Solution to analysis in Home Assignment 4

QunZhang + qunz (cid)

Analysis

In this report I will present my independent analysis of the questions related to home assignment 4. I have discussed the solution with Lizi Teng, Tianyi Zhang,Mingxiang Zhao but I swear that the analysis written here are my own.

1 Smoothing

1.1 a

• As what did in the assignment 3, choosing the CKF as the favorite filter, the figure 1 is the plot of the CKF filter. As we can see, the covariance is a bit big at the beginning and the filter performance is not so good in the turning domain.However, after using the smoothing(backward), the curve becomes smoother and fits the real curve better, which can be seen in the figure 2.

$$\text{Backward: } \begin{cases} \mathbf{G}_k = \mathbf{P}_{k|k}\mathbf{A}_k^T\mathbf{P}_{k+1|k}^{-1} \\ \hat{\mathbf{x}}_{k|K} = \hat{\mathbf{x}}_{k|k} + \mathbf{G}_k\left(\hat{\mathbf{x}}_{k+1|K} - \hat{\mathbf{x}}_{k+1|k}\right) \\ \mathbf{P}_{k|K} = \mathbf{P}_{k|k} - \mathbf{G}_k\left[\mathbf{P}_{k+1|k} - \mathbf{P}_{k+1|K}\right]\mathbf{G}_k^T \end{cases} \tag{1}$$

• As is shown in the figure 3, which is the position error of CKF filter and smoothing. Smoothing techniques help in refining the estimated states by incorporating information from both past and future measurements. This leads to a more accurate estimation of the system's true state, reducing the effects of measurement noise and improving overall accuracy. In the figure

3, when the vehicle is turning(around 200s and 400s), the CKF is quite noisy and covariance is high. However, after smoothing, the error decrease obviously.

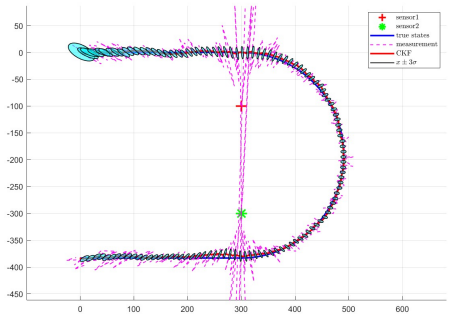

Figure 1: plot with CKF filter

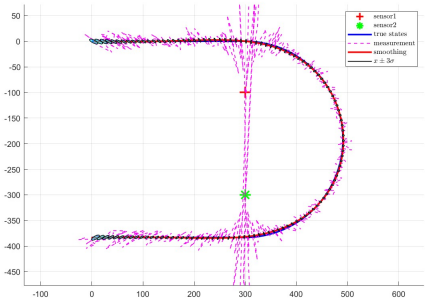

Figure 2: plot with using smoothing

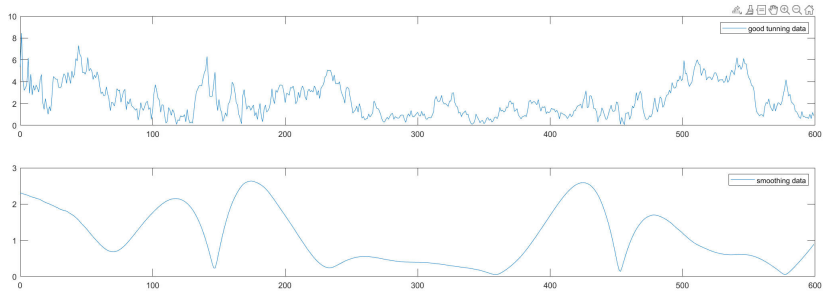

Figure 3: Differences between filtering and smoothing error covariances

1.2 b

At k=300, adding a noise into the measurement, the filter performs really bad at that time in the figure 4. However, after the smoothing, the noisy deviation is reduced, as is shown in the figure 5.
That is because smoothing increases robustness to Measurement Noise: Smoothing techniques take advantage of multiple measurements over a time window, which helps in mitigating the adverse effects of measurement noise. By averaging or weighting measurements, the impact of individual noisy measure-

ments is reduced, resulting in more reliable and robust state estimates.
As is shown in the figure 6, due to using the whole measurement and motion model, even through affecting by the outlier, the smoothing still performs better than the CKF-the curve becomes smoother and fits the real curve better.

