Localization of Mobile Robot using Classical Bayesian Filtering and Sensor Data

Abhinav Agrahari, Andrew Barlow, Sameeksha Naik, Stephanie Skarica

***Abstract*—This paper documents a sensor data fusion method using Bayesian filtering techniques, tested on a long-term autonomy dataset collected at the University of Michigan. This data consisted of Inertial Measurement Unit (IMU) readings, GPS readings, GPS Real Time Kinematic readings, Fiber Optic Gyroscope (FOG) readings, and wheel encoder readings. States tracked include robot position in latitude and longitude, velocity in the x and y directions (measured in the local frame), robot heading about the z axis (robot yaw), and angular velocity. A differential drive motion model was derived for the robotic platform of interest, and corresponding measurement models were derived to allow for state vector prediction correction. Inputs to the derived motion models included acceleration in the x and y directions and angular velocity, both of which were measured by the on-board Inertial Measurement Unit, as well as wheel velocities provided by the wheel encoders. Ultimately, this paper found that the Extended Kalman Filter method of localization could localize the robot with accuracy comparable to existing SLAM and computer vision techniques, however, significant error was found in regions where GPS signal was lost, including in areas of dense infrastructure, or inside of large buildings.**

***Index Terms*— Bayesian Filtering, Extended Kalman Filter, GPS, GPS-RTK, IMU, Localization, Mobile Robotics, Sensor Fusion**

I. Introduction

THIS paper focuses on the implementation and evaluation of sensor data fusion techniques, employed specifically for the purposes of mobile robot pose estimation. The Bayesian Extended Kalman Filter (EKF) was implemented using Python, and the University of Michigan North Campus Long-Term Vision and Lidar Dataset (NCLT) [1] was used as the primary test dataset due to its accessibility, detailed documentation, and inclusion of both indoor and outdoor scenes.

The dataset utilizes a remotely operated differential drive mobile robot platform pictured in Figure 1, with Real Time Kinematic (RTK) GPS (1), an Omni-directional camera (2), 3D LiDAR (3), IMU (4), Consumer-grade GPS (5), FOG (6), and 2D LiDAR (7) sensors. Wheel odometry was also captured using the on-board motor encoders. The primary issues to be addressed using the implemented sensor fusion method are the relative inaccuracy of the consumer-grade GPS when compared to the more expensive GPS-RTK, signal loss and divergence inside and near dense urban infrastructure, and signal drift of individual sensors (wheel slippage for encoders and quadratic positional error in IMUs). Further, existing SLAM methods may be computationally restrictive for every setting.

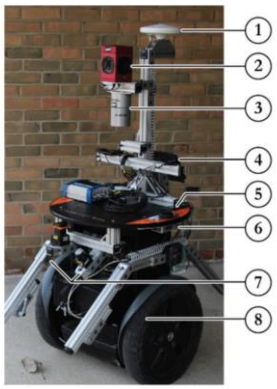

Figure 1. University of Michigan Segway Robot [1]

Pictured below in Figure 2 and Figure 3 is the University of Michigan campus throughout which the data of interest was collected. Regions of note include the long stretches of uninterrupted walkways and roads which will be referenced to determine the efficacy of the sensor fusion method in ideal conditions (that is, with minimal GPS drift) to determine if the method meets or exceeds the performance of existing methods. Other regions of interest include varying paths through large buildings (pictured in Figure 3, highlighted in purple) which will be used to assess the efficacy of the sensor fusion method when the GPS signal becomes significantly unreliable.

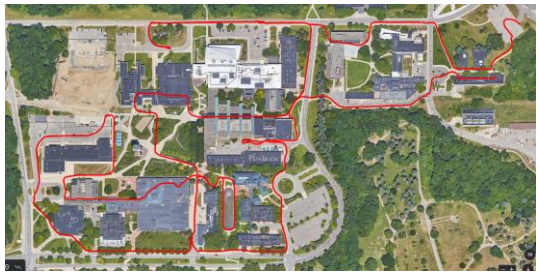

Figure 2: Satellite view of University of Michigan campus with a ground truth path visualized in red

Figure 3: Satellite view of university buildings outlined in purple

The ground-truth measurements included in the dataset have been generated using a "preprocessed large SLAM solution" [1]. This path combined LiDAR scan matched images (seen in Figure 4) and GPS-RTK measurements, allowing for robot pose localization with a significant degree of accuracy. As such, this method can be used as an accurate benchmark with which the Bayesian filtering fusion methods can be quantified against. However, despite the accuracy of this method, some manual post-processing of the data was required [1], particularly with respect to LiDAR scan matching, which can result in a degree of unreliability should this method be used to localize the robot in real-time. As such, there is a need for an accurate sensor fusion method that is robust to signal loss while being competitive with GPS-RTK that can be used in real-time.

Figure 4. LiDAR Point Clouds overlaid onto sample images [1]

II. Background Knowledge and Information

When determining the methodologies and sensor fusion philosophy to be used to localize the robot throughout its path, existing literature was referenced to ensure that the proposed methods were feasible. Firstly, GPS and IMU sensor fusion methods were analyzed [2].

The first method reviewed utilized a low-cost navigation algorithm, implementing both an Extended Kalman Filter and an Extended Information Filter, with their efficacies compared to determine the most effective and reliable estimation methods. This sensor fusion philosophy relied primarily on a microcontroller for sensor fusion and real-time state estimation, a consumer-grade GPS, Inertial Measurement Unit (IMU), and wheel velocity encoders [2]. The methods proposed and tested in the paper fused IMU and consumer-grade GPS measurements together using the Bayesian filtering techniques and were able to achieve real-time state estimation at a rate of 50 Hz. With respect to system modelling, the state estimation techniques utilized the nonholonomic constraints of the robotic platform and linear velocity measurements simultaneously to improve the accuracy of their estimates [2]. The proposed algorithm is illustrated in Figure 5.

