Lab 4 Report: Dead Reckoning

Mostafa Ibrahim

Ibrahim.mos@northeastern.edu

Submitted to:

Prof. Hanumant Singh

Jasen Levoy

Vishnu Rohit Annadanam

Methodology

Data Collection

The data was collected by first going in circles around the Ruggles square to be able to visualize and calibrate the magnetometer data correctly. After collecting that data, we drove through Boston for around 2.4mi. The IMU was firmly fixed in the middle of the front part of the car while the GPS was stuck to the roof of the car.

Limitations

During the data collected in circles, few buses came in the way which have resulted in distorted circles around Ruggles square as well as a very low speed. The driving data was collected during rush hour which led to more than 40 minutes of data with frequent stops which should theoretically affect the data analysis.

Results & Analysis

Yaw Estimation: Magnetometer Calibration

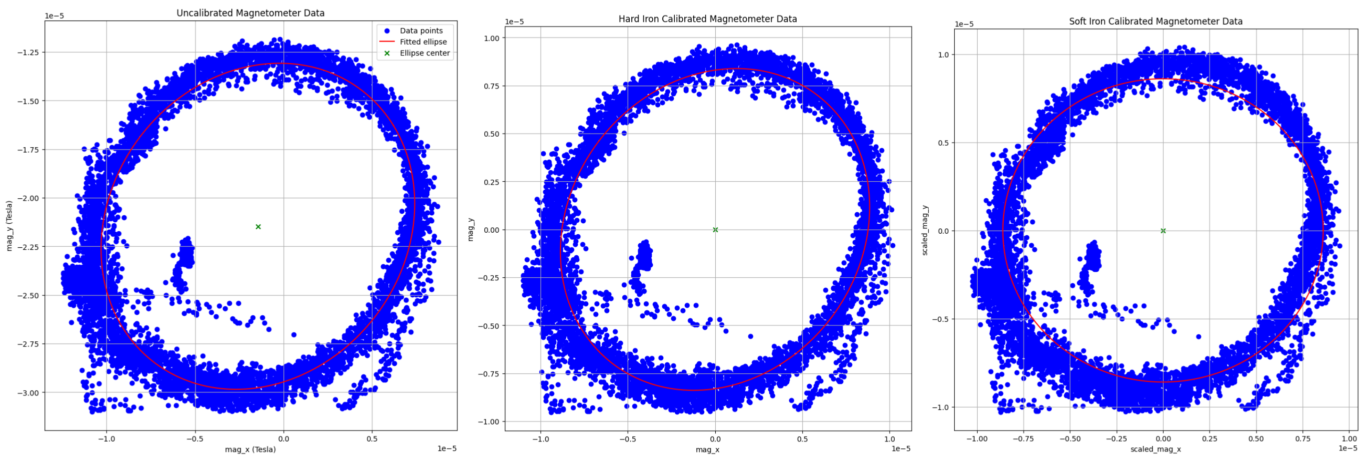

Figure 1: Uncalibrated Data (Left), Hard Iron Calibrated Data (Middle) and Hard & Soft Iron Calibrated Data (Right)

To calibrate the magnetometer data and exclude outliers, I fit an ellipse on the data points and plotted the ellipse with its center. As seen in Figure 1, the left plot shows Hard Iron Distortion given by the center not being at 0 and the ellipse shape indicates the presence of some Soft Iron distortion due to the scale incorrections. To fix the Hard Iron distortions, I simply subtracted the ellipse's center from all data points. The corrected plot is shown in Figure 1 (middle). For the soft iron correction, the ellipse parameters give the longest and shortest radii a and b respectively from which the mean radius was calculated. The scale factor was calculated as follows: The corrected mag data was calculated by scaling the hard iron corrected data giving us the circle shown in Figure 1 (right). The scaling doesn't seem to have changed the data drastically; however, the minor change in the scale helped to get better results in the yaw estimation.

