Probabilistic Filter Comparison on *nuScenes by Aptiv* Dataset*

Arnav Sharma[1], Benjamin Dion[2], Eric Yu[3], and Saurabh Sinha[4]

***Abstract*— In the current autonomous vehicle research realm, simultaneous localization within Simultaneous Localization and Mapping (SLAM) algorithms remain a key topic of study. The current industry standard is to localize the vehicle by using Global Positioning System (GPS) and Inertial Measurement Unit (IMU) sensor data. The work demonstrated in this paper aims to address intermittent losses of GPS satellite communication. By comparing Particle Filter (PF) and Invariant Extended Kalman Filter (IEKF) results, the paper investigates opportunities to reduce the localization covariance and gain robustness in state estimation. Aptiv's nuScenes publicly available dataset is used to compare the results against benchmarks provided in the nuScenes benchmark suite.**

I. INTRODUCTION

Localization and mapping methods are key to the development of reliable autonomous vehicles and advanced driver assist technologies. Advancing reliable pose estimation and mapping capabilities have been dependent upon a convergence of several technological improvements. Computing speeds and efficiency have continued to increase as costs for equivalent computing speeds have decreased. Decreased costs and improved accuracy are increasing proprioceptive sensor proliferation within modern automobiles. Such sensors as Inertial Measurement Unit (IMU), Global Positioning System (GPS), wheel speed, and steering angle sensors are now widely incorporated, and increasingly common are camera technologies capable of performing visual odometry. Light Detection and Ranging (LiDAR) systems are now small enough to fit onto an experimental compact car [1]. Lastly, advancements in robotic research are utilizing the improved computing power and sensor data to enable pose estimation [4], [5], [10], [11].

To be successful in real-world applications, autonomous vehicles must be capable of robustly navigating dynamically changing environments and situations in which sensor data quality may not be ideal. Dense urban environments and underground routes may obscure GPS data; weather may lead to obscured cameras or wheel slip, resulting in abnormal vehicle data; and construction or unexpected obstacles may interfere with typical vehicle route planning [6], [11]. Additionally, sensor signal bias and noise can increase system variance and may change over time [7]. The safety of passengers and bystanders depends upon consistent, reliable localization, so understanding and managing error propagation is critical to a successful autonomous vehicle. To assess the capability and limitations for pose estimation, we compare the Invariant Extended Kalman Filter (IEKF) and Particle Filter (PF) to the nuScenes vehicle data set.

A. Related Work

The foundation for the project is guided by Barrau and Bonnabel's case study using IEKF with noisy GPS and idealized odometry [5]. Applying IEKF to nuScenes data and comparing with PF results is meant to establish the framework for assessing pose estimation quality, which can then be used to assess localization in adverse conditions and using varying sensor signal availability.

B. nuScenes Dataset [3]

The nuScenes dataset is an expansive dataset with over a thousand 20-second driving scenes in various locations around Boston and Singapore. The full dataset includes variations in weather, time-of-day, construction zones, and many other scenarios that human drivers are expected to navigate on a daily basis. The data relevant for this project is as follows:

- Controller Area Network (CAN) bus
- Maps data
- Metadata (e.g. scene information, ego pose data)
- Semantic annotations
- Camera images
- LiDAR sweeps

The CAN data is provided at several different frequencies. For example, the ego pose (pseudo-GPS) information - which includes position, rotation quaternion, rotation rate, and velocity - are provided at 50 Hz, while the IMU provides data at 100 Hz. Also, it is important to note that some data is provided in the global frame while other data is provided in the local frame. Appropriate conversions are necessary when employing methods that use data in different frames. Simple quaternion to Euler angle conversion functions can be developed to extract the desired 3D rotation angles.

The data available for each scene is vast and conveniently linked with a universal timestamp, *utime*, available for all data. However, it is very important to note that the *utime* of the data source Electronic Control Units (ECU), and the sensors within them, are not perfectly aligned to the beginning of a scene. Alignment using the *utime* values is necessary to successfully run incremental probabilistic filters.

*Also known as 'where-dyss'

[1]Arnav Sharma is a Master of Automotive Engineering student at University of Michigan, Ann Arbor, MI, USA `arnavsha@umich.edu`

[2]Benjamin Dion is a Master of Automotive Engineering student at University of Michigan, Ann Arbor, MI, USA `bedion@umich.edu`

[3]Eric Yu is a Master of Automotive Engineering student at University of Michigan, Ann Arbor, MI, USA `ericyu@umich.edu`

[4]Saurabh Sinha is a Master of Automotive Engineering student at University of Michigan, Ann Arbor, MI, USA `sinhasau@umich.edu`

The Maps data (provided in .json files) includes multiple layers such as lane dividers, road dividers, walkways, pedestrian crossings, driveable road areas, traffic lights, among others. They are color-coded and can be displayed using the built-in methods in the *NuScenesMap* class. An example of such a map is provided in Figure 1. The maps allow the estimated pose data to be superimposed to validate the performance of the filter against the ground truth.

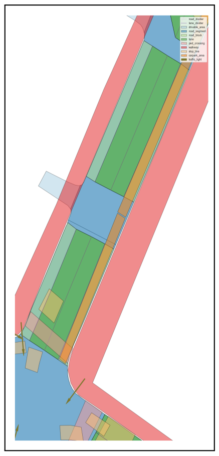


Fig. 1. nuScenes Fancy Map output of Scene-0249 rendering different map layers in Boston Seaport

The Metadata download ($\sim$ 0.5 GB) contains data in .json file format for the scenes. This includes samples, ego pose, annotations and their categories, etc. This file download from the nuScenes website is critical to run the developed filters. The data structure for these files is well documented and described in the nuScenes website through tutorials and README webpages. Parsing through this data is time-consuming, but also crucial to executing this experiment.

