Solution to analysis in Home Assignment 3

QunZhang + qunz (cid)

Analysis

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

1 Approximations of mean and covariance

1.1 a

As is shown in the figure 1,2,3, which are the comparison of samples, EKF, UKF and CKF. As is shown in figure 1(a), 2(a) and 3(a), which are estimated by 10000 samples. Because he took a lot of samples to approximate, according to the law of large numbers, we think his mean and variance are very accurate and reliable.

1.2 b

The EKF algorithm(as is shown in equation 1) expands the nonlinear function by Taylor series, retains the first-order item to realize the linearization of the nonlinear function, and then uses the Kalman filter algorithm as a framework to calculate the state estimation value and variance of the system. As is shown in figure 1(b), 2(b) and 3(b), when the degree of nonlinearity is not so high, EKF perform well (figure 1(b)). However, if the nonlinearity is high, compared with sample estimate, it performs bad. That is because the

neglect of high-order terms, when the degree of nonlinearity is high or the initial error is large, the filtering accuracy will be greatly reduced.

$$\begin{aligned} \mathbb{E}\{\mathbf{y}\} &\approx \mathbf{g}(\hat{\mathbf{x}}) \\ \mathrm{Cov}\{\mathbf{y}\} &\approx \mathbf{g}'(\hat{\mathbf{x}})\mathbf{P}\mathbf{g}'(\hat{\mathbf{x}})^T \end{aligned} \tag{1}$$

	sample mean		EKF mean		UKF mean		CKF mean		sample covariance		EKF covariance		UKF covariance		CKF covariance	
case1	0.1986		0.1974		0.1983		0.1983		0.0017	0.0015	0.0017	0.002	0.0017	0.0015	0.0017	0.0015
	1.3746		1.3734		1.3743		1.3743		0.0015	0.006	0.0015	0.006	0.0015	0.0059	0.0015	0.0059
case2	2.3273		2.3562		2.3269		2.3265		0.0565	0.0104	0.05	0.01	0.06	0.0108	0.0566	0.0105
	2.355		2.3562		2.355		2.355		0.0104	0.002	0.01	0.002	0.0108	0.002	0.0105	0.002
case3	-0.5941		-0.588		-0.5949		-0.5948		0.0095	-0.011	0.0092	-0.01	0.0099	-0.0112	0.0097	-0.011
	2.1517		2.1588		2.1524		2.1523		-0.011	0.0149	-0.0111	0.015	-0.0112	0.0151	-0.0112	0.015

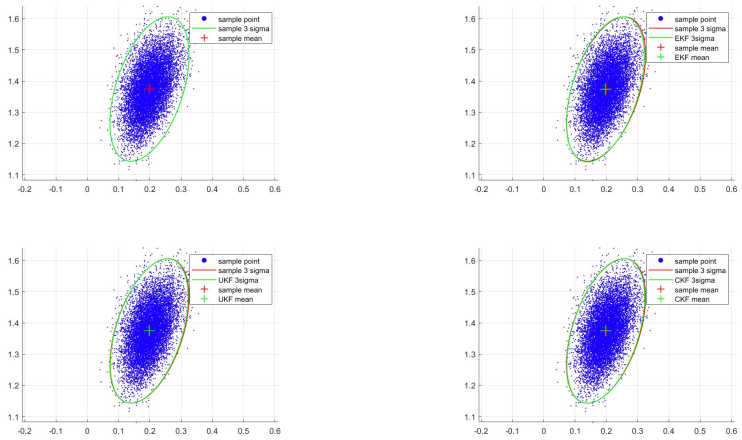

Figure 1: Comparison of samples,EKF, UKF, and CKF

As is shown in figure 1(c), 2(c) and 3(c), they are calculated by UKF. The UKF algorithm (as is shown in equation 2-3) uses the UT transformation to obtain the Sigma point set, and then passes through the nonlinear function to convert the linearization problem of the nonlinear function into an approximation of the probability density distribution of the system state quantity, and then realizes the filtering problem based on the Kalman algorithm framework. Compared with EKF, it performs well even on nonlinear

situation. Because UKF will not introduce linearization error, the accuracy can reach the second-order accuracy of Taylor series expansion, no need to calculate Jacobian matrix.