Yaw Estimation: Sensor Fusion

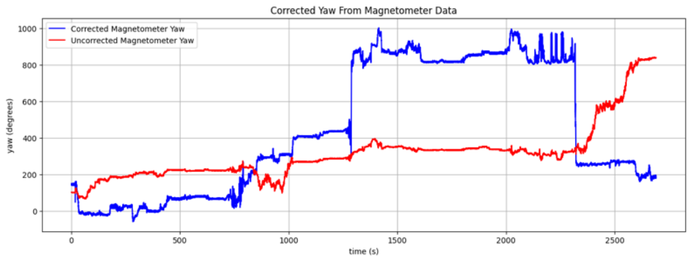

Figure 2: Corrected Vs Uncorrected Yaw

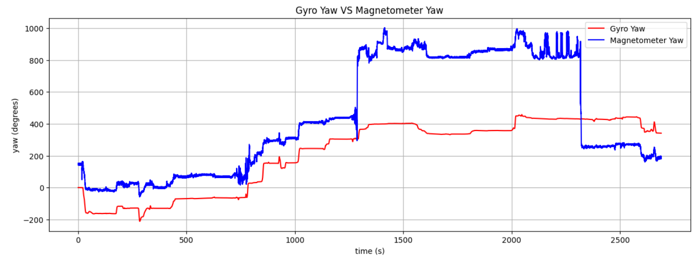

Figure 3: Gyro Yaw vs Corrected Magnetometer Yaw

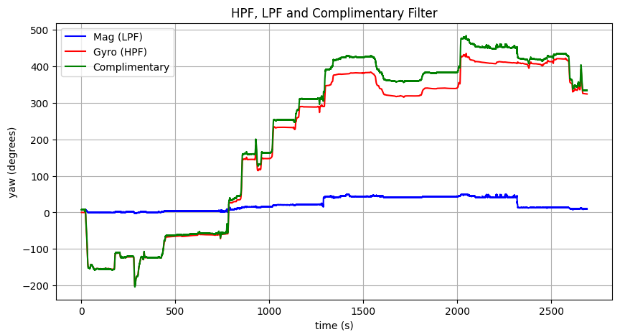

Figure 4: High Pass Filter Gyroscope Yaw, Low Pass Filter Magnetometer Yaw and Complimentary Filter

The complementary filter was used by complementing the high noise magnetometer yaw data with the much less noisy gyro yaw data. However, since the magnetometer yaw was found noisy, we want to pass the readings through a low-pass filter to eliminate the high noise while using most of the low noise gyroscope data by passing it through a high pass filter. The complemented signal is the addition of both filtered noise and is represented as: $CF = \alpha \times Gyro + (1 - \alpha) \times Mag$ where α is the cutoff percentage. After experimenting with some α 0.95 was shown to give good results.

Since the magnetometer data is very noisy, I would trust yaw integrated from the gyro as it is less noisy and less prone to biases (they still have biases).