The semantic annotations are particularly helpful when using landmarks in the filter. They offer identification, as well as a state description, of the object. For example, upon identifying a pedestrian or vehicle, the annotation further describes if the object is standing, sitting, stopped, moving, etc. The categories of objects used in this project include, but are not limited to, animals, various types of vehicles (normal buses, bendy buses, cars, trucks, construction vehicles), pedestrians (including adults and children), traffic cones, and plastic construction barriers. This data is provided at 2 Hz.

The camera images for each scene can be rendered as a video to visualize the annotations, the route, weather, type of road, etc. The videos of the scenes chosen for this project were analyzed in detail for content such as bumps, cones, and road quality, among many other observations. The analysis is provided in the GitHub repository website.

LiDAR sweeps at 20 Hz can provide vehicle position in relation to the semantic landmarks. The project utilizes this vehicle position as localized-GPS information that was extracted from a LiDAR map-based localization algorithm, as described in [1]. Note: In this paper, the localized-GPS data will be referred to as GPS for simplicity.

For more information regarding this high quality multi-sensor layout and its specifications, please visit `https://www.nuscenes.org/data-collection`.

C. *Our Contribution*

In this paper, two probabilistic filters were developed to compare their performance on the nuScenes dataset. The full working open-source code can be found on our GitHub page [`https://arnavsharma.github.io/where-dyss/`]. The presentation of the project can be found on YouTube [`https://youtu.be/hRwTPL3oU74`]. The setup of each of the filters, the Invariant Extended Kalman Filter and the Particle Filter, are discussed in Section II. Section III elaborates the testing used to validate our filters on one of the scenes. Section IV highlights our findings, and Section V concludes and proposes future work.

II. PROPOSED ALGORITHM

This paper investigates two different types of probabilistic filters for their effectiveness in robot pose estimation. The framework of each algorithm implementation can be described in the following steps:

1) Extract relevant data from the dataset.
2) Process data in either Python or MATLAB to create a user-friendly dataset for MATLAB filters.
3) Conduct filtering in MATLAB.
4) Save data and transfer back to Python for viewing on map.

Each algorithm utilizes a combination of different information provided in the dataset. Ego pose (GPS from LiDAR), IMU data, and semantic landmarks observed by camera and LiDAR are extracted. The use of the proposed algorithms will present challenges if implemented in real-world autonomous vehicles. These challenges will be elaborated in the conclusion of this paper.

A. *Invariant Extended Kalman Filter*

The Invariant Extended Kalman Filter (IEKF) capitalizes on the estimation error being invariant under the action of a matrix Lie Group. It combines the benefits of the Extended Kalman Filters (EKF) with symmetry filters that utilize natural symmetry for nonlinear systems. The main benefit with IEKF is that the gain and covariance equations converge to constant values on a much larger set of trajectories than equilibrium points, which results in a better convergence of the estimation [2]. In this application, the filter is used to estimate vehicle state using IMU and GPS sensors only.

There are two different versions of IEKF - Right IEKF and Left IEKF; Right IEKF is invariant in the body frame while Left IEKF is invariant in the global frame. Left IEKF is used in this application since the GPS readings are left-invariant and in the global frame.

In the nuScenes dataset, as previously discussed, the IMU data frame rate is 100 Hz but GPS is 50 Hz. To account for the faster rate of IMU data, the algorithm predicts twice and corrects once using a method similar to a counter. Figure 2 provides the process flowchart of the filter implementation, where 'C' is a counter.

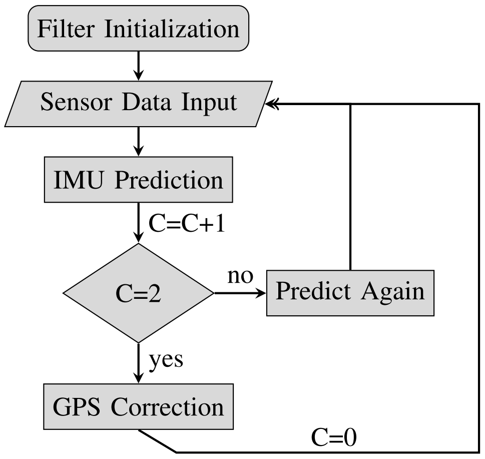


Fig. 2. IEKF Process Flowchart - IMU (100Hz) & GPS (50Hz)

1) Filter Initialization: The prior state estimation used to initialize the filter was the first frame of GPS data. The signals included in the initialization are global x, y, and z positions, rotation angles in the body frame via quaternion rotation vector, and longitudinal velocity (x-axis pointing forward) in the body frame. To maintain the longitudinal velocity value in the global frame, its x and y components are calculated using the global yaw angle.

2) Sensor Data Input: The IMU sensor (viewed from the body frame) as described above provides rotation rate in x, y, and z directions, a 4-dimensional quaternion rotation vector, and linear acceleration values in x, y, and z directions. The quaternion vector can be converted to Euler angles (roll ϕ, pitch θ, and yaw ψ). The conversion equations are provided in the Appendix, as well as the repository. The GPS, interestingly, provides quite a few data points from the body frame, as well as the global frame. In the global frame, a 3-dimensional Cartesian position vector is provided. The rotation rates in the three-axis body frame from GPS sensors are also provided.

