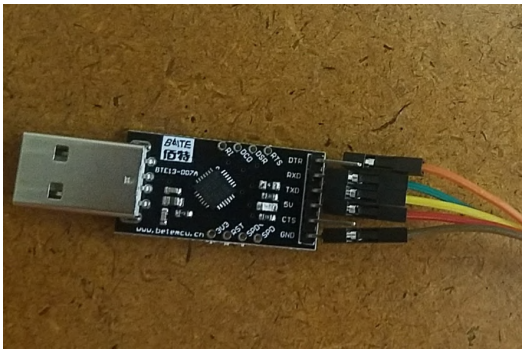

Figure 3: USB-UART

```
for (p = buf; rdlen-- > 0; p++){
    current = *p;
    if(current == 0x59)
        count = 1;
    else if(count==1 && current != 0x59 && prev==0x59){
        b3 = current;
        count++;
    }
    else if(count == 2 && current != 0x59 && prev == b3 && prev2 == 0x59){
        b4 = current;
        count++;
    }
    else if(count == 3 && current != 0x59 && prev == b4 && prev2 == b3){
        b5 = current;
        count++;
    }
    else if(count == 4 && current != 0x59 && prev == b5 && prev2 == b4){
        b6 = current;
        count++;
    }
    else if(count == 5 && current != 0x59 && prev == b6 && prev2 == b5){
        b7 = current;
        count++;
    }
    else if(count == 6 && current != 0x59 && prev == b7 && prev2 == b6)
    {
        count++;
        b4<<=8;
        b4+=b3;
        b6 <<= 8;
        b6 += b5;
        printf("%d,%d,%d\n",i++,b4,b6);

    }
    else
        count = 0;
    prev2 = prev;
    prev = current;
}
```

Figure 4: The C code used to read postion data

5 Initial Data Collection

1. Our first attempt at data collection was using the Lidar Camera and USB-UART adapter connected to a personal laptop. We set up the

laptop and Lidar camera with enough space for the subject to walk about 2.5-3 meters toward the camera. This worked, but there was a serious issue. The Lidar sent the distance data back in an inconvenient fashion and a parsing algorithm had to be created to read the data correctly. Until this issue was fixed, we attempted another way to read the data.

2. Our second attempt was using the Lidar and and a Arduino Uno Board. This would theoretically fix the way the data was being sent back so we use the data. The company that sells of the Lidar camera, Seeed, has a wiki that has many examples using Arduino boards so we anticipated the setup to be seamless. A problem arose when we discovered the Arduino Uno only had 1 UART sender/receiver and 2 were needed by the Lidar. One for receiving the data, and one for sending the data to a monitor. This required another change.

3. We switched our design to use the Arduino Mega Board instead of the Uno. This board had the required hardware and once connected, we were ready to record more data. At this time, the parsing algorithm was finished. We decided on going back to the USB-UART adapter since we needed to re-record the data anyway and we were more comfortable with the Adapter than the Arduino.

4. We re-recorded all the data that was unusable using the USB-UART and everything worked as planned. We each were recorded 40 times. 35

were for training and 5 were for testing. Using the parsing algorithm, we were able to print out the distance data to new files to be sent off to MATLAB to be analyzed.

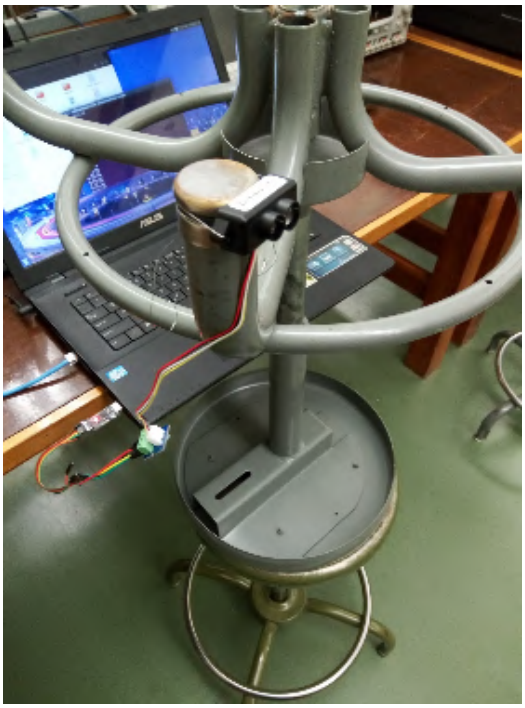

Figure 5: Prior to using a tripod, this is how we set up the camera

6 Extracting Acceleration Information (MATLAB)

Using MATLAB, our parsed position data was used in order to obtain a plot of acceleration vs. frequency through the use of a discrete fourier