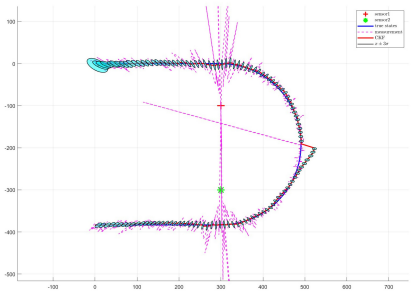

Figure 4: plot with CKF filter

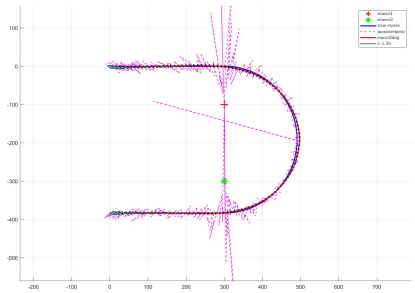

Figure 5: plot with using smoothing

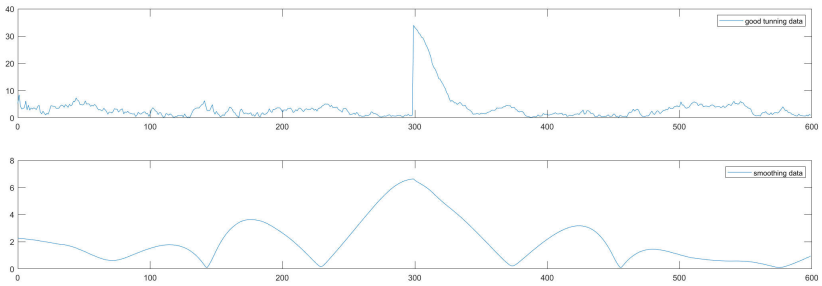

Figure 6: Differences between filtering and smoothing error covariances

2 Particle filters for linear/Gaussian systems

2.1 a

2.1.1

As is shown in the figure, the PF without resampling still performs bad and can not fit the kalman curve until we add particles to 10000. However, the PF with resampling performs very well. And related MSE could be shown

in the table 1.
It means that PF without resampling will lead to large error compared with kalman filter or PF with resampling when the number of particles is small. However, if we increase the particles of PF without resampling to get a accurate result, that will lead to a high computational complexity. Thus, resampling is fairly necessary in PF and can help avoid Particle Degeneracy and improve Estimation Accuracy.

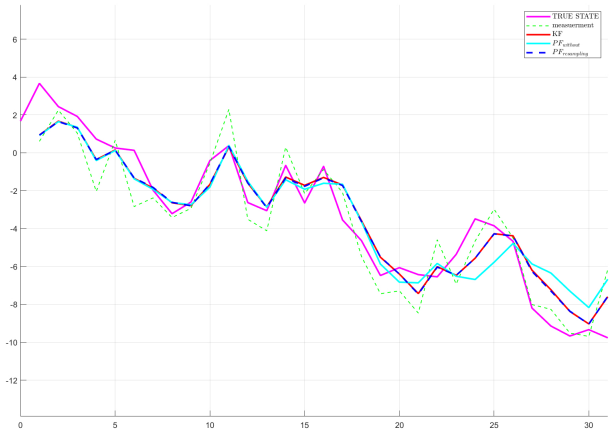

Figure 7: Comparison of KF, PF with resampling and PF without resampling

	N = 25	N = 100	N = 4000	N = 10000
KL	1.3922	1.3922	1.3922	1.3922
PF-no resample	7.6872	3.5696	2.2221	1.3122
PF-resample	1.5204	1.4160	1.3763	1.3864

2.1.2

As is shown in the figure 8, which is the comparison of Errorbar with KF, PF with resampling and PF without resampling. From the picture, we can see that the PF without resampling performs worse than the other two.

