Solution to analysis in Home Assignment 4

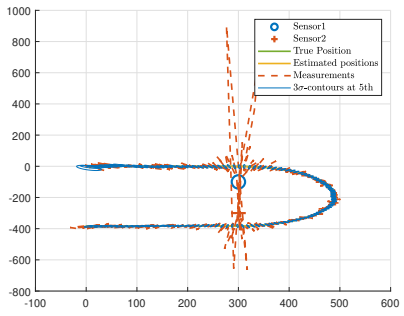
Yuhong Zhou + yuhong

Analysis

In this report I will present my independent analysis of the questions related to home assignment 1. I have discussed the solution with Xinying Wang but I swear that the analysis written here are my own.

1 Smoothing

(a) As motivated by the **HA3**, we choose to filter the measurement sequence using the CKF. And we mainly use the code from the **HA3** to get the filtered result which shown as in the Figure 1.1. Then we use the MatLab function **nonLinRTSsmoother** to generate the smoothed position trajectories as illustrated in Figure 1.2.



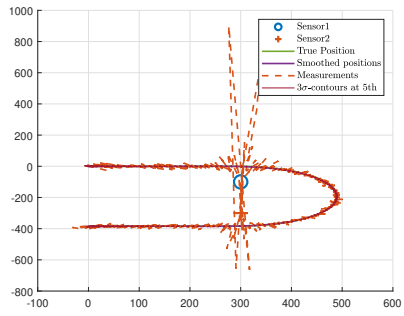


Figure 1.1: Filtered position trajectories. *Figure 1.2: Smootheded position trajectories.*

To better compare the results of the filtered and smoothed trajectories, we focused on enlarging the section of the turn, as depicted in Figure 1.3. This adjustment allowed for a more detailed examination of the trajectories' behavior within this critical maneuver.

From the Figure 1.3, we can find that both the filtered and smoothed trajectories fits the true trajectory well. But the shape of smoothed trajectory is more closer to the shape of true trajectory than the one of filtered trajectory. And the shape of the smoothed trajectory is much smoother. The smoother estimate the position by integrating information from both past and future measurements. This comprehensive approach enhances the accuracy of the system's true state estimation, effectively mitigating the impact of measurement noise and ultimately enhancing overall precision.

From the Figure 1.3, we can also obtained that the smoothing error covariances are much smaller than the filtering error covariances in the same time instance. Since the smoothed positions will have less uncertainty.

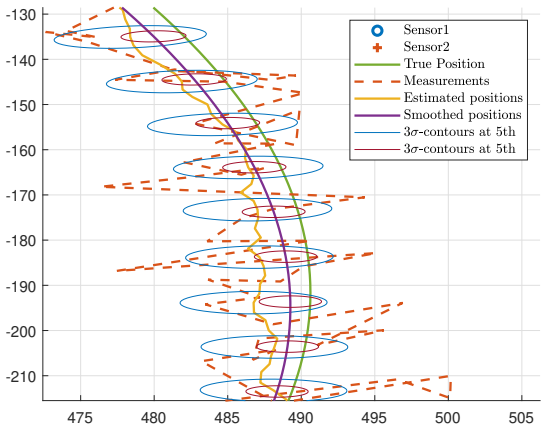


Figure 1.3: Enlarged picture.

(b) The resulting trajectories introduced an outlier as illustrated in the Figure 1.4. We can find that the filter performs really bad when cope with the introduced outlier data. While the smoother can still performs well and get a result that really close to the true trajectory.

Since the smoother estimate the position at the time instance by integrating information from both past and future measurements, which helps mitigate the adverse effects of measurement noise, ensuring a more robust and reliable estimation process. Therefore, the impact of individual outlier data is reduced, resulting in more reliable and robust estimation.

