Ground Vehicle IMU and GPS Sensor Fusion Filter Comparison

Michael Swafford
IIHR-Hydroscience & Engineering
The University of Iowa
Iowa City, USA
michael-swafford@uiowa.edu

***Abstract*—Summary of motivation, summary of lit review, summary of methodology, summary of results and findings. <200 words.**

***Index Terms*—data fusion, navigation, Kalman filter**

I. Introduction

Autonomous robots are becoming more prevalent in both research, industry, and consumer life. For autonomous ground robots the accuracy of the positioning system is integral to its ability to localize within the larger operational environment Caron et al. [1]. For small mobile ground robots this localization can either be done through processes like Simultaneous Localization and Mapping Bailey and Durrant-Whyte [2] or more simply and where possible can be accomplished with an internal sensor like and IMU or an external sensor like GPS. Unfortunately, for small ground mobile robots the available space is at a premium meaning that sensors cannot be large, energy consumption is limited by the size of the battery, and computational effort. Because of this many small robots rely on low-end IMU and GPS sensors to determine their pose. These sensors must be used in conjunction because IMU's on their own are prone to measurement drift over time since current measurements on based on the previous measurement Sukkarieh [3]. Likewise, low-end GPS units have limited positional accuracy and suffer from errors due multi-path signal acquisition or signal latency Sukkarieh [3]. This is why many have turned to filter algorithms to fuse these two measurements to get the best pose estimate, however, it is not clear among the many filters which is the best estimator for the mobile ground robot application.

Sukkarieh, Sukkarieh [3], pioneered work in the year 2000 publishing his thesis on a low-cost data fusion between GPS and inertial sensors using a Kalman filter for ground vehicles. Sukkarieh tested his algorithm on a 65 tonne autonomous straddle carrier Sukkarieh [3]. Sukkareih also theorized encoder data could also be added to further reduce estimation errors. Later in 2004 Caron et al, Caron et al. [1], took Sukkarieh's work further by improving data rejection for both the GPS and IMU based on contextual logic as well as adding the ability for more sensors to be added to the algorithm. Lastly Matsson, Matsson [4], showed that an unscented Kalman filter could be applied to the data fusion problem and tested on a small autonomous R/C car. Matsson's car navigated by following lane lines on a track and utilized wheel encoders in addition to the GPS and IMU data Matsson [4].

What has not been made clear in the literature is a comprehensive comparison of several fusion filters for the same vehicle and trajectory. The objective of this project was to construct a simulation process to test and compare different data fusion filters for IMU and GPS sensor data for pose estimation of a ground vehicle. Specifically, in this project the extended Kalman filter (EKF) and the error-state Kalman filter (ESKF) are used to estimated the position and orientation of a tenth scale remote control (R/C) car. The filters were compared at both high and low GPS sensor refresh rates of 10Hz and 1Hz to simulate a real world environment as well as a lower end GPS sensor.

II. Methodology

A. Mechanical System

While any small mobile robot could have been chosen for IMU/GPS sensor fusion comparison, a one-tenth scale R/C car was chosen for several reasons. First, an R/C car has a kinematic model more closely resembling most ground vehicles used for transportation, one motor to drive a set of wheels and steering by turning the front set of wheels. Second, an R/C car is an application where an operator or scientist may want to have high quality pose information for either performance improvement or dynamic understanding. Lastly, an R/C car is not only an easy case to numerical simulate but is also a cost effective and simple case to conduct real-world validation tests. An example of one of these cars is shown in Figure 1. These cars typically have a wheel base between 220mm and 280mm, have one electric motor to drive the rear wheels, have independent front and rear suspension, and have one servo motor to turn the front wheels. The car simulated in this report had a wheel base of 260mm, drove at a constant velocity of 8.33 m/s (30 kph) and drove the nonlinear super-ellipse path around the Old Capitol in Iowa City, Iowa as shown in Figure 2.

B. Process Model

The motion model used for the R/C car is shown in Figure 3 and the discrete state-space model is shown in equation 1, where k is the discrete time-step.