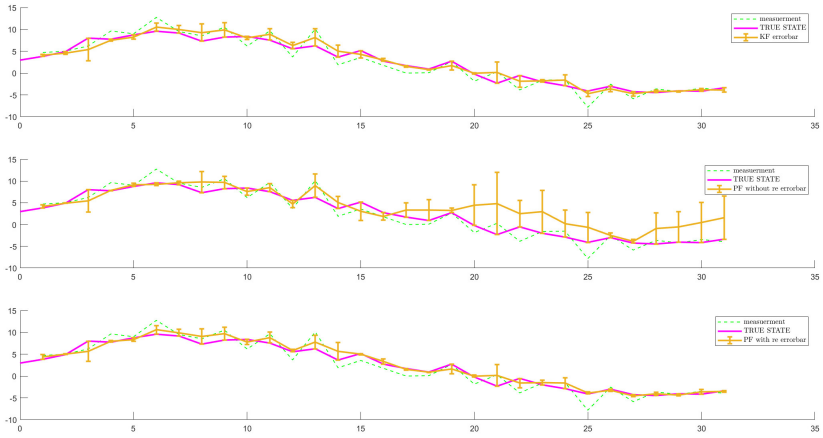

Figure 8: ErrorBar of KF, PF with resampling and PF without resampling

2.1.3

As is shown in the figure 10, which are the posterior densities of those two PFs. After tunning the without resampling PF sigma kernel sizeas 0.5, 0.8, 0.5, it will get a reasonable illustration as PF with resampling.

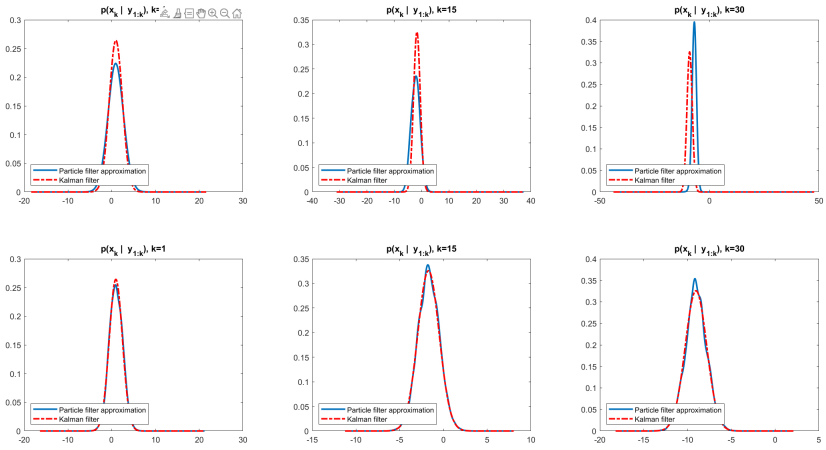

Figure 9: posterior densities of PF with resampling and PF without resampling with 4000 particles without tunning, pictures above are without resampling, pictures below are with resampling

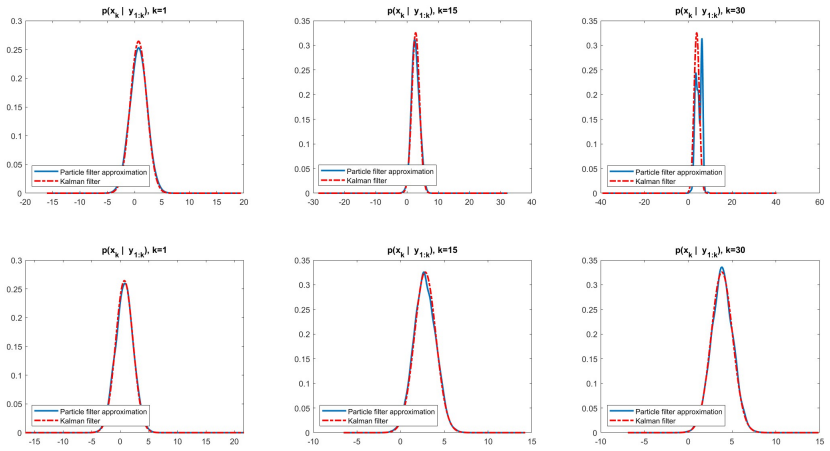

Figure 10: posterior densities of PF with resampling and PF without resampling with 4000 particles, pictures above are without resampling, pictures below are with resampling

Decreasing the kernel size would result in narrower and sharper representations of individual particles in the visualization. This can provide a more detailed view of the particle locations and their contribution to the posterior density.
PF without resampling, with the decreased kernel size, produces a plot that closely matches the KF plot, it suggests that the PF is estimating the true state with a similar level of accuracy as the KF. The PF is effectively capturing the dynamics of the system and incorporating the available measurements to generate accurate state estimates, comparable to those of the KF.
However, the performance of PF highly related to the number of particles. If the number of particles is low, even though we tune it (such as 100 particles), it will not get a expected result like resampling one.

2.2 b

As is shown in figure 11, PF with resampling can recover and provide reasonable state estimates over time, as resampling allows accurate particles to propagate and survive, mitigating the impact of the incorrect prior.
On the other hand, PF without resampling may struggle to recover from an incorrect prior as particles representing less accurate hypotheses persist

without corrective adjustments. This can lead to poor performance and divergence from the true state.
In summary, the PF with resampling has better potential to handle a wrong prior compared to the PF without resampling.