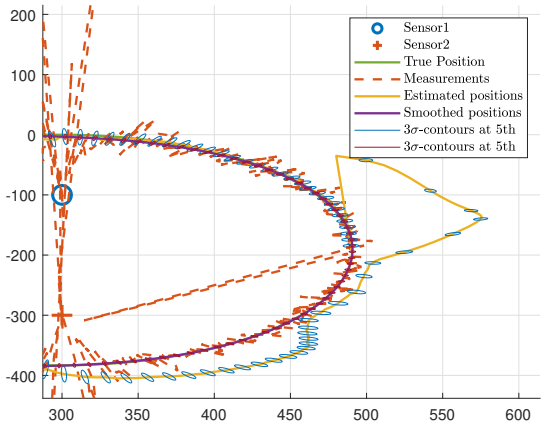


Figure 1.4: Enlarged picture introduced an outlier.

2 Particle filters for linear/Gaussian systems

(a) The mean square error (MSE) of the KF, the PF without resampling and the PF with resampling with different number of particle are illustrated in table 1. By comparing the MSE of the two PFs and the KF, we can find that the PF with resampling can perform approximately as well as the KF with a small number of the particle. While we need more than 100 particles in order for the two PFs to perform approximately as well as the KF.

This implies that Particle Filtering (PF) without resampling may result in significantly larger errors compared to Kalman filtering or PF with resampling, especially when dealing with a small number of particles. While increasing the number of particles can indeed enhance the accuracy of estimation without resampling, it unavoidably escalates computational complexity. Therefore the using of resampling in the Particle filter will be very helpful.

Table 1: Mean square error

	N = 20	N = 50	N = 100	N = 1000
KF	1.9944	0.8865	1.2362	1.1012
PF without resampling	7.2044	1.5405	1.8274	1.1987
PF with resampling	1.6670	0.9618	1.2596	1.1255

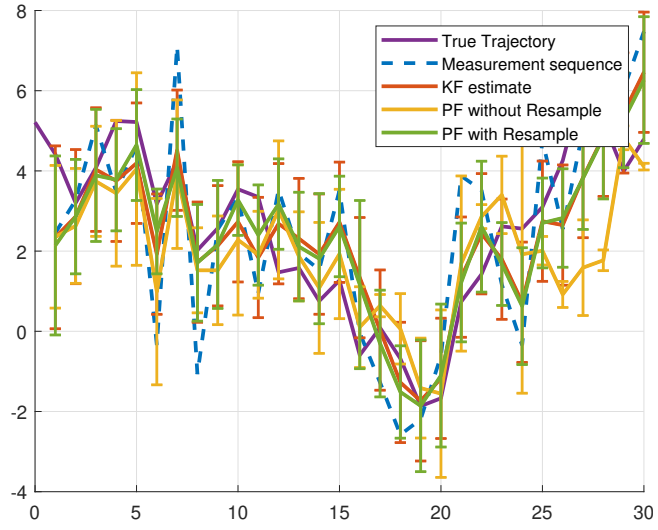


Figure 2.1: Two PFs with the KF.

The Figure 2.1 shows the true trajectory and the measurement sequence as well as $\mathbb{E}[x_k|y_{1:k}]$ and $\mathrm{Cov}[x_k|y_{1:k}]$ for the three filters over time.

We choose three time instance: one time step near the beginning(k = 1), one time step near the end(k = 28) and one time step somewhere in between(k = 14) to illustrate the approximations to the posterior densities. The results of the PF without resampling shown as in Figure2.2, Figure 2.3 and Figure 2.4. The results of the PF with resampling shown as in Figure2.5, Figure 2.6 and Figure 2.7.

Form the figures, we can find that the PF with resampling performs similar as the KF. This suggests that the Particle Filter (PF) with resampling estimates positions with a level of accuracy comparable to that of the Kalman Filter (KF). While the PF without resampling performs not similar as the KF, it has a smaller covariances. It approximates the mean with higher probability, even though the quality of the estimation is the worst among all three filters.

