Assignment IV: Advanced Robot Navigation (RBE 595)

Pradnya Sushil Shinde

April 25, 2024

1 Particle Filter

The objective of the project is to implement a Particle Filter to estimate the pose of a quadcopter drone. The Measurement Model and Process Model remain the same as used in the previous assignment of Non-Linear Kalman Filter.

In this problem, we will model the pose using a typical fifteen-state model used in inertial navigation. This state vector will consist of the three-axis components of the linear position, orientation, linear velocity, gyroscope bias, and accelerometer bias as shown below:

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \dot{\mathbf{p}} \\ \mathbf{b}_g \\ \mathbf{b}_a \end{bmatrix} \tag{1}$$

Additionally, we will define the orientation using a Z-X-Y Euler angle parameterization corresponding to rolling, pitching, and yawing $\mathbf{q} = [\phi\theta\psi]$. In rotation matrix form these yield:

$$R(\mathbf{q}) = \begin{bmatrix} \cos\psi\cos\theta - \sin\phi\sin\psi\sin\theta & -\cos\phi\sin\psi & \cos\psi\sin\theta + \cos\theta\sin\phi\sin\psi \\ \cos\theta\sin\psi + \cos\psi\sin\phi\sin\theta & \cos\phi\cos\psi & \sin\psi\sin\theta - \cos\psi\cos\theta\sin\phi \\ -\cos\phi\sin\theta & \sin\phi & \cos\phi\cos\theta \end{bmatrix} \tag{2}$$

2 Introduction

2.1 System Models

Let us define the System Models that will help us evaluate the Process and Observations for the filter model.

2.1.1 Process Model

The continuous time process model is defined as:

$$\dot{x} = \begin{bmatrix} \dot{\mathbf{p}} \\ G(\mathbf{q})^{-1}\mathbf{u}_w \\ \mathbf{g} + R(\mathbf{q})\mathbf{u}_a \\ \mathbf{n}_{bg} \\ \mathbf{n}_{ba} \end{bmatrix} \tag{3}$$

where $\mathbf{u}_w$ and $\mathbf{u}_a$ are the commanded angular velocity and linear acceleration, $\mathbf{g}$ is the gravity vector, and $G(\mathbf{q})$ is defined as:

$$G(\mathbf{q}) = \begin{bmatrix} \cos\theta & 0 & -\cos\phi\sin\theta \\ 0 & 1 & \sin\phi \\ \sin\theta & 0 & -\cos\phi\cos\theta \end{bmatrix} \tag{4}$$

The size of $R(q)$ and $G(q)$ will vary for the number of particles being sampled.

2.1.2 Observation Model

Our observation model is relatively simple. We will be using a computer vision-based technique that measures the position and orientation:

$$\mathbf{z} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} + \mathbf{v} \tag{5}$$

Our measurement model is dependent on the estimates of the position and orientation that we will obtain in Task I. Essentially z is an array of $[x, y, z, roll, pitch, yaw]$ estimates.

3 Project Implementation

3.1 TASK I: Particle Filter Implementation

Particle filtering uses a set of particles (also called samples) to represent the posterior distribution of a stochastic process given the noisy observations. The following process explains the implementation:

1. **Initialise Particles**
   We have the estimates retrieved through the Measurement Model based on the Computer Vision technique. We will initialize particles around these estimates to process an initial state for the filter model. We will generate random samples from the uniform distribution.

2. **Predict Step**
   For each particle, we will take the original particle, sample from the noise distribution, and use the measured inputs plus noise to determine the future state of that particle using the process model. The process model remains the same as used in Non-Linear Kalman Filter as described in Section 2.
   $$x_t^{[i]} \approx (x_t | x_{t-1}^{[i]}, u_t)$$

3. **Update Step**
   The particle filter refines the uncertainty of the distribution by re-weighting and resampling the particle distribution. Give each particle a new weight according to the measurement model.
   $$w^{[i]} = g(z_t | x_t^{[i]})$$

4. **State Estimation**
   Particles that more accurately estimate the truth state will receive higher weights. We can use Weighted Average/Average Particle Weights Estimation to implement state estimation on existing particles.

5. **Resampling**
   After reweighting, resample the particles in proportion to their weight. We will use a low-variance sampling method to do this.