Using the GPS and IMU inputs, the state estimation, X, is a 5-by-5 matrix as shown below, where R_k is a 3-by-3 rotation matrix around the z-axis (Equation (1)), v_k is a 3-dimensional linear velocity vector, and p_k is a 3-dimensional Cartesian position vector.

$$X = \begin{bmatrix} R_k & v_k & p_k \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \in SE_2(3)$$

$$R_k = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

3) IMU Prediction: Using the previous time-step state estimation X_{k-1} and the latest IMU input, the algorithm predicts the next state, $\bar{X}_k$. Assuming a zero-order hold on the incoming IMU measurements between times t_k and t_{k+1}, the rotation matrix R_{k+1}, the velocity vector, v_{k+1}, and the pose vector p_{k+1} can be calculated using Equations (2), (3), and (4), respectively. Here, $\bar{\omega}_k \triangleq \bar{\omega}_k$ - $\bar{b}_t^g$ and $\bar{a}_k \triangleq \bar{a}_k$ - $\bar{b}_t^a$ are the "bias-corrected" inputs, where g is gyroscope and a is accelerometer. In addition, the formulas for $\Gamma_0, \Gamma_1, \Gamma_2$ for the equations are provided in the Appendix.

$$R_{k+1} = R_k \Gamma_0(\bar{\omega}_k \Delta t) \tag{2}$$

$$v_{k+1} = v_k \Gamma_1(\bar{\omega}_k \Delta t)\bar{a}_k \Delta t + g\Delta t \tag{3}$$

$$p_{k+1} = p_k + v_k \Delta t + R_k \Gamma_2(\bar{\omega}_k \Delta t)\bar{a}_k \Delta t^2 + \frac{1}{2} g \Delta t^2 \tag{4}$$

$\bar{\Sigma}_{t_k}$ and Σ_{t_k} are 9-by-9 matrices with the variance values along the diagonal. The diagonal values correspond to the variances (σ^2) of roll, pitch, yaw; velocity in x, y, and z directions; and pose with the x, y, and z coordinates. To calculate $\bar{\Sigma}_{t_k}$, A_t^l matrix must be constructed using the gyroscope and accelerometer (IMU) inputs, which are then used to find the state transition matrix, Φ_t^l. To construct A_t^l, the skew operator ($\wedge$) was applied to the input vectors $\tilde{\omega}_t$ and $\tilde{a}_t$ to create skew-symmetric matrix using Equation (25) in the Appendix. See below for A_t^l matrix and refer to Equation 5 to calculate the Φ_t^lx. Using Φ_t^l, and the additive sensor noise Q, $\bar{\Sigma}_{t_k}$ can be updated using Equation (6). The structure of Q is also provided in the Invariant EKF section of the Appendix. Due to unavailability of bias information in the nuScenes dataset, the team assumed a bias of 0 in this algorithm implementation.

As previously discussed, the prediction step was performed twice before each correction, due to the difference in the data acquisition rate.

$$A_t^l = \begin{bmatrix} -\hat{\tilde{\omega}_t} & 0 & 0 \\ -\hat{\tilde{a}_t} & -\hat{\tilde{\omega}_t} & 0 \\ 0 & I & -\hat{\tilde{\omega}_t} \end{bmatrix}$$

$$\Phi_t^l = \exp(A_t^l \Delta t) \tag{5}$$

$$\bar{\Sigma}_{t_k} = \Phi_t^l \Sigma_{t_{k-1}} {\Phi_t^l}^\top + Q \tag{6}$$

4) GPS Correction: GPS updates are used in this IEKF algorithm implementation to correct the prediction. The GPS data was converted into a 5-dimensional vector, Y_{t_k}, as described below in Equation (7). b in Equation (8) is a 5-dimensional vector to define the desired innovation.

$$Y_{t_k} = [GPS_x, GPS_y, GPS_z, 0, 1]^\top \tag{7}$$

$$b = [0, 0, 0, 0, 1]^\top \tag{8}$$

Next, the Kalman Gain, L_{t_k}, was calculated using the known Y_{t_k}, b, $\bar{X}_{t_k}$, $\bar{\Sigma}_{t_k}^l$, the Jacobian H provided in Equation (9), along with the Equations (10), (11), and (12). The Jacobian, H, was defined such that $H\xi = \xi^\wedge b$.

$$H = \begin{bmatrix} 0_{3\text{x}3} & 0_{3\text{x}3} & I_{3\text{x}3} \end{bmatrix} \tag{9}$$

When calculating $\bar{N}_k$, V_k is the noise of the GPS reading in x, y, and z directions. For this application, V_k was a diagonal covariance matrix padded with two zeros for matrix dimensions. Due to the GPS reading being three dimensional, the top-left 3x3 rotation matrix, with a block diagonal of 0_{2x2} added on, could be substituted into $\bar{X}_{t_k}$ since velocity was not being calculated here. Knowing this, Equation (10) was used to calculated $\bar{N}_k$, with a resultant 5-by-5 matrix. $\bar{N}_k$ was reduced to the top-left 3-by-3 matrix relating to rotation, as the rest of the entries are 0.