Forward Velocity Estimation

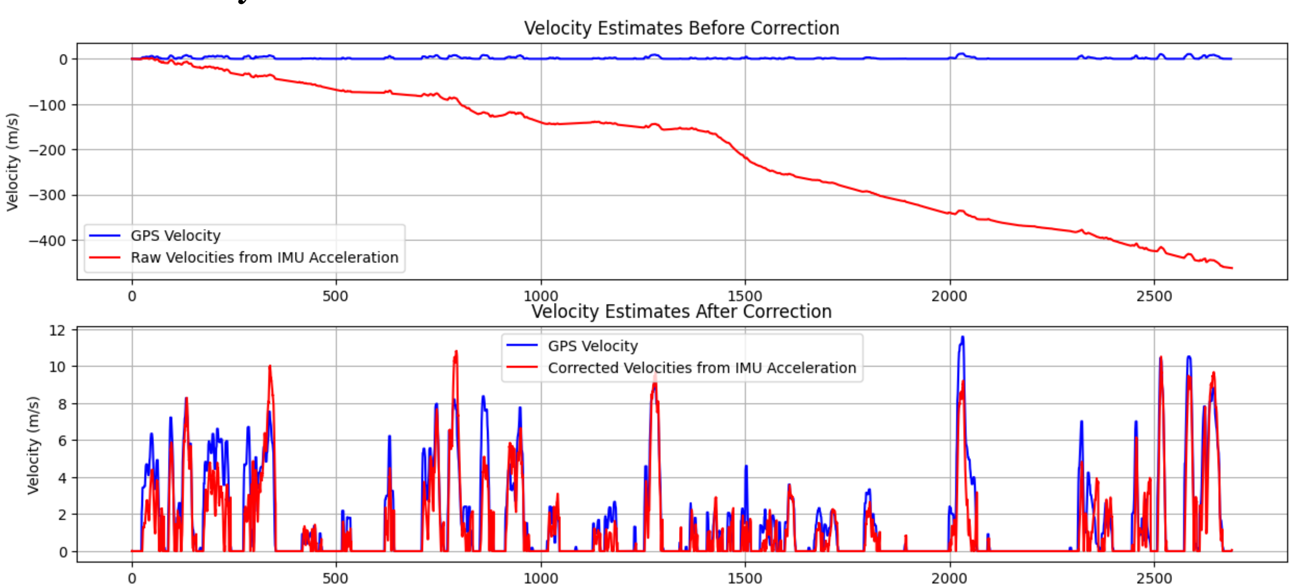

Figure 5: GPS Calculated Velocities and Raw Acceleration (Top). GPS Velocities Compared to IMU Integrated Acceleration Velociteis after correction (Bottom)

The velocity estimate as seen in Figure 5 (top) shows how acceleration bias can completely drift the velocity obtained by integration. In our case, this is explained by the long drive time we had to take in addition to long stop times which allowed the acceleration bias to accumulate as time goes. To better estimate the velocity, I followed three steps. First, adjust the acceleration bias by running a convolving window over the data and subtracting the results from the acceleration data. The first few data points are skipped as they seem to have the least bias. Second, the acceleration is integrated to find the velocity. Third, detect stops in the velocity data by setting a threshold that, if a velocity is less than, the said velocity is subtracted from the following velocity data to remove the bias inflicted by this stop point. **Note:** I had three ways of finding the stops, one being using the GPS data, another being recording timestamps where we stopped, and finally using the velocity data. The three methods yielded acceptable results. The main discrepancy between estimating the velocity from an accelerometer and GPS is the bias. The GPS velocity needed no correction and was more than zero once integrated, while the accelerometer velocity estimate had negative results even though the car never reversed. The bias adds up as you go, and it makes the data alignment a more difficult task.

Dead-Reckoning

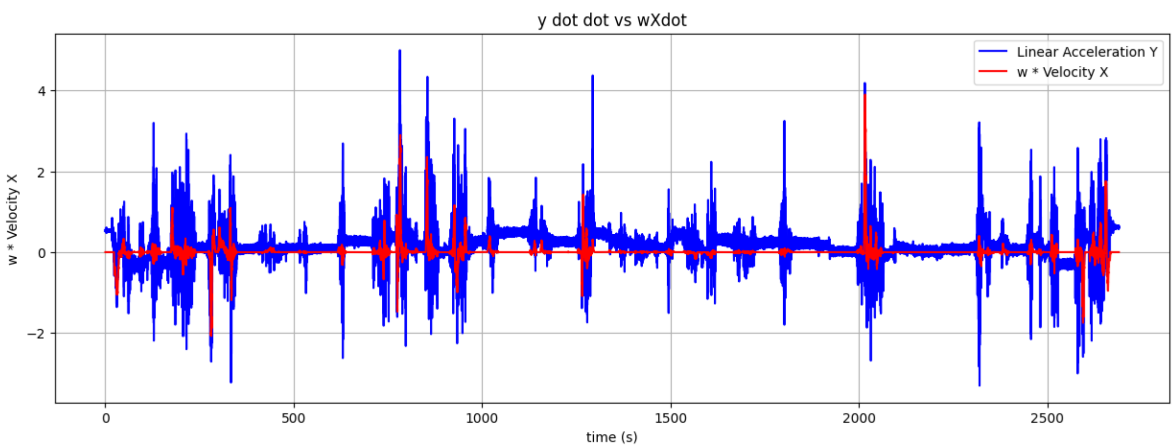

Figure 6: Observed Linear Acceleration (Y-axis) Compared to Computed Linear Acceleration (Y-axis) from w and GPS Velocities

The ωX data, which was calculated by multiplying the angular velocity in the z direction to the GPS velocities showed the same trends when plotted against the linear acceleration in the y-axis. However, the acceleration data shows noise and drifts as the bias wasn't corrected on this data nor it was filtered.

