Activity Classification

Rafael Delwart
University of California, Santa Cruz
ECE167 Professor Colleen Josephson

3/17/2024

Git Commit ID: 5795f4a640e3d4db4bc82ec4b3db5c73d5ba3c64

1 Overview

Function of this activity classifier will be to track rock climbing routes by measuring the overall displacement throughout the route. This involves double integrating the acceleration in each axis to find the displacement over time in each axis and visualizing the movements of a climber on the wall. In order to track the climbing I utilize the 9-DOF IMU sensor.

2 Wiring and Diagrams

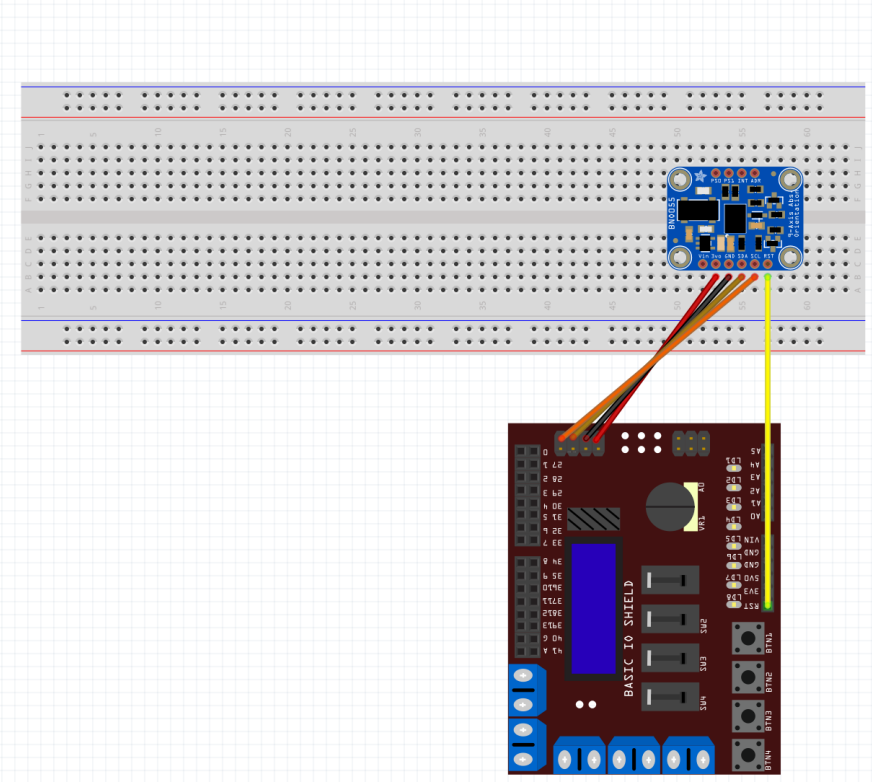

Figure 1: Wiring Diagram for the BNO055 sensor and the MCU

3 Project Design

3.1 Rock Climbing

The BNO055 is a versatile 9-axis absolute orientation sensor, incorporating a triaxial 14-bit accelerometer, a triaxial 16-bit gyroscope with a range of ±2000 degrees per second, and a triaxial geomagnetic sensor. This sensor is particularly suited for applications requiring precise orientation detection and motion sensing. The objective was to measure acceleration to then determine displacement using the BNO055 in its 9 Degrees of Freedom (9DOF) mode, which combines accelerometer, gyroscope, and magnetometer data to provide a comprehensive understanding of the sensor's spatial orientation and movement.

To efficiently manage the data collection process from the BNO055, a simple state machine was implemented. This state machine comprised three primary states: Calibration, Recording, and Waiting, and can be seen in figure 2.

Calibration State: In the BNO055 library a function was added called `getCalibration` this function checks the addresses to see if each of the 3 types of sensors are calibrated, only once all of them are calibrated does it move to the next state. The BNO055 was calibrated this way to ensure accuracy in the measurements. This step was critical for minimizing errors due to sensor biases and environmental factors. Users were instructed to move the sensor

through various orientations to achieve full calibration. Recording State: Once calibrated, the system transitions to the Recording state if a button is pressed. In this phase, the BNO055 was set to output the linear acceleration data, which is available in the 9DOF mode, capturing real-time acceleration. This mode was chosen for its comprehensive data output, providing a detailed picture of the sensor's motion and orientation. Waiting State: The Waiting state served as a stop between recording sessions. It allowed for any necessary adjustments to the setup or to prepare for data analysis. The system could be manually triggered to re-enter the Waiting or Recording states as needed.

To record data from the BNO055 was transmitted to a computer via a serial connection and captured using CoolTerm. CoolTerm was configured to receive the sensor's output data stream, which included timestamped measurements of acceleration, gyroscope, and magnetometer readings in a structured format. This setup facilitated the continuous monitoring of the data feed in real-time and the direct recording of the data stream into a text file (.txt format). The recorded data file was then imported into MATLAB for detailed analysis. MATLAB's powerful data processing and visualization tools allowed for the comprehensive examination of the acceleration patterns and movements captured by the BNO055. Custom scripts were developed to parse the text file, extracting relevant data points for further analysis.

Reading and Preprocessing Data: The code begins by reading gyroscopic data from a text file using `readmatrix`. This data contains time stamps and acceleration readings in three axes (X, Y, Z).

Debounce Filter: A simple moving average filter is applied across all columns (except the first, which is time) with a window size of 200. This acts as a debounce mechanism, smoothing out the data to reduce noise and fluctuations.

Bias Removal: The code calculates the average bias for each axis (X, Y, Z) using a subset of the data (rows 50 to 100) and sets initial and final 100 readings of each axis to zero, to remove edge artifacts from pressing the record.

Low and High-Pass Filters: The script applies a low-pass filter to each axis of the accelerometer data to remove high-frequency noise and then a high-pass filter to remove any DC offset or slow-moving trends. This filtered acceleration data is used for subsequent velocity and displacement calculations.

Velocity Calculation: For each axis, the script calculates velocity by integrating the acceleration over time. The integration is performed by summing the product of acceleration and the time difference (deltaTime) between consecutive readings. Displacement Calculation: Displacement is calculated by integrating the velocity over time, again using the sum of the product of velocity and the time difference between readings.

Data Visualization: Unfiltered Accelerometer Data: The code plots the raw accelerometer data for each axis over time, providing a visual representation of the movements captured by the sensor.

Filtered Accelerometer Data: It also plots the debounced accelerometer data, showcasing the effect of the initial smoothing filter.

FFT Magnitude Spectrum: The Fast Fourier Transform (FFT) of the acceleration data is computed and plotted, showing the frequency components of the movement.