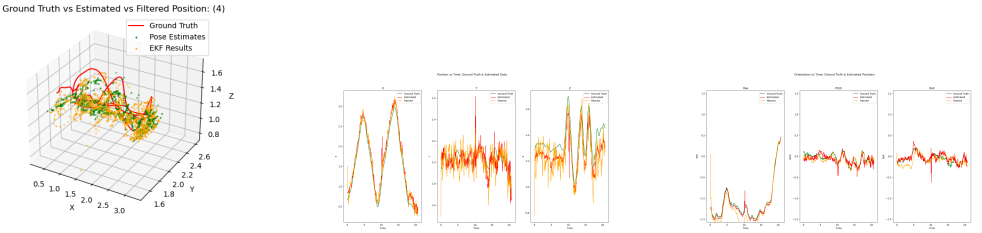
$$\begin{aligned}
&w_{total} = \sum_{i=1}^{M} w^{[i]} \\
&r = \text{rand}\left(0, \frac{w_{total}}{M}\right) \\
&X = [\ \ ];\ i = 1;\ c = w^{[i]}; \\
&\text{for } i = 1{:}M \\
&\quad u = r + \frac{(k-1)w_{total}}{M} \\
&\quad \text{while } u > c \\
&\quad\quad i++ \\
&\quad\quad c \mathrel{+}= w^{[i]} \\
&\quad \text{append}\left(x^{[i]}\right) \rightarrow X
\end{aligned}$$

Figure 1: Sampling Algorithm

3.2 TASK II: Navigation solution and Particle Count Investigation

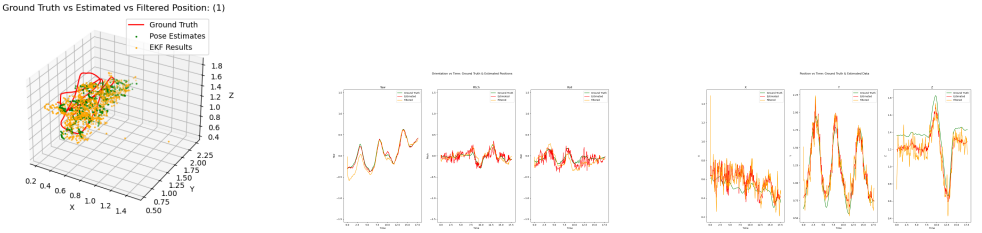
3.2.1 Particle Sampling

Particle filter solely relies on Particle Distribution to filter the estimates. To determine a singular position estimate, we will process the position estimates as the highest weighted particle, the average over all the particles, and the weighted average over all particles. Below are visualization plots that define the above sampling methods and their behavior on the provided dataset.



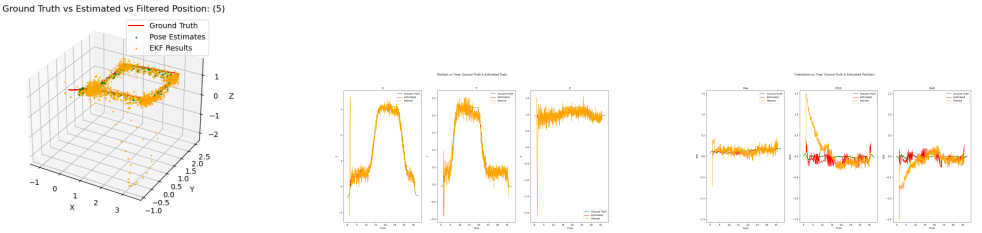
(a) Trajectory Plot (b) Position Comparison (c) Orientaion Comparison

Figure 2: **Student Data 4: Average Weights Particle Sampling**



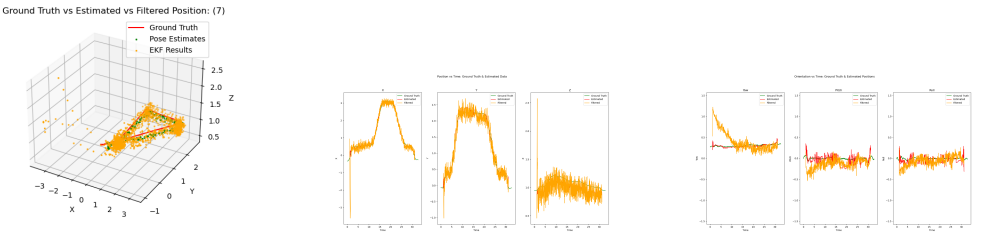
(a) Trajectory Plot (b) Position Comparison (c) Orientaion Comparison