Figure 11: PF with resampling and PF without resampling with wrong prior

2.3 c, d

Figure 12 illustrates the behavior of the Particle Filter (PF) without resampling, while Figure 13 represents the PF with resampling. In the absence of resampling, particles tend to disperse relative to the true trajectory due to the inherent uncertainty associated with the motion model's covariance. Subsequent updates in the filtering process cause particles to deviate from the true state, resulting in lower weights assigned to particles that are distant from both the true state and the measured data. This phenomenon, known as degeneracy, significantly undermines the filter's performance.
In contrast, resampling effectively tackles the degeneracy problem. Particles with lower weights are more likely to be replaced by particles with higher weights during the resampling step, leading to a balanced distribution of weights at each time step. Consequently, the resampled particles consistently maintain close proximity to the true state, thereby accurately representing the probability distribution of the true state.
Through the incorporation of resampling, the PF with resampling addresses the degeneracy issue, ensuring that the particles remain informative and representative of the true state. This improvement in particle distribution

enhances the filter's performance and enables more accurate estimation of the true state.

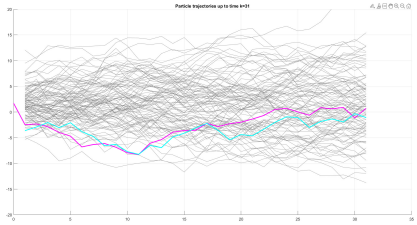

Figure 12: PF without resampling, pink one is true trajectory, cyan one is filter

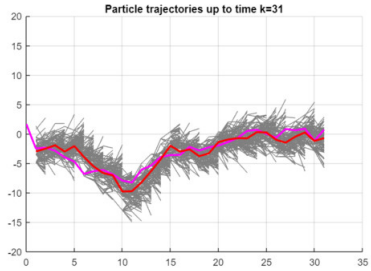

Figure 13: PF with resampling, pink one is true trajectory, red one is filter

3 Bicycle tracking in a village

3.1 a

As is shown in figure 14, it is the trajectory generated by M apP roblemGetP oint.m.

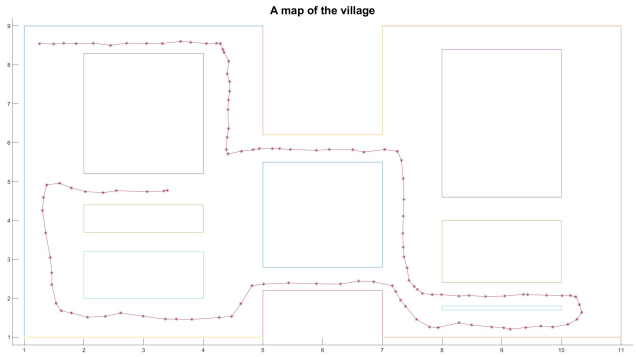

Figure 14: trajectory generated by M apP roblemGetP oint.m

3.2 b

As is shown in figure 15, it is the figure of this velocity measurement sequence.

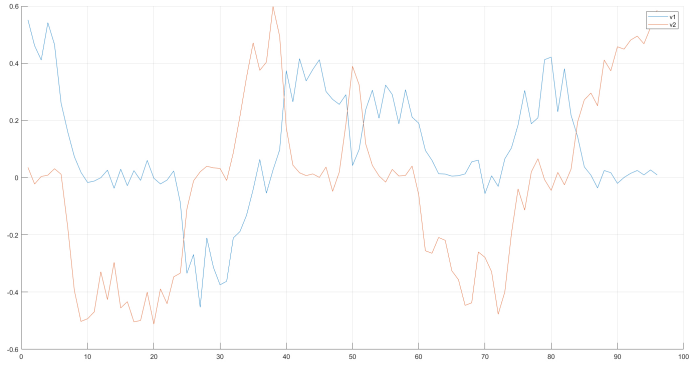

Figure 15: velocity measurement sequence

3.3 c

In the filter, the motion model is fairly important, and we can choose CV model(equation 2) and CT model(equation 3) as our motion model. The CV model will be cheaper compared with CT model, because CT model lead to higher computational complexity. However, CT model will lead more accurate result, which will be shown in question d.