Velocity and Displacement Over Time: The script plots the calculated velocity and displacement for each axis over time, illustrating how the sensor's position and speed change.

3D Displacement Plot: Finally, a 3D scatter plot visualizes the displacement in all three dimensions, colored by time to provide insight into the trajectory and timing of the movements. Examples of the plots for measuring the displacement in the y-axis for 1 meter can be seen in figures **??** - **??**.

Initially, to verify the functionality of the device, displacement was measured for 1 meter and back along each axis, conducting 10 trials for each axis. This preliminary testing allowed to calculate the error rate associated with each axis. The results of these trials are depicted in Figures 9 through 11. Based on these trials, the average error rate was determined to be 0.1556m for the X-axis, 0.1031m for the Y-axis, and 0.1489m for the Z-axis.

Subsequently, real-world testing began, which involved climbing with the device mounted on a GoPro chest strap, connected by long wires. This setup posed significant challenges and introduced considerable error. Achieving movement strictly along the intended axis while climbing was particularly difficult. The device is already susceptible to noise, so activities with a lot of complex motion are very difficult to track accurately, a lot of time was spent refining the data collection and filtering methods so thus there weren't as many climbs. Although attaching the device to a gimbal was considered as a solution to mitigate movement errors, time constraints made this option impossible. Instead, climbing as smoothly as possible proved to be a viable option. The best outcomes of the these climbing trials are illustrated in figures 12 and 13. These experiments revealed that error rates increased with the duration of the climbs. Additionally, the unavoidable movement across other axes while climbing was identified as a major source of error and can be seen as exaggerated in the plots due to the filtering.

During the development of this project, numerous challenges were encounterd whilst collecting accurate data. The BNO055 sensor, being an affordable IMU option, is highly sensitive to noise, where minor movements can lead to significant data spikes. This sensitivity presents a substantial hurdle when measuring displacement by double integrating acceleration, often resulting in considerable inaccuracies. A significant portion of my efforts was dedicated to eliminating these noisy artifacts. Various filtering techniques were explored, including the exponential moving average, debouncing, and setting thresholds to exclude unrealistic data, to devise a functioning system.

A complication arose concerning the data output frequency of the BNO055. The datasheet indicated that the maximum output rate in fusion mode was 100Hz. Operating under this assumption for nearly a week, it was discovered through manual timing of data collection periods that the actual output frequency was approximately 230Hz. A thorough examination of the data showed no apparent repetitions or patterns. Conversations with Adam Korycki led to the adoption of a new strategy that completely ignored this issue: incorporating a timer value from the timer library into the data collected for MATLAB analysis. This approach not only provided a more accurate representation of the time spans for data sets but also, surprisingly, improved the accuracy of the integration results.

The final methodology involved debouncing the acceleration data, transforming it into the frequency domain, and applying both high and low-pass filters. This strategy was based on the premise that it would be possible to eliminate unwanted acceleration frequencies, smoothing out abrupt changes in the data and further reducing noise. Attempts to identify desired acceleration frequencies using an app gave an initial starting point for the desired cutoffs but it required manual adjustments. Additionally, applying a secondary layer of

filtering on the integrated velocity data was experimented with. However, this introduced more errors than it resolved, leading to simply integrating the velocity directly after the initial filtering step.

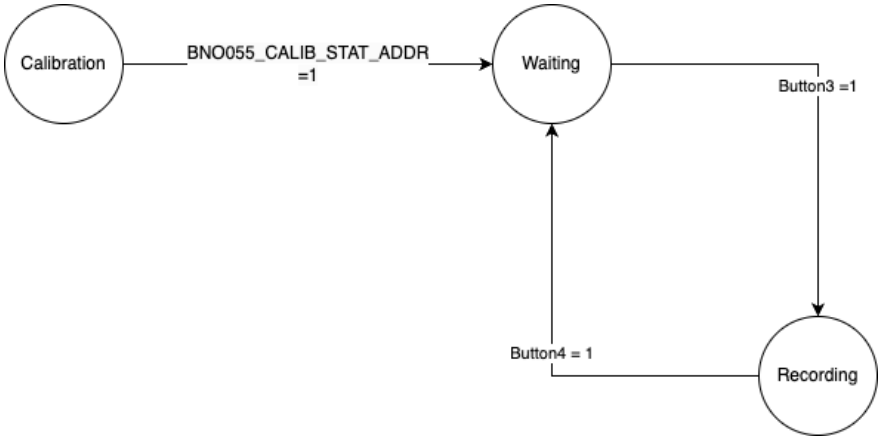

Figure 2: State Machine for recording data

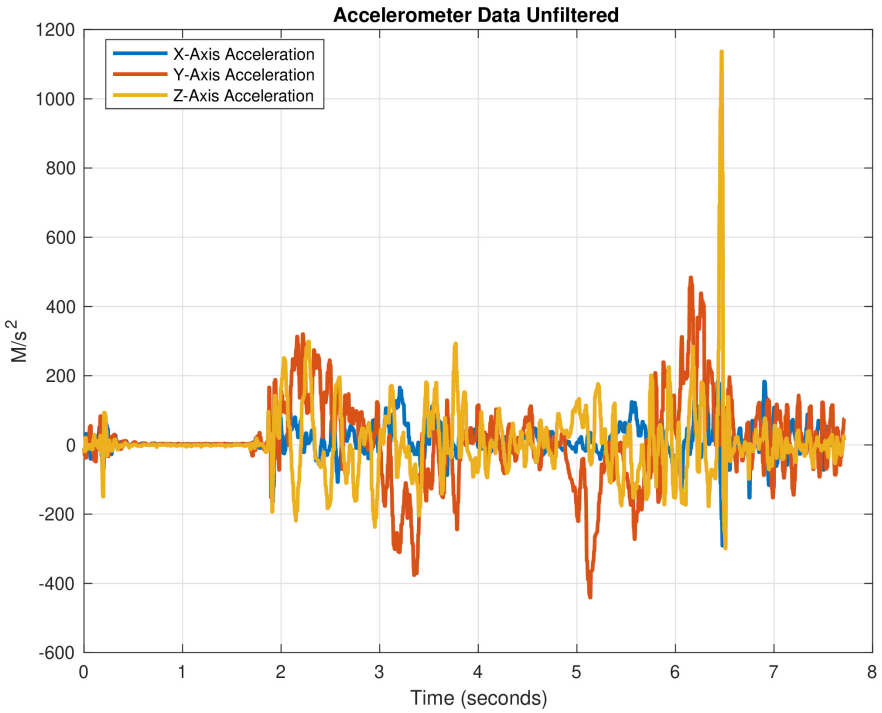

Figure 3: Unfiltered Accelerometer data Over Time(1 meter of displacement in the y-axis and then back to the original position)