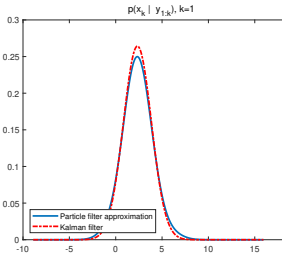


Figure 2.2: K = 1 Without Resampling.

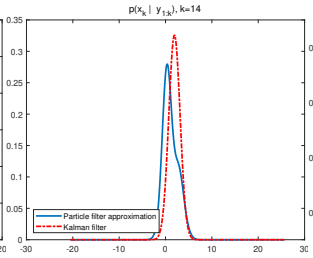


Figure 2.3: K = 14 Without Resampling.

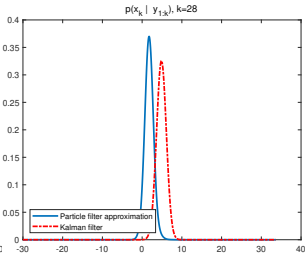


Figure 2.4: K = 28 Without Resampling.

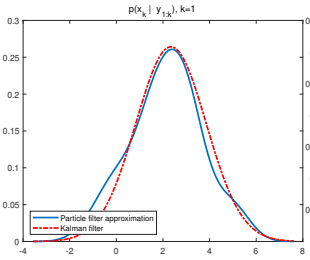


Figure 2.5: K = 1 With Resampling.

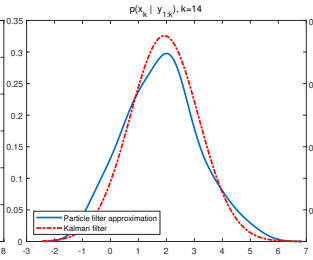


Figure 2.6: K = 14 With Resampling.

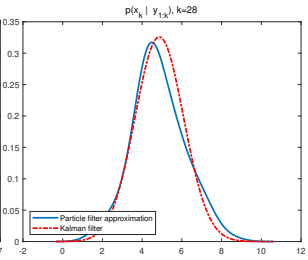


Figure 2.7: K = 28 With Resampling.

(b) We run the three filters with incorrect initial prior $p(x_0) = \mathcal{N}(x_0; -20, 2)$, and the results illustrated as in Figure 2.8. We can find that it takes KF 5 timesteps to converge to the true trajectory. It takes the PF with resampling 10 timesteps to converge to the true trajectory. Thus the KF converges to the true trajectory faster than the PF with resampling. While the PF without resampling can not approach to the true trajectory.

The Kalman filter (KF) excels in linear Gaussian systems, swiftly correcting its estimations and converging to the true trajectory even in the presence of initial incorrect priors. On the other hand, the Particle Filter (PF) with Resampling can also refine its estimations over time, albeit requiring more iterations to converge to the true trajectory. The PF without resampling fails to approach the true trajectory altogether. This outcome likely stems from the particle degeneracy issue, where a few particles dominate the representation of the belief distribution, leading to a loss of diversity and accuracy in estimation.

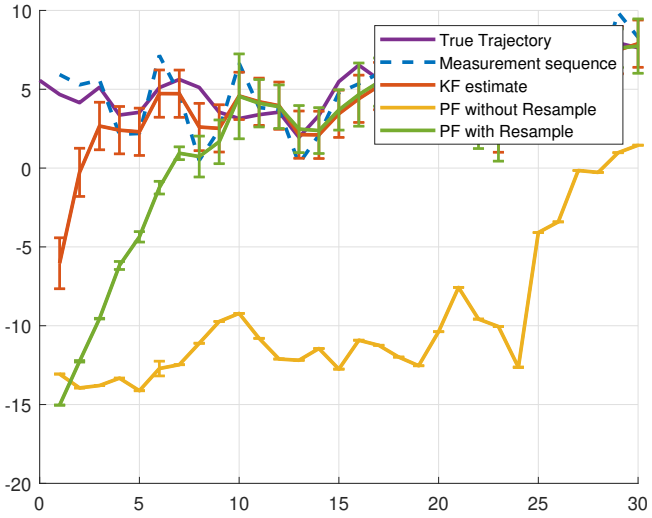


Figure 2.8: Three filters with incorrect initial prior.