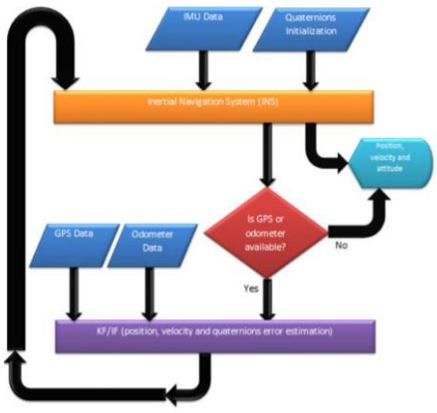

Figure 5. Inertial Navigation System and GPS filtering algorithm [2]

Other sensor fusion methods were also reviewed, with a focus on further improving the accuracy and performance of inexpensive GPS receivers in real-time. The second method reviewed incorporates stereo-vision to estimate 'frame-to-frame motion' [3] to allow for the collection of visual odometry. This proposed method also uses inertial measurements for situations where the visual-odometry method fails and said motion measurements are then fused with GPS using a standard Kalman Filter. The efficacy of this proposed method can be seen pictured in Figure 6 wherein the fused method is considerably more accurate than the raw odometry.

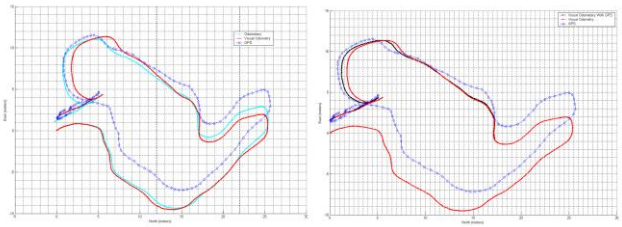

Figure 6. (left) raw odometry state estimation versus (right) fused visual odometry and GPS [3]

This proposed method of sensor fusion and state estimation ultimately resulted in a significant reduction in percent error throughout the routes tested empirically. Without sensor fusion, percent error was on the order of 1.3% to 31.0%, while GPS and visual odometry fusion by way of Kalman Filter resulted in the range of percent error decreasing to the order of 0.3% to 2.0% for the same routes travelled [3].

Another reviewed method consisted of the self-localization of an underwater Remotely Operated Vehicle (ROV) that utilized optical, pressure, and inertial sensors alongside an EKF for robust and precise localization underwater [4]. This method also utilizes a variation of EKF multi-sensor fusion, wherein localization drift over time is minimized. The proposed state-estimation algorithm operates using a velocity correction model, utilizing Euler angle information provided by the on-board IMU to correct visual-odometry velocity estimates. Simulations in the ROS-based Gazebo software were then used to assess the validity of the proposed fusion technique. The general structure of the underwater localization and fusion method is illustrated in Figure 7.

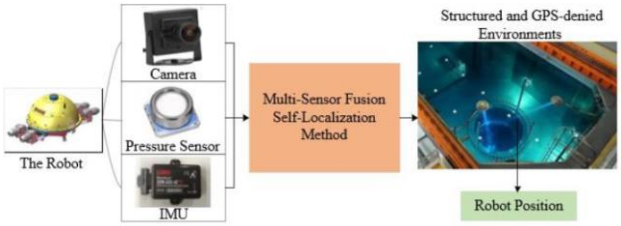

Figure 7. Diagram of ROV sensing suite and fusion algorithm [4]

The proposed methods reviewed primarily serve to validate the sensor fusion philosophy that will be developed in this paper. As can be seen in the reviewed methods, common sensor themes and strategies include employing visual or classical odometry and inertial measurements to correct consumer-grade GPS data outages and drift during state estimation. As such, these methods will be implemented using a Bayesian EKF to

accurately localize the differential drive robot pictured in Figure 1.

III. Methods and Materials

Although the robot platform includes several LiDAR sensors and a 360° camera, the sensor fusion method discussed was limited to using the Consumer-grade GPS, GPS RTK, IMU, and Wheel Encoders for position estimation. The data of interest from the consumer GPS and GPS RTK were the global coordinates of the robot. The use of an Extended Kalman Filter (EKF) was investigated to fuse these accessible sensors with the consumer-grade GPS to determine the position of the robot at least as accurately as GPS RTK, while also attempting to estimate the position of the robot indoors when GPS signal is lost. It was determined that the motion and measurement models were nonlinear, hence the use of an EKF will provide accurate and real-time estimation performance.

The dataset contains recorded data from the robot sensors when driven in 21 different paths around the campus. The most recent path from "2013-04-05", which included several periods of GPS outage when the robot travelled inside buildings, was used to investigate and tune the EKF algorithm. The remaining 20 test paths are used as test cases for determining the average performance of the EKF.

During the modelling of sensor data and robot kinematics, additive gaussian noise was assumed on sensor data. Additionally, although the uncertainty in GPS measurements may change over time due to signal divergence when navigating near dense buildings, constant GPS uncertainty for the duration of a path was also assumed. These are assumptions needed to utilize an EKF, which requires Gaussian Noise on states and measurements.

A. Solution Architecture

The EKF solution proposed involves pre-processing relevant sensor data, implementing the EKF for sensor fusion, and computing performance of the EKF system. The dataset is available in the form of several CSV files, where each piece of sensor data has a Unix timestamp of when the data was received. The rate of update of each sensor is shown in Table 1. Of note is that the EKF receives sensor updates at different rates. This implies the use of time-syncing measurements, whereby the EKF incorporates sensor data (either through Motion or Measurement model updates) only if available.

Table 1: Sensor update rates

Sensor	Update Frequency [Hz]
GPS RTK	2
GPS	6
Wheel Velocity	37
IMU	47

Figure 8 outlines this process; at each timestep the EKF algorithm will perform a state prediction using the inputs and motion model, and only perform a state correction if the associated measurement is available. For example, not all timesteps will have an associated GPS measurement available for use as state correction.