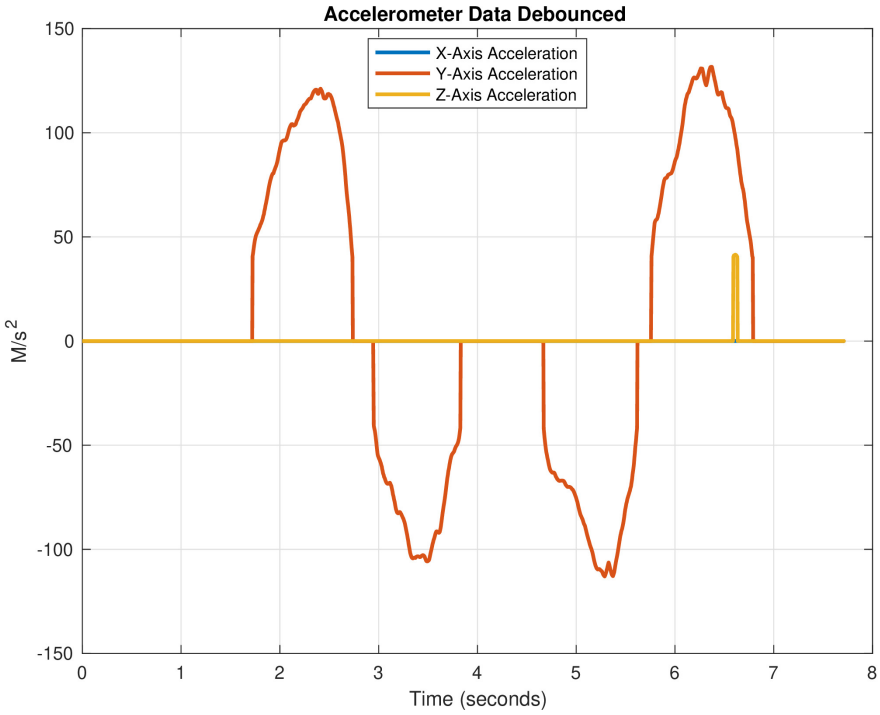

Figure 4: Accelerometer Data After Debouncing(1 meter of displacement in the y-axis and then back to the original position)

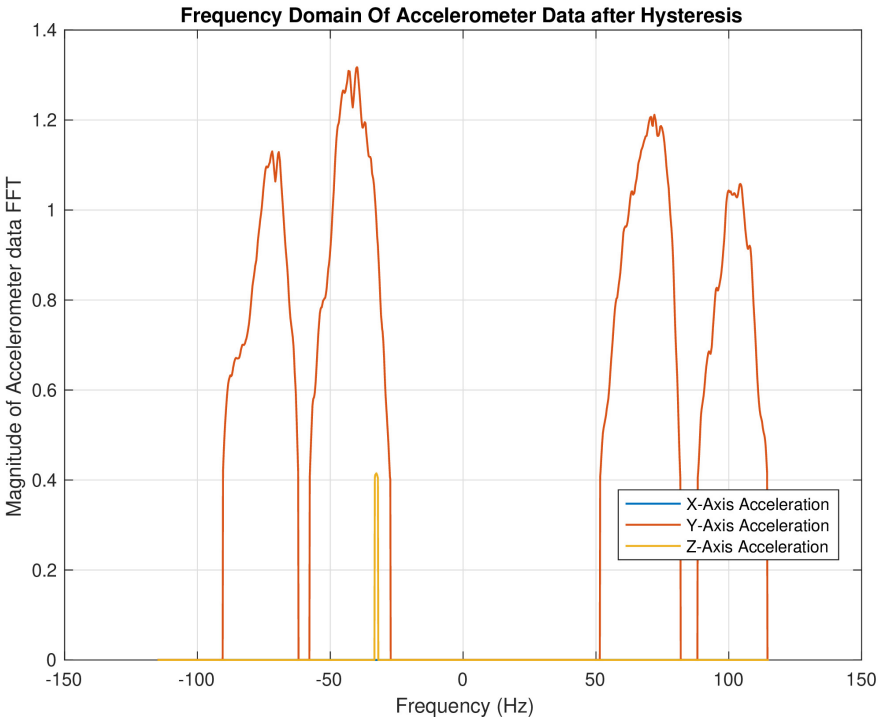

Figure 5: Magnitude Spectrum of Accelerometer Data(1 meter of displacement in the y-axis and then back to the original position)

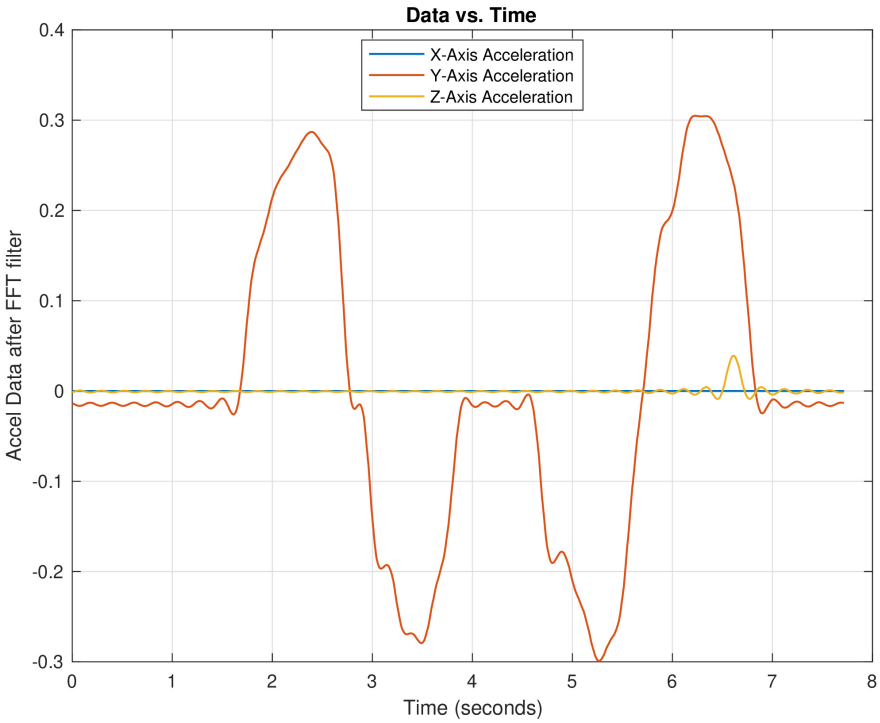

Figure 6: Filtered Accelerometer Data Over Time(1 meter of displacement in the y-axis and then back to the original position)

