Tamim Faruk
October 1, 2016

Dephy Software Controls Challenge

This document will outline the steps and thought process to solving the software controls problem. All the figures and code are included in the zip folder.

The Initial Approach

At first I was stumped with this problem. But to write a proper algorithm, the data provided had to be deciphered and understood to impart meaning into my code.

Having no prior proficiency in the gait cycle before, the first thing I noticed is shown in figure 1.

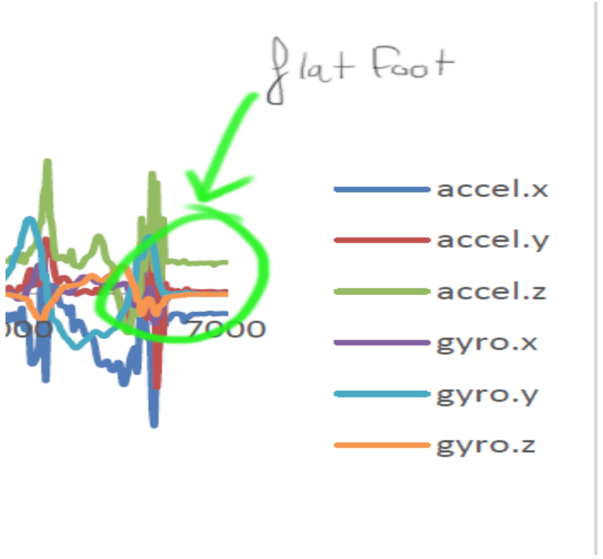

Figure 1: Signals end on a flat foot

The instructions stated that the person walking ends off having completed 6 steps and has stopped walking (i.e foot is flat). Therefore, the phase right before that must have been when the heel strikes the ground. I started considering the Z-accelerometer. I noticed there are 11 peaks, but a heel strike/cycle would only have happened 6 times. And since the acceleration when a heel strikes should intuitively have a large magnitude, it was clear that every other peak was a heel strike, and that after every heel strike, occurred the flat foot phase.

What I also noticed was that the Y-Accelerometer, though less so than the Z-Accelerometer in strength, would only hit a negative peak during a heel strike. So for the sake of indicating the heel strike, I utilized these two sensors.

I graphed the datasets for further analysis. The second dataset was far more unkempt than the first dataset, but after researching into the sensor, it seems as if the MPU-6500 already has a built in low pass filter[1], and based on an eye balled effort, I felt the data was clean enough as it was to interpret.

Assumptions

There were a number of key assumptions that I made to devise my algorithm:

1) The placement of the sensor during each trial is the same.
2) The height and mass of each test subject are very similar to avoid considering major differences in acceleration magnitudes
3) The test subjects are healthy, have symmetrical gait cycles on each foot, and are not suffering from any sort of disability
4) That the sensor readings from the MPU-6500 are consistent in terms of the sensor's sensitivity, the quality of it's built in filter, the frequencies it reads, etc. - so my algorithm works for both datasets, and should also work for future datasets presented by this sensor.

Using these assumptions, it is possible to analyze the data and conclude thresholds that can be used to interpret the data. Also, to develop an even more general model would require many more datasets and specific data about the conditions of each test. After researching the standard practices for how thresholds are actually concluded, I learned about convolution, which involves comparing a snippet of the data to the entire set[2], in pursuit of a proper threshold, but I believe the data was clear enough to conclude these thresholds by inspection with some trial and error required to better refine/approximate these key values.

For the Z-Accelerometer, the threshold was based of the value of the smallest peak in the positive direction. For the Y-Accelerometer, the threshold was based of the value of the smallest peak in the negative direction. Since the two datasets are different in this respect, I assumed the threshold based on how much the data would overlap.

The Algorithm

Note: The code provided is an updated version which takes account of and attempts to track the other phases. Aside from a couple of lines, everything else should be the same.

In this section, I'll describe how the code I wrote works with some snippets to compliment my explanations. There are more comments inside the actual source code.

The algorithm I used to decipher the phases of the gait cycle is a very simple *Weighted Moving Average*. For the purposes of testing, I simply initialized a file stream to read in the data, and I initialized variables corresponding to all the inputs as provided by the Dropbox datasets. For the purposes of the heel strike, the relevant variables are accelY and accelZ, and their respective arrays yArray[10] and zArray[10].

