Underground Vehicle Navigation

EECE-5698 Team Project Final Report

Jake Messner, Amanda Zhu,
Darshan Jain, Shaoshuang Jian

April 10, 2018

Hardware Used:
GPS : GlobalSat BU-353-S4 USB GPS Receiver
IMU: VN-100
Northeastern University Car of the Future
Encoder: Duralast Vehicle/Transmission Speed Sensor

Repo: https://gitlab.com/zhu.yil/Tunnel-Navigation-Project

All group members have completed TRACE review

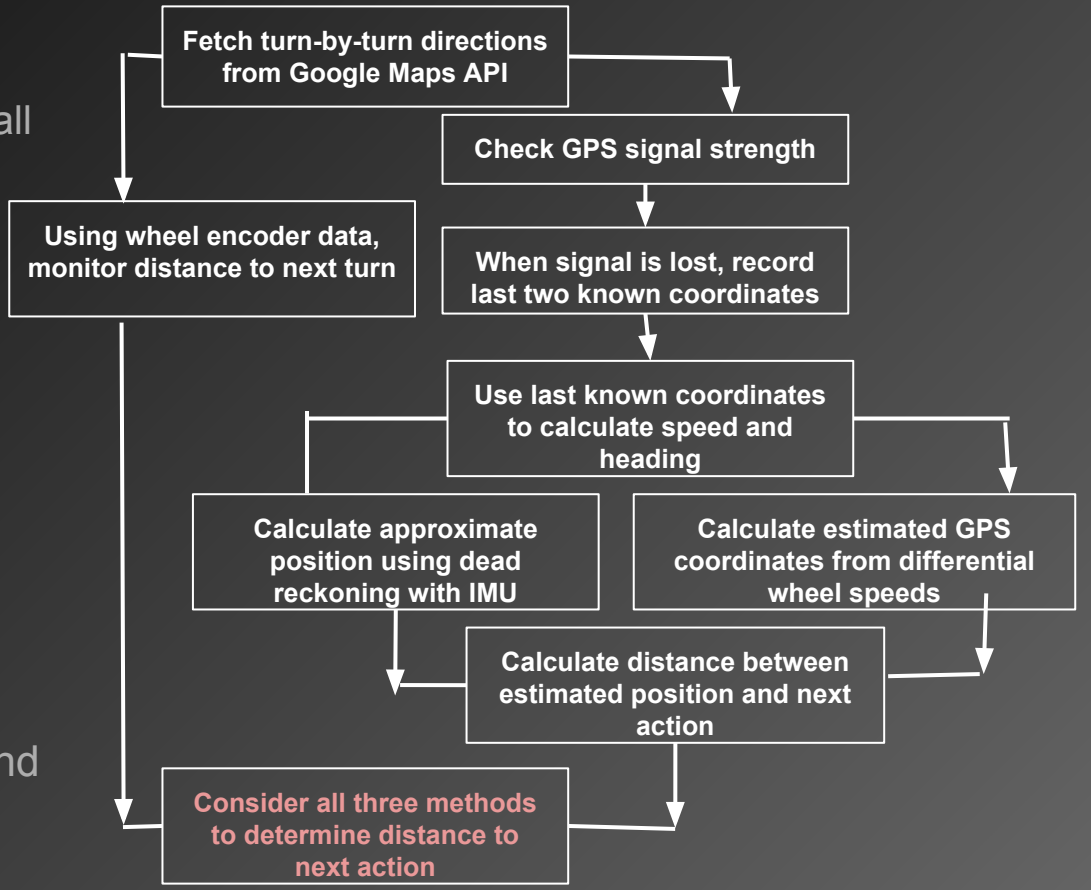
Problem

Traditional navigation systems lose GPS connectivity when heavy sky cover is present such as in tunnels, beside many tall buildings, and in dense forest. This prevents applications from sending turn-by-turn navigation in such areas.

Our Approach

Using the Northeastern University autonomous car, we will utilize sensor data to supply users with real-time turn-by-turn directions in areas with no satellite view. We have identified three possible methods, which we will test and combine to produce accurate results.