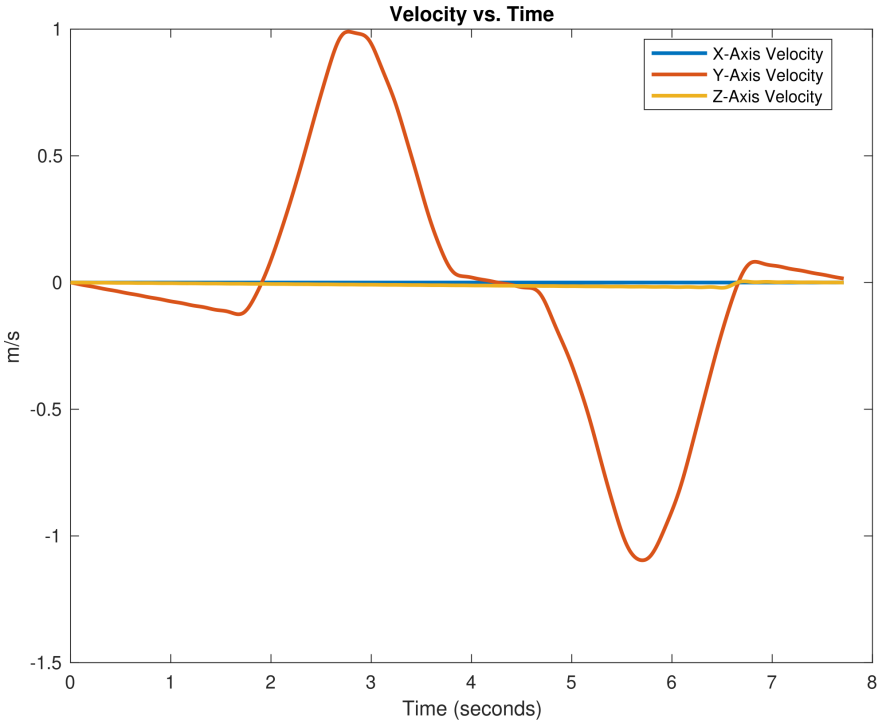

Figure 7: Velocity Over Time Derived from Accelerometer Data(1 meter of displacement in the y-axis and then back to the original position)

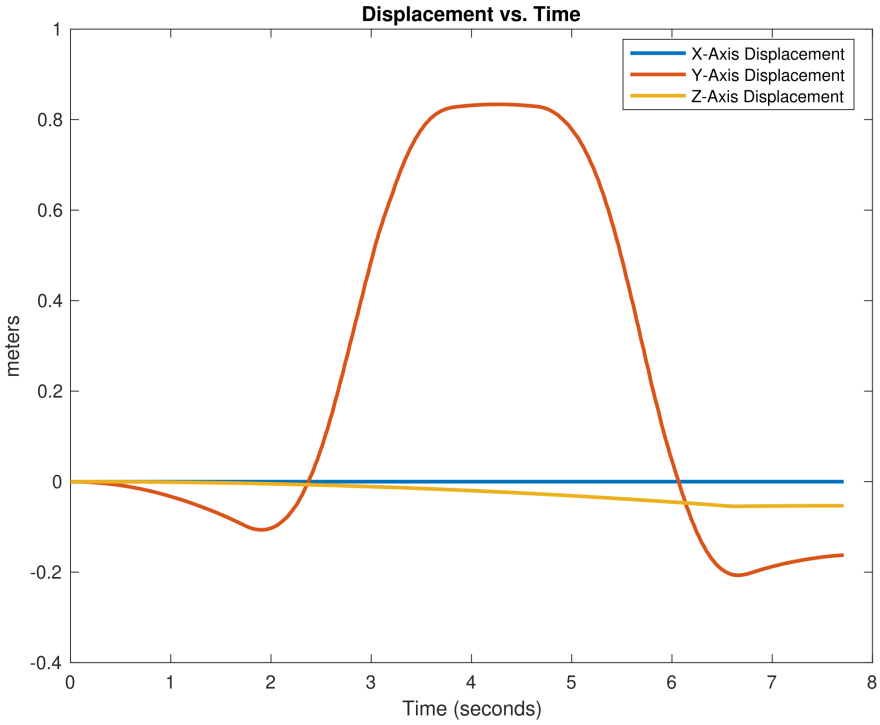

Figure 8: Displacement Over Time Derived from Velocity Data(1 meter of displacement in the y-axis and then back to the original position)