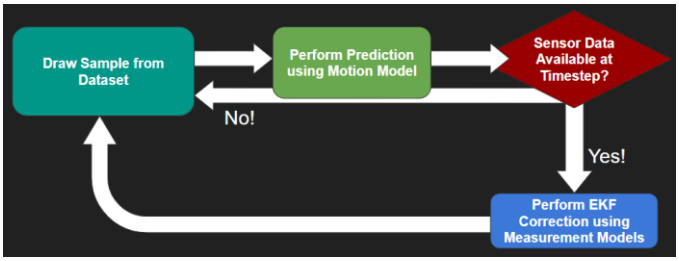

Figure 8: EKF Prediction and Correction Mechanism

B. Characterizing GPS Data

The first sensor to characterize is the GPS. Both the consumer-grade GPS and GPS-RTK provide robot positions in the form of (timestamp, latitude, longitude). As the GPS coordinate system (latitude, longitude) is spherical, these coordinates were linearized about the center of the University of Michigan campus using the method from [1], resulting in a global cartesian coordinate system of (x,y,θ) where x is the meters in North direction, y in the East, and θ the heading angle with respect to North.

Figure 9 shows a comparison between the Ground Truth, GPS RTK measurements, and Consumer-Grade GPS measurements for the 2013-04-05 robot path. Areas of GPS outage in GPS RTK and GPS plots correspond to locations where the robot path crosses through the buildings shown in Figure 3. The difference in accuracy between GPS and GPS-RTK data is clearly visible. While GPS-RTK provides positional accuracy of up to 10 cm, Consumer-grade GPS like those in cellular devices have an accuracy of approximately 10 meters [5]. Additionally, it was found the Consumer-grade GPS periodically provided false-positive positions outside the University Campus area, hence preprocessing was performed on the Consumer-grade GPS data to constrain readings to within the campus boundaries.

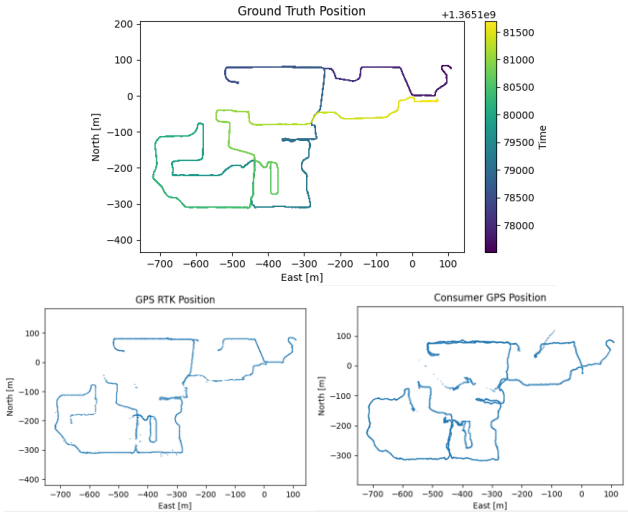

Figure 9: Ground Truth vs GPS Data Comparison

C. Characterizing IMU

The IMU used provides linear accelerations in the local x,y,z directions, angular velocities in the local x,y,z frame, magnetometer readings in the local x,y,z frame, and yaw with respect to North, which is internally estimated by the IMU using its own filtering method [1]. The data of interest was the linear accelerations in the x and y directions, the angular velocity

along the z axis, and the internally estimated Euler angle about the z axis. This raw data, along with an applied rolling average with window size of 1000 data points, can be seen plotted below in Figure 10 for the 2013-04-05 path.

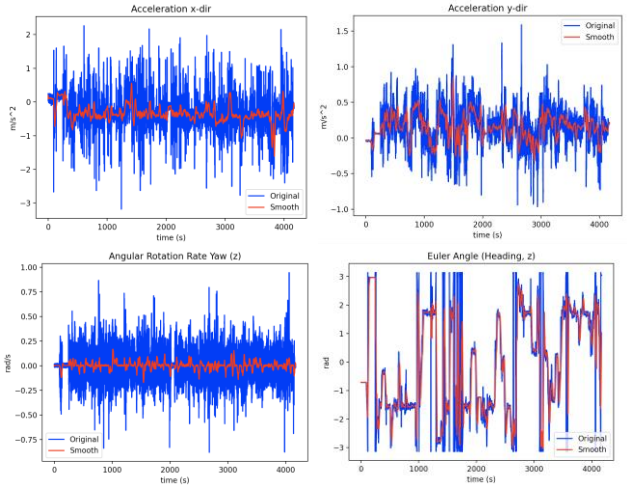

Figure 10: Raw (blue) and averaged (red) IMU data.

As seen in Figure 10, the IMU data is significantly noisy with the rolling average resulting in a noticeable smoothing effect. The noise profiles of the IMU data were obtained by subtracting the rolling average (mean signal) from the raw data at each sample point and can be seen plotted in the histograms in Figure 11 below.

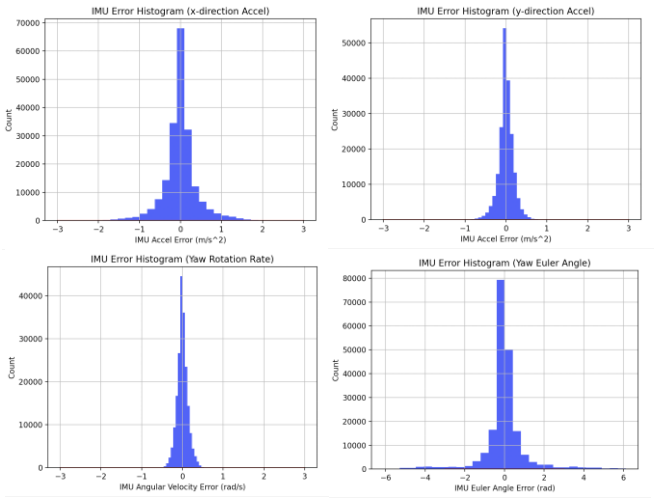

Figure 11: IMU Noise histograms for relevant data.

As seen in Figure 11, all data of interest provided by the IMU fits to a normal distribution (thus deeming the original additive Gaussian noise assumption correct). The standard deviations of the x-acceleration, y-acceleration, yaw rotation rate, and yaw Euler angle are 0.42 m/s^2, 0.18 m/s^2, 0.13 rad/s, and 1.10 rad respectively.