transform. Using this, we were able to obtain plots of acceleration vs time and velocity vs. time. We found that acceleration vs time plot only changed in 1 m/s^2 increments, which was unusual (See Figure 9).

```
clc
clear

[num,T,vT]=xlsread('NawafWalk1.xlsx');


Distance = num(:, 3);

Fs = 100;
dt = 1/Fs;
l = length(Distance);

%t = (0:dt:l-1);
t = ((0:l-1)*dt)';

plot(t,Distance)
title({'Distance vs. Time';'x(t)'})
xlabel('Time (seconds)')
ylabel('Distance(cm)')
figure

NFFT = 2^nextpow2(l);


f = Fs*linspace(0,.5,NFFT/2+1);

x = fft(Distance,NFFT)/l;

xmag = abs(x);
Power=xmag.^2;

plot(f,Power(1:NFFT/2+1))
%plot(f,2*abs(x(1:NFFT/2+1)))
title('Single-Sided Amplitude Spectrum of x(t)')
xlabel('Frequency (Hz)')
ylabel('|x(f)|')
figure
```

Figure 6: Old MATLAB Code pt 1

```
for j = 1:257
    Acceleration(j) = x(j)' .* ((2*pi*1i*f(j)).^2);
end
%Acceleration = (x) .* ((2*pi*1i*f).^2);

Amag = abs(Acceleration);
%Amag = sqrt((real(Acceleration)).^2+(imag(Acceleration).^2));

plot(f,Acceleration)
title('Acceleration vs. Frequency')
xlabel('Frequency (Hz)')
ylabel('Acceleration')
figure

plot(f,Amag)
title('Magnitude of Acceleration vs. Frequency')
xlabel('Frequency (Hz)')
ylabel('Magnitude of Acceleration')
figure


% y is the acceleration in time domain
y = diff(diff(Distance));

l1 = length(y);
t1 = ((0:l1-1)*dt)';
plot(t1,y)
title('Acceleration vs. Time')
xlabel('Time (seconds)')
ylabel('Acceleration')
```