$$\bar{N}_k = \bar{X}_{t_k}^{-1} \mathrm{Cov}[V_k] \bar{X}_{t_k}^{-\top} \tag{10}$$

$$S = H \bar{\Sigma}_{t_k}^{l} H^\top + \bar{N}_k \tag{11}$$

$$L_{t_k} = \bar{\Sigma}_{t_k}^{l} H^\top S^{-1} \tag{12}$$

Using Equations 13 and 14, the state estimation was then calculated. This state estimation was used in the next iteration.

$$X_{t_k} = \bar{X}_{t_k} \exp(L_{t_k}(\bar{X}_{t_k}^{-1} Y_{t_k} - b)) \tag{13}$$

$$\Sigma_{t_k}^{l} = (I - L_{t_k} H) \bar{\Sigma}_{t_k}^{l} (I - L_{t_k} H)^\top + L_{t_k} \bar{N}_k L_{t_k}^\top \tag{14}$$

B. Particle Filter (PF)

Particle Filter (PF) algorithms utilize particles, χ, to obtain the best estimated pose of a vehicle by utilizing landmark locations along with range and bearing measurements. When functional, the particles converge toward the vehicle pose through each iteration. The overall structure of the PF algorithm is summarized in the Figure 3 flowchart, and the algorithm is detailed in Figure 11 in the Appendix. The implementation of this filter for this pose estimation application is discussed in detail below.

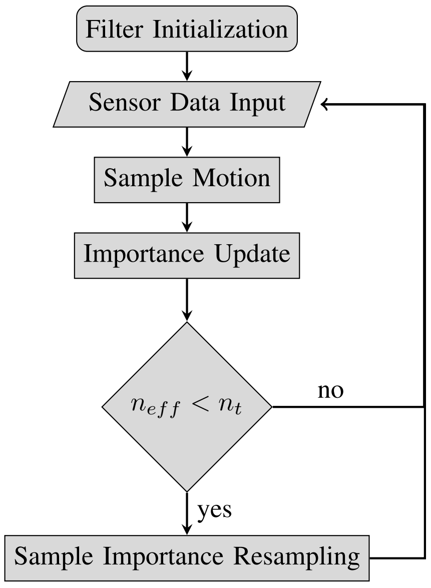


Fig. 3. Particle Filter Process Flowchart

1) Initialize Filter: A 10-by-10 even grid, with 100 particles of equal importance, was used to initialize the PF. The grid was overlaid on a portion of the map near the first GPS sensor output, which became the prior state estimate for the filter. Also, for sample size, using 100 particles provided optimal performance through fast calculations and near-peak filter performance.

2) Sensor Data Input: To run the particle filter, the landmark observations must be converted into usable measurement value, z_χ. For this application, z_χ consists of range, r_χ, and bearing, θ_χ, with zero-mean Gaussian noise, η, added. The Gaussian noise can be calculated using Equation (15), where L is the lower Cholesky decomposition. The equation to calculate range and bearing are provided below in Equation (16), where l_x, and l_y are the Cartesian coordinates of the landmark, and x and y are the coordinates of the particle. As a result, z_χ can be calculated using Equation (17) and is a 2-dimensional vector corresponding to the range and bearing in this application.

$$\eta = L * \mathcal{N}(\mu, \sigma^2) \tag{15}$$

$$\begin{bmatrix} z_r \\ z_\theta \end{bmatrix} = \begin{bmatrix} \sqrt{(x - l_x)^2 + (y - l_y)^2} \\ \arctan\left(\frac{l_y - y}{l_x - x}\right) \end{bmatrix} \tag{16}$$

$$z_\chi = \begin{bmatrix} z_r \\ z_\theta \end{bmatrix} + \begin{bmatrix} \eta_{z_r} \\ \eta_{z_\theta} \end{bmatrix} \tag{17}$$

3) Sample Motion: For this experiment, the algorithm employed a random walk for the motion model. For this random walk, lower Cholesky factored Gaussian noise was introduced into each particle as described in Equation (15), and the particle was then propagated.

4) Importance Update: The importance weight of each particle was based on the range and bearing of itself from the observed landmark. This was calculated based on the difference between the particle's range and bearing to each landmark, z_χ, and the robot's measurements of its range and bearing to each landmark, z_l. The total weight of these particles was then normalized. the number of effective particles, n_{eff}, was then calculated based on the weight as shown in Equation (18). If n_{eff} was less than the sample threshold n_t, then the Sample Importance Resampling step was conducted. Else, then the filter returned to the Sensor Data Input step for the next iteration, For additional implementation information, refer to the Appendix PF Algorithm or the `/where-dyss/readme_files/pfREADME.md` document on GitHub.

$$n_{eff} \leftarrow \frac{1}{\sum_{i=1}^{n} (\tilde{\omega}_k^i)^2} \tag{18}$$

5) Sample Importance Resampling (SIR): As the particle filter moved through iterations, the number of effective particles reduced over time, which in turn reduces the efficiency of the filter. To address this concern, new particles with higher weights were added to the sample pool when the effective sample size drops below a certain threshold n_t. For this project, n_t was one-third of the initialized sample size of the particles. This threshold had proven to retain the performance of the filter in the team's prior experience in other projects. Upon resampling, the filter returned to the Sensor Data Input step for the next iteration. For additional information on how the team calculated the SIR in this experiment, refer to `/where-dyss/output/Matlab/PF_To_Python/particle_filter.m` MATLAB class.