It is important to note that when characterizing the IMU, significant drift in the displacement estimate was observed. This can be seen in Figure 12, wherein velocity and position are obtained by integrating the raw accelerations.

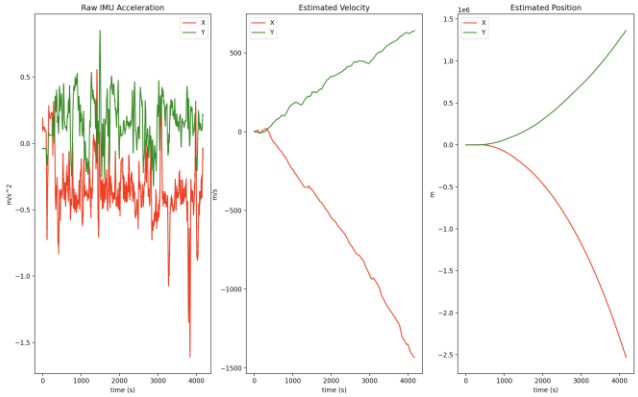

Figure 12: Estimating velocity and position from IMU acceleration.

As seen in Figure 12, there is significant drift in the estimated velocity and position, with both values reaching highly unreasonable values by the end of the sampling period. This drift behaviour was considered when incorporating IMU data into the EKF.

D. Characterizing Wheel Encoders

The wheel encoder data from the robot provided the velocities of the left and right wheels, both of which were of interest. This raw data, along with an applied rolling average with window size of 1000 data points, can be seen plotted below in Figure 13 for the robot while driving the 2013-04-05 path.

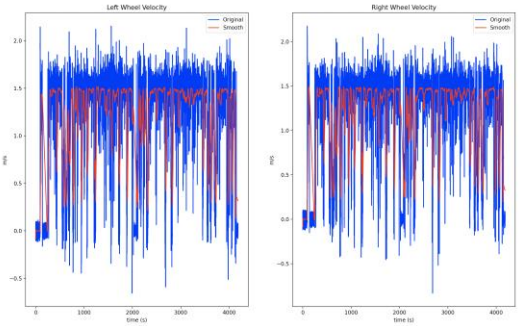

Figure 13: Raw (blue) and averaged (red) encoder data.

As seen in Figure 13, the encoder wheel velocity data is slightly noisy, with major spikes most likely being attributed to the robot slowing down to turn. The noise profiles of the left and right wheel encoders were obtained by subtracting the rolling average from the raw data at each sample point and can be seen plotted in the histograms in Figure 14 below.

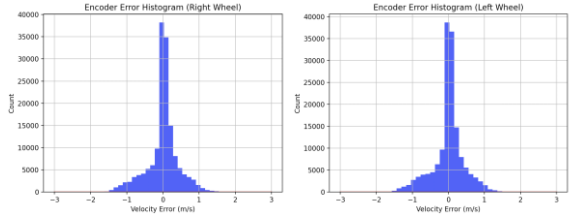

Figure 14: Wheel encoder noise histograms for wheel velocities.

As seen in Figure 14, the data noise provided by the wheel encoders approximately fits a normal distribution, like the IMU data. The standard deviation of both the right and left wheel velocity noise is 0.43 m/s.

E. Motion Model

The raw data from the sensors are measured in the IMU's reference frame, which closely aligns with the robot's reference frame. To effectively predict and correct the state vectors, the IMU data must be transformed to the global frame. Using Figure 15, the coordinate frame conversion between the global frame and the IMU frame can be derived. A similar process is used to transform wheel encoder data to the global frame.

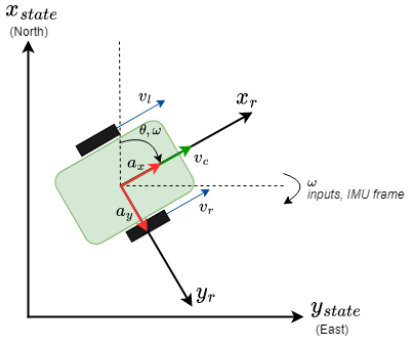

Figure 15: The robot's frame of reference within the global frame.

The state vector of the system in the global frame is:

$$x_k = [x \quad y \quad \dot{x} \quad \dot{y} \quad \theta \quad \omega]^T \tag{1}$$

where x, y are the robot's position in the global frame, $\dot{x}, \dot{y}$ are the respective velocities, and θ and ω are the heading angle and angular velocities respectively.

As there are 2 sensors (wheel encoders and IMU) that can be used as inputs to the EKF motion model, 2 separate motion models were created: an IMU-based and a wheel-velocity-based motion model. These nonlinear motion models are used independently, depending on the configuration of the EKF to generate a state prediction. Comparing these EKF implementations enables the characterization of the effect of IMU drift observed earlier.

1) IMU Based Motion Model

The IMU-based motion model uses the IMU's estimated heading θ, angular velocity ω and the linear accelerations a_x and a_y as control inputs u_{k-1}. From this, acceleration of the robot in the global frame can be calculated using an inverse Rotation Matrix as:

$$a_{gx} = a_x \cos(-\theta) - a_y \sin(-\theta) \tag{2}$$

$$a_{gy} = a_x \sin(-\theta) - a_y + \cos(-\theta) \tag{3}$$

The IMU motion model (in the global frame) using forward kinematics is then:

$$\hat{x}_{k|k-1} = f(\vec{x}_{k-1}, u_{k-1}) = \begin{bmatrix} x_{k-1} + \dot{x}_k \Delta t + 0.5 a_{gx} \Delta t^2 \\ y_{k-1} + \dot{y}_k \Delta t + 0.5 a_{gy} \Delta t^2 \\ \dot{x}_{k-1} + a_{gx} \Delta t \\ \dot{y}_{k-1} + a_{gy} \Delta t \\ \theta_k \\ \omega_k \end{bmatrix} \tag{4}$$

where Δt is the sampling time of the EKF.