Figure 7: Old MATLAB Code pt 2

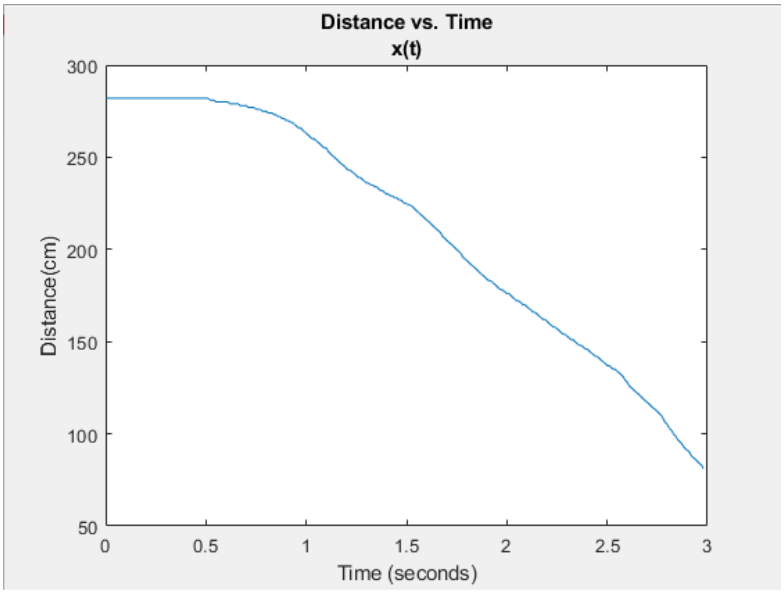

Figure 8: Distance vs Time

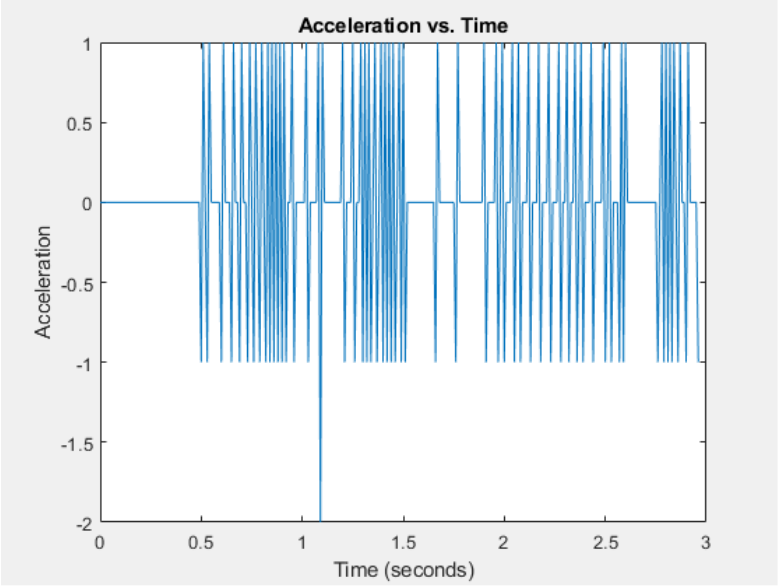

Figure 9: Acceleration vs Time

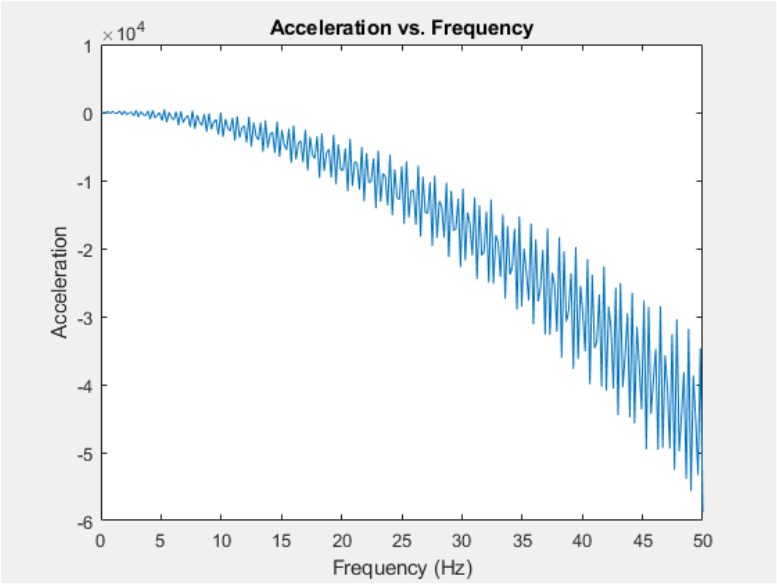

Figure 10: Acceleration vs Frequency

7 New Data Collection and Analysis

After initial MATLAB analysis, we consulted Dr. Goggans who helped us edit our MATLAB code in order to obtain smoother plots for acceleration vs time (See Figure 16). Dr. Goggans provided us with a tripod for use with our LiDAR camera, as seen in Figure 11, which was used to re-record data in which the subject was not in the camera's line of sight.

A problem with the data was from how the camera recorded data. If the subject walked out of the infrared beam of the LiDAR, the beam took the position data of the wall. This made the data unusable.

Figure 11: Tripod LiDAR camera

```
clc
clear

%
%[num,T,vT]=xlsread('NawafWalk2.xlsx');
%clear('T','vT')

filename = 'J1.txt';
delimiterIn = ',';
headerlinesIn = 1;
Data = importdata(filename,delimiterIn,headerlinesIn);
num = Data.data;
num(:, 3) = [];
%
L = length(num);
Offset = 20;
N = 2^(nextpow2(L+Offset));
%x = zeros(N,1);
xRaw = zeros(N,1);
x = zeros(N,1);
xRaw(1+Offset:L+Offset,1) = num(1:L, 2)/100; % Measured Distance in meters
xRaw(1:Offset,1) = ones(Offset,1)*num(1,2)/100;
Fs = 100; % Sampling Frequency in Hz
Ts = 1/Fs;  % Sampling Time in seconds
% The position data is measured in integer numbers of centimeters.
% To determine a smooth estimate of the acceleration via
% the finite difference approximation the
% position data is low pass filtered before using the difference formula.
lpFilt = designfilt('lowpassiir','FilterOrder',2, ...
        'DesignMethod','butter',...
         'HalfPowerFrequency',10, ...
         'SampleRate',Fs);
xTemp = filter(lpFilt,xRaw);
% Remove first Offset samples
x(1:N-Offset,1) = xTemp(Offset+1:N,1);
%
figure(1)
```