$$\begin{aligned}
&\mathcal{X}_{k-1}^{(0)} = \hat{\mathbf{x}}_{k-1|k-1} \\
&\mathcal{X}_{k-1}^{(i)} = \hat{\mathbf{x}}_{k-1|k-1} + \sqrt{\tfrac{n}{1-W_0}} \left(\mathbf{P}_{k-1|k-1}^{1/2} \right), \quad i = 1, 2, \ldots, n, \\
&\mathcal{X}_{k-1}^{(i+n)} = \hat{\mathbf{x}}_{k-1|k-1} - \sqrt{\tfrac{n}{1-W_0}} \left(\mathbf{P}_{k-1|k-1}^{1/2} \right)_i, \quad i = 1, 2, \ldots, n, \\
&W_i = \tfrac{1-W_0}{2n}, \quad i = 1, 2, \ldots, 2n
\end{aligned} \tag{2}$$

$$\begin{aligned}
\hat{\mathbf{x}}_{k|k-1} &\approx \sum_{i=0}^{2n} \mathbf{f} \left(\mathcal{X}_{k-1}^{(i)} \right) W_i \\
\mathbf{P}_{k|k-1} &\approx \mathbf{Q}_{k-1} + \sum_{i=0}^{2n} \left(\mathbf{f} \left(\mathcal{X}_{k-1}^{(i)} \right) - \hat{\mathbf{x}}_{k|k-1} \right) (\cdot)^T W_i
\end{aligned} \tag{3}$$

As is shown in figure 1(d), 2(d) and 3(d), they are calculated by CKF. The CKF algorithm (equation 4) selects cubature points through the volume rule, and then transfers these cubature points through a nonlinear function, and then weights the cubature points after the nonlinear function transfer to approximate the state posterior mean and covariance. As is shown in the comparison, in the nonlinear situation, it performs better than EKF too.

$$\begin{aligned}
\mathcal{X}_{k-1}^{(i)} &= \hat{\mathbf{x}}_{k-1|k-1} + \sqrt{n} \left(\mathbf{P}_{k-1|k-1}^{1/2} \right)_i, \quad i = 1, 2, \ldots, n, \\
\mathcal{X}_{k-1}^{(i+n)} &= \hat{\mathbf{x}}_{k-1|k-1} - \sqrt{n} \left(\mathbf{P}_{k-1|k-1}^{1/2} \right)_i, \quad i = 1, 2, \ldots, n, \\
W_i &= \tfrac{1}{2n}, \qquad\qquad i = 1, 2, \ldots, 2n \\
\hat{\mathbf{x}}_{k|k-1} &\approx \sum_{i=1}^{2n} \mathbf{f} \left(\mathcal{X}_{k-1}^{(i)} \right) W_i \\
\mathbf{P}_{k|k-1} &\approx \mathbf{Q}_{k-1} + \sum_{i=1}^{2n} \left(\mathbf{f} \left(\mathcal{X}_{k-1}^{(i)} \right) - \hat{\mathbf{x}}_{k|k-1} \right) (\cdot)^T W_i.
\end{aligned} \tag{4}$$

1.3 C

The required plot are shown in figure 1-3.

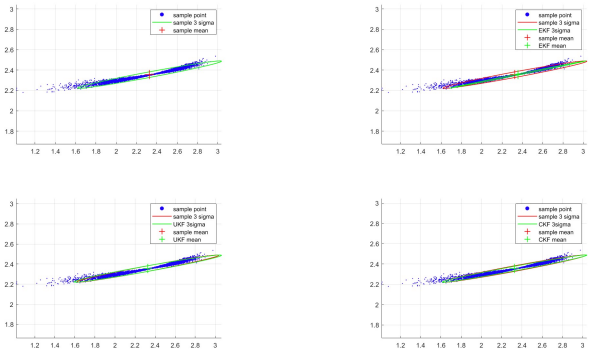

Figure 2: Comparison of samples,EKF, UKF, and CKF

1.4 d

In conclusion, EKF, UKF and CKF are all aimed at nonlinear Gaussian systems, and they all aim to solve the problem that Gaussian weighted integrals in nonlinear Gaussian filtering are difficult to obtain accurate analytical solutions(case 2 and case 3).(details of comparison seeing the 1.1b above)

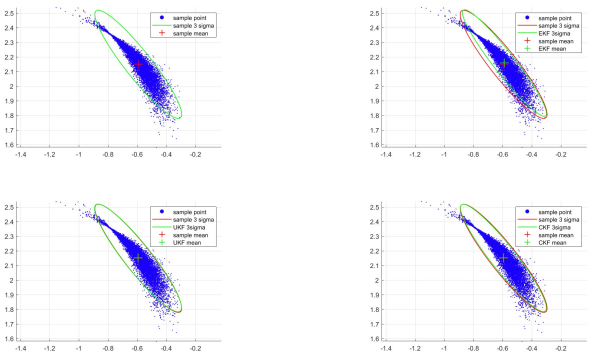

Figure 3: Comparison of samples,EKF, UKF, and CKF

However, because EKF ignores high-order terms, the accuracy is much

lower than UKF and CKF in the case of strong nonlinearity. The trade-off when using the UKF (Unscented Kalman Filter) or CKF (Cubature Kalman Filter) compared to the EKF lies in their ability to capture non-linearities more accurately. Both the UKF and CKF employ sigma-point methods, which involve selecting a set of representative points around the mean of the distribution and propagating them through the non-linear transformation. These methods provide a better approximation of the true non-linear behavior, resulting in improved estimates of the mean and covariance. However, these algorithms, including sampling estimation, try to fit through Gaussian estimation, so they cannot completely fit all points.