Algorithm Execution



Data collection

- 5 attempts
- Conclusions:
  - All sensors in car have to be plugged in for complete data collection.
  - The GPS on the car itself produces continuous data in tunnel
    - It is possible the Lincoln MKZ is automatically utilizing a suite of sensor data to perform dead reckoning and reporting these results as GPS data. It is also possible the car is interfacing over LTE with a real-time mapping API that delivers estimated GPS updates based on the position trajectory of the road on which the car is travelling. We were skeptical about the accuracy of these coordinates and implemented our own methods to validate their accuracy.
  - Use our own GPS (from lab 1) to collect GPS Instead
  - Issue ran into when running GPS LCM driver: LCM logger automatically omits data as if set as float(0)
- Tunnels travelled:
  - Thomas O'Neil Tunnel (The "Big dig")
  - Summer/Callahan Tunnel Tunnel (Boston <-> East Boston)
  - Ted Williams Tunnel (Boston <-> Boston Logan International airport)

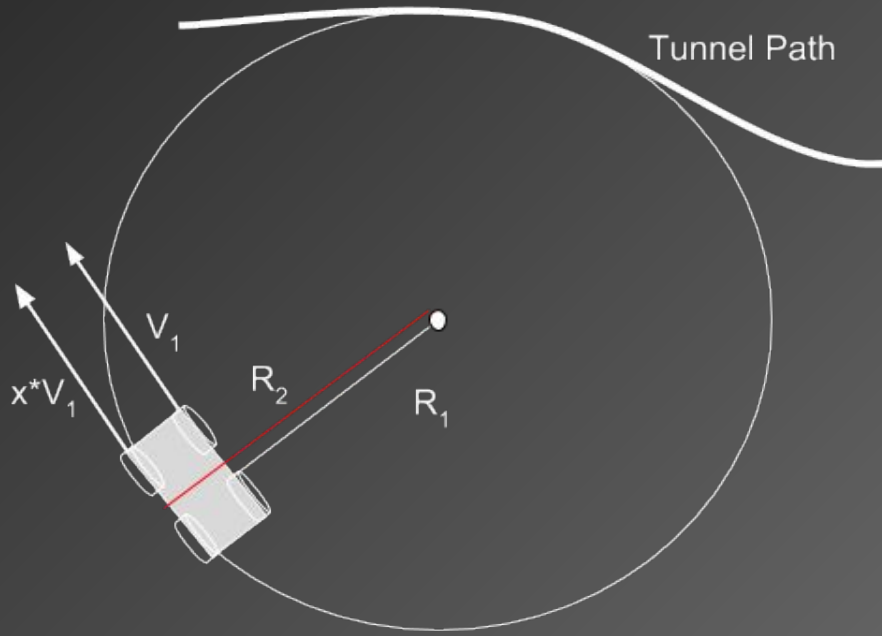
Data Collection in Different Traffic Conditions

- We recorded one dataset in light traffic conditions
  - 13 minutes to drive from Somerville into Chinatown
  - This data was our cleanest dataset and required the least filtering
- We recorded one dataset in extremely heavy rush hour traffic
  - 45 minutes from Logan Airport to Northeastern University
  - This data had accurate GPS and wheel speed measurements, though the IMU data tended to drift more significantly

Position Tracking via Analysis of Differential Wheel Speeds

$$r_1 = \tfrac{1}{2}\,(3.3v_1/(v_2-v_1))$$

1. Calculate ratio of inside and outside wheel speeds
2. Using wheel speed, calculate turn circle radius over small time interval
3. Using average wheel speed, calculate distance travelled of car center of mass
4. Update estimated car heading
5. Using heading and distance travelled, estimate new GPS position
6. Plot on map



Relevant Equations Used

Entrance Heading and Update GPS equations from https://www.movable-type.co.uk/scripts/latlong.html

Calculate Tunnel Entrance Heading:

Formula: $\theta = \text{atan2}(\sin \Delta\lambda \cdot \cos \varphi_2 , \cos \varphi_1 \cdot \sin \varphi_2 - \sin \varphi_1 \cdot \cos \varphi_2 \cdot \cos \Delta\lambda)$

where φ_1,λ_1 *is the start point,* φ_2,λ_2 *the end point (*$\Delta\lambda$ *is the difference in longitude)*

Calculate Turn Radius: $d_1 = 3.3v_1/(v_2-v_1)$

Update Estimated GPS Position:

Formula: $\varphi_2 = \text{asin}(\sin \varphi_1 \cdot \cos \delta + \cos \varphi_1 \cdot \sin \delta \cdot \cos \theta)$

$\lambda_2 = \lambda_1 + \text{atan2}(\sin \theta \cdot \sin \delta \cdot \cos \varphi_1, \cos \delta - \sin \varphi_1 \cdot \sin \varphi_2)$

where φ *is latitude,* λ *is longitude,* θ *is the bearing (clockwise from north),* δ *is the angular distance* d/R*;* d *being the distance travelled,* R *the earth's radius*

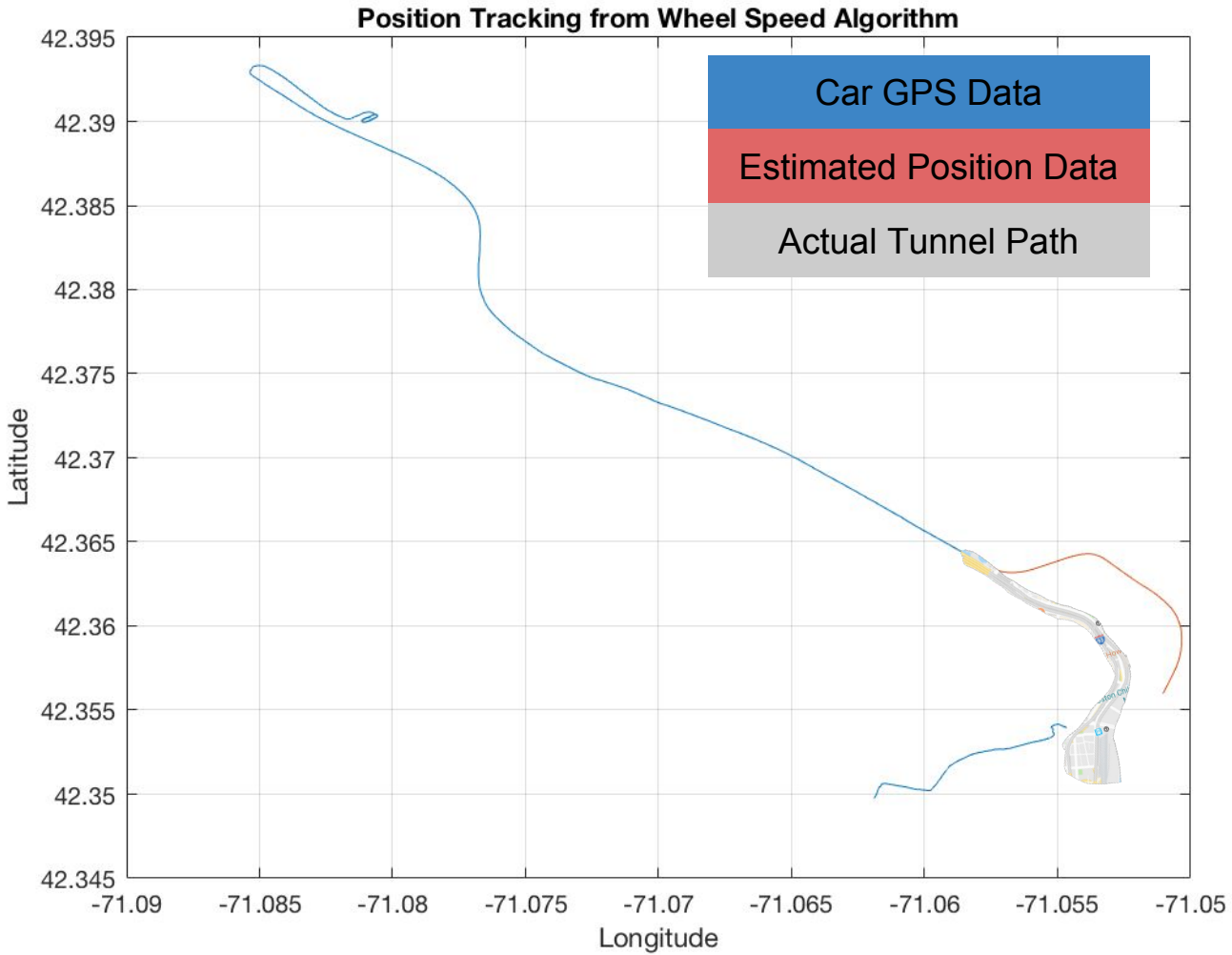
Update Estimated Heading:

New_heading = (dist/radius)*2pi

Challenges and Sources of Error

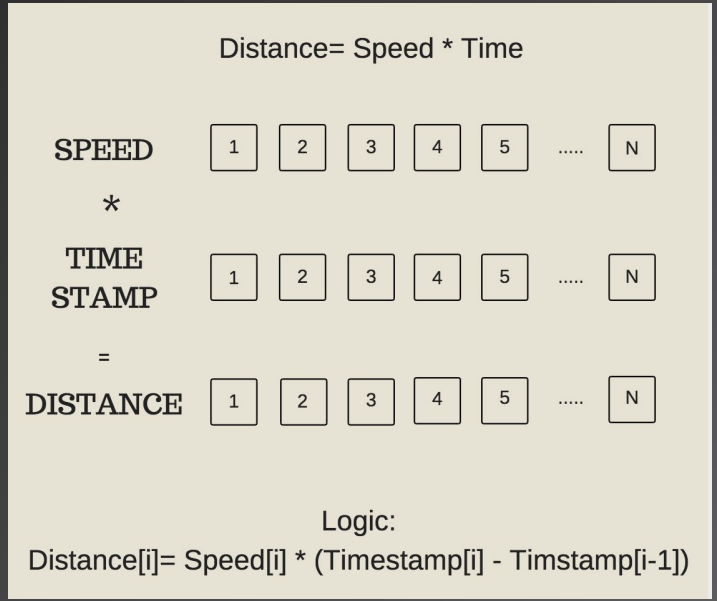
- Accounting for wheel speed variations such as potholes, slipping, and changing lanes
  - Use moving average of last 25 wheel speed points
- No accounting for elevation changes
- Converting between many different units for different formulas

0.25 Scaling was added to heading function to fit to data


Position Tracking from Wheel Speed Algorithm
Car GPS Data
Estimated Position Data
Actual Tunnel Path
Latitude
Longitude
42.395
42.39
42.385
42.38
42.375
42.37
42.365
42.36
42.355
42.35
42.345
-71.09
-71.085
-71.08
-71.075
-71.07
-71.065
-71.06
-71.055
-71.05