I use a While Loop which is where the magic happens. The while loop ends when the data stream is completed.

The main skeleton of the code is composed of an if-else statement.

In the first if statement:

```
while (sensorValues.good()) {
        if (flag == true) {
                for (int i = 0; i < 10; i++) {
                        yArray[i] = 0;
                        zArray[i] = 0;
```

```
                    sensorValues >> a >> b >> time>> c >> accelY >> accelZ >> c >>
                    gyroY >> ignoreLine;
                    plotValues << GREEN << " " << accelY << " " << accelZ << endl;
                    lines++;
                    }
                    flag = false;
            }
```

I use the variable flag as a buffer period for every time a heel strike occurs. The reader will still read in the data, but the data is not processed to avoid false positives. The ignoreLine variable is a template variable which will skip a line or the rest of a line. In this case, as it is read after the other parameters, it skips the rest of the lines[3]

In the else-statement, I actually assign all the variables I initialized, their respective values from the data stream. These values are stored in their respective variables, and each variable is stored in a circular buffer array – implemented using a modulus division of number of read in lines by the array size. This ensures every value in the array is a recent, relevant value for the forthcoming average calculation.

```
} else { // flag == true, need to find next heel strike
      sensorValues >> a >> b >> time>> c >> accelY >> accelZ >> c >> gyroY >> ignoreLine;
            zArray[lines%10] = accelZ;
            yArray[lines%10] = accelY;
                    double zAvg = 0;
                    double yAvg = 0;
```

After every value, these arrays taken, and in a quick for loop, their values are accumulated into an average. To reiterate, these thresholds were assumed based on eyeballed data analysis and pinpointed during trial and error executions of the code.

```
for (int i = 0; i < 10; i++) {
        if (zArray[i] >= 20000)
                zArray[i] *= 1.25;
        zAvg += zArray[i];
        if (yArray[i] <= -9000)
                yArray[i] *= 1.5;
        yAvg += yArray[i];
}
```

Finally, in another if statement, the averages are actually taken and compared to their respective thresholds, Y_THRESHOLD and Z_THRESHOLD. I concluded that if the average for the y values was less than the Y_THRESHOLD and the z values were more than the Z_THRESHOLD (i.e both conditions are true), then a heel strike must have occurred. If a heel strike occurred, then the time, and line of the event is stored in a variable and printed out, and the number of heel strikes is allotted. Flag is also set to true to start the buffer period.

Results

The charts were plotted using Python Pychart from the Matlib library.