Figure 12: New MATLAB Code pt 1

```
t = ((0:N-1)*Ts)';
plot(t,x)
title({'Distance vs. Time';'x(t)'})
xlabel('Time (seconds)')
ylabel('x (meters)')

v = zeros(N,1);
a = zeros(N,1);
v(2:L-1,1) = ( x(3:L,1) - x(1:L-2,1))/(2*Ts) ;
a(2:L-1,1) = ( x(1:L-2,1) + x(3:L,1) - 2*x(2:L-1,1))/Ts^2 ;
figure(2)
plot(t,v)
xlabel('Time (seconds)')
ylabel('v (meters/second)')
figure(3)
plot(t,a)
xlabel('Time (seconds)')
ylabel('a (meters/second-squared)')

A = fft(a)*Ts;  % m/s
f = (0:N-1)'*Fs/N;
fa = f;
for I=(N/2 +1):N
    fa(I)=f(I)-Fs;
end
figure(4)
hold on
plot(fftshift(fa),fftshift(real(A)))
plot(fftshift(fa),fftshift(imag(A)))
xlabel('Frequency (Hz)')
ylabel('real(X), imag(X) (meters/second^2-Hz)')
figure(5)
hold on
plot(fftshift(fa),fftshift(abs(A)))
xlabel('Frequency (Hz)')
ylabel('abs(X) (meters/second^2-Hz)')
```