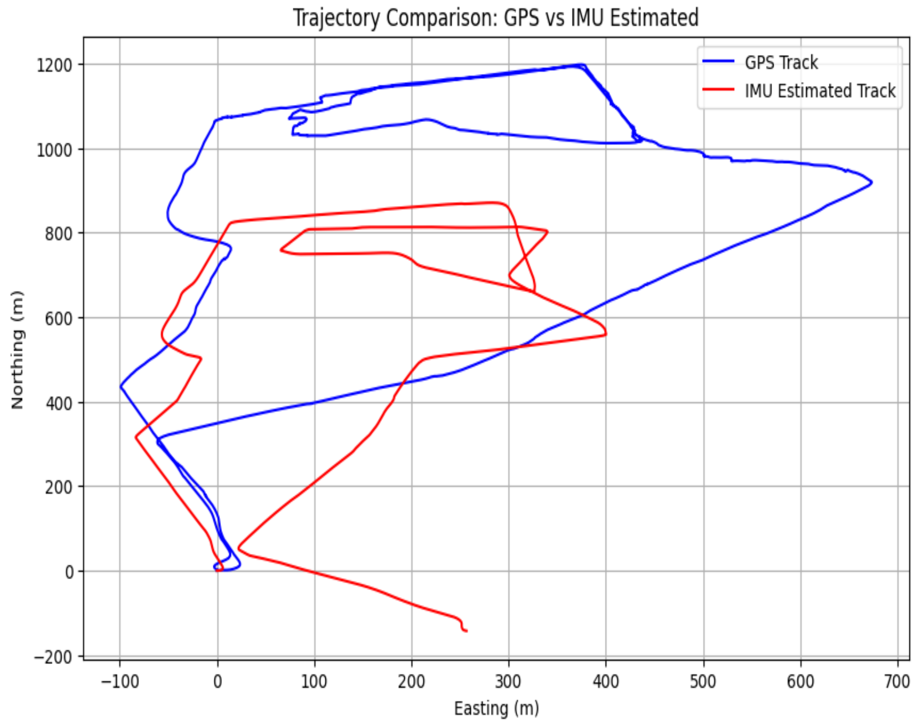

Figure 8: Trajectory from GPS Coordinates vs Trajectory from IMU

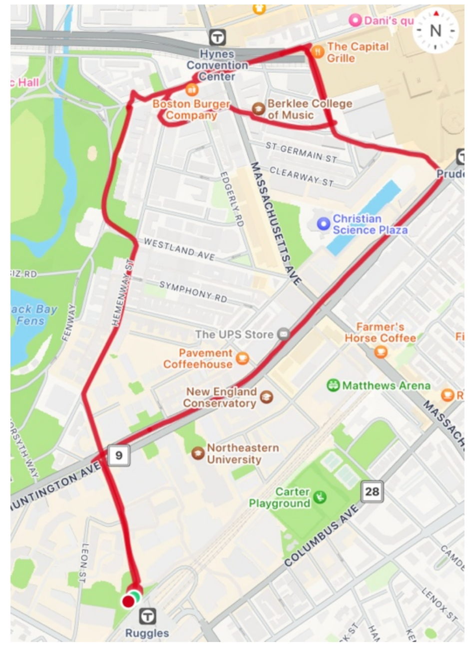

Figure 7: Trajectory on MapMyRun

A an angle of 137⁰ was used to turn the IMU heading Clockwise to align with the GPS trajectory

Based on my trajectory, the two trajectories aligned closely for the first leg which was a period of exactly three minutes. However, I believe it can do a limited range over that since our data had an issue of being long with many stops. The IMU trajectory seen in Figure 7 shows a correct over all headings however, it didn't align with the GPS trajectory except for 3 minutes. Additionally, since the data collection was lengthy, the end part of the trajectory shows excessive bias.

Xc

$$\ddot{x} = \dot{v} + \omega \times v = \ddot{X} + \dot{\omega} \times r + \omega \times \dot{X} + \omega \times (\omega \times r)$$

From the given equation we can simplify to have x_obs = Y_dot + ωX_dot + ω^2 x_c. We got this equation by focusing taking the Y component from the equation. Rearranging would give xc = x_obs – Y_dot – ωX_dot/ ω^2

Since we are only focusing on the x component we will assume the Velocity Y_dot is zero. Hence, we get xc = x_obs– ωX_dot/ ω^2