III. Experiment

We applied the proposed probabilistic filters on multiple scenes in this dataset. Descriptions of the chosen scenes may be found in the GitHub repository in the `/where-dyss/readme_files/dataREADME.md` file. This paper will illustrate the performance of the filters on *Scene-0249* of the nuScenes by Aptiv dataset [3]. In this scene, the vehicle turned right after exiting a bridge onto a narrow street filled with parked cars on both sides, along with several pedestrians on the sidewalk, and one person crossing the street in a crosswalk. Additionally, there were several traffic cones on the right side of the road protecting parked cars and unloading trucks. The weather conditions were cloudy but dry, and the time of day was mid to late afternoon.

The filters were implemented as class objects and run sequentially in their respective `run_me` files.

For the IEKF, the CAN data was processed and aligned in MATLAB from the provided .json files. The algorithm was developed in MATLAB as well.

For the Particle Filter, the semantic annotations (landmarks) including range and bearing calculations from the scene data were all processed in Python. The data from Python was saved into .mat files to be used in MATLAB, as the filter was implemented in MATLAB.

For both filters' results, the estimated states were saved in .mat files, and visualized in Python on nuScenes' Fancy Map as shown in Figure 1.

Calibrations

Invariant EKF: The lower Cholesky decomposition of the following values in a diagonal zero-mean Gaussian noise matrix was added to the readings of IMU gyrometer and accelerometer data: $[(3\text{e}{-}4)^2, (3\text{e}{-}4)^2, (3\text{e}{-}4)^2]$.

Q matrix (as defined in the Appendix) comprises diagonal entries in sequence (Higher values denote high variance and low information. Lower values denote low variance and high information.):

$$[(1\text{e}{-}5)^2, (1\text{e}{-}5)^2, (1\text{e}{-}5)^2,\\ (1\text{e}6)^2, (1\text{e}6)^2, (1\text{e}6)^2,\\ (1\text{e}{-}1)^2, (1\text{e}{-}1)^2, (1\text{e}4)^2]$$

V_k is the vector of noise in GPS readings: $[5\text{e}{-}5; 5\text{e}{-}5; 5\text{e}{-}5; 0; 0]$.

$\text{Cov}[V_k]$ (or Σ_{V_k}) is a diagonal matrix with variance values along the diagonal. The variance in the GPS readings are very low as the algorithm in [1] is robust: $[(1\text{e}{-}6)^2; (1\text{e}{-}6)^2; (1\text{e}{-}6)^2; 0; 0]$.

Particle Filter: The lower Cholesky decomposition of the following values in a diagonal zero-mean Gaussian noise matrix was added to the readings of semantic annotations' range and bearing values: $[(2\text{e}{-}3)^2, (5\text{e}{-}3)^2]$.

Motion noise covariance matrix, Q, is $9I_2$.

Measurement noise covariance matrix, R, is a diagonal matrix with the values $[1.2^2, 0.7^2]$ along the diagonal.

The results of this experiment can be reviewed in the following section.

IV. Result

A. Invariant EKF

Filter Performance without GPS Data Loss

The YouTube presentation link provided in 'Our Contribution' section contains the IEKF active localization and pose estimation results in a video format, beginning at time 1:03. The resultant final pose estimation for IEKF filter run is provided in Figure 4. The red line in the figure is the IEKF pose estimation on the street map, and the red rectangle represents the pose of the vehicle. The black line representing pose ground truth is no longer visible, showing the filter localization performance was nearly perfect.