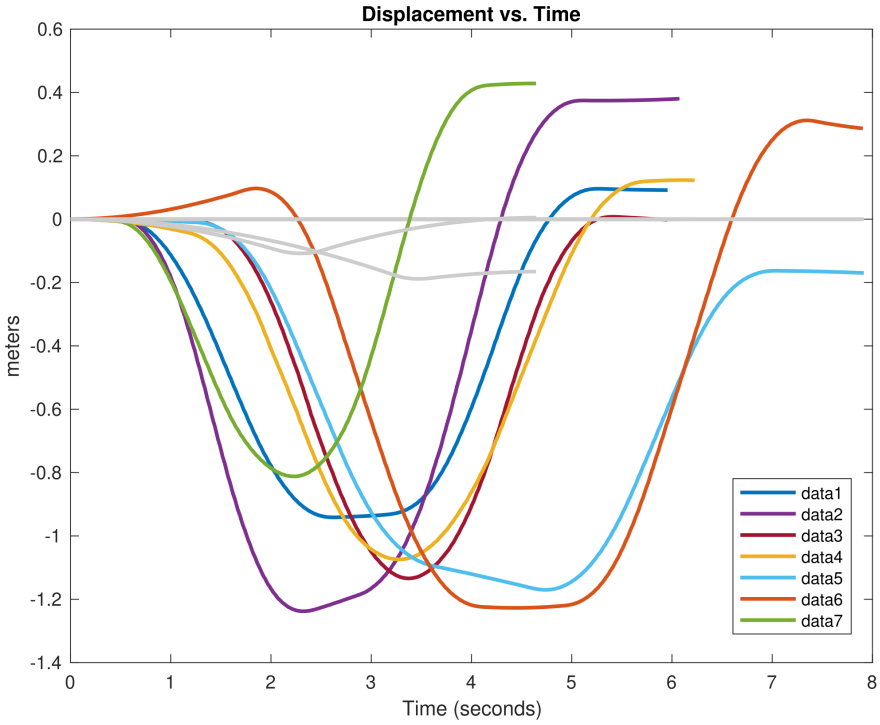

Figure 9: 10 trails moving 1 meter in the x-axis

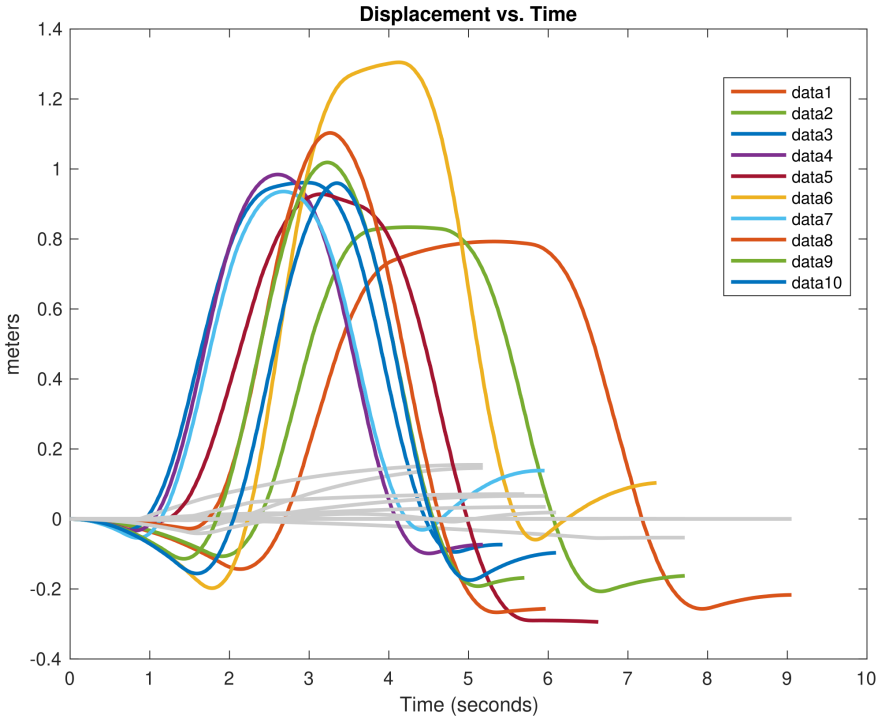

Figure 10: 10 trails moving 1 meter in the y-axis

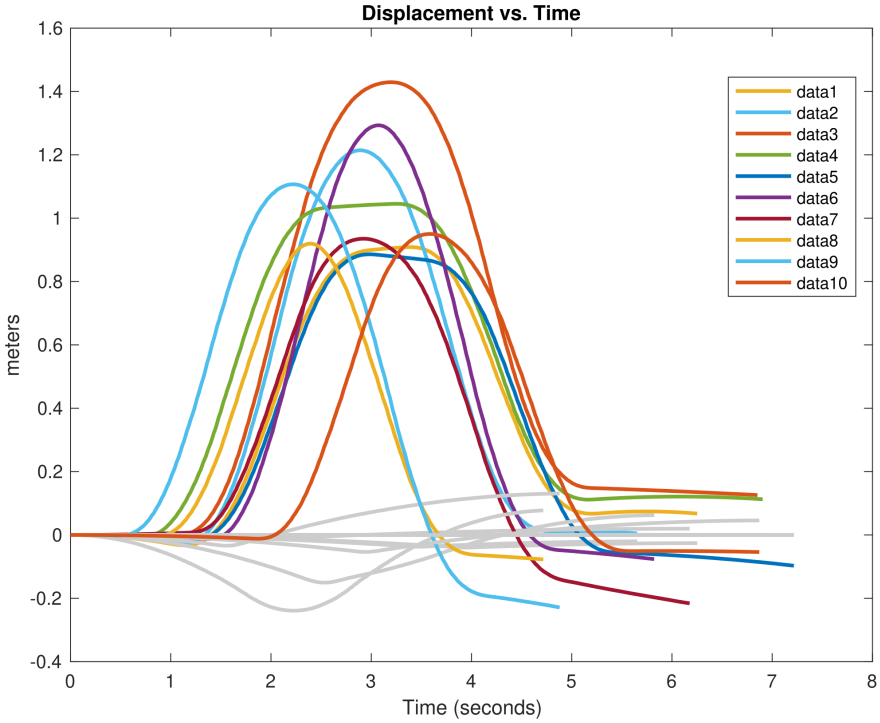

Figure 11: 10 trails moving 1 meter in the z-axis

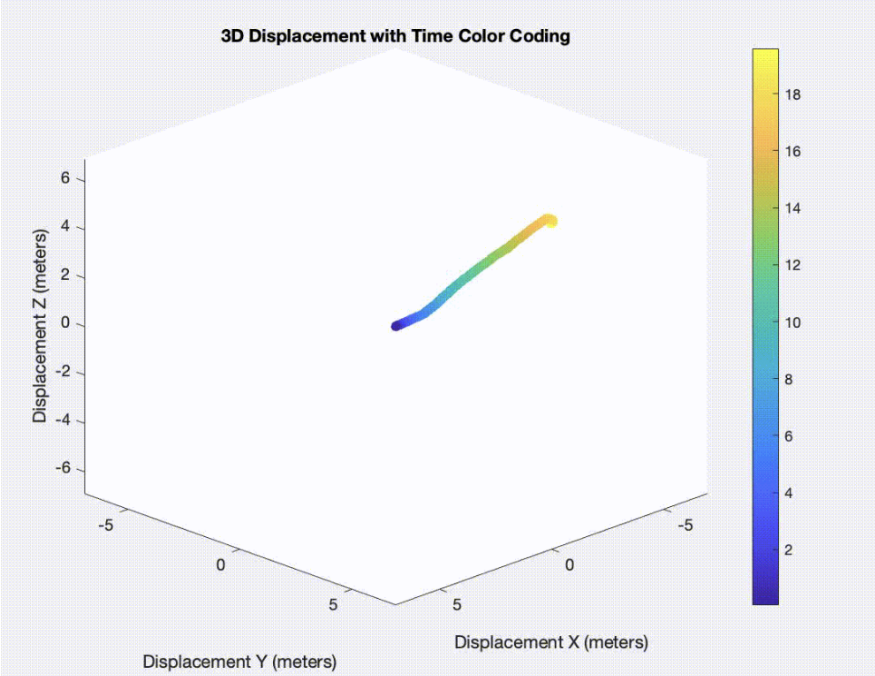

Figure 12: Traverse wall resulting displacement graph, 6 meters horizontal climbing