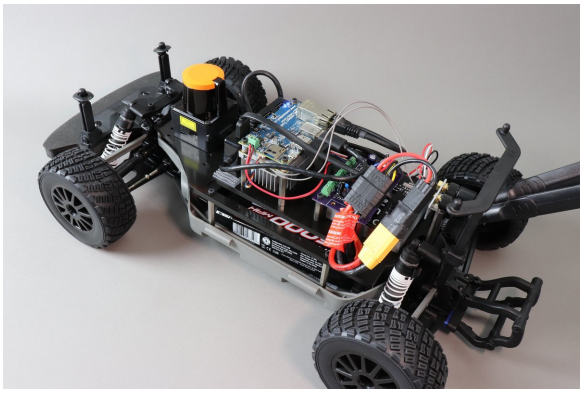

Fig. 1: A typical $1/10^{th}$ R/C Car. This exact example comes from the F1Tenth racing series, an autonomous racing series that use AI and controls algorithms to race around predefined race tracks.

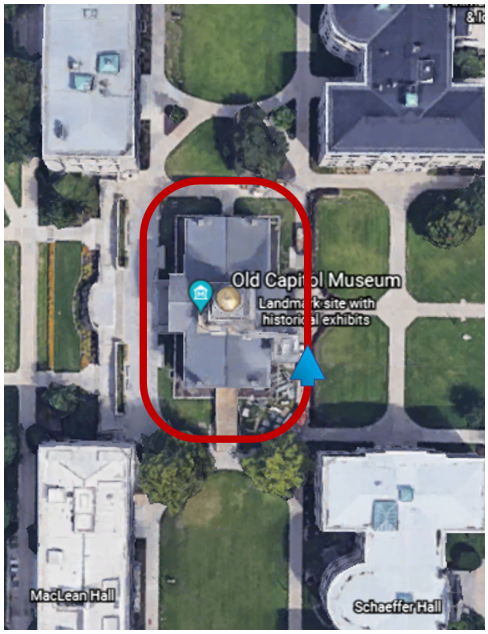

Fig. 2: The planned objective trajectory around the Old Capitol in Iowa City, Iowa. The path is shown in red and the blue arrow shows the starting position and orientation of the vehicle.

$$X_{k+1} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \\ \phi_k \end{bmatrix} + \begin{bmatrix} Vcos(\theta_k)\Delta t \\ Vsin(\theta_k)\Delta t \\ \frac{V}{L}tan(\phi_k) \\ \dot{\phi}_k \end{bmatrix} \tag{1}$$

The car pose is composed of the x-position, the y-position, and the heading angle. The state space equation takes these into account as well as the steering angle ϕ. Equation 2 represents the input u to the motion model and is made up of the total velocity V and the steering angle rate $\dot{\phi}$.

$$u_k = \begin{bmatrix} V \\ \dot{\phi} \end{bmatrix} \tag{2}$$

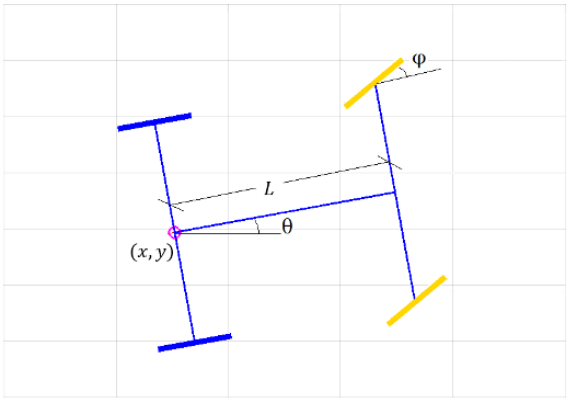

Fig. 3: R/C car motion model diagram. L is the wheel base, θ is the heading, and ϕ is the steering angle.

Both inputs are subject to zero-mean Gaussian process noise shown in equations 3 and 4.

$$V \sim \mathcal{N}(\mu = 0, \sigma = 0.05\frac{m}{s}) \tag{3}$$

$$\dot{\phi} \sim \mathcal{N}(\mu = 0, \sigma = 0.00175\frac{rad}{s}) \tag{4}$$

These standards of deviation were chosen to represent hobby grade components, where the motor has the ability to control the velocity within a standard of deviation of 5 $\frac{cm}{s}$ $(0.05\frac{m}{s})$ of the desired one. The steering actuator motor has the ability to control the desired steering angle within a standard of deviation of 0.00175 $\frac{rad}{s}$ $(0.1\frac{deg}{s})$.