Figure 13: New MATLAB Code pt 2

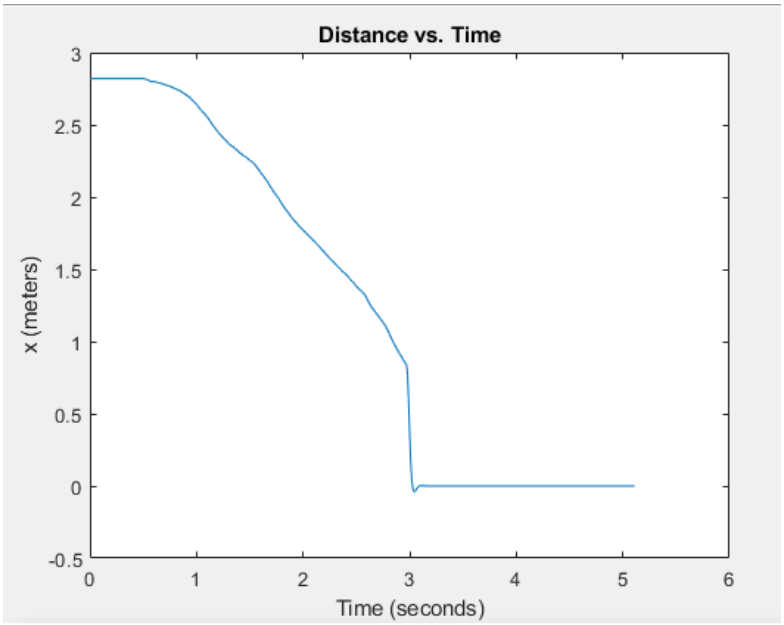

Figure 14: Distance vs Time

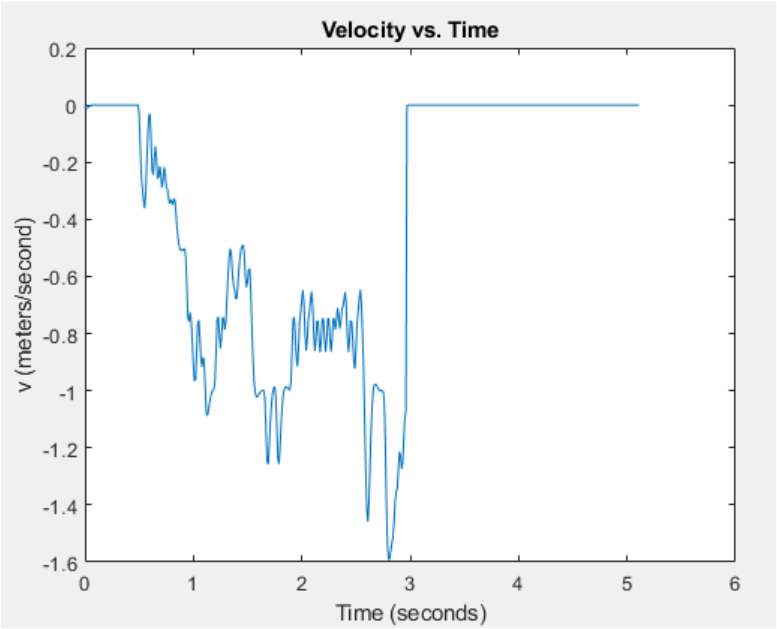

Figure 15: Velocity vs Time

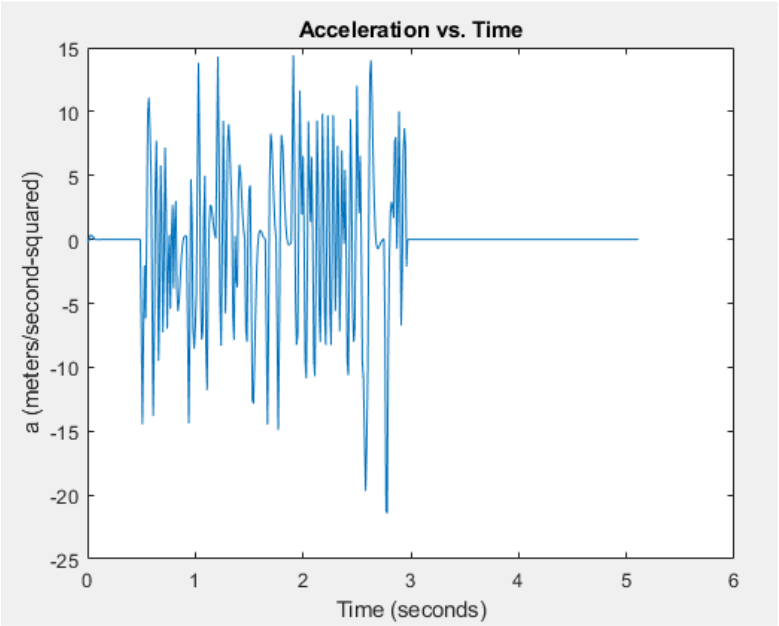

Figure 16: Acceleration vs Time

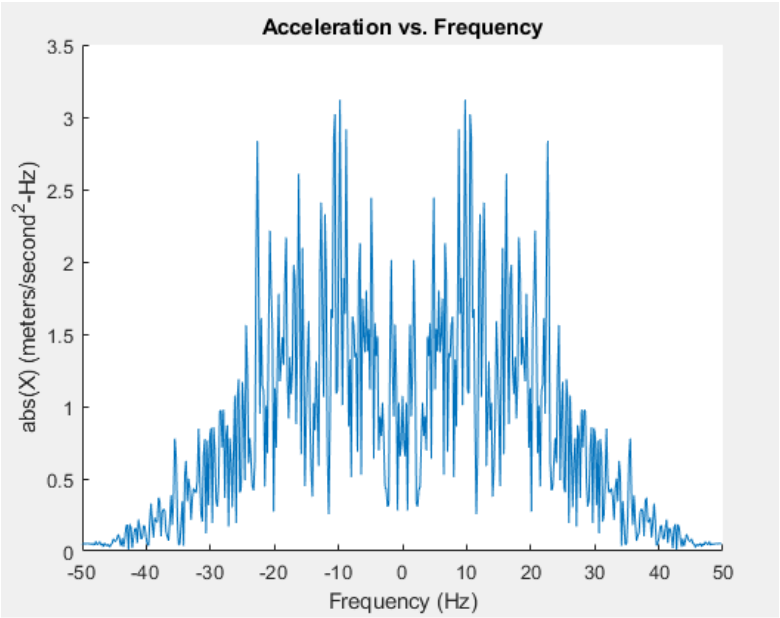

Figure 17: Acceleration vs Frequency

8 Neural Networks and LiDAR

It was at this point we needed to discuss with Dr. Jones. At our first meeting with him, he taught us how to process the data and what he recommended. He explained the basics of what a neural network is, and showed us some python examples to help guide us in creating a neural network. He also told us about using just 2 targets instead of all 3 to help the network make a decision. Since there were only two targets, the decimal values used for each person's target would be either 0.0 or 1.0, and the neural network's predicted value should be very close to one of these two values. As we created

our neural network, we encountered a few issues.