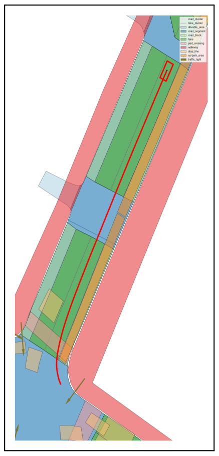


Fig. 4. Invariant EKF Result - nuScenes Fancy Map output of Scene-0249

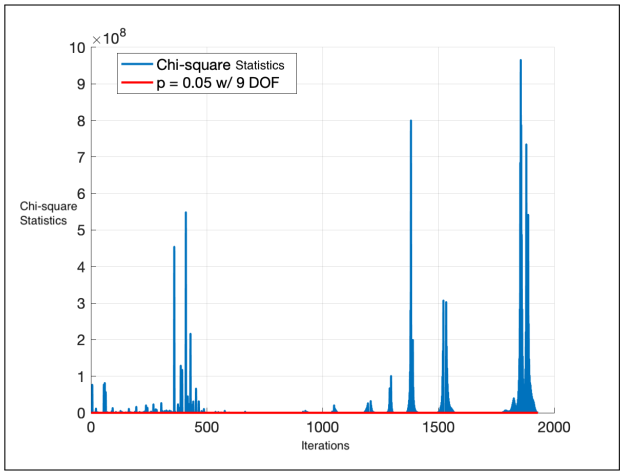


Fig. 5. Invariant EKF - Mahalanobis Result

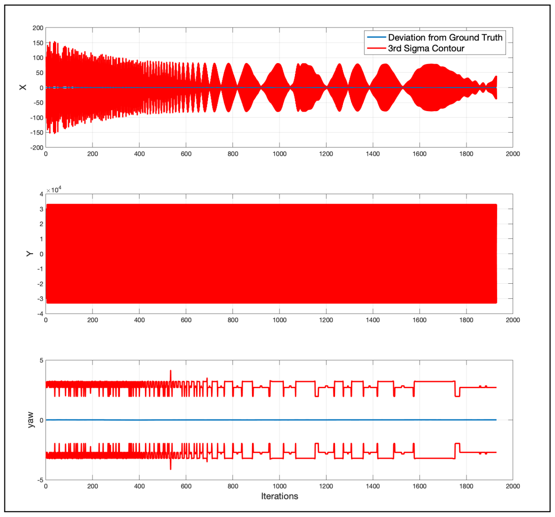


Fig. 6. Invariant EKF - Covariance Result

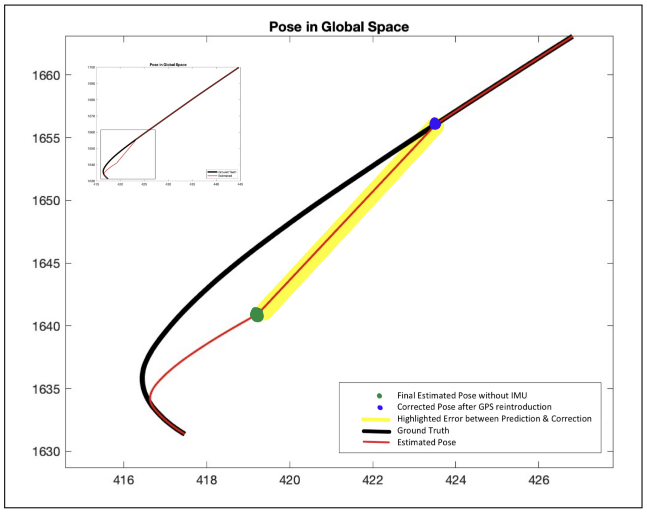


Fig. 7. MATLAB Map output of Scene-0249's Invariant EKF run with GPS Loss between $t = 1.5s$ and $t = 8s$

Figures 5 and 6 show the Mahalanobis and covariance calculations throughout the IEKF filter run. These figures are extremely noisy, and the team's attempts to correct it within the allotted time were unsuccessful. The Mahalanobis data values significantly exceed the set p threshold. This does not match the output of the IEKF result, which had nearly no error. The covariance plot also shows large oscillations in the 3 sigma contour, and doesn't provide valuable information unless compared to the covariance plot of the IEKF run without GPS in Figure 9. The comparison shows that with GPS, the IEKF run's 3rd sigma contour values are much smaller and evenly spread out than the run without GPS. This shows that the covariance values remained similar throughout the iterations and relatively low due to the correction step.

Filter Performance with GPS Data Loss

The resultant final pose estimation for IEKF filter performance without GPS for a period of time is provided in Figure 7. The top-left box with the figure shows the full pose estimation result. The black rectangle box in this box is the zoomed in portion we see on the main plot. The red line in the figure is the pose estimation on the street map, and the black line represents the ground truth. As the IEKF algorithm loses the ability to correct between $t = 1.5s$ and $t = 8s$, it relies on the IMU for localization. This result shows that within a few iterations of the turn, the IMU-only prediction veers the vehicle away from the ground truth. When correction is introduced into the algorithm, we see the large correction highlighted in the figure.

Figures 8 and 9 show the Mahalanobis distance and covariance calculations for the IEKF filter performance without GPS correction for a period of time. Similar to above, these figures are extremely noisy, and the team's attempts to correct it within the allotted time were unsuccessful. The Mahalanobis data values significantly exceed the set p threshold. This does not match the output of the IEKF result, which had nearly no error. The covariance plot also shows large oscillations in the 3 sigma contour for X but a clean exponential increase for Y. Again, it doesn't provide valuable information unless compared to the covariance plot of the IEKF run with GPS correction in Figure 6. The comparison shows that without GPS correction, the IEKF run's 3-sigma contour increases exponentially. This also shows that the covariance values increased throughout the iterations without GPS correction, but it returned to "normal" after GPS was reintroduced.

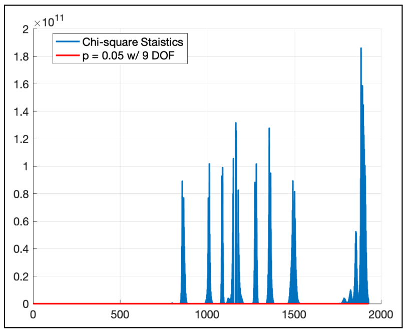


Fig. 8. Mahalanobis Result of Invariant EKF run with GPS Loss between $t = 1.5s$ and $t = 8s$

B. Particle Filter

The YouTube presentation link provided in 'Our Contribution' section contains the PF active localization and pose estimation results in a video format, beginning at time 1:57. The resultant final pose estimation for the PF run is provided in Figure 10. In the figure, the PF pose estimation is represented by a white line and is superimposed onto the pose ground truth (black line). The result shows the filter localizing closely but not perfectly.

V. CONCLUSIONS

A. Invariant EKF and Particle Filter Comparison

Invariant EKF filters improve the prospective of stable and reliable algorithms to be used in autonomous vehicle development. Although the particle filter is an industrialized algorithm in the field of robotics, it does not provide performance

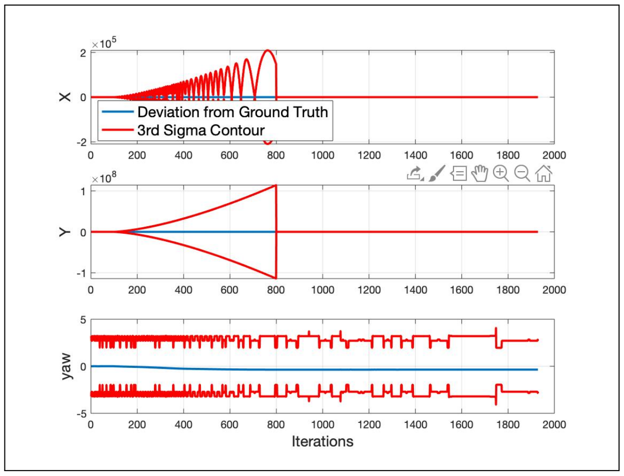


Fig. 9. Covariance of Invariant EKF run with GPS Loss between $t = 1.5s$ and $t = 8s$