2) Wheel Based Motion Model

The Wheel-based motion model uses the IMU's estimated heading θ, angular velocity ω and the wheel velocities of the left and right wheel, v_l and v_r respectively, as control inputs u_{k-1}. The robot's forward velocity v_c is calculated from the individual velocities of the left and right wheel (v_l and v_r, respectively) with the following equation:

$$v_c = 0.5\,(v_r + v_l) \tag{5}$$

The motion model of the wheel in the global frame is:

$$\hat{x}_{k|k-1} = f(\vec{x}_{k-1}, u_{k-1}) = \begin{bmatrix} x_{k-1} + v_c \cos\theta\, \Delta t \\ y_{k-1} + v_c \sin\theta\, \Delta t \\ v_c \cos\theta \\ v_c \sin\theta \\ \theta_k \\ \omega_k \end{bmatrix} \tag{6}$$

Note that the states representing $\dot{x}$ and $\dot{y}$ are redundant but are kept for ease of implementation and consistency in the size of matrices. Through experimentation, it was found that θ and ω from the IMU are more accurate and closely match the ground truth, than those given by wheel velocities.

F. Measurement Model

The use of wheel velocity and GPS measurements are used as additional sensors to refine the predicted position. For each of these, a separate measurement model was created. These corrections are then fused independently into the EKF estimate, depending on the sensor data available at a given timestep.

1) Wheel Measurement Model

The wheel measurement model is active only when using an IMU-based motion model and produces a prediction of the expected left and right wheel velocities based on the predicted state. From Figure 15, this can be formulated as:

$$\hat{z}_{k|k-1} = h(\hat{x}_{k|k-1}) = \begin{bmatrix} \widehat{v_l} \\ \widehat{v_r} \end{bmatrix} = \begin{bmatrix} \sqrt{\dot{x}_k{}^2 + \dot{y}_k{}^2} - \dfrac{T\omega_k}{2} \\ \sqrt{\dot{x}_k{}^2 + \dot{y}_k{}^2} + \dfrac{T\omega_k}{2} \end{bmatrix} \tag{7}$$

where the width of the robot base is T= 0.562 m from the SolidWorks CAD model of the robot in [1].

2) GPS Measurement Model

The GPS measurement model produces a prediction of the expected GPS measurements based on the predicted state. As the GPS provides measurements of the global position of the robot, expected measurements are given directly by:

$$\hat{z}_{k|k-1} = C\hat{x}_{k|k-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \hat{x}_{k|k-1} \tag{8}$$

IV. Results

A. EKF Implementation

The complete EKF algorithm for an IMU-based motion model with corrections from Wheel Velocity and GPS Measurement models is shown below.

$$\hat{x}_{k|k-1} = f_{imu}(x_{k-1}, u_{k-1})$$
$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_{IMU}$$
IF Wheel Measurement available: perform correction
$$\hat{z}_{k|k-1} = h_{wheel}(\hat{x}_{k|k-1})$$
$$K_k = P_{k|k-1} H_k^T \left(H_k P_{k|k-1} H_k^T + R_{wheel}\right)^{-1}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - \hat{z}_{k|k-1})$$
$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$
IF GPS Measurement available: perform correction
$$\hat{z}_{k|k-1} = C\hat{x}_{k|k-1}$$
$$K_k = P_{k|k-1} C^T \left(C P_{k|k-1} C^T + R_{GPS}\right)^{-1}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\left(z_k - \hat{z}_{k|k-1}\right)$$
$$P_{k|k} = (I - K_k C) P_{k|k-1}$$

The complete EKF algorithm for a Wheel-velocity-based motion model with corrections from the GPS Measurement models is shown below:

$$\hat{x}_{k|k-1} = f_{wheel}(x_{k-1}, u_{k-1})$$
$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_{Wheel}$$
IF GPS Measurement available: perform correction
$$\hat{z}_{k|k-1} = C\hat{x}_{k|k-1}$$
$$K_k = P_{k|k-1} C^T \left(C P_{k|k-1} C^T + R_{GPS}\right)^{-1}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - \hat{z}_{k|k-1})$$
$$P_{k|k} = (I - K_k C) P_{k|k-1}$$

The EKF algorithm was implemented in Python. Jacobians of the motion model (F) and measurement models (H) are computed analytically using the Sympy Python library with respect to the predicted states $\hat{x}_{k|k-1}$, and the predicted date is substituted at each timestep.

Several configuration modes of the EKF were tested, as outlined in Table 2. This was done to determine the efficacy of the wheel-based motion model versus the IMU-based motion model, as well as the estimation performance with and without Consumer-grade GPS measurement corrections. The GPS and GPS RTK Modes in Table 2 are not estimated with the EKF, but act as a baseline for performance comparison.

Table 2: EKF Modes

EKF Mode	Motion Model Input	Measurement Models
GPS-RTK-Only	N/A	N/A
GPS-Only	N/A	N/A
Wheels-Only	Wheel-velocity	None
Wheels-with-GPS	Wheel-velocity	Consumer GPS
IMU-with-GPS	IMU Acceleration	Consumer GPS
IMU-with-Wheels-and-GPS	IMU Acceleration	Wheel-velocity and Consumer GPS

B. Tuning R and Q matrices

Through trial and error, it was found that the wheel velocities are quite accurate in determining forward velocity, but not angular velocity. Complimentary to this, the IMU is adept at determining a heading estimate and angular velocity, but not a linear velocity as observed in Figure 12.