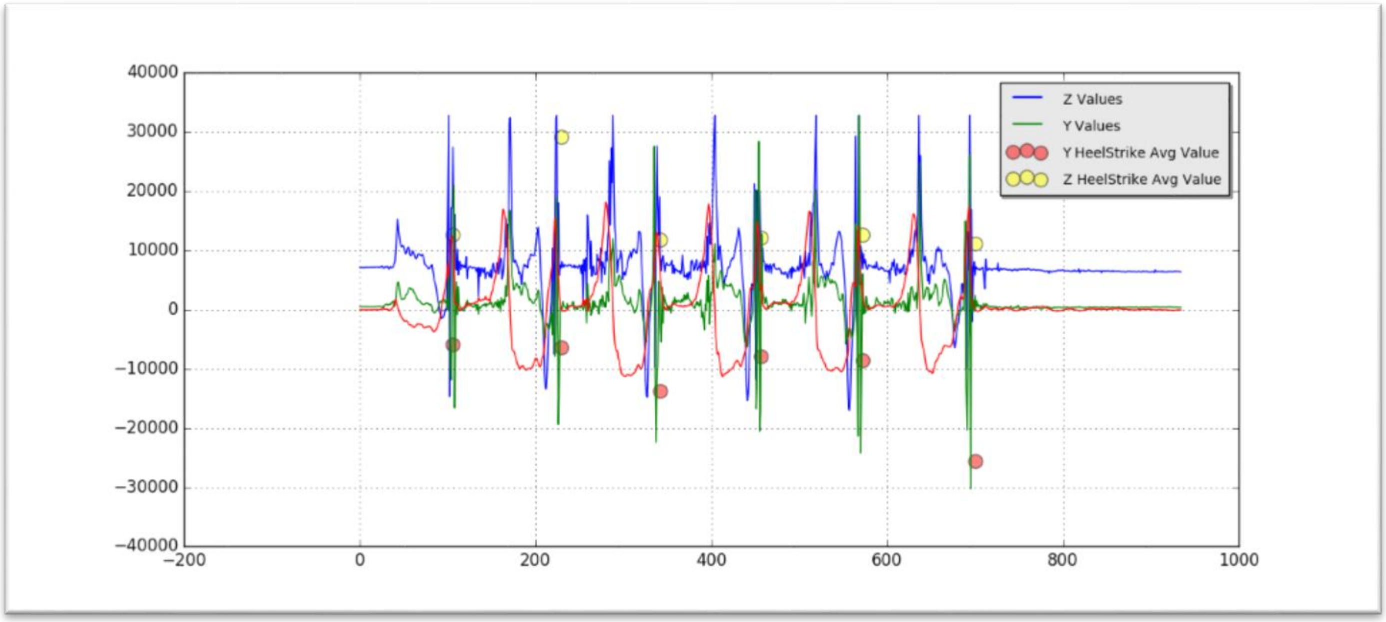

Figure 2: Results for Dataset 1

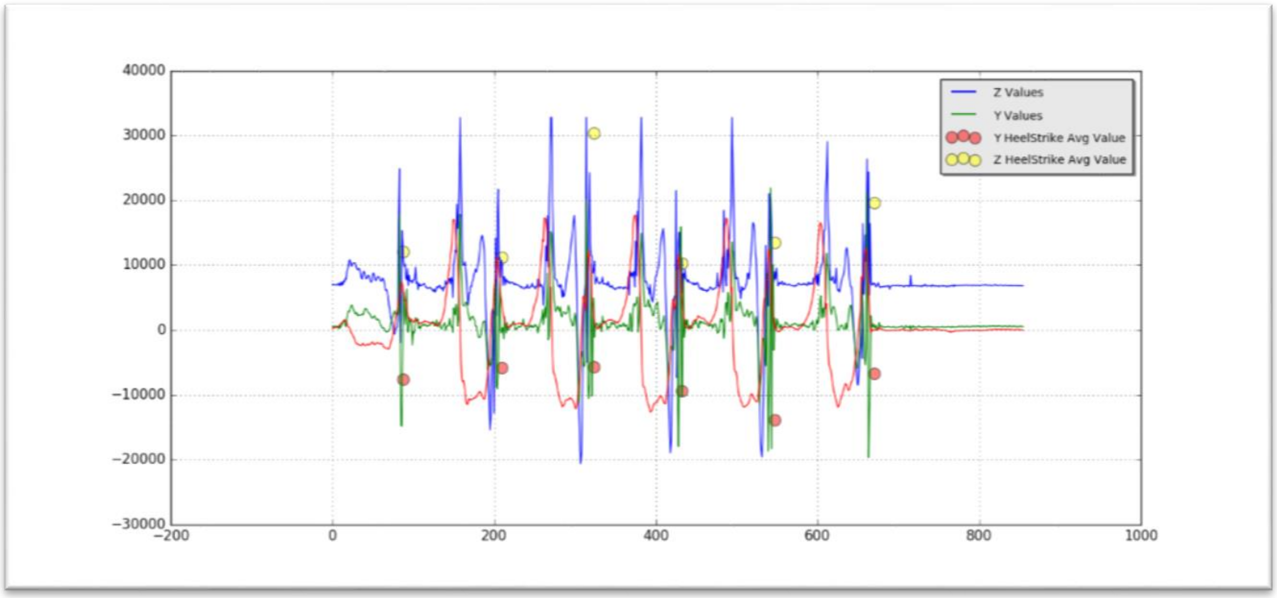

Figure 3: Results for Dataset 2

Figure 2 and 3 indicate the markers for the heel strikes. The red and yellow markers are in line and share the same x axis, except that the red marker represents the Y-accelerometer average value that indicated a heel strike occurred, and the yellow marker represents the Z-accelerometer average value.