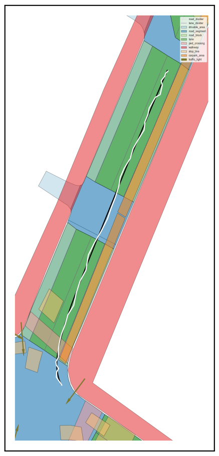


Fig. 10. Particle Filter Result - nuScenes Fancy Map output of Scene-0249

equivalent to IEKF for pose estimation and localization. The IEKF filter localized the vehicle almost perfectly using IMU prediction and GPS correction. However, it was extremely sensitive to an extended duration of GPS data loss as seen in Figure 7.

As for the particle filters, there were slight deviations from the ground truth. The overall tracking may be acceptable for a closed loop robot in a controlled environment but is unacceptable for a safety-critical pose-estimation application with low margin for error. The PF also presents an over-information challenge. If the number of landmarks used at each time-step was too high, there would be a loss of diversity in particles, causing the particle distribution to collapse into one particle and potentially estimate the incorrect ground truth pose. Reduced number of landmark observations can resolve this issue. The particle filter's overall performance can also be improved by increasing the motion noise and the number of particles, or decreasing the number of landmark observations.

For a practical implementation, both algorithms present their own set of challenges. For both filters, all data will be asynchronous. Additional delays may be required to synchronize the data. There may also be bias in the IMU that would require correction within in each calculation, if IMU is used in the implementation. For the particle filter, identifying a landmark may also present an additional challenge. Without landmark observations available, the particle filter fails to perform effectively. In addition, identifying landmarks introduce an additional time delay. This could have catastrophic consequences in rapidly changing environments.

Overall, the IEKF localized the vehicle significantly better than the PF, but the PF was more robust to single landmark loss than the IEKF was to the loss of GPS for extended periods of time. To elaborate more on the concepts of both probabilistic filters, future work would need to be conducted as outlined below.

B. Future Work

Due to the limited time and scope of this project, the team was unable to apply novel ideas but did learn how to establish probabilistic filters to evaluate real-world datasets. The team also discussed opportunities to expand upon the concepts illustrated in this paper.

To start, incorporating the steering wheel angle, steering feedback, and wheel speed data from the CAN bus into the correction step of the IEKF should provide further improved results. By using this vehicle data which is already available, there are improvement opportunities in path prediction for autonomous vehicles, especially in a system experiencing sensor signal losses. There is an additional research opportunity for the team to use the steering angle sensor and wheel speed sensors to model yaw rate; use steering feedback to model road noise; and use using wheel speed data to model vehicle velocity and pose estimation. By combining this data with machine learning algorithms, the team can create new dynamic motion models that continuously adjust to vehicle and environment changes and improve filter performance.

Additionally, expanding on the understanding and improved implementation of the Mahalanobis distance and 3-sigma plots will be key to validate that the proposed probabilistic filters will meet target performance.

APPENDIX

Invariant EKF Equations

Gamma Formulas - $\Gamma_0(\phi)$; $\Gamma_1(\phi)$; $\Gamma_2(\phi)$

$$\Gamma_0(\phi) = I + \frac{\sin(\|\phi\|)}{\|\phi\|}(\phi^\wedge) + \frac{1-\cos(\|\phi\|)}{\|\phi\|^2}(\phi^\wedge)^2 \tag{19}$$

$$\Gamma_1(\phi) = I + \frac{1-\cos(\|\phi\|)}{\|\phi\|^2}(\phi^\wedge) + \frac{\|\phi\| - \sin(\|\phi\|)}{\|\phi\|^3}(\phi^\wedge)^2 \tag{20}$$

$$\begin{aligned}\Gamma_2(\phi) =& \frac{1}{2}I + \frac{\|\phi\| - \sin(\|\phi\|)}{\|\phi\|^3}(\phi^\wedge) \\ &+ \frac{\|\phi\|^2 + 2\cos(\|\phi\|) - 2}{2\|\phi\|^4}(\phi^\wedge)^2\end{aligned} \tag{21}$$

Quaternion to Euler *- Roll ϕ ; Pitch θ ; Yaw ψ*

$$\phi = \arctan\left(\frac{2*(\mathrm{w}*\mathrm{x}+\mathrm{y}*\mathrm{z})}{1-2*(\mathrm{x}*\mathrm{x}+\mathrm{y}*\mathrm{y})}\right) \tag{22}$$

$$\theta = \arcsin(2*(\mathrm{w}*\mathrm{y}-\mathrm{z}*\mathrm{x}))\begin{cases}1 & \theta > 1\\ -1 & \theta < -1\end{cases} \tag{23}$$

$$\psi = \arctan\left(\frac{2*(\mathrm{w}*\mathrm{z}+\mathrm{x}*\mathrm{y})}{1-2*(\mathrm{y}*\mathrm{y}+\mathrm{z}*\mathrm{z})}\right) \tag{24}$$