The R (sensor noise) matrices of the EKF system can be seen in equations (9) and (10). These R matrices were determined using empirical data as well as completing experimental tuning. The wheel velocity R values were set to be very low to ensure this data was given higher priority in the EKF compared to the less reliable GPS data, especially when signals were dropped indoors. The GPS R values were chosen based on the accuracies of consumer-grade GPS of 10m [5].

$$R_{wheel} = \begin{bmatrix} R_{left} & 0 \\ 0 & R_{right} \end{bmatrix} = \begin{bmatrix} 0.00001 & 0 \\ 0 & 0.00001 \end{bmatrix} \quad (9)$$

$$R_{GPS} = \begin{bmatrix} R_x & 0 \\ 0 & R_y \end{bmatrix} = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix} \quad (10)$$

Two unique Q (process noise) matrices were developed for the EKF implementation, one for the IMU-based model and the other for the wheel-velocity-based model. These Q matrices can be seen in equations (12) and (13).

$$Q_{general} = \begin{bmatrix} Q_x & 0 & 0 & 0 & 0 & 0 \\ 0 & Q_y & 0 & 0 & 0 & 0 \\ 0 & 0 & Q_{\dot{x}} & 0 & 0 & 0 \\ 0 & 0 & 0 & Q_{\dot{y}} & 0 & 0 \\ 0 & 0 & 0 & 0 & Q_\theta & 0 \\ 0 & 0 & 0 & 0 & 0 & Q_\omega \end{bmatrix} \quad (11)$$

$$Q_{IMU} = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.01 \end{bmatrix} \quad (12)$$

$$Q_{wheel} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.001 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.01 \end{bmatrix} \quad (13)$$

The Q matrices can be represented as diagonal matrices as additive Gaussian noise is assumed which is uncorrelated between states. Similar to the R matrices, the Q values were determined using both empirical sensor data as well as experimental tuning. For example, the process noise values corresponding to x and y velocity in Q_{IMU} were both set to a high value of 10000 m/s due to the significant IMU drift observed in Figure 12. On the other hand, the angular rotation rate process noise was set to a low 0.01 rad/s due to its low error variance of 0.13 rad/s from Figure 11, thus making it a more reliable measurement.

C. Localization Results

Figure 16 to Figure 18 showcase position estimates by the EKF compared to the ground truth for the 2013-04-05 path used

for tuning EKF parameters. In each of these plots, the EKF's estimated position at each timestep is shown in blue, while the ground truth position is shown in red. These plots were generated by overlaying a list of estimated and ground truth positions onto the satellite view of Google Earth.

Both EKF formulations in Figure 16 and Figure 17 use Consumer-grade GPS measurements for correction, so the localization performance when the robot is outdoors appears quite similar. However, the Wheel-velocity-based motion model shown in Figure 17 is noticeably smoother along straight paths as well as abrupt turns.

Of further interest is the estimation performance when the robot travels through or near dense buildings and loses GPS signal. During these times, from Figure 16 and Figure 17, it appears that the IMU-acceleration-based motion model has large amounts of drift during GPS outage, while the position estimate using the Wheel-velocity-based motion model undergoes a more reasonable amount of drift.

Figure 18 shows the same path, estimated using only wheel-velocity without any correction updates from the GPS. It appears that estimated position has the same shape as the ground truth path but is slightly rotated due to an error in the heading angle. This indicates that the wheel velocity measurements by encoders on the robot are quite accurate but require an additional sensor for heading drift correction.

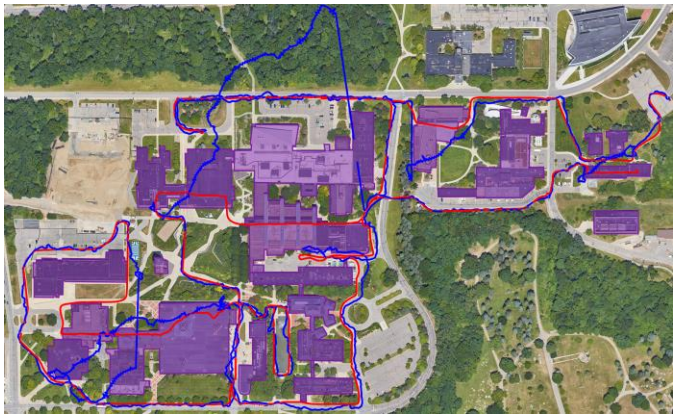

Figure 16: IMU with Wheels and GPS. Blue: Estimated Position. Red: Ground Truth Position

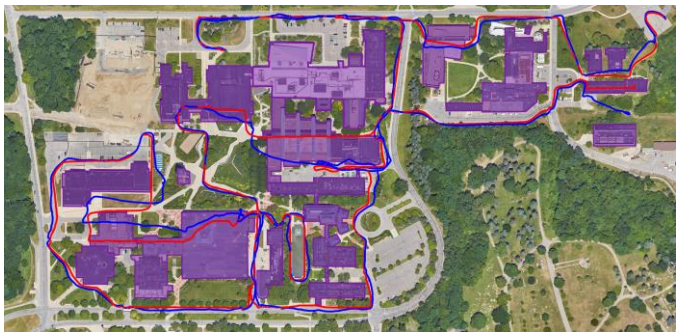

Figure 17: Wheels with GPS. Blue: Estimated Position. Red: Ground Truth Position

The x and y global position states from the estimate in Figure 17 are plotted over time in Figure 19, with respect to the ground truth. Figure 20 shows a section of time where the robot travelled through a building and GPS signal was lost. The red dashed lines in Figure 20 represent uncertainty bounds of the estimated state, based on the corresponding 1st and 2nd elements of the State covariance matrix $P_{k|k}$.

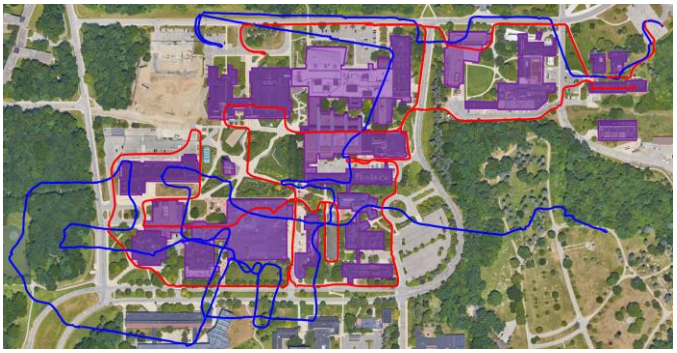

Figure 18: Wheels Only. Blue: Estimated Position. Red: Ground Truth Position

During the time of GPS outage in Figure 20, it is clear that the EKF-predicted estimate became increasingly uncertain until the EKF received a GPS signal near the end of the plot in Figure 20. Interestingly, although the uncertainty in the state grew, the uncertainty bounds on the state were not large enough to also contain the true x coordinate after about 550 seconds. This indicates that the EKF estimate diverged from the ground truth, until a GPS signal was regained.