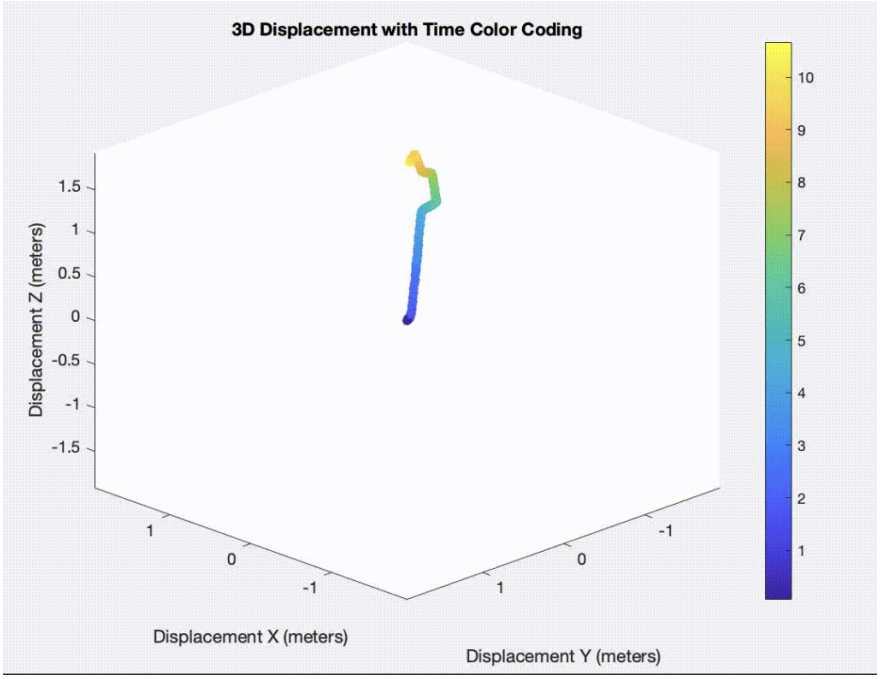

Figure 13: Rock climbing gym wall resulting displacement graph, 2 meters of vertical climbing

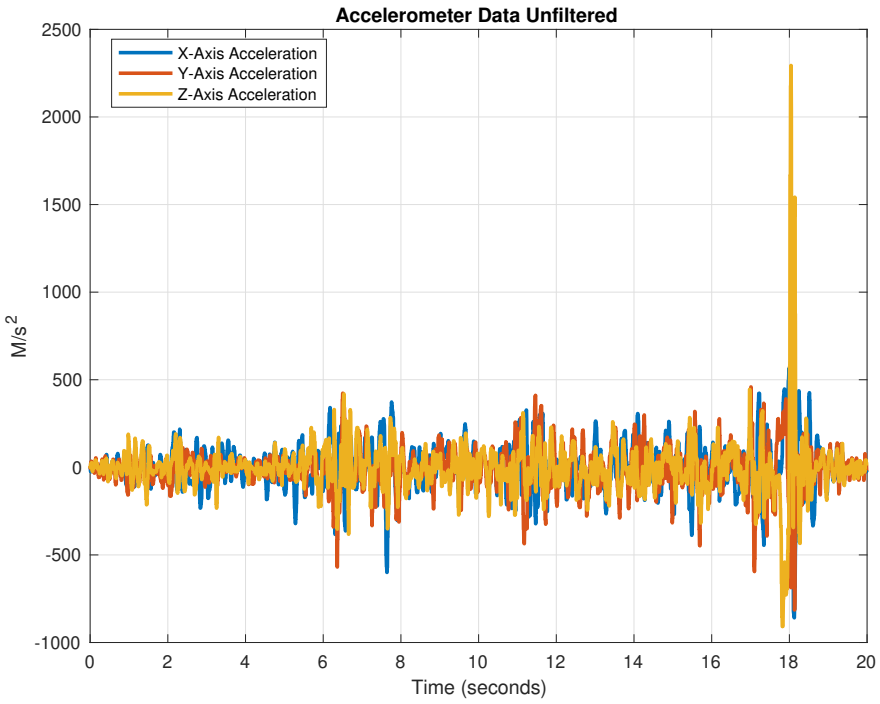

Figure 14: Traverse wall raw acceleration data, 6 meters horizontal climbing

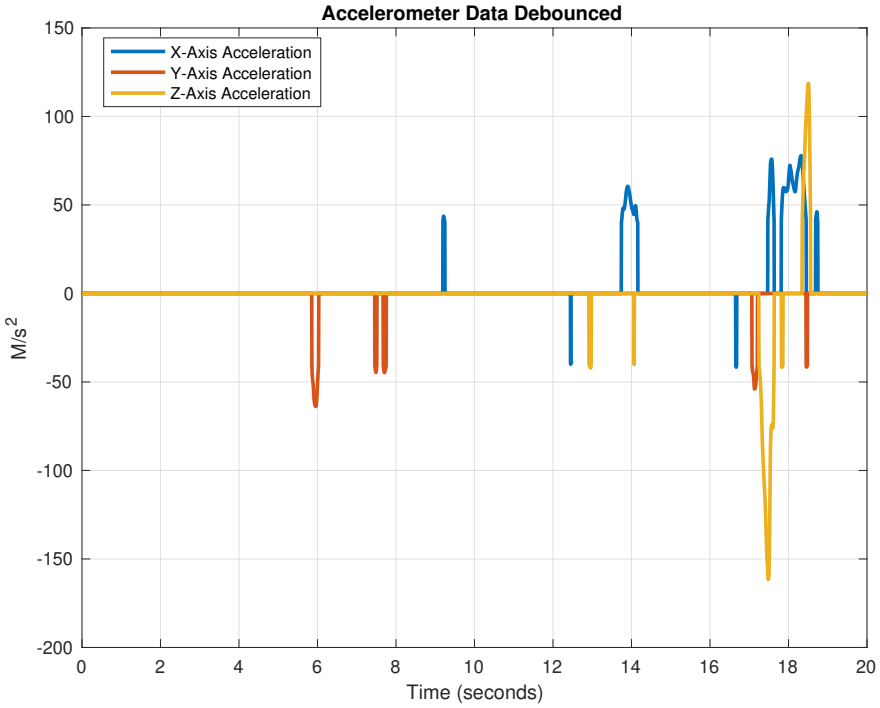

Figure 15: Traverse wall debounced acceleration data, 6 meters horizontal climbing