Skew Symmetric Matrix for any 3D Vector, d

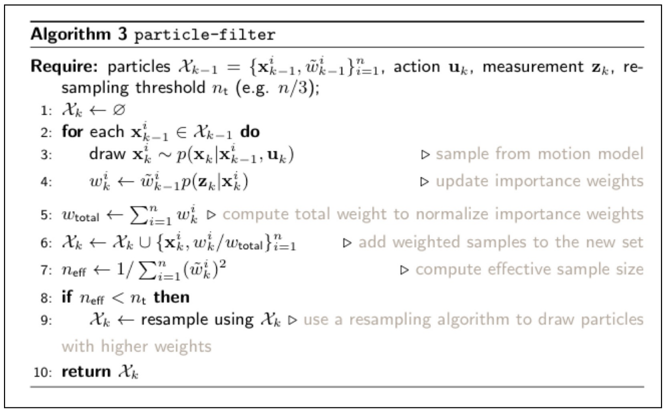
$$d^\wedge = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \tag{25}$$

Q Matrix *- Additive Sensor Noise Matrix structure*

$$Q = \begin{bmatrix}
\sigma_\phi^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \sigma_\theta^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \sigma_\psi^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \sigma_{\mathrm{v}_x}^2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \sigma_{\mathrm{v}_y}^2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \sigma_{\mathrm{v}_z}^2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \sigma_{\mathrm{x}}^2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{\mathrm{y}}^2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{\mathrm{z}}^2
\end{bmatrix}$$

Particle Filter Equations

Basic Particle Filter Algorithm in Figure 11



```
Algorithm 3 particle-filter
Require: particles $\mathcal{X}_{k-1} = \{\mathbf{x}_{k-1}^i, \tilde{w}_{k-1}^i\}_{i=1}^n$, action $\mathbf{u}_k$, measurement $\mathbf{z}_k$, re-
    sampling threshold $n_t$ (e.g. $n/3$);
 1: $\mathcal{X}_k \leftarrow \varnothing$
 2: for each $\mathbf{x}_{k-1}^i \in \mathcal{X}_{k-1}$ do
 3:     draw $\mathbf{x}_k^i \sim p(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{u}_k)$          ▷ sample from motion model
 4:     $w_k^i \leftarrow \tilde{w}_{k-1}^i p(\mathbf{z}_k|\mathbf{x}_k^i)$            ▷ update importance weights
 5: $w_{\text{total}} \leftarrow \sum_{i=1}^n w_k^i$ ▷ compute total weight to normalize importance weights
 6: $\mathcal{X}_k \leftarrow \mathcal{X}_k \cup \{\mathbf{x}_k^i, w_k^i/w_{\text{total}}\}_{i=1}^n$     ▷ add weighted samples to the new set
 7: $n_{\text{eff}} \leftarrow 1/\sum_{i=1}^n (\tilde{w}_k^i)^2$              ▷ compute effective sample size
 8: if $n_{\text{eff}} < n_t$ then
 9:     $\mathcal{X}_k \leftarrow$ resample using $\mathcal{X}_k$ ▷ use a resampling algorithm to draw particles
    with higher weights
10: return $\mathcal{X}_k$
```

Fig. 11. Basic Particle Filter Algorithm in Robotics

ACKNOWLEDGMENTS

This project would not have been successful without the aid and guidance of Professor Maani Ghaffari Jadidi and Graduate Student Instructor Peter Westra of the University of Michigan, Ann Arbor. We thank you for your continued support throughout the semester, especially for the time taken to guide us during a weekend and throughout the COVID-19 pandemic.

REFERENCES

[1] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. arXiv preprint, arXiv:1903.11027, 2019.

[2] Silvère Bonnabel, Philippe Martin and Erwan Salaun. Invariant Extended Kalman Filter: theory and application to a velocity-aided attitude estimation problem. *Proceedings of the 48th IEEE Conference on Decision and Control*, pp: 1297 - 1304. 2010. 10.1109/CDC.2009.5400372.

[3] nuscenes.org. 2020. nuScenes. [online] Available at: ¡https://www.nuscenes.org¿ [Accessed 1 March 2020].

[4] Axel Barrau and Silvère Bonnabel. Invariant Kalman filtering. *Annual Reviews of Control, Robotics, and Autonomous Systems*, Vol 1, pp: 237-257, 2018.

[5] Axel Barrau, Silvère Bonnabel. Navigating with highly precise odometry and noisy GPS: a case study. 2016. hal-01267244

[6] Christopher R. Carlson. Estimation with Applications for Automobile Dead Reckoning and Control. PhD Dissertation, Department of Mechanical Engineering, Stanford University, CA. 2004.

[7] Manon Kok, Jeroen D. Hol and Thomas B. Schön. "Using Inertial Sensors for Position and Orientation Estimation." *Foundations and Trends in Signal Processing*: Vol. 11: No. 1-2, pp: 1-153. 2017.

[8] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. Massachusetts Institute of Technology Press. 2006.

[9] Timothy D. Barfoot. *State Estimation for Robotics*. Cambridge University Press. 2019.

[10] Axel Barrau and Silvère Bonnabel. Invariant filtering for Pose EKF-SLAM aided by an IMU. In IEEE Conference on Decision and Control (CDC), 2015.

[11] Martin Brossard, Silvère Bonnabel. Learning Wheel Odometry and IMU Errors for Localization. International Conference on Robotics and Automation (ICRA), May 2019, Montreal, Canada.