8.1 Data Preprocessing

Before data could be used to train neural networks, it had to be processed such that it could be used effectively. The values for position/acceleration were normalized, and the first 50 values of each recording were skipped such that only movement data was being analyzed. The inputs for the neural network consisted of 60 different recordings, 30 per subject, with 30 samples taken per recording initially, which was later raised to 100. The targets consisted of 40 values, each value a decimal corresponding to a walking subject. 5 recordings for each of us were used to test the network.

8.2 Difficulties with LiDAR Sensor

One of our python libraries, num.py, was not working as intended, which led to a second meeting with Dr. Jones in the VR Lab. The neural network was trained at different repetition values, but no changes could change the predicted value, which was stuck at 0.79 as seen in Figure 18. We showed him our python code and we tried to debug it as a group. Some errors were corrected, but the network was still behaving incorrectly. This led to the possible conclusion that the data from the camera was not enough to train a neural network. We decided to see if our initial plan to try to create and train a neural network using gait data was still possible using data from the

VR Lab. The equipment used in the VR Lab had better resolution so it could possibly make a difference. We took some initial data, and decided on a day where we could collect all the data we needed.

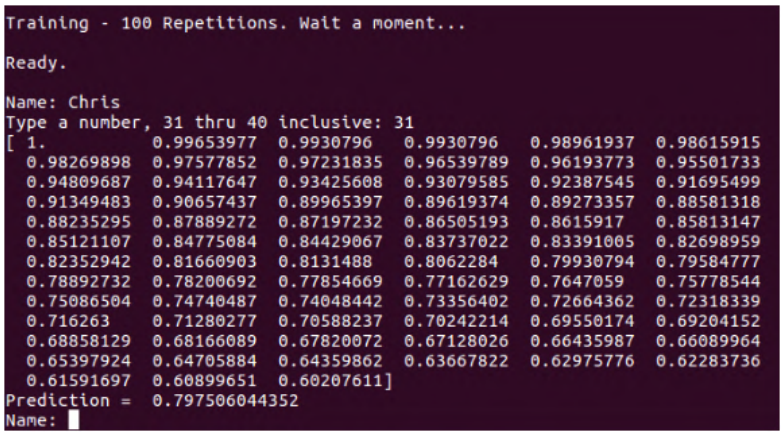

Figure 18: The neural network did not seem to want to work with the LiDAR data

9 Final Data Collection (HTC Vive)

Under the supervision of Dr. Jones, we set up for data collection in the VR Lab. The equipment used for the data collection was a HTC Vive controller placed via a small backpack at the base of the neck, seen in Figure 19. We set up a 4 meter walkway, making sure it was inside the area where the Vive's data could be received. We each had 25 files. 20 for training and 5 for testing. The Vive sent back the position data in 3 dimensions. In relation to the direction we walked, one was the movement left and right

(sway). Another dimension measured movement up and down (bob). The third dimension measured movement forward and backward. Upon reviewing the data we noticed the sway to be very unique between all three of us. We decided to train a neural network using the sway data to see if it could successfully determine who was the person walking when it is given some data.

Figure 19: The HTC Vive remote was worn on our backs in a drawstring bag

10 Neural Networks and HTC Vive

Initial review of the HTC Vive position data recordings revealed distinctions between our Y-coordinate, or sway characteristics, as can be seen in Figures 20 and 21. A neural network was trained using 40 recordings of 300 samples each for the inputs, and 40 values for targets, with 20 recordings and

20 values per subject. It was found that the neural network could accurately determine the identity of a subject 96.7 percent of the time, or 29 out of 30 total tests. An example of a test run of the neural network can be seen in Figure 22, in which Jared's identity (1.0) is corrected guessed with a value of 0.97.