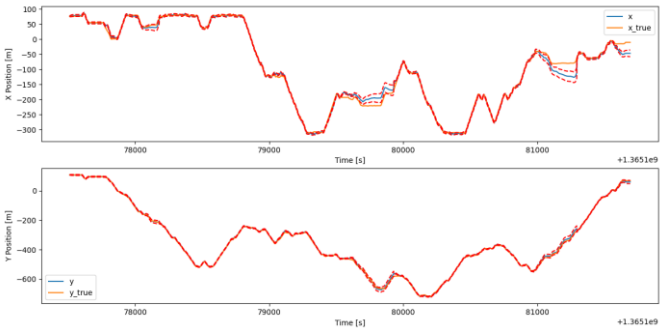

Figure 19: Estimated position from Wheel-based and GPS-corrected EKF vs Ground Truth

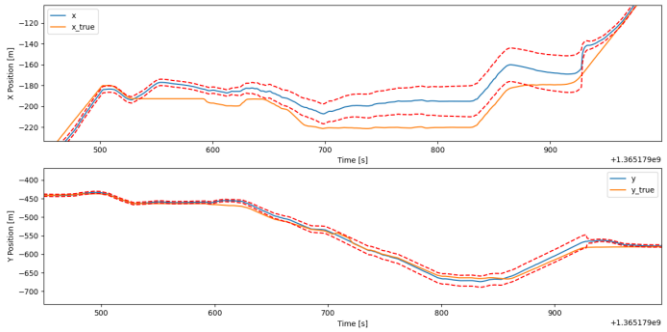

Figure 20: Zoomed in view of Figure 19. Red dashed lines represent uncertainty bounds using corresponding elements of State covariance matrix (P).

To quantify the error of the various EKF modes tested, the Euclidean distance between the ground truth position and EKF estimated position was calculated at each timestep. Figure 21 shows this error in Euclidean distance over time for the 2013-04-05 path when the EKF is in Wheel-based-GPS-corrected

mode. The peaks in this error graph correspond to sections of loss of GPS signal inside buildings, where only wheel odometry is utilized.

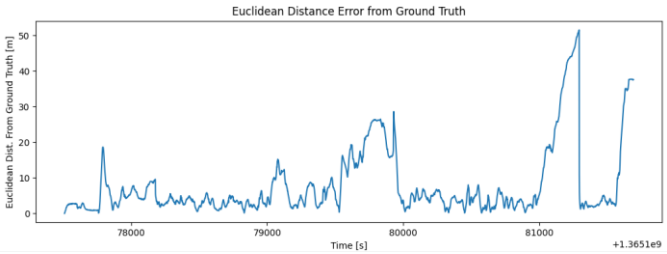

Figure 21: Euclidean Distance Error over Time for Wheel-based, GPS-corrected EKF.

A histogram of this error is shown in Figure 22, and appears to follow a normal distribution. Note that since the Euclidean distance error metric is an absolute-valued error, the Normal distribution does not appear symmetric in Figure 22. As the error is normally distributed, the error over time can be characterized through the mean and standard deviation of the error over time.

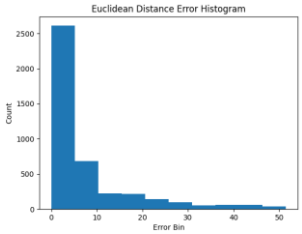

Figure 22: Histogram of Euclidean error in Figure 21.

The resulting mean and standard deviation of error for the 2013-04-05 training path is summarized below for each EKF mode in Table 3.

Table 3: Error Metrics across 2013-04-05 path

EKF Mode	Mean Error [m]	Std. Dev Error [m]
GPS-RTK-Only	18.22	38.57
GPS-Only	27.60	55.10
Wheels-Only	105.99	53.59
Wheels-with-GPS	7.98	9.80
IMU-with-GPS	803.29	2430.13
IMU-with-Wheels-and-GPS	20.05	46.31

As shown in Table 3, the best performing EKF mode (lowest mean and standard deviation in error) is the "Wheels with GPS" mode of operation. The Wheels-with-GPS mode significantly outperforms IMU-with-Wheels-and-GPS; this is due to the greater inaccuracy of the IMU as seen when using only the IMU-with-GPS in the EKF (mean error of 803.29m). Wheels-with-GPS also outperforms GPS-RTK-Only, GPS-Only, and Wheels-Only; This indicates that the wheel velocity data fused with the lower quality consumer-grade GPS via the EKF produces more accurate positional estimations than the high-quality GPS-RTK alone. This is due to the ability to estimate the robot's position when indoors, leading to an overall lower error.

Figure 23 shows position estimates by the EKF compared to the ground truth for the 2012-05-11 test path used for validating the EKF algorithm. The results are like that of the 2013-04-05 training path. As shown in Figure 24, the EKF is able to accurately estimate the position of the robot even when navigating through buildings without a GPS signal.

Figure 23: Wheels-with-GPS EKF mode on 2012-05-11 test path. Blue: Estimated Position. Red: Ground Truth Position

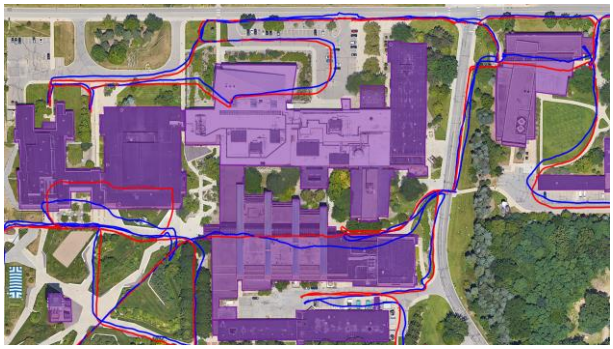

Figure 24: Zoomed in view of the top of Figure 23. Sections where the red Ground Truth path intersects with purple buildings are indoor paths.