Bonus

Figure 4 illustrates some of the bonus phases. The phases identified for the bonus challenge were the toe raised and midswing phases of the gait cycle. The orange "+" symbols indicate the midswing regions, and the yellow squares indicate the Z-average values which detected the Toe Up phase. The foot flat and midstance positions were also identified, in purple and grey respectively, but not effectively so due to the overlap of parameters of both phases. In terms of the foot's the differences of both phases mainly occur in the knee and hips, so finding a distinguishing factor that occurred in the foot was not feasible at this time. Also, another problem with this bonus solution is that the beginning portion does not identify the midswing and toe up in the beginning, but this could easily be adjusted given a bit more time.

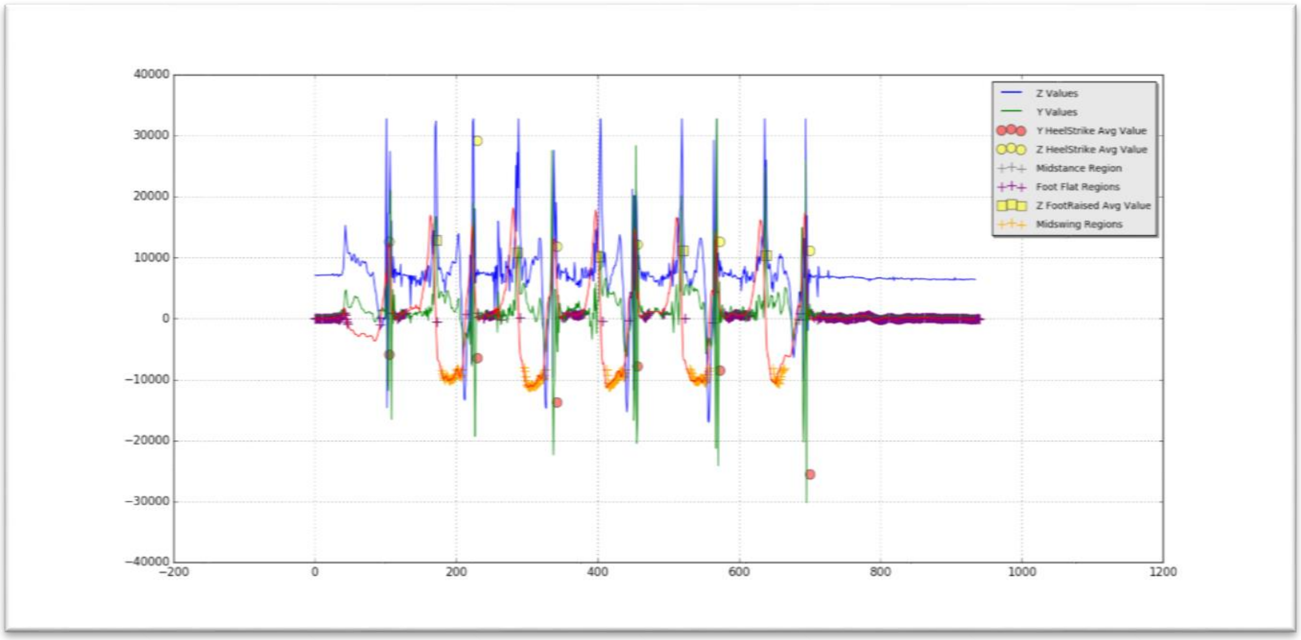

Figure 4: Bonus Phases of Foot's Gait Cycle

Tamim Faruk
October 1, 2016

References

[1]https://www.invensense.com/products/motion-tracking/6-axis/mpu-6500/

[2]https://www.quora.com/How-do-I-calculate-a-threshold-not-arbitrarily-in-a-signal-processing-cross-correlation-of-an-audio-file-of-animal-calls

[3]http://www.java2s.com/Tutorial/Cpp/0100__Development/Manipulatorthatskipsuntilendofline.htm [3]

[4]http://www.physio-pedia.com/Gait

[5]http://matplotlib.org/1.3.0/examples/pylab_examples/legend_demo.html