(c) The Figure 2.9 illustrates the particle trajectories for a PF without resampling along with the true trajectory and the estimation in the same figure.

In the absence of resampling, particles tend to gradually disperse relative to the true trajectory because of the inherent uncertainty linked to the motion trajectory. Subsequent updates during the filtering process lead particles to gradually drift away from the true trajectory, consequently diminishing their weights. As a result, this can lead to a decline in the performance of the Particle Filter.

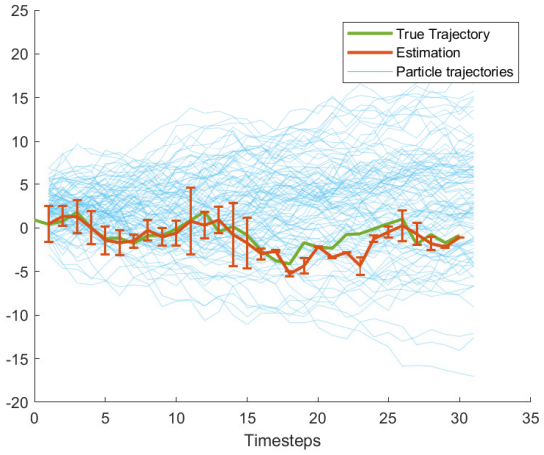


Figure 2.9: Particle trajectories for a PF without resampling.

(d) The Figure 2.10 illustrates the particle trajectories for a PF with resampling along with the true trajectory and the estimation in the same figure.

From the figure, we can find that the particle trajectories always stay close to the true trajectory. During the resampling step, particles with higher weights have a higher probability of being selected multiple times. This ensures particle diversity and more accurately represents the shape and characteristics of the posterior distribution, thereby enhancing the performance of the particle filter.

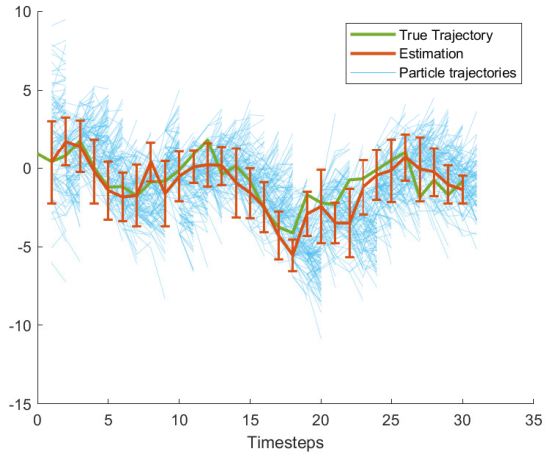


Figure 2.10: Particle trajectories for a PF with resampling.

3 Bicycle tracking in a village

(a) The Figure 3.1 illustrates the position sequence x_k^p generated by the Matlab function *MapProblemGetPoint.m*.

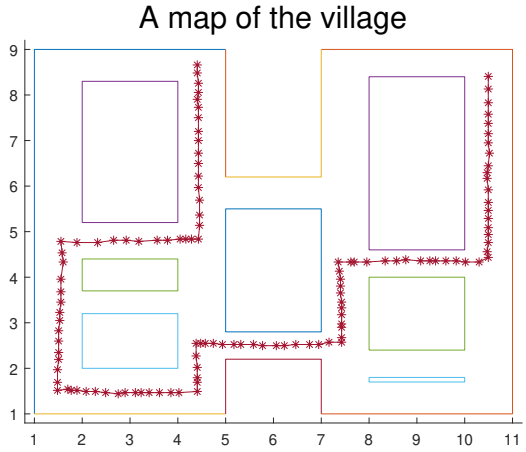


Figure 3.1: Generated position sequence.