$$\begin{bmatrix} p_k \\ v_k \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}}_{\tilde{A}} \begin{bmatrix} p_{k-1} \\ v_{k-1} \end{bmatrix} + \tilde{\mathbf{q}}_{k-1} \tag{2}$$

$$\underbrace{\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{v}(t) \\ \dot{\phi}(t) \\ \dot{\omega}(t) \end{bmatrix}}_{\dot{\mathbf{x}}(t)} = \underbrace{\begin{bmatrix} v(t)\cos(\phi(t)) \\ v(t)\sin(\phi(t)) \\ 0 \\ \omega(t) \\ 0 \end{bmatrix}}_{\tilde{\mathbf{a}}(\mathbf{x}(t))} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}}_{\Gamma} \underbrace{\begin{bmatrix} q_c^v(t) \\ q_c^\omega(t) \end{bmatrix}}_{\mathbf{q}_c(t)} \tag{3}$$

$$\begin{bmatrix} x_k \\ y_k \\ v_k \\ \phi_k \\ \omega_k \end{bmatrix} = \begin{bmatrix} x_{k-1} + Tv_{k-1}\cos(\phi_{k-1}) \\ y_{k-1} + Tv_{k-1}\sin(\phi_{k-1}) \\ v_{k-1} \\ \phi_{k-1} + T\omega_{k-1} \\ \omega_{k-1} \end{bmatrix} + \mathbf{q}_{k-1} \tag{4}$$

Adding map information to the measurement model involves incorporating it into the likelihood calculation, while incorporating it into the motion model integrates the map constraints into the particle propagation step.
In terms of computational cost, adding map information to the measurement model is generally cheaper compared to incorporating it into the motion model. This is because updating the likelihood typically involves straightforward calculations using the map information, whereas integrating map constraints into the motion model requires more complex computations to ensure that particles adhere to the spatial constraints imposed by the map (a bit like constraint model predict control or constraint programming, and those contraints need to keep changing).
To add the map information into the PF filter as measurement information, we can use the bayes rule, as is shown in the equation 4. We can add the info into measurement model or motion model.

$$
\begin{aligned}
p(x_k|Y,M) &= \frac{p(M|x_k,Y) * p(x_k|Y)}{p(M|Y)} \\
p(x_k|Y,M) &\propto p(M|x_k) * p(x_k|Y)
\end{aligned}
\tag{5}
$$

3.4 d

To construct a PF algorithm, we need to choose the motion model. As is shown in figure 16-19, they are the comparison of CV model and CT model.

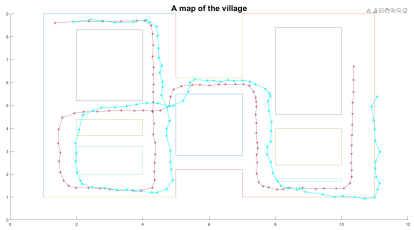

Figure 16: tuned CV model without map info

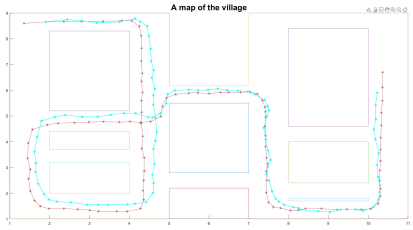

Figure 17: tuned CV model without map info

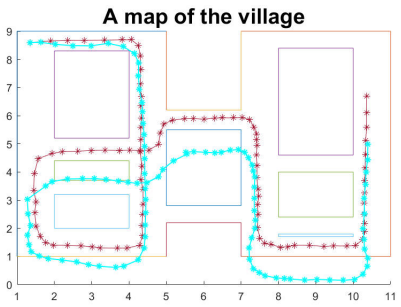

Figure 18: tuned CT model without map info

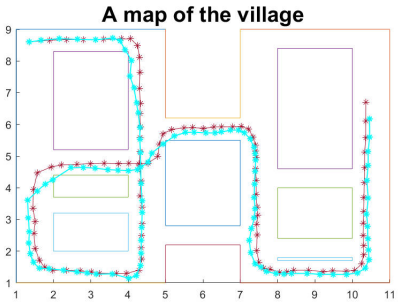

Figure 19: tuned CT model with map info

From the picture, we can get that CT model, we can see that the CT model performs really especially around the corner. Even through CT model will take more computational resource, CT will get more accurate result. We choose the CT model as our motion model, with the characters shown below:

$$\sigma_r = 0.05,\ \sigma_v = 0.1,\ ,\sigma_w = \frac{10 * \pi}{180} \tag{6}$$

As is shown in the figure 20 (every 10 time step) and figure 21, which are the iteration and some details of the particles. In the particles, black particles are those particles being given 0 weight by map information, because they are inside the building. Those colorful particles left are particles with weights, and used to calculate the filtered value.