C. Measurement Model

The measurement model for the R/C car is shown in equation 5, where $Z_{k,i}$ represent the measurement at time-step k from sensor i. H_i represents the measurement system matrix for sensor i, X_k is the current state. The measurements available come from GPS and IMU data where GPS is able to provide the x-position, y-position, z-position, and ground speed and the IMU is able to provide the linear accelerations and angular velocities.

$$Z_{k,i} = H_i X_k + b_{k,i} \tag{5}$$

$$b_{k,GPS} \sim \mathcal{N}(\mu = 0, n\sigma^2) \tag{6}$$

$$b_{k,IMU} \sim \mathcal{N}(\mu = 0, \sigma) \tag{7}$$

The measurement noise is taken into account in $b_{k,i}$. The GPS measurement noise is a Gaussian random walk, shown in equation 6 and the IMU measurement noise zero mean Gaussian white noise represented in equation 7. The simulated sensors are shown in Figure 4, both are low-cost sensors with a price tag under $25. The Grove GPS module can sample up to 10Hz, has a horizontal and vertical accuracy of 2.5m, and a velocity accuracy of $0.1\frac{m}{s}$. The IMU sensor samples at 100Hz, representing nearly continuous data, and was run on its level 2 setting. The accelerometer has a maximum linear acceleration range of up to 4g's, linear acceleration resolution of $1\frac{mm}{s^2}$

Fig. 4: Left: Grove - IMU 9DOF v2.0 - MPU-9250. Right: Grove - GPS Module

$(0.001\frac{m}{s^2})$, and a noise density of $2.9\frac{mm/s^2}{\sqrt{Hz}}$ $(0.0029\frac{m/s^2}{\sqrt{Hz}})$. The gyroscope has a maximum angular velocity of 500 $\frac{deg}{s}$, angular velocity resolution of $0.015\frac{deg}{s}$, and angular velocity noise density of $0.01\frac{deg}{s}$.

D. Simulation

To conduct the simulation a predetermined super-ellipse trajectory was generated as shown in figure 5. The path is 123m long and 48m wide with fast corners that resemble the real trajectory that a ground vehicle would take versus a squared off corner. A circular path was chosen to see how much drift would be present at the end of the experiment. This

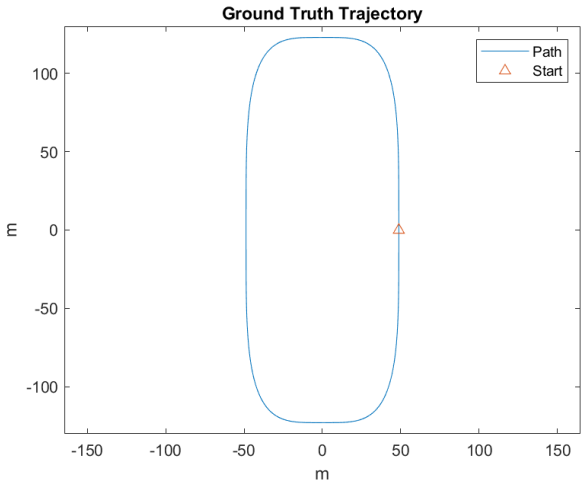

Fig. 5: Super-ellipse path in blue and starting location and direction shown as the orange triangle.

path and the resulting orientation were then used to determine the steering rate required to properly navigate so that the process model could correctly be implemented in MATLAB. The navigation toolbox in MATLAB was then used to fully simulate both sensors based on the information from their data sheets. The next step is to setup the filter, both the EKF and ESKF come from MATLAB's tracking and data fusion toolbox. The simulation was run at the IMU sampling rate, at each step the ground truth trajectory gave a true set of data (position, orientation, velocity, etc.) that was fed into the IMU to simulate what the sensor would receive. The acceleration and gyroscope data from the simulated IMU was then fed into the prediction step of the filter being tested, after which the estimated pose was found. If the time stamp matched when the GPS would receive a sample the ground truth position and velocity was fed to the simulated GPS. This would then be fed to the update step of the filter to correct the estimate. The output from the simulation was an animated video showing the true and estimated path as well as the true and estimated orientation. The simulation also reported the time history of error for the x, y, and z position as well as the quaternion distance. A graph of the proposed path(without process noise), the true path, and the estimated path from the filter was also available at the end of the simulation to understand the effectiveness of the filter. Lastly the end-to-end root-mean-square (RMS) of the position and the quaternion are generated for comparison.

