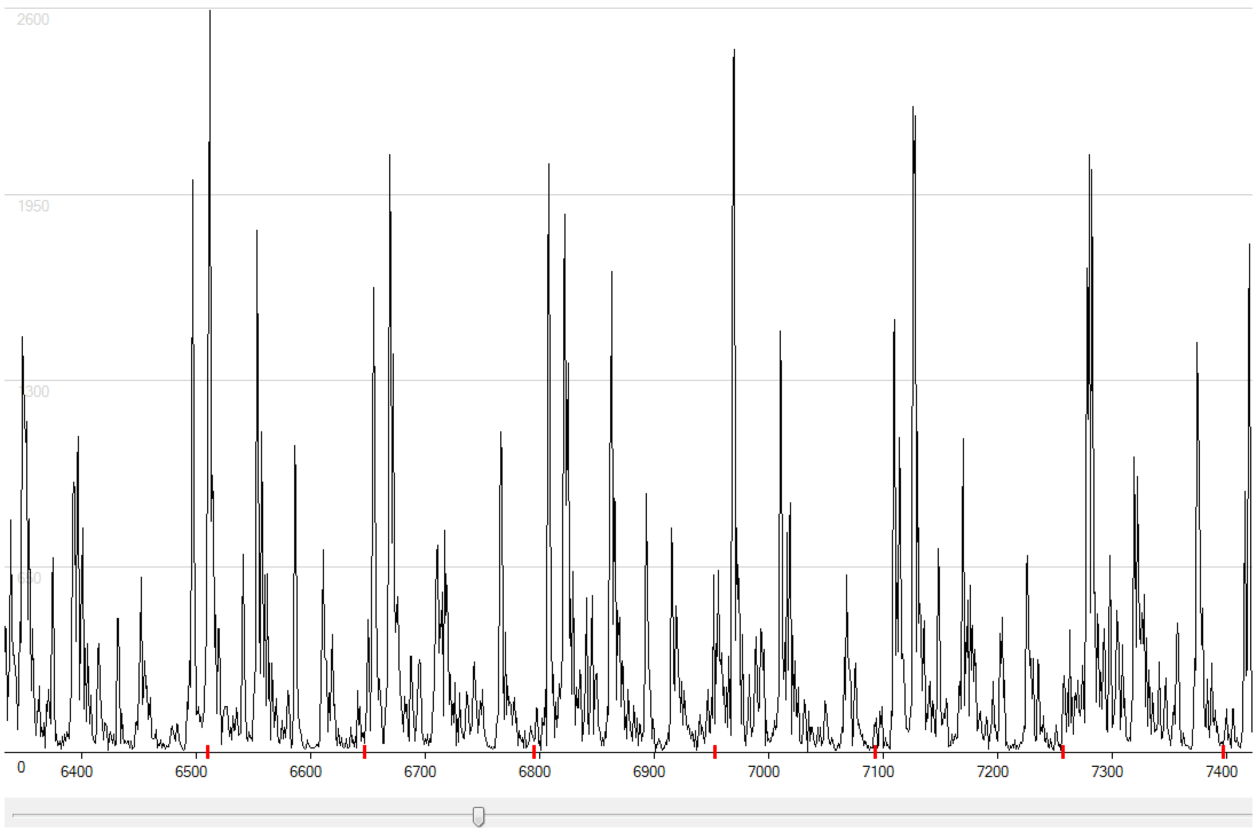
The BetterSpeedBagCounter Algorithm

Initially I viewed the data in MS Excel to get an idea of what the data looked like. As others have pointed out the gravity vector is present in the z axis and motion is seen in all three axis (x, y, and z). To preserve as much of the motion information as possible while also removing the gravity vector, removing any inherent accelerometer DC bias, and to reduce the data size, I decided to combine the x, y, and z values by using the sum of the absolute value of the delta between the current and previous x, y, and z axis values. The calculation is shown below:

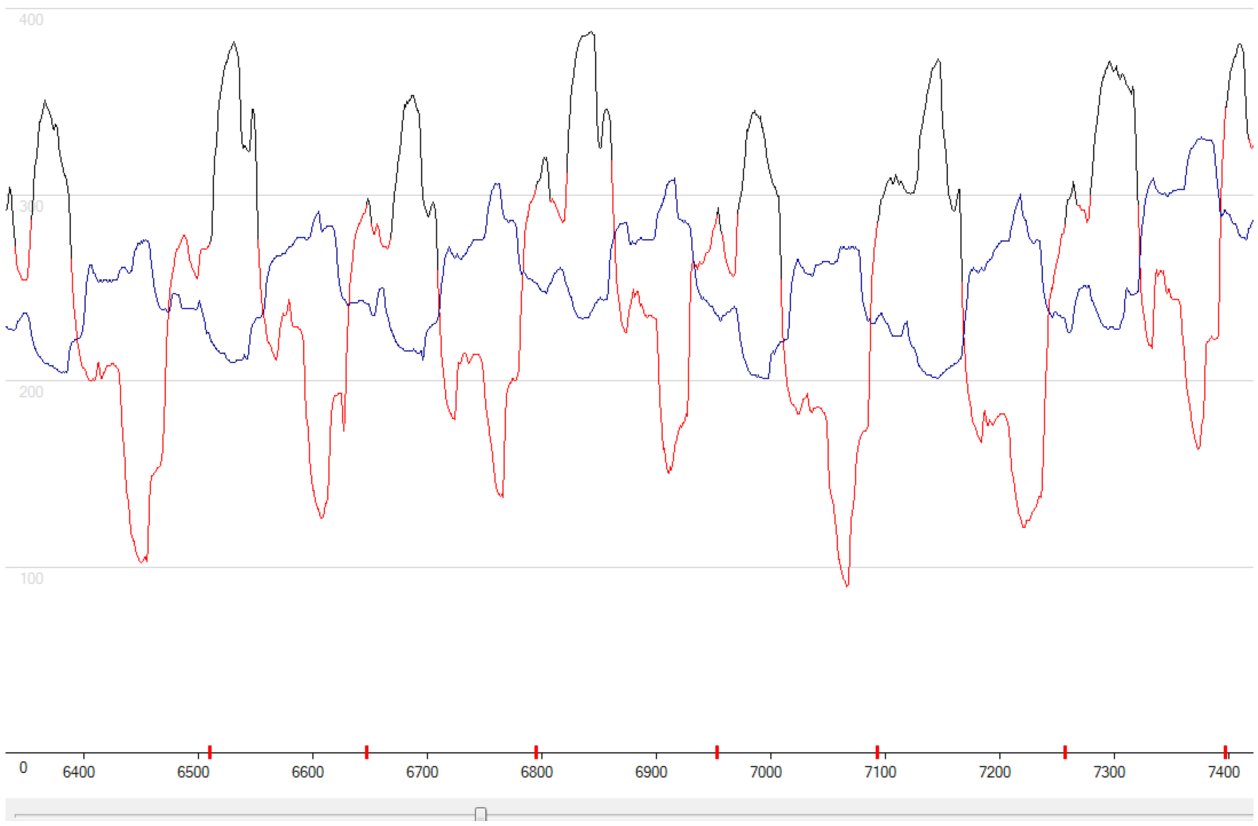
```
absoluteDeltaSum = abs(currentSample.x – previousSample.x) +
                   abs(currentSample.y – previousSample.y) +
                   abs(currentSample.z – previousSample.z);
```

This gives us a single 16-bit unsigned value for each sample that essentially contains the overall motion in all three axes. Below is graph of the absolute delta sums vector over a range of samples while the bag is moving.



Next, I run two low pass filters over the absolute delta sums to generate a “magnitude” and “initial threshold” value for each sample. Writing embedded software for over 20 years, I am constantly aware of how efficiently things can be implemented, so I decided to use simple rolling average filters to generate both of these values. The number of samples averaged to calculate the “magnitude” value is simply less than the number of samples averaged to calculate the “initial threshold”. However, the samples used are aligned so that the middle sample used to calculate both averages is the same.

Below is graph showing the “magnitude” (i.e. red/black line) and “initial threshold” (i.e. blue line), for the same range of samples that were shown above in the absolute delta sums plot.