Distance Tracking

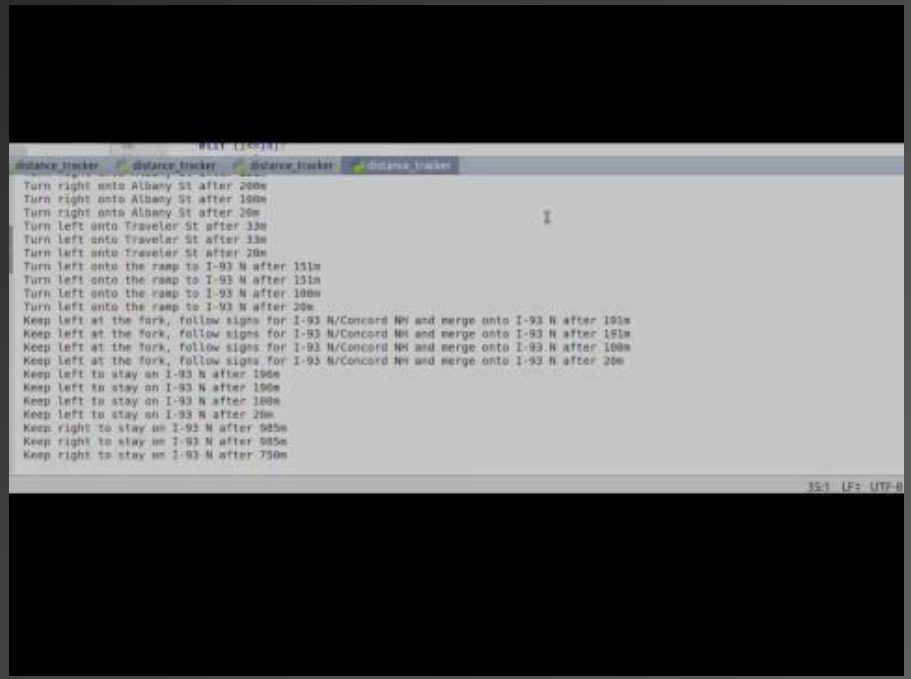
1. Subscribe to rostopic /vehicle/twist which gives the velocity of the car and yaw rate (50hz)
2. Calculate the circumference of the wheel
3. Using the circumference of wheel,velocity of the car, find the distance travelled.
4. Get the sub routes from the origin address to the destination address
5. The distance tracker keeps a track of the distance travelled and notifies the user with distance left before turn



Google Maps API Direction Fetch

```
Origin Address:  75 Mystic Ave, Somerville, MA 02145, USA

Destination Address:  Jeharis Family Center for Biomedical and Nutritional Sciences,

Origin location:  lat: 42.3898161, long: -71.0799681

Destination location:  Jeharis Family Center for Biomedical and Nutritional Sciences

Leg start location:  lat: 42.3898161, long: -71.0799681
Leg end location:  lat: 42.3899148, long: -71.0800895
instruction:  Head northwest
distance: 15 m


Leg start location:  lat: 42.3899148, long: -71.0800895
Leg end location:  lat: 42.3895335, long: -71.0805893
instruction:  Turn left toward Mystic Ave
distance:  59 m


Leg start location:  lat: 42.3895335, long: -71.0805893
Leg end location:  lat: 42.3892322, long: -71.0801792
instruction:  Turn left toward Mystic Ave
distance: 48 m
```

Active Distance Tracking Demonstration



Challenges and Sources of error

- Wheel radius measurement error alters distance travelled
- Decrease in Tire-pressure can change wheel radius slightly
- Style of driving (With or without hard-braking) can introduce error

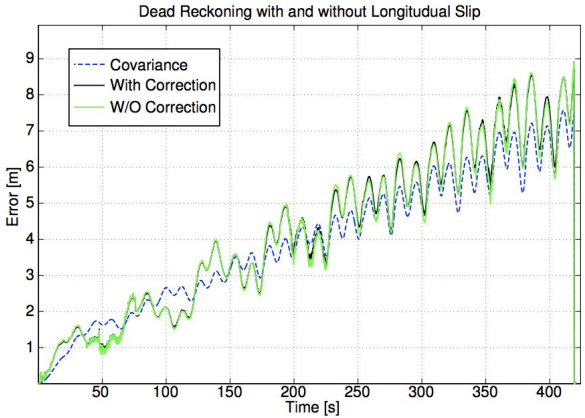


Figure 3.6: Error accumulation during dead-reckoning. The vehicle which uses brakes to decelerate accumulates similar errors to one which does not.