E. Performance Metrics

The filters will be analyzed by three different metrics, first will be a qualitative evaluation of the ability for the filter to estimate the correct path by looking at the last corner of the trajectory. Second, the time histories of the error will be directly compared to understand the behavior of each filter. Lastly, the RMS end-to-end error will be compared for each filter, the equation for which is shown in equation 8.

$$RMS = \sqrt{[\sum_{k=1}^{K}(\hat{X} - X_{true})]^2} \tag{8}$$

III. Results

Results from the simulation were generated for the EKF and ESKF at both a 10Hz and 1Hz GPS sampling rate. Results for the higher 10Hz rate are shown first followed by the 1Hz rate. Figure 6 shows a snapshot of the path and orientation animation.

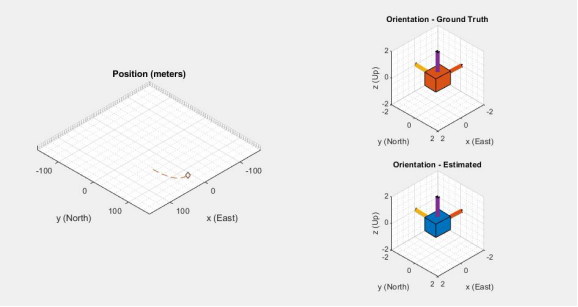

Fig. 6: Extended Kalman filter IMU/GPS fusion animation for 10Hz GPS rate.

Comparing the EKF and ESKF final corner path's in Figures 7 and 8 it is observed that the EKF does a better job at estimating the true path. The ESKF appears to have an offset through the corner but converges to the true path once the vehicle straightens out its trajectory. Both are offset from the ideal planned path due to the process noise that exists in the motion model. Analyzing the 10Hz time histories of position and orientation error in Figure 9 it is seen that for the quaternion distance both filters behave similarly, accumulating a low order of magnitude of error. It is seen that the ESKF

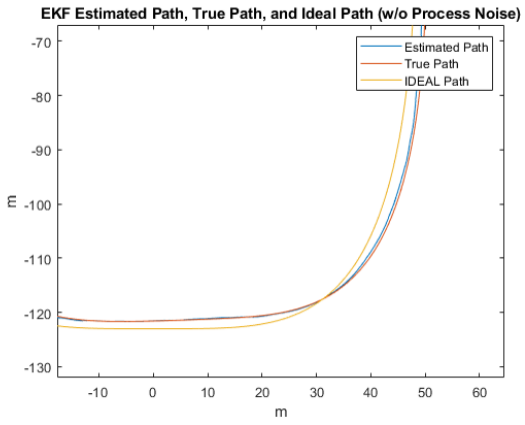

Fig. 7: EKF 10Hz final corner path. Notice that the EKF does a good job staying with the true path.

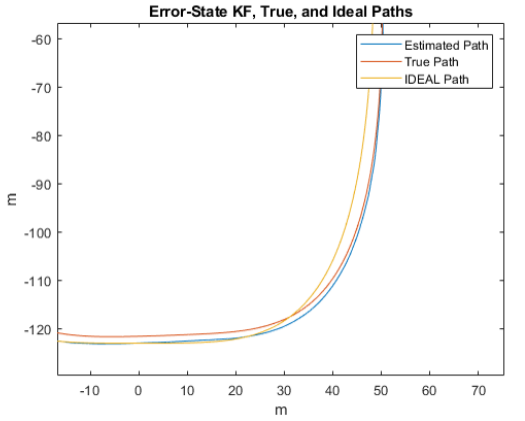

Fig. 8: ESKF 10Hz final corner path. Notice that the ESKF stays offset through the constant y-position traverse but then converges after the corner.

has higher noise frequency but with a lower bias offset, where as the EKF varies throughout the entire interval but corrects to lower values after navigating through a corner. For the x-position, the EKF was better at not accumulating a large amount of error. The ESKF varies in a sinusoidal manner when the x-velocity changes sign. With a majority of the simulated path traversed through y-position change it is not at all surprising to see the largest estimation errors here. EKF here again does a better job and again it is seen that the ESKF varies through a sinusoid. Lastly, with no changes in z-position both filters remain low in error magnitude. Summing up what has been observed in the previous figures for the 10Hz GPS sample rate Figure 10 shows the end-to-end RMS error for the EKF and ESKF. It is observed that the EKF does a superior job estimating the position whereas both filters do a good job estimating the orientation with a slight edge given to the ESKF.