(b) The Figure 3.2 illustrates velocity measurements sequence: $x_k^v = x_k^p - x_{k-1}^p$.

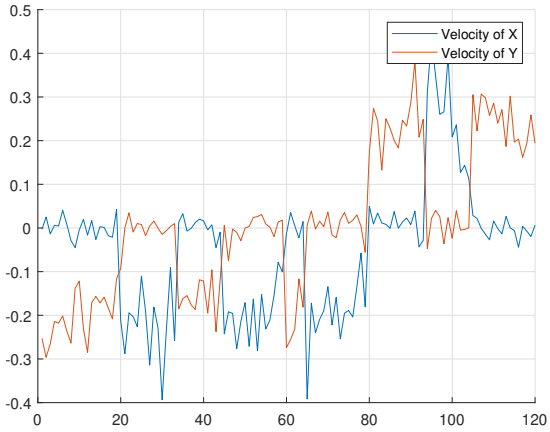


Figure 3.2: Generated velocity measurements.

(c) In this Task, we choose the constant velocity model (CV model) as our motion model shown as in equation (1).

$$\begin{bmatrix} p_k \\ v_k \end{bmatrix} = \begin{bmatrix} I & T \\ 0 & I \end{bmatrix} \cdot \begin{bmatrix} p_{k-1} \\ v_{k-1} \end{bmatrix} + q_{k-1} \tag{1}$$

Incorporating map information into the measurement model involves integrating it into the likelihood calculation, while including it in the motion model integrates map constraints into the particle generating step.

In terms of computational cost, incorporating map information into the measurement model is typically less expensive than integrating it into the motion model. Since the likelihood calculations is straightforward using the the map information. However, integrating map constraints into the motion model necessitates more complex computations to ensure that particles adhere to the constraints imposed by the map information.

(d) We choose the constant velocity model (CV model) as our motion model to construct the PF algorithm with the ability to track both the position and the velocity of the bicycle. The estimated position of the bicycle shown as in the Figure 3.3.

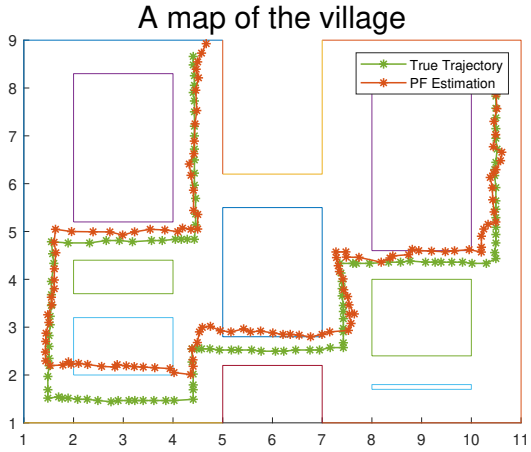


Figure 3.3: Position estimation.

The estimated velocity of the bicycle shown as in the Figure 3.4 and the Figure 3.5.

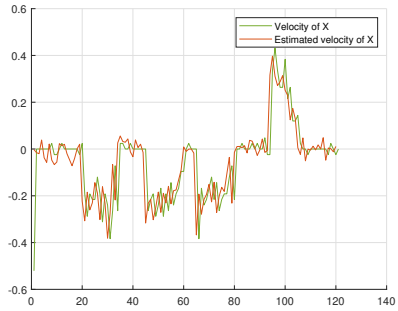


Figure 3.4: Velocity estimation of X.

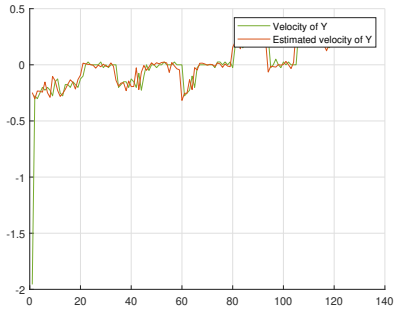


Figure 3.5: Velocity estimation of Y.