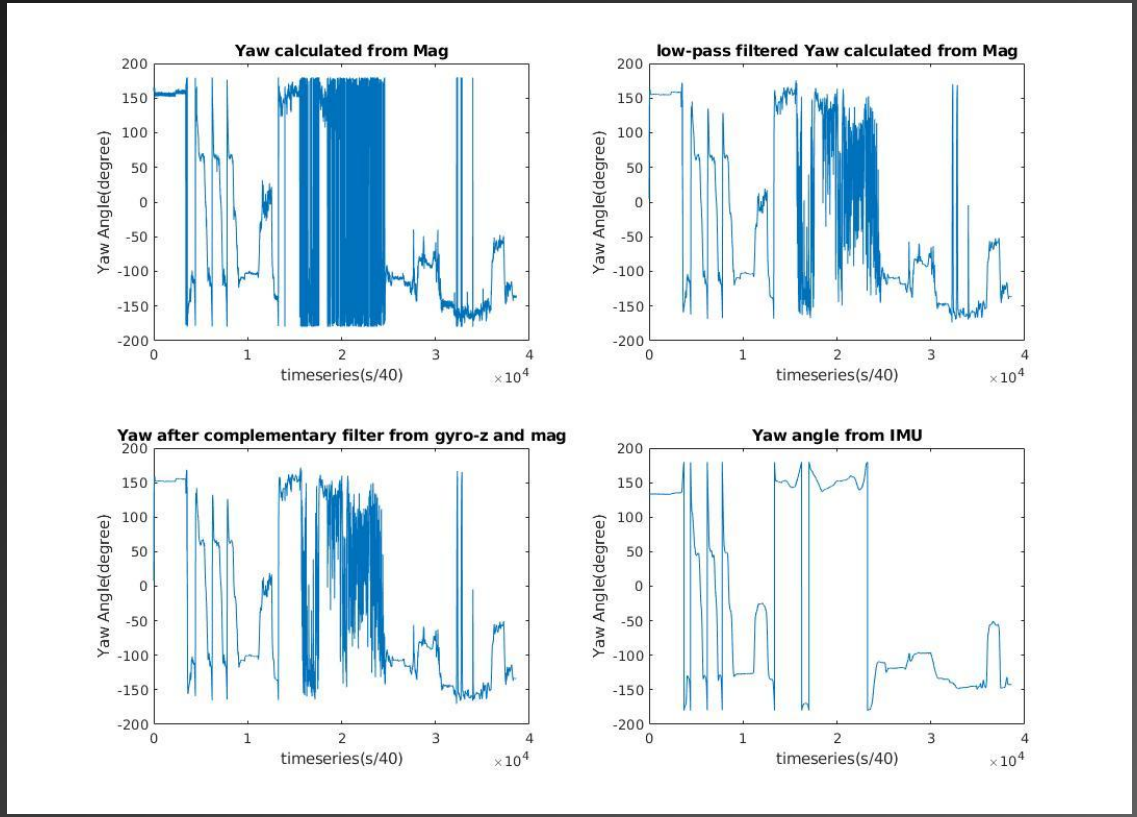
Practical position and yaw rate estimation with GPS and differential wheelspeeds- by Christopher Carlson

Dead Reckoning (Position Tracking - IMU)

1. Calibrate magnetometer reading from hard and soft iron effects.
2. Implement complementary filter with magnetometer yaw and gyro yaw, adjust yaw to get estimated bearing for estimation.
3. Integrate accelerometer-x to obtain car velocity in the tunnel.
4. Adjust velocity with initial speed calculated from wheel speed when entering the tunnel.
5. Estimate latitude,longitude in the tunnel according to equation on page 6 with heading, latitude, longitude of last point.
6. Compare estimated route with other position estimations.