Now analyzing the 1Hz GPS sampling rate it is observed that the GPS update corrections are now visible on both final corner paths as shown in Figures 11 and 12. Comparing to the

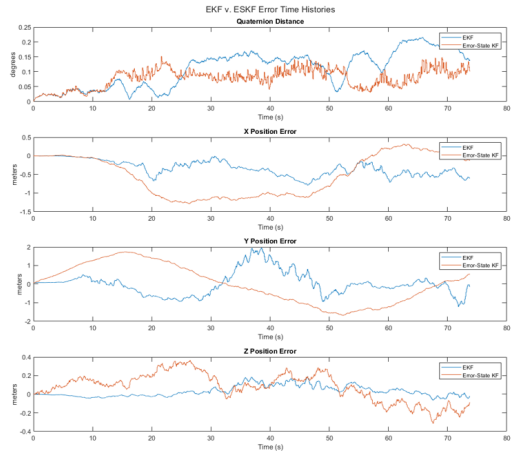

Fig. 9: EKF v. ESKF 10Hz Error Time History. Notice that there is a higher frequency of noise on the EKF as opposed to the ESKF, however, the EKF does a better job correcting to zero error.

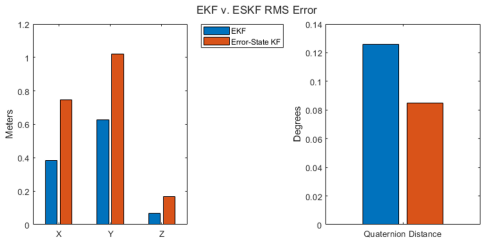

Fig. 10: Comparison of end-to-end root mean square error for the 10Hz GPS update. Notice that for position the EKF is superior, but for ESKF the ESKF is better.

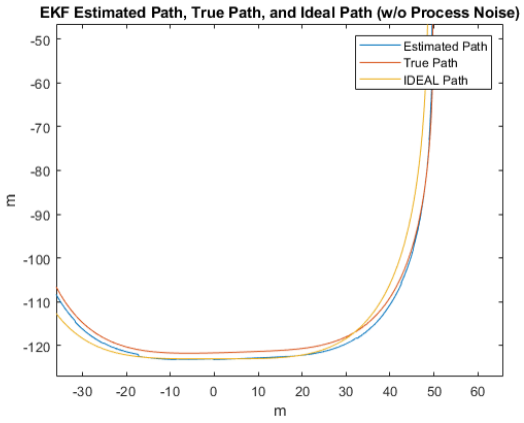

Fig. 11: EKF 1Hz final corner path. Notice that the GPS updates are visible, and that the EKF does a relatively good job estimating the true path.

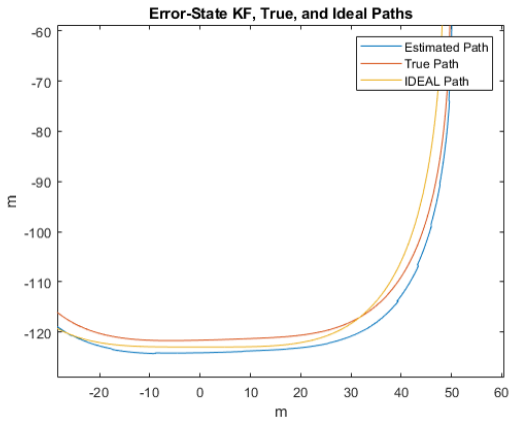

Fig. 12: ESKF 1Hz final corner path. Notice that the GPS updates are visible, again the ESKF stays offset through the constant y-position traverse but then eventually converges after the corner.

higher sample rate both filters operate deficiently at the lower rate, however, the EKF still edges out the ESKF being visually much closer to the true path and converges much quicker after the corner. Figure 11 shows the 1Hz GPS sampling

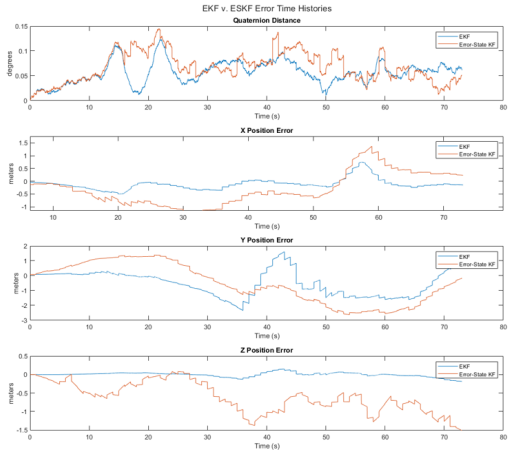

Fig. 13: EKF v. ESKF 1Hz Error Time History. Both filters are much noisier than the 10Hz sample rate but very similar behaviors are still present.