As an engineer, the choice of filter depends on the specific application and the characteristics of the system being modeled. If the system exhibits strong non-linearities, and accuracy is crucial, the UKF or CKF would be preferred due to their better handling of non-linear transformations. However, it's important to consider the computational complexity of these filters, as they require more operations compared to the EKF. If real-time processing or limited computational resources are a concern, the EKF may still be a reasonable choice, especially if the non-linearities in the system are relatively mild and the linear approximation is sufficient for accurate estimation.

Additionally, the EKF is more stable, because the high dimension matrix calculation of the UKF will lead negative-defined matrix which makes our system collapsed. So, sometimes in the engineering domain, particle filter or EKF may be a better choice.

2 Non-linear Kalman filtering

2.1 a,b

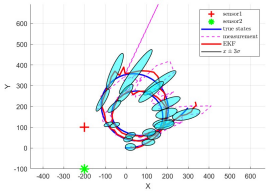

Figure 4: case 1 EKF

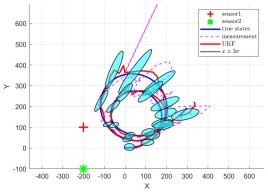

Figure 5: case 1 UKF

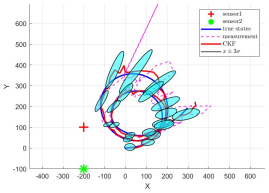

Figure 6: case 1 CKF

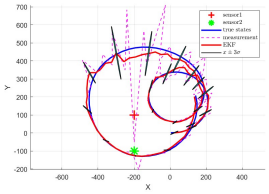

Figure 7: case 2 EKF

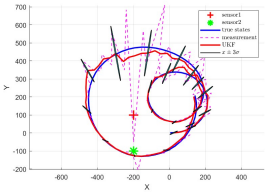

Figure 8: case 2 UKF

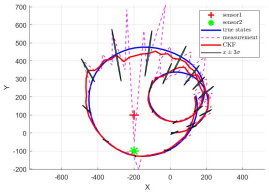

Figure 9: case 2 CKF

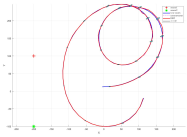

Figure 10: case 3 EKF

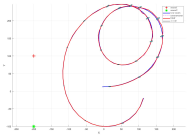

Figure 11: case 3 UKF

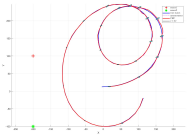

Figure 12: case 3 CKF

As is shown in figure 4-12, they are comparisons of three filters in there cases. In figure 4-9(case 1 and case 2), the measurement noise is quiet large which make measurement perform very bad. For case 1, due to large noise affect on both sensors, measurement of two directions are not good. For case 2, the first direction performs bad. However, for case 3, since the noise is small, the measurement perform better than both case 1 and case 2.
Additionally, When two such sensors measure angles towards the same object, the intersection of the lines described by the sensor positions and the measured angles is the (noisy) position of the object. In such a condition,

When two sensors are measured on a line, the measurement error will be particularly large.
However, after using filters, the filtered trajectories(EKF, UKF and CKF) of case 1 and case 2fit the real trajectories well and are all within 3 sigma levels, which means the error covariance represent the uncertainty very well. For the case 3, since its measurement noise is fairly small, after the filtering, the trajectories also perform well and successfully represent the uncertainty of the system.
It is hard to say which filter is better in those plots, because the scale here is very large, which makes it indistinguishable. However, from the data of the next question (mean and covariance), we can say the UKF and CKF behaviors better than EKF. UKF and CKF perform better than EKF because they can handle non-linear transformations more accurately, thanks to their sigma-point methods that capture non-linear behavior more effectively. This allows for improved estimation of mean and covariance compared to EKF linearization approximation.

2.2 c

As is shown in the pictures shown below, figure 13-30 describes the performance of different filters in different cases. The mean and covariance of UKF and CKF performs very well(mean and standard variance are shown above each figure), better than the EKF. The mean and covariance of EKF are obviously bigger than the UKF and CKF, which verifies its bad performance. The histogram can not fit with the Gaussian distributions.Because after the nonlinear distribution, it can not be norm distribution any more, but the mean keeps same.
In the CKF or UKF (we choose the filter that performs well), the histogram of x behaviors better than y in each case(smaller mean and covariance). And the performance of y is worse when the noise in the sensors become large.
That is because the position of sensors and its detect theory(sensors detect the intersection of lines) makes it can get better data in X direction.