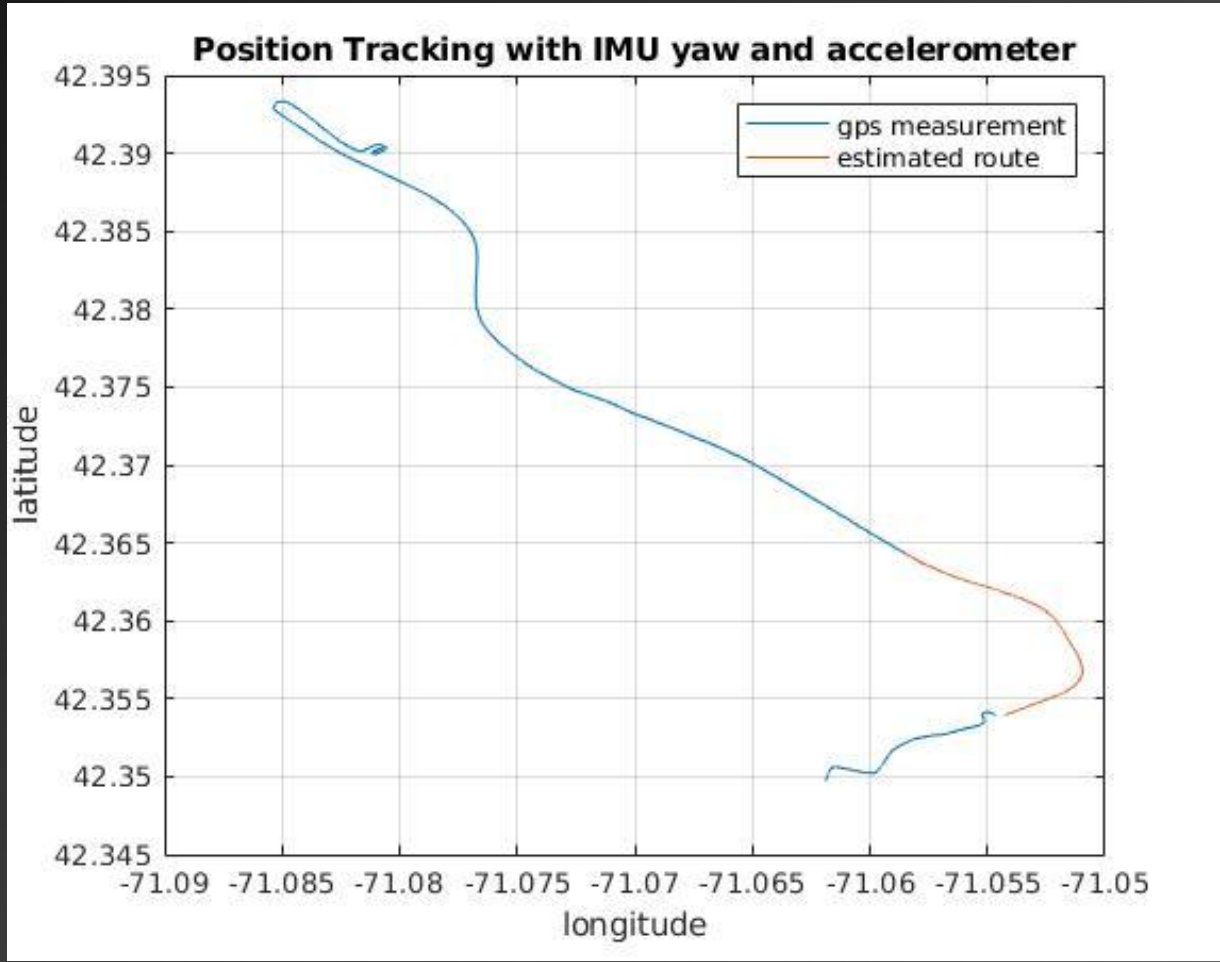
Magnetometer Correction

- Large portion of high frequency noises in Mag yaw.
- Might be caused by unevenly distributed original magnetometer readings.
- Used Yaw from IMU as heading for estimation.