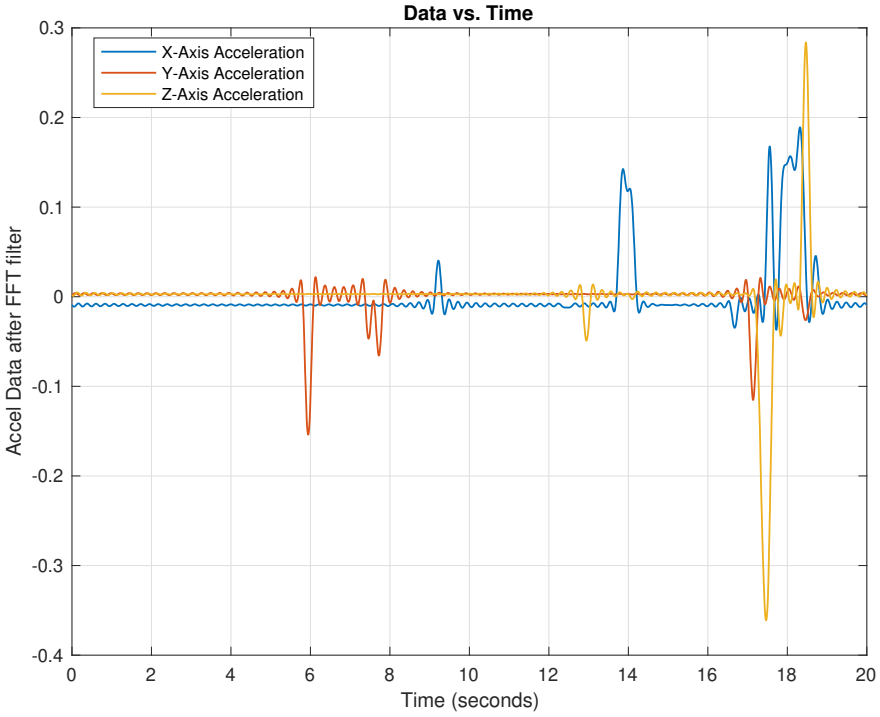

Figure 16: Traverse wall filtered acceleration data, 6 meters horizontal climbing

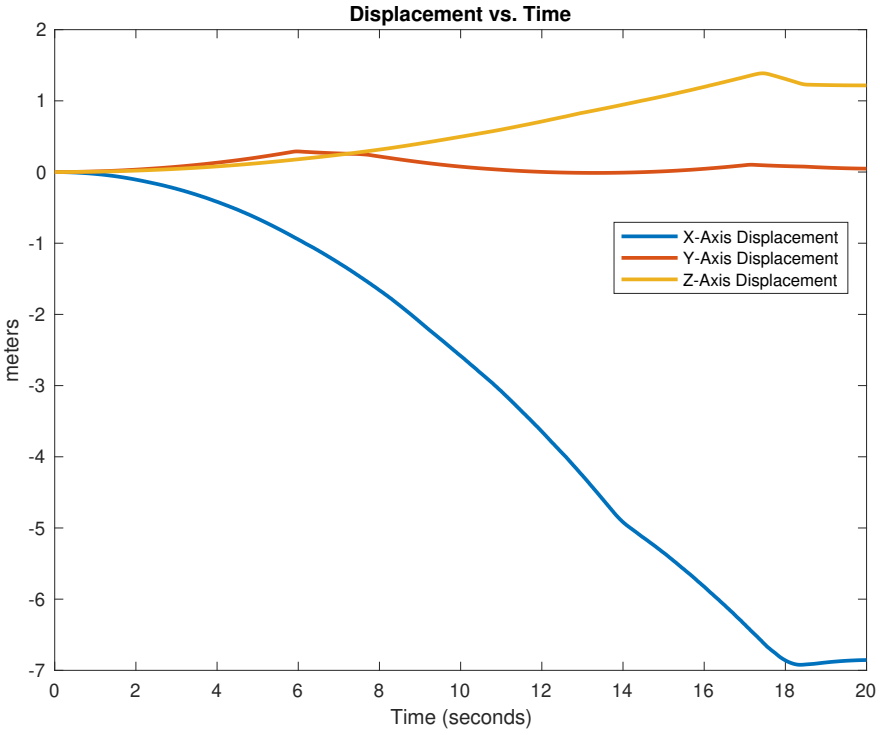

Figure 17: Traverse wall displacement, 6 meters horizontal climbing

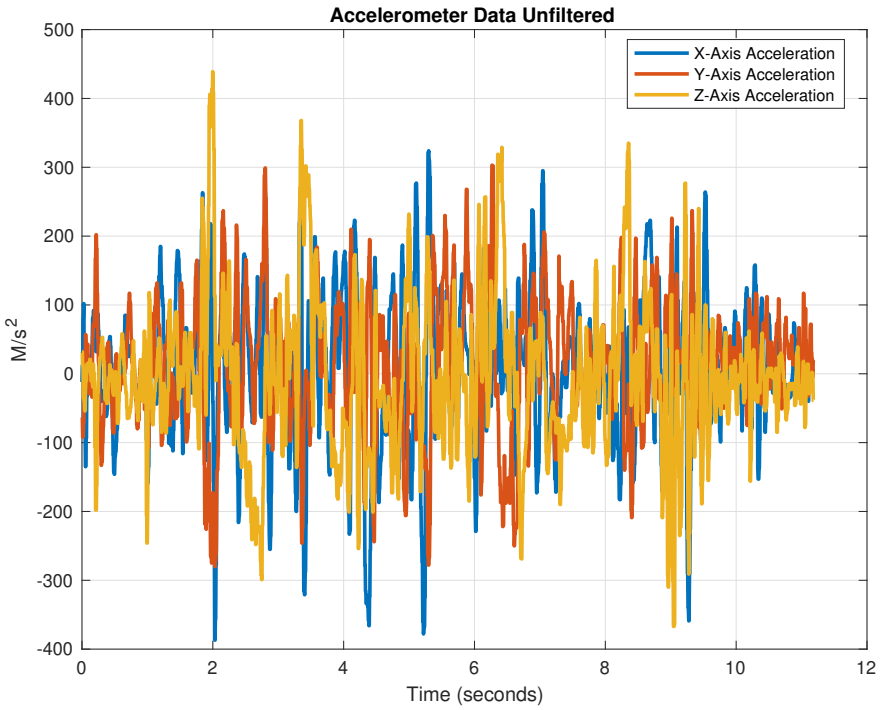

Figure 18: Rock climbing gym wall raw acceleration data, 2 meters vertical climbing