Figure 3: **Student Data 1: Average Weights Particle Sampling**



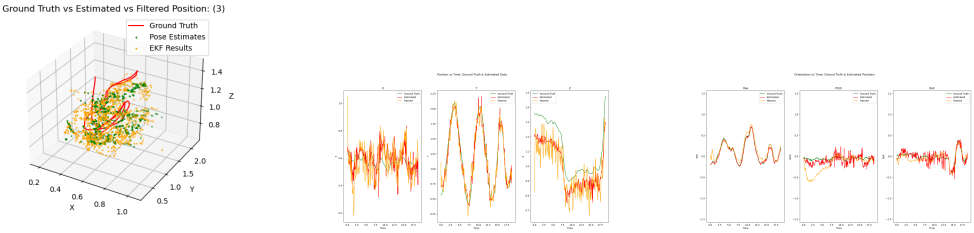
(a) Estimated Pose (b) Ground Truth Pose (c) Positions Comparison

Figure 4: **Student Data 5: Highest Weighted Particle Sampling**



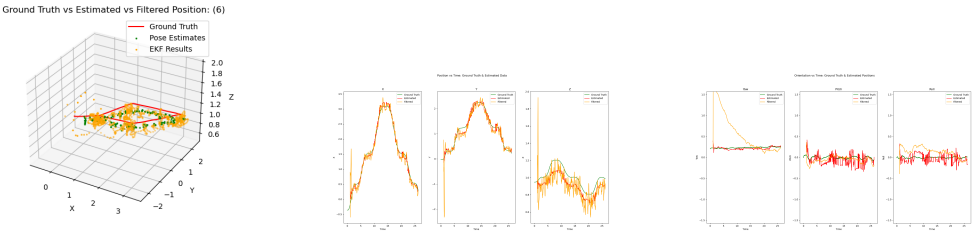
(a) Estimated Pose (b) Ground Truth Pose (c) Positions Comparison

Figure 5: **Student Data 7: Highest Weighted Particle Samlping**



(a) Estimated Pose (b) Ground Truth Pose (c) Positions Comparison

Figure 6: **Student Data 3: Weighted Average Sampling**



(a) Estimated Pose (b) Ground Truth Pose (c) Positions Comparison

Figure 7: **Student Data 6: Weighted Average Sampling**

3.2.2 RMSE Evaluation

RMSE is calculated as:

$$\text{position\_estimates\_error} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - x_i)^2}$$

where **position_estimate_error** is the position estimate error.
$\mathbf{X_i}$ is the i^{th} ground truth position value.
$\mathbf{x_i}$ is the i^{th} estimated position value.
n is the number of entries in the arrays.

The RMSE has been calculated for three different sampling methods: Highest Weighted Particle, Average over all Particles, and Weighted Average over all particles.

1. Maximum Weight Estimation

$$\text{max\_weight\_idx} = \arg\max_i(\text{weights}[i])$$

$$\text{estimatedState} = \text{particles}[\text{max\_weight\_idx}]$$

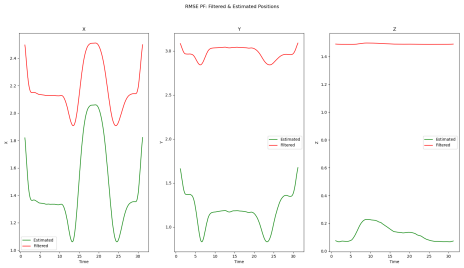
2. Average Estimation

$$\text{estimatedState} = \frac{1}{N}\sum_{i=1}^{N}\text{particles}[i]$$

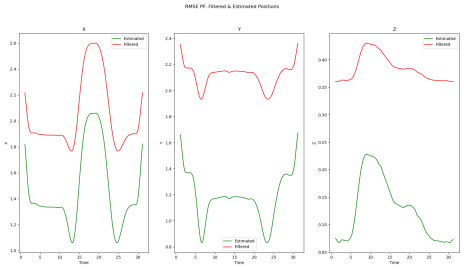
3. Weighted Average

$$\text{estimatedState} = \sum_{i=1}^{N}\text{particles}[i] \times \text{weights}[i]$$

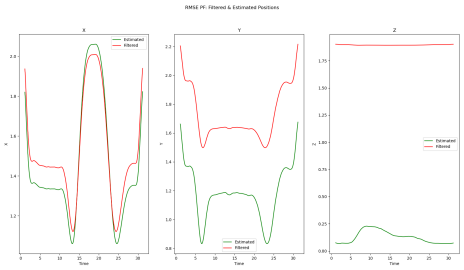
The performance visualization is shown below for one of the datasets. We can conclude that the Weighted Average Sampling Method performs better than the other two methods, as evident from the plots below.