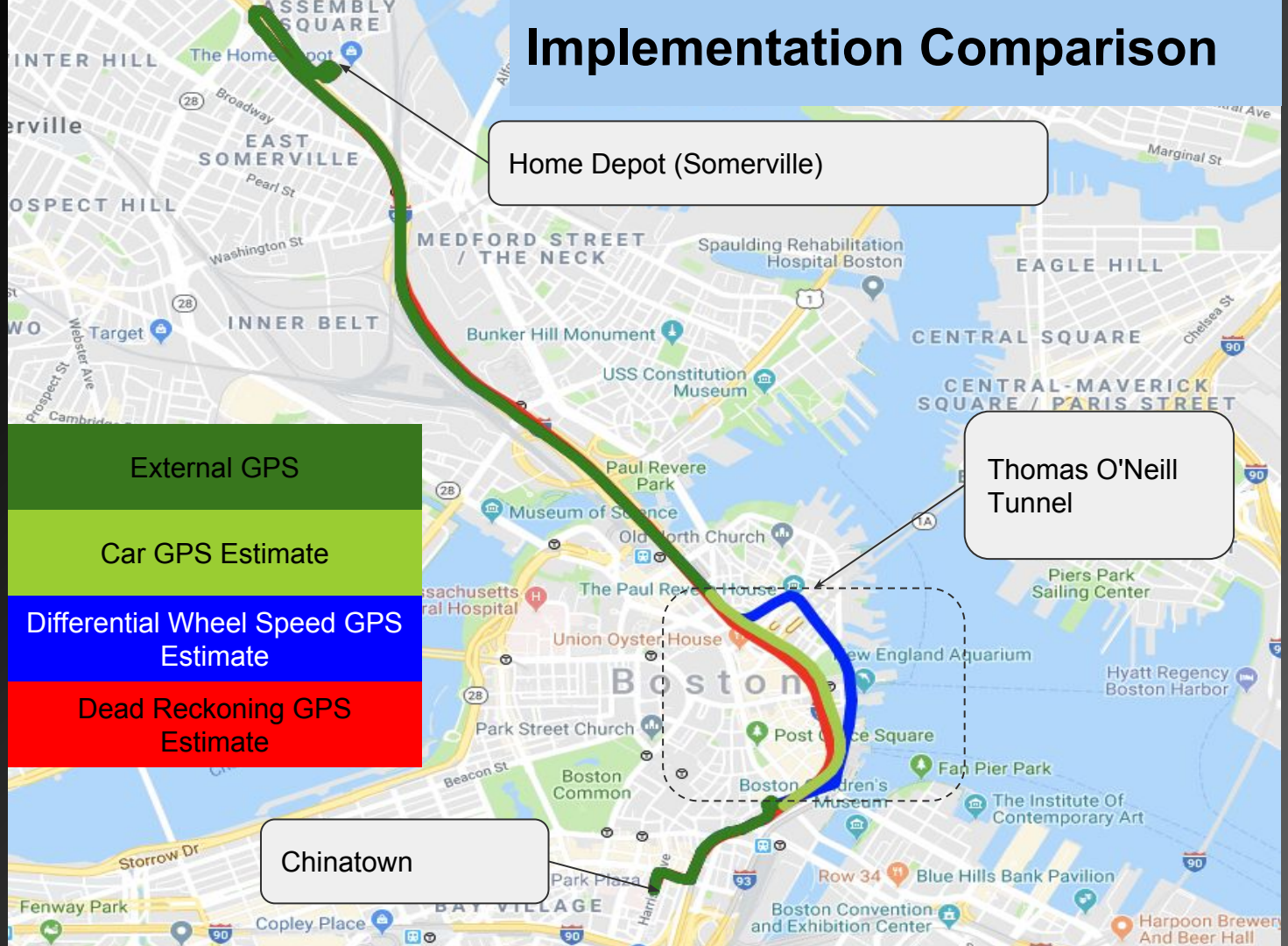


Route Plotting based on IMU data

- Initial heading: 140 degree
- Adjust Yaw from IMU with initial heading for bearing estimation.
- From wheel speed get Initial speed when entering the tunnel: 22.339m/s



Implementation Comparison



Future Work

- **Dead Reckoning ground truth at tunnel entrances and exits (implemented)**
- **Improved averaging of differential wheel speed data (implemented)**
- **Cleaner merge of all three implementations for a seamless user experience (partially implemented)**
- **Better interface with professional mapping API (hard to visualize without high-quality map overlays) (implemented, Google Maps)**
- Implementation of real-time mapping updates without cellular connectivity
- Real-time IMU data filtering for dead reckoning