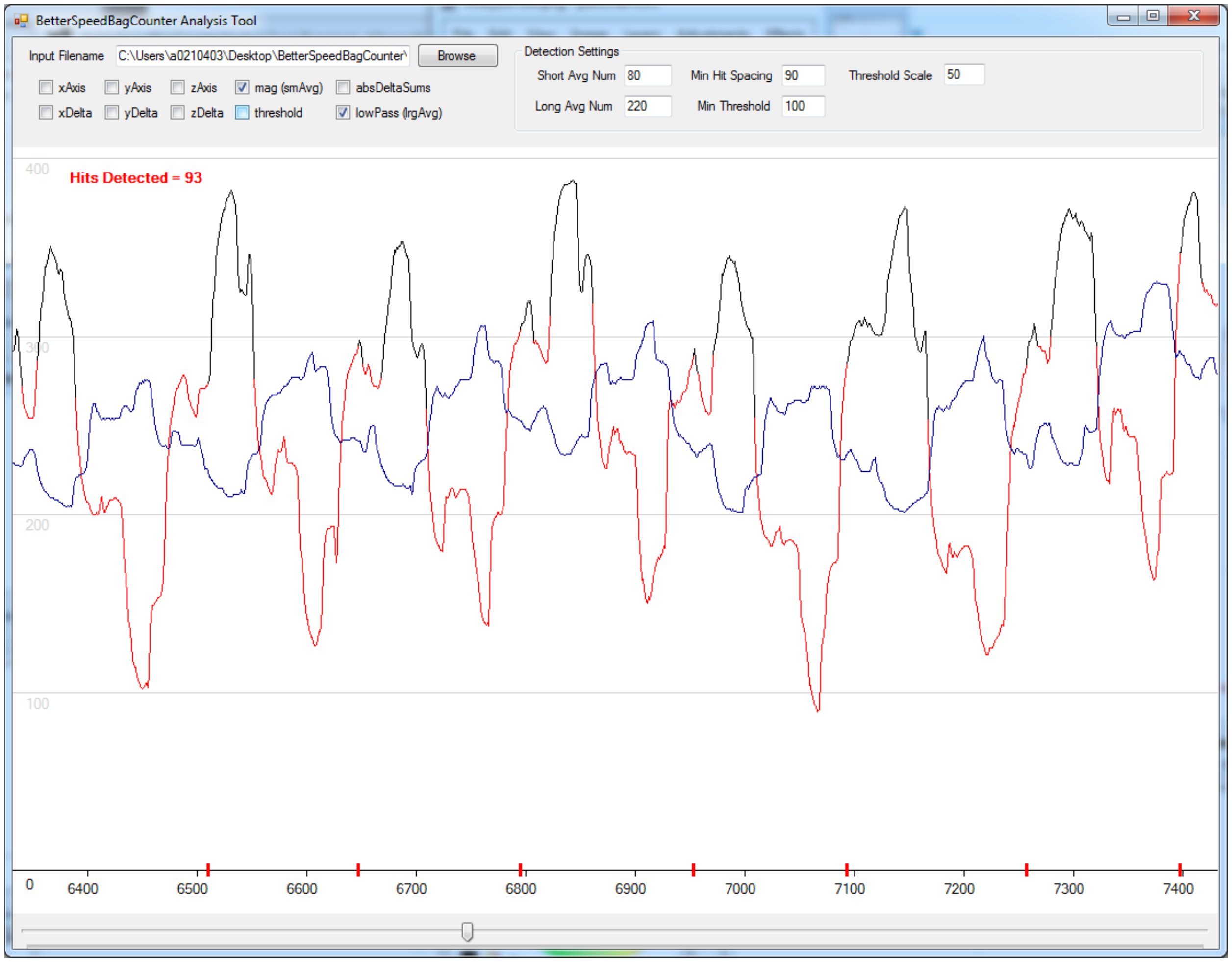


Finally, the “initial threshold” vector is used along with a “minimum threshold” and “threshold scale” value to generate the final threshold value for each sample. The final threshold is calculated by increasing the “initial threshold” value by the “threshold scale” and limiting it to not be less than the “minimum threshold”. The “minimum threshold” adjustment allow me to prevent small signals (i.e. jitter / noise) during inactive time from being counted as hits; while, the “threshold scale” is used to shift the threshold value up and down based on the relative size of the “magnitude” and “initial threshold” low pass filters. **Note: In the graph above the black pixels on the “magnitude” line represent samples that exceeded the final threshold for that sample.**

The final check is a “minimum hit spacing” value that prevents us from counting hit too quickly. When this is applied on the data above, samples that are considered as hits are indicated by the red ticks on the x-axis of the graph.