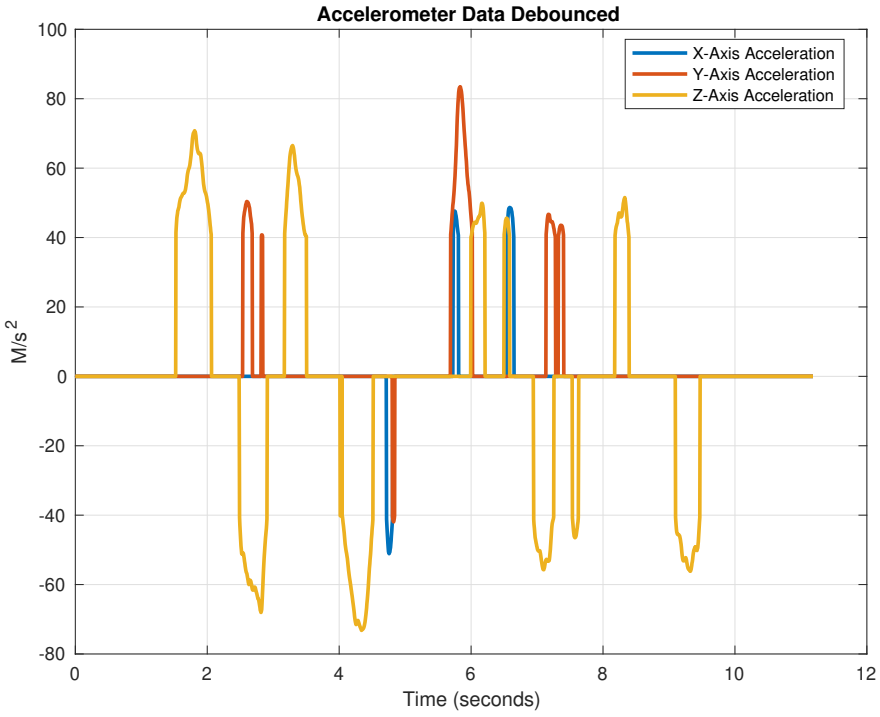

Figure 19: Rock climbing gym wall debounced acceleration data, 2 meters vertical climbing

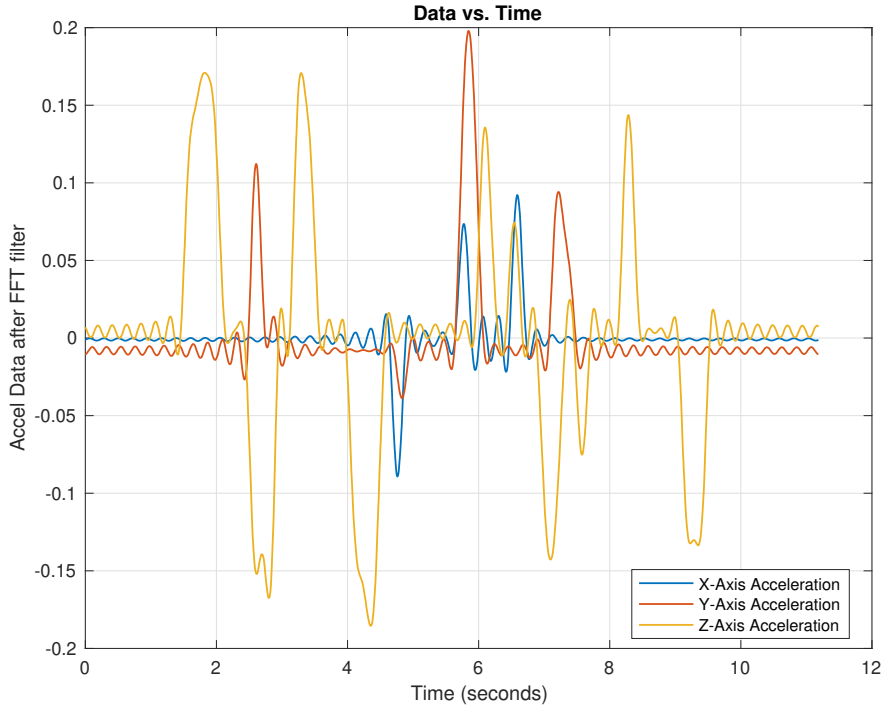

Figure 20: Rock climbing gym wall filtered acceleration data, 2 meters vertical climbing

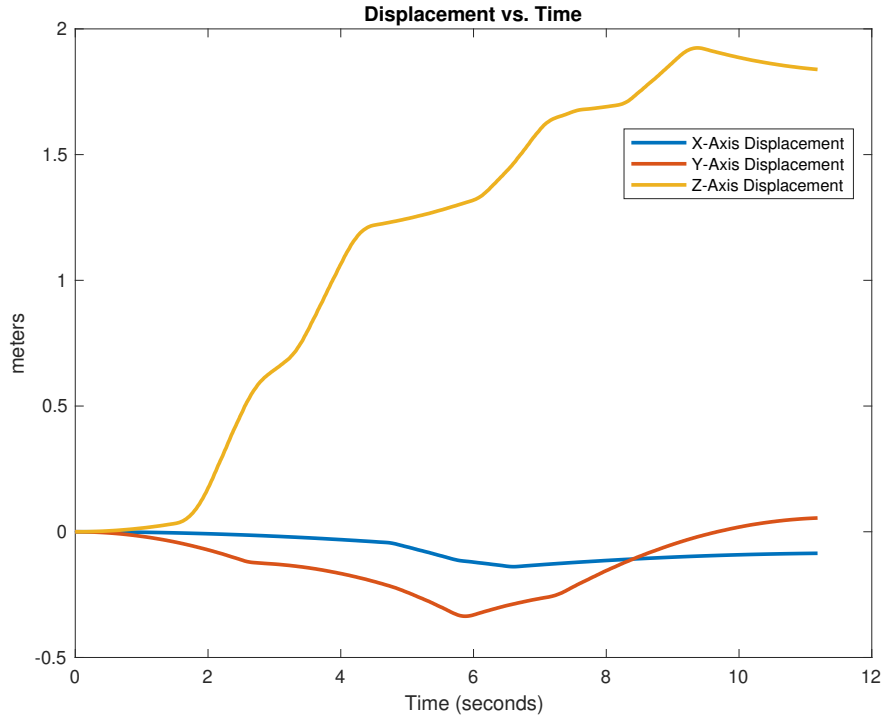

Figure 21: Rock climbing gym wall displacement, 2 meters vertical climbing

References

[1] Digilent Inc. *uC32™ Board Reference Manual*. Revised June 29, 2017.

[2] Microchip Technology Inc. *PIC32MX3XX/4XX Data Sheet*. 2011.

[3] Bosch Sensortec. *BNO055 Intelligent 9-axis absolute orientation sensor*. Revision 1.2, November 2014, BST-BNO055-DS000-12.