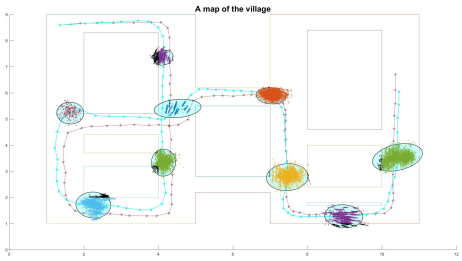

Figure 20: iteration of the CT PF algorithm

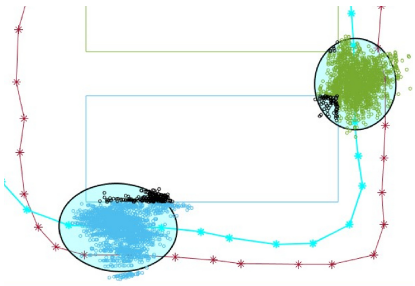

Figure 21: details of particles

3.5 e

without any knowledge about its initial position, we can use random function inside the matlab to generate a random value as our initial condition.

However, since we've known the information of the map, what we have to do is to generate points on the road and select one point randomly, as is shown in figure 22.

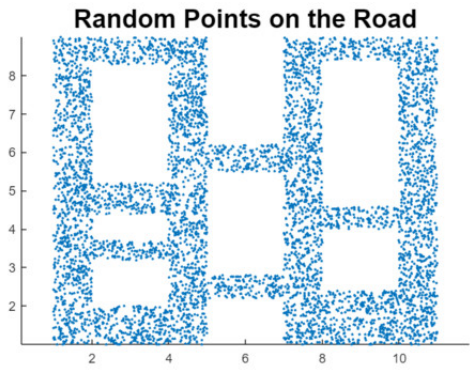

Figure 22: 10000 random points on the road

After randomly selecting a point, we tested the points, as shown in Figures 23-26, they are filtered trajectory without initial condition.

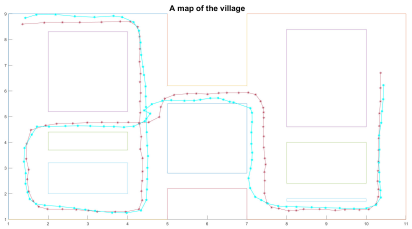

Figure 23: CT PF algorithm with initial condition near the correct position, without map

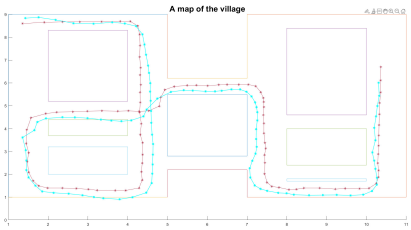

Figure 24: CT PF algorithm with initial condition near the correct position, with map

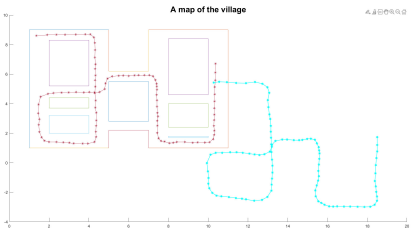

Figure 25: CT PF algorithm without initial condition, without map

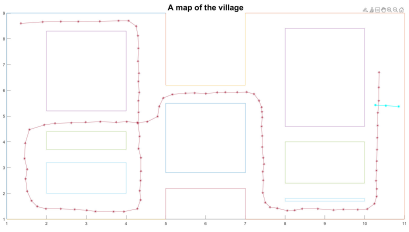

Figure 26: CT PF algorithm without initial condition, with map

If the starting point happens to be near the exact initial point, as shown in Figure 23, 24, the filter can complete the trajectory prediction, and the trajectory effect is good. However, if with wrong initial condition, the algorithm crashed when reached to the building or obstacle, as is shown in figure 26.

Using only velocity measurements may not provide sufficient information to accurately estimate the bicycle's position. Additional measurements such as position, orientation, or other sensor data should be considered to improve estimation accuracy.

Additionally, maybe adding another resampling in the crashed condition will help to solve such a problem. Unfortunately, I tried several times and did not succeed in avoiding the algorithm crash. All data becomes NAN after the moment of collision.

Thus, enough measurement and correct initial condition are fairly important in PF algorithm.