BetterSpeedBagCounter Analysis Tool

I created a MS Windows Application to optimize the filter and threshold parameters based on the known datasets. Below is a screen grab of the application.



Input Filename: C:\Users\a0210403\Desktop\BetterSpeedBagCounter\

[ ] xAxis [ ] yAxis [ ] zAxis [✓] mag (smAvg) [ ] absDeltaSums
[ ] xDelta [ ] yDelta [ ] zDelta [ ] threshold [✓] lowPass (lrgAvg)

Short Avg Num: 80 Min Hit Spacing: 90 Threshold Scale: 50
Long Avg Num: 220 Min Threshold: 100

Hits Detected = 93

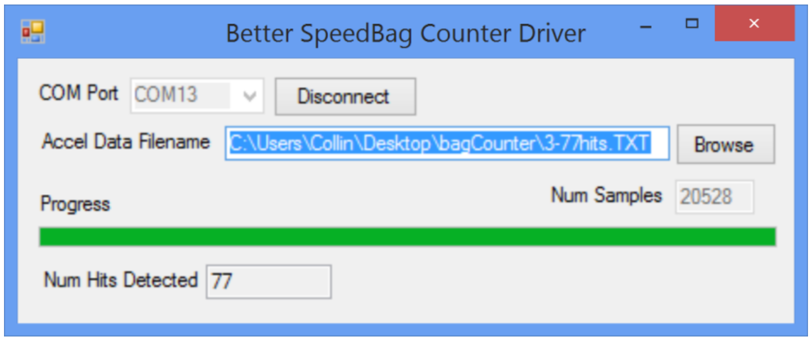
The application allows you to load any of the captured datasets and run the BetterSpeedBagCounter algorithm while changing the filter and threshold parameters. The modifiable parameters are accessible at the top and include the number of samples to average for both the short and long averages, the minimum samples between hits, the minimum threshold, and the threshold scale to use. Any of these parameters can be changed and both the graphs and detected hit count will be updated.

Graphs of a variety of values (i.e. the raw x, y, and z values, the delta X, delta Y, and delta Z values, the short average value (i.e. magnitude), the long average value (i.e. base threshold), the absolute delta sums, and the final (i.e. adjusted) threshold can be displayed/hidden based on the checkboxes at the top. A scrollbar at the bottom allows you to slide through and view the entire dataset worth of values. In addition, the number of hits that were detected is displayed at the top and red tick marks appear on the horizontal axis where individual hits were detected.

The application executable is provided for anyone that would like to play around with the parameters.

Arduino Hardware Testing

I ported the algorithm from the BetterSpeedBagCounter Analysis Tool to the BetterSpeedBagCounter.ino file. In addition to the processAccelerometerData() function, this Arduino sketch contains a serial test interface that communicates with another MS Windows Application called "Better SpeedBag Counter Driver". Via the serial interface this driver application sends the captured accelerometer datasets to Arduino board which processes the dataset for hits. Once the entire dataset has been processed the number of detected hits is sent back to the application and displayed. Below is a screen capture of the driver application.



Better SpeedBag Counter Driver

COM Port: COM13

Accel Data Filename: C:\Users\Collin\Desktop\bagCounter\3-77hits.TXT

Progress

Num Samples: 20528

Num Hits Detected: 77

I ran the BetterSpeedBagCounter on an Arduino Leonardo board I had laying around, but verified the sketch would compile for basically any Arduino board (i.e. Uno, Mini, Nano, etc. regardless of the microcontroller ATmega168, ATmega328).

> **Note:** **I could not receive data over the "Serial" (i.e. USB interface) on my Arduino Leonardo so I used "Serial1" and connected a FTDI breakout board to the DIO 0 and 1 pins on the Leonardo board. If you target a different Arduino board you will likely want to change the #define SERIAL_PORT Serial1 to use Serial!**

Using an oscope connected to DIO pin 13 connected to the LED I was able to time the processAccelerometerData() function. The time varied between 110-130us. This should leave plenty on time to handle the interface with the accelerometer and the display. Furthermore, there should be plenty of time left to improve this approach to create a BestSpeedBagCounter!