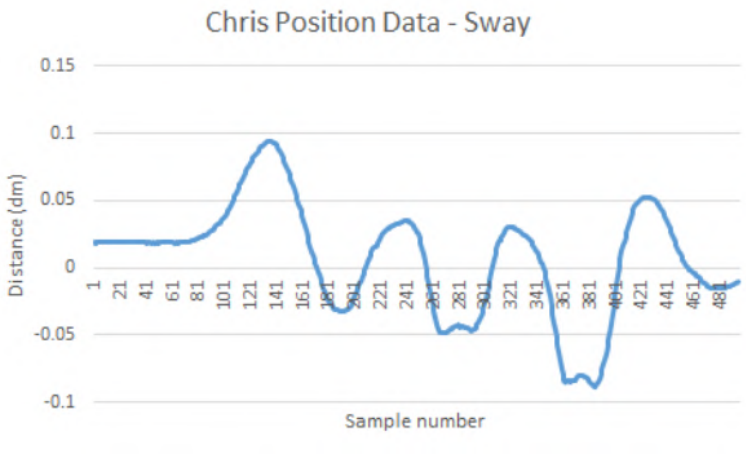

Figure 20: Chris's sway plot

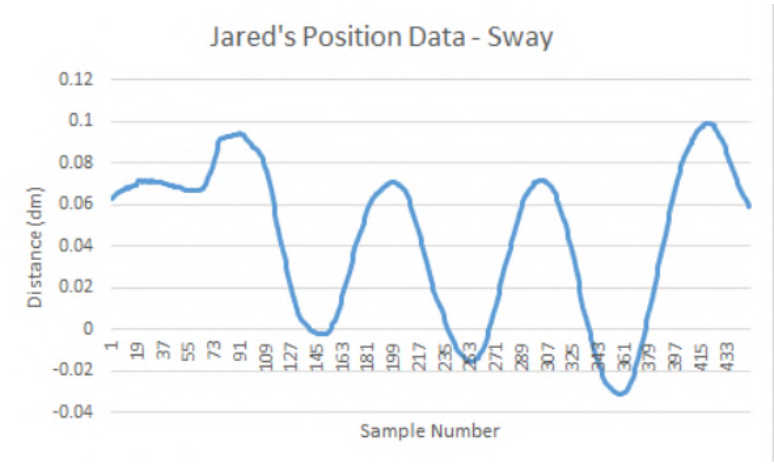

Figure 21: Jared's sway plot

```
 -2.41235281 -2.56242628 -2.68595753 -2.79757652 -2.89152801 -2.98622343
 -3.07969642 -3.17687528 -3.29349235 -3.40375514 -3.51071602 -3.61129197
 -3.73150809 -3.82157713 -3.89910999 -3.97557059 -4.04944095 -4.12321032
 -4.19607873 -4.27825014 -4.34888097 -4.41754493 -4.49958415 -4.54643962
 -4.57644183 -4.61104505 -4.58768632 -4.58457898 -4.58469847 -4.54006103
 -4.49483088 -4.44365094 -4.38253142 -4.28760283 -4.16532466 -4.03974511
 -3.89388244 -3.68690607 -3.51419294 -3.33457144 -3.13712542 -2.97722604
 -2.85466553 -2.7084243  -2.53923304 -2.43918635 -2.37315017 -2.33407814
 -2.32922493 -2.33894403 -2.32778784 -2.27859328 -2.16957776 -2.08581805
 -1.97286931 -1.8023869  -1.68996794 -1.59466079 -1.56111027 -1.52128849
 -1.49413507 -1.4848067  -1.47049928 -1.46685634 -1.47171589 -1.44766981
 -1.44025121 -1.40187286 -1.37108284 -1.35233463 -1.33466133 -1.31235529
 -1.3101616  -1.30269277 -1.31405674 -1.33439009 -1.39236473 -1.44818374]
Prediction =  0.970268795457
Name:
```

Figure 22: The working running Python script

11 Conclusions

While the LiDAR infrared camera did not seem to have the resolution required for the neural network to be be able to identify a subject, we have proven that it is indeed possible for a neural network to identify someone based on their position data using the HTC Vive. Out of 30 tests, there

was only one instance in which the network mistook Chris's identity for Jared. Future adjustments to this project's design may include using a higher resolution infrared camera, and possibly, and possibly utilizing more neural network package tools for data analysis.

12 References

Vrigkas, M., Nikou, C. and Kakadiaris, I. (2018). A Review of Human Activity Recognition Methods. [online] www.frontiersin.org. Available at https://www.frontiersin.org/articles/10.3389/frobt.2015.00028/full [Accessed 1 Mar. 2018]