(a) Average Weight RMSE



(b) Max Weight RMSE



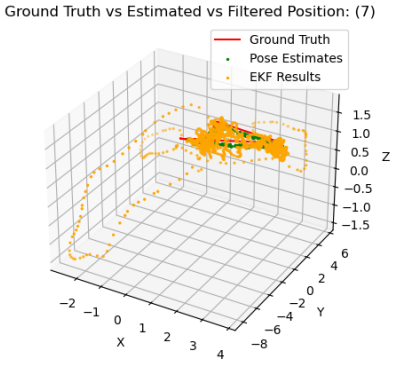
(c) Weighted Average RMSE

Figure 8: **Student Data 7: RMSE Postion**

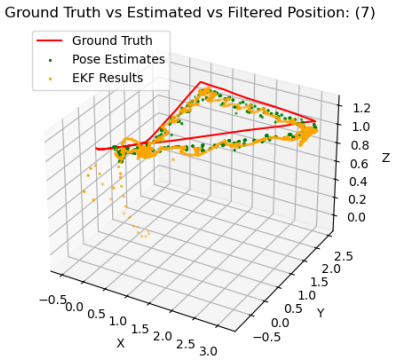
3.2.3 Does Particle Count affect the performance of the particle filter?

Particle count directly affects the speed of convergence. As you increase the particle count, it takes more computations to perform filtering. Although increasing particle count is computationally expensive for the performance of Particle Filter, there are a few advantages too.

1. Generally, a higher number of particles leads to a more accurate estimation of the system state. This is because more particles provide better coverage of the state space, allowing the filter to capture the underlying probability distribution more effectively. 100 particles, as shown in the plots, leave behind redundant filter particles, thus covering less trajectory space.



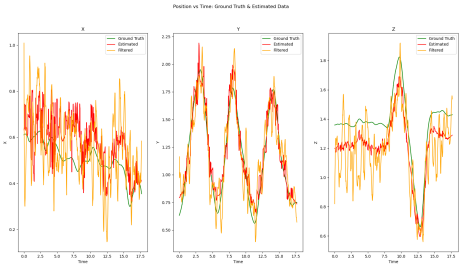
(a) 100 Particles



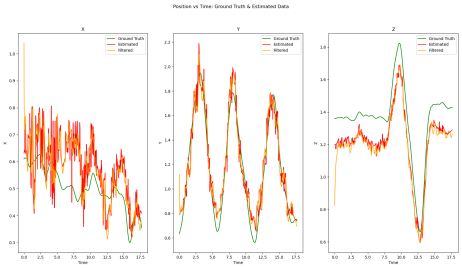
(b) 5000 Particles

Figure 9: **Particle Count and Trajectory**

2. Increasing the particle count also helps in reducing the noise and smoothes the trajectory overall. Below are the position comparisons for 100 and 5000 particles. We can see 5000 particles have smoothened the position plots.



(a) 100 Particles



(b) 5000 Particles

Figure 10: **Particle Count and Position**

3.3 Task III: Comparison to EKF

We will compare the performance of the Extended Kalman Filter and Particle Filter based on the data given to us.

3.3.1 Ease of Implementation

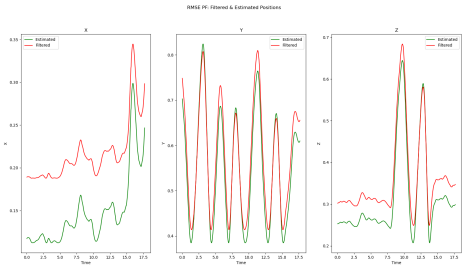
- Extended Kalman Filter was easy to code due to the pre-defined parameters that are easy to tune. Particle Filters can incorporate various sampling techniques as well as various state estimation techniques. This results in a randomized sampling approach, which demands variations in writing the code.
- The parameter tuning in the Extended Kalman Filter was relatively simpler as compared to the Particle Filter. This is because EKF uses noise covariances which can have fixed values, however, PF uses noise values that are defined by random uniform sampling around a particular value. Again the random nature makes the tuning a little complicated.