A similar trend in positional accuracy was observed for the remaining test paths, as shown in **Error! Reference source not found.** and Table 5. The average/standard deviation in error was calculated across all test paths' mean/standard deviation Euclidean error. As the lowest average mean/standard deviations of error are of the Wheels-with-GPS EKF Mode, it can then be concluded that the EKF investigated in this report was successful in outperforming the highly accurate GPS RTK sensor. This again can be attributed to the wheel velocities compensating for the loss of GPS signals within buildings.

Table 4: Avg. Error Metrics across Entire Dataset

EKF Mode	Avg. of Mean Error [m]	Avg. of Std. Dev Error [m]
GPS-RTK-Only	12.37	25.75
GPS-Only	15.91	30.45
Wheels-Only	116.68	53.63
Wheels-with-GPS	7.88	10.84
IMU-with-GPS	341.96	1140.37
IMU-with-Wheels-and-GPS	15.26	30.48

Table 5: Std. Dev. Error Metrics across Entire dataset

EKF Mode	Std. Dev. of Mean Error [m]	Std. Dev. of Std Dev Error [m]
GPS-RTK-Only	3.83	8.81
GPS-Only	9.44	19.91
Wheels-Only	29.31	17.05
Wheels-with-GPS	2.92	6.57
IMU-with-GPS	602.48	1876.32
IMU-with-Wheels-and-GPS	9.27	20.25

D. Limitations

The benefit of performing sensor fusion via an EKF for localization of the mobile robot is that this method does not rely on the computationally intensive SLAM post-processing method. The EKF can run on a real-time system. Additionally, this sensor fusion can attempt a more accurate indoor positional estimation than any of the sensors alone.

However, as the motion and measurement models are non-linear, the linearization through Jacobian matrices in an EKF removes the guarantee of optimality of state estimation provided by the standard Kalman Filter algorithm. Another limitation of the current EKF formulation is that by using the current sensor suite, there is no way to correct the robot's heading state. It is currently relying solely on the IMU's estimate of heading θ, and angular velocity ω. A low-quality IMU can thus cause significant error during times of GPS outage. One way to improve the estimated angular heading is by using the Fiber Optic Gyro (FOG) available on the robot, which is a much more expensive and higher quality sensor than MEMS based IMU Gyroscopes [5].

Although the Consumer-grade GPS and other sensors have a fast update rate as outlined in Table 1, the EKF was run at a rate of 1 Hz due to computational power and resource constraints. Each path in the dataset is approximately a 2 hour and 30-minute session, and so running the EKF at a higher rate for the entire dataset will inherently take much longer. The most computationally intensive aspect of the EKF algorithm is the calculation of the F Jacobian matrix which is recomputed at every timestep. Increasing the efficiency of this computation and using a GPU for the matrix multiplications part of the EKF can increase the rate at which the EKF can run at, which may lead to higher accuracy results in times of GPS outage.

Lastly, it was initially assumed that the GPS uncertainty remains constant. However, the GPS data in Figure 9 indicates that there is varying uncertainty in the Consumer-grade GPS measurements due to signal divergence when navigating near or through dense buildings. To account for this effect, the R_{GPS} noise matrix could be formulated as time-dependant, with increasing uncertainty based on the proximity to campus buildings as calculated from the 3D LiDAR sensor.

V. Conclusions and Recommendations

In conclusion, various EKF's were successfully implemented and analyzed to fuse GPS, IMU, and Wheel Encoder sensor data to accurately track the position of the Segway Mobile robot throughout the university campus, both in and around campus buildings. Through error quantification, it was determined that the best performing EKF mode was where wheel velocity data from the robot encoders are fused with the consumer-grade GPS. This EKF mode outperformed the highly accurate GPS RTK model, with an average mean error of 7.88m across all data sets. Ultimately, incorporating the IMU data into the state estimate as well hindered performance due to its drifting behaviour. Based on these results, the overall purpose of this paper was achieved. However, future improvements such as correcting heading angle estimates, increasing efficiency of Jacobian Matrix computation, and GPS uncertainty modelling are still recommended to produce even better performing EKF models.

VI. References

[1] N. Carlevaris-Bianco, "University of Michigan North Campus Long-Term Vision and Lidar Dataset," *International Journal of Robotics Research*, pp. 1023-1035, 2016.

[2] K. Saadeddin, "Estimating Vehicle State by GPS/IMU Fusion with Vehicle Dynamics," *Journal of Intelligent & Robotic Systems*, no. 74, pp. 147-172, 2013.

[3] M. Agrawal, "Real-time Localization in Outdoor Environments using Stereo Vision and Inexpensive GPS," in *The 18th International Conference on Pattern Recognition (ICPR'06)*, Menlo Park, 2006.

[4] H. Xing, "A Multi-Sensor Fusion Self-Localization System of a Miniature Underwater Robot in Structured and GPS-Denied Environments," *IEEE Sensors Journal*, vol. 21, no. 23, pp. 27136-27146, 2021.

[5] A. Ahmed, "Multi-Sensor Fusion for Navigation of Ground Vehicles," Carleton University, Ottawa, 2021.

VII. Appendix

Here, we outline the Python files created for working with the dataset, and implementing the Extended Kalman Filter.

- EKF.py:
  - This file contains the implementation of the Extended Kalman Filter, computing Jacobian Matrices, and time-synchronizing the sensor data
  - The parameters at the top of this file determine the mode the EKF runs in i.e. using the wheel-velocity-based vs IMU-acceleration-based motion model.
- read_imu.py, read_wheels.py, read_gps.py, read_ground_truth.py, read_FOG.py
  - These files are used to read and pre-process the various sensors included in the dataset.

- IMU_processing.py
    - This file is used to determine noise distribution and noise characteristics of the IMU sensor data
- wheels_processing.py
    - This file is used to determine noise distribution and noise characteristics of the Wheel Encoder sensor data
- utils.py
    - This file converts between GPS and linearized coordinate frames, generates plots, exports visualizations to Google Earth, as well as calculates Euclidean Error metrics.