rate time history of error for the position and orientation. Starting with the quaternion distance it is seen that both filters accumulate more error than the 10Hz example but remain at a relatively low overall magnitude compared to the other errors. X and y-position follow similar trends as the higher sample counterparts, however, the GPS corrections are much more visible. The EKF again does the better job at not accumulating significant error. For z-position the EKF remains at a low error, but the ESKF appears to diverge after 30 seconds. More investigation into this behavior should be undertaken. Finally the 1Hz end-to-end RMS error can be viewed in Figure 14. It is

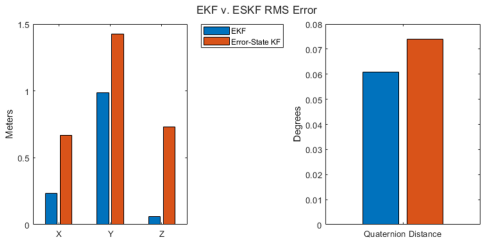

Fig. 14: Comparison of end-to-end root mean square error for the 1Hz GPS update. Notice that the EKF is far superior at the lower measurement sampling rate.

clearly seen that the EKF performs much better than the ESKF at the lower sampling rate for both position and orientation. Notably though both filters still perform better than the GPS alone since the errors are both within 2.5m.

IV. Interpretation and Reflection

Relating back to the Probability Inference and Estimation for Mechanical Systems class (ME:6240) the fundamental course concepts revealed were as follows: the successful estimation of a dynamic state vector, working with a non-linear motion model and linear measurement model, and the successful implementation of the EKF and ESKF. Successfully being able to estimate the dynamic state vector with the included process noise of the R/C car greatly increased the understanding of the stochastic nature of any mechanical system and the importance of being able to develop and apply these estimators. This project also helps to summarize numerically the strengths and weaknesses of two common estimators, the EKF and the ESKF.

The key improvements to be made are to further investigate elevation effects and to compare the unscented Kalman and Particle filters. The investigating elevation would improve the holistic comparison of the two filters studied and would improve transfer to the real world. Especially interested in how the ESKF would behave if elevation is instructed since it struggled at the lower GPS sampling rate. Furthermore, speaking to the comprehensiveness of the study, the unscented Kalman filter and Particle filter should also be added to the study to understand which filter is truly the best for this data fusion application. Unfortunately, this was attempted at the start of the project, but the filters were not working properly in time for the project deadline.

Of the two filters tested the EKF performed the better of the two with overall lower magnitude time history error and end-to-end RMS error. The EKF also was far superior at the lower GPS sampling rate of 1Hz allowing for low power operation. However, both the EKF and the ESKF performed better than the GPS alone at being able to estimate the pose.

REFERENCES

[1] Francois Caron, Emmanuel Duflos, Denis Pomorski, and Philippe Vanheeghe. GPS/IMU data fusion using multisensor Kalman filtering: Introduction of contextual aspects. *Information Fusion*, 7(2):221–230, 2006. ISSN 15662535. doi: 10.1016/j.inffus.2004.07.002.

[2] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (slam): part ii. *IEEE Robotics Automation Magazine*, 13(3):108–117, 2006. doi: 10.1109/MRA.2006.1678144.

[3] Salah Sukkarieh. Low Cost high integrity aided inertial navigation systems for autonomous land vehicles. , (March):212, 2000.

[4] Fredrik Matsson. Sensor fusion for positioning of an autonomous vehicle Design and implementation of an unscented. 2018.

APPENDIX

Running the code: Before you can run the code you must download and install the MATLAB Navigation Toolbox and the Tracking and Data Fusion Toolboxes. The main code is in Project.mlx, it requires two helper functions for the animations to work, HelperScrollingPlotter.m and HelperPoseViewer.m that must be in the same directory to function without errors. The code is extensively commented to aid user understanding and improvements if they wish to make them. By default the code runs at the higher 10Hz GPS sampling rate but can be changed by simply toggling which *gpsRate* is commented out. Running the code is as simple as pressing *Run* in MATLAB and then when prompted saving the animation files under the corresponding filter name. The code is available here for download: https://github.com/Swaffles/Class-work/blob/main/ME6240/ME6240_Project.zip