3.3.2 Speed of Code

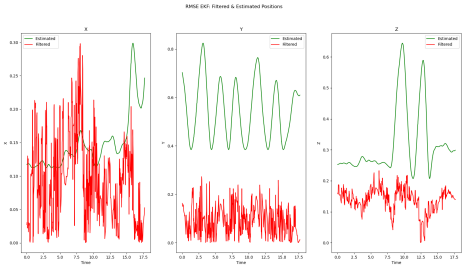
- In terms of the speed of code, the computational efficiency was relatively the same for both EKF and PF. However, as one increases the number of particles to be sampled in PF, the speed of code is reduced marginally.
- The speed matters because in real-time, if we wish to avoid noticeable lags in the dynamics of a robot/system, we must have data that is processed effectively faster for the robot to act and react smoothly.

3.3.3 Accuracy of Results

In terms of accuracy, EKF performed better. To compare RMSE values, refer to the figures below. We can observe that its RMSE values are comparatively lower as compared to the RMSE produced by Particle Filter.

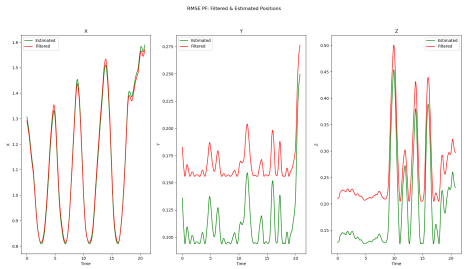


(a) Particle Filter

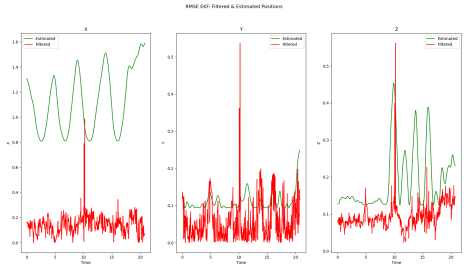


(b) EKF

Figure 11: **Student Data 1: RMSE Postion**

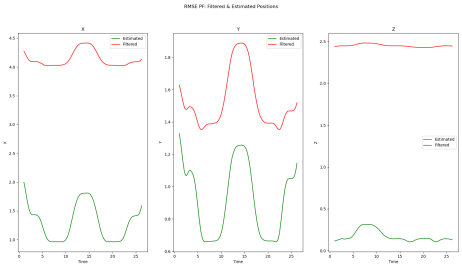


(a) Particle Filter

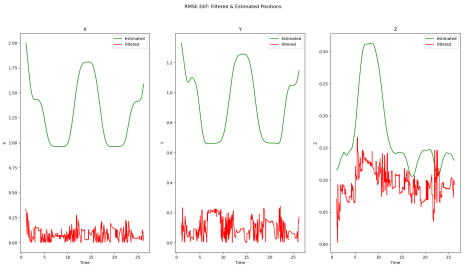


(b) EKF

Figure 12: **Student Data 4: RMSE Postion**

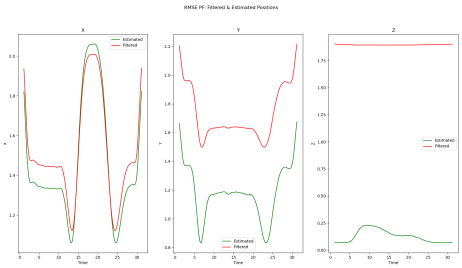


(a) Particle Filter

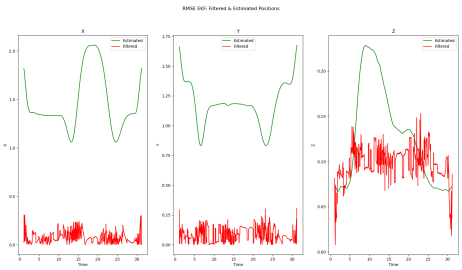


(b) EKF

Figure 13: **Student Data 6: RMSE Postion**



(a) Particle Filter



(b) EKF

Figure 14: **Student Data 7: RMSE Postion**

4 Conclusion

Implementing Particle Filters had a few Pros and Cons. We will list them to conclude on the performance of the Particle Filter.

1. Particle Filter provides an approximation of the posterior distribution through sampling methods which can be reliable. However, it can also be computationally expensive, especially when the number of particles to be sampled increases.

2. Particle Filter does not require linearization, making it more suitable for highly non-linear systems. However, it can suffer from particle impoverishment, especially in high-dimensional state spaces.