From the figures, we can find that the filter performs well. Therefore, our PF algorithm has the ability to track both the position and the velocity of the bicycle.

(e) As motivated in the 2b), the PF with resampling can converge to the true trajectory with incorrect initial position. So if we have no knowledge about the bicycle's initial position, we can set the initial position as a random point. Since we know the map information, we can get the random positions on the road in the map shown as in Figure 3.6 and then pick up one random position on the road to be our initial position.

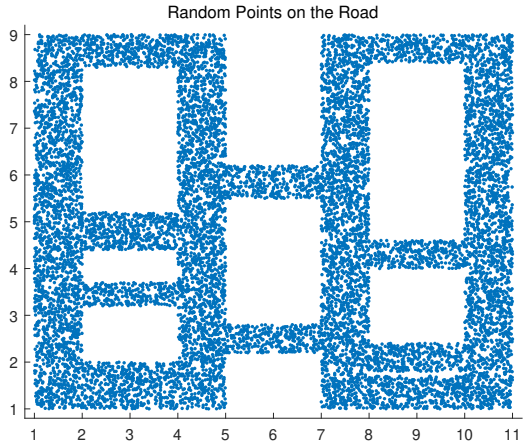


Figure 3.6: Random points on the road.

The estimation of the bicycle position without any knowledge about its initial position shown as in the Figure 3.7. We can find that the estimation trajectory can follow the true trajectory. But the filter is unable to position the bicycle on the road in the map.

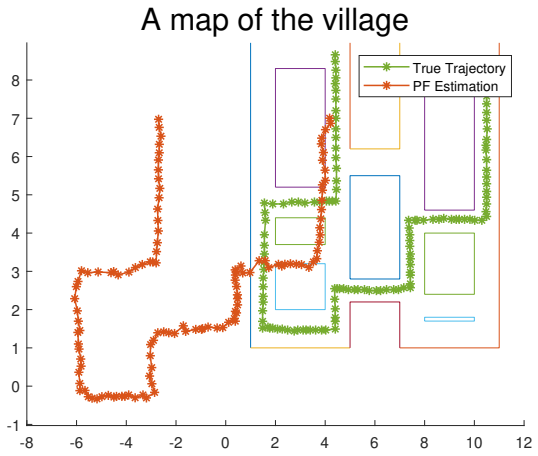


Figure 3.7: Position estimation with random initial position.

The estimation of the bicycle's velocity without any knowledge about its initial position shown as in the Figure 3.7. We can find that the velocity estimation fits the true velocity well.

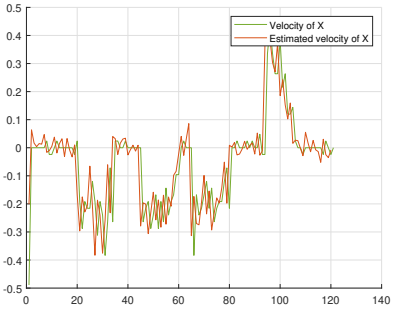


Figure 3.8: Velocity estimation of X with random initial position.

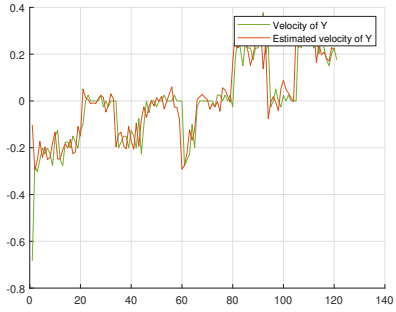


Figure 3.9: Velocity estimation of Y with random initial position.

Only the velocity measurements is not enough for the PF without any knowledge about the bicycle's initial position to position the bicycle. Our filter need more information to position the bicycle on the road in the map.

When a filter successfully rules out large chunks of particles, it usually occurs during the resampling step. This step aims to maintain diversity among particles while discarding poorly performing particles and replicating well-performing ones.