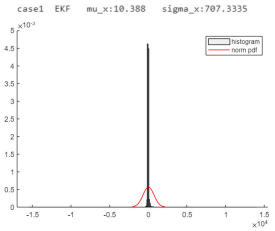

Figure 13: case 1 EKF x

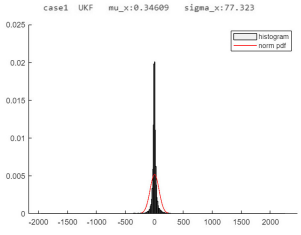

Figure 14: case 1 UKF x

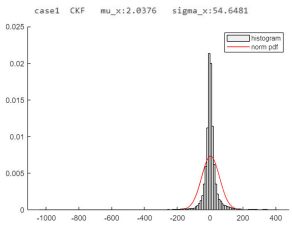

Figure 15: case 1 CKF x

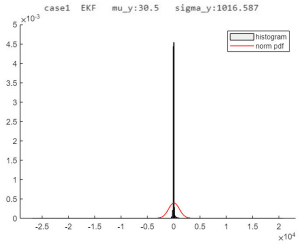

Figure 16: case 1 EKF y

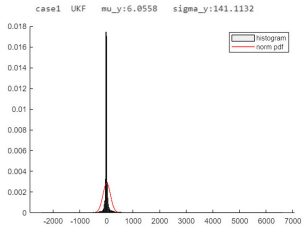

Figure 17: case 1 UKF y

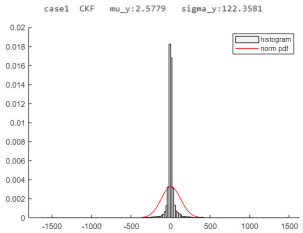

Figure 18: case 1 CKF y

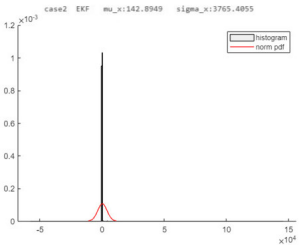

Figure 19: case 2 EKF x

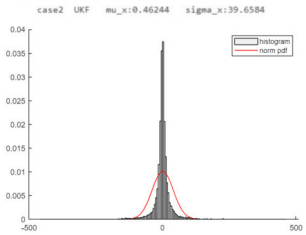

Figure 20: case 2 UKF x

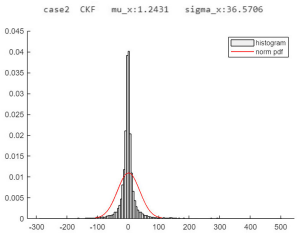

Figure 21: case 2 CKF x

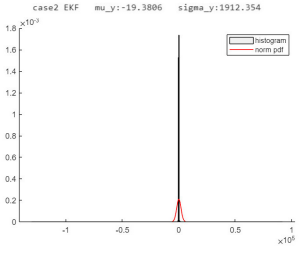

Figure 22: case 2 EKF y

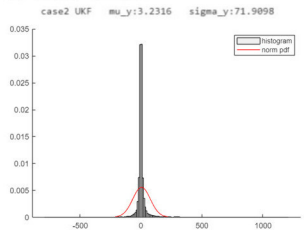

Figure 23: case 2 UKF y

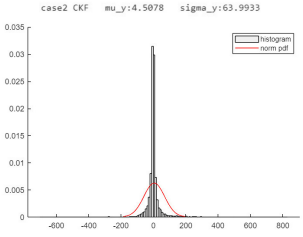

Figure 24: case 2 CKF y

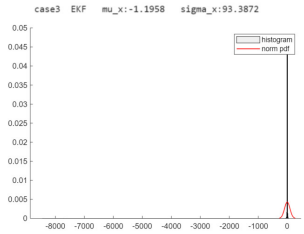

Figure 25: case 3 EKF x

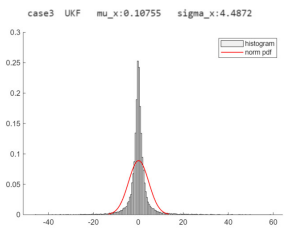

Figure 26: case 3 UKF x

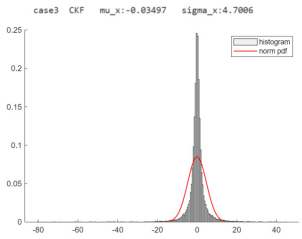

Figure 27: case 3 CKF x

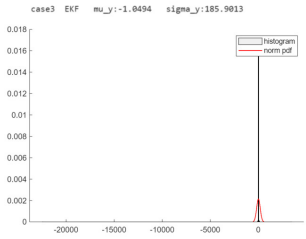

Figure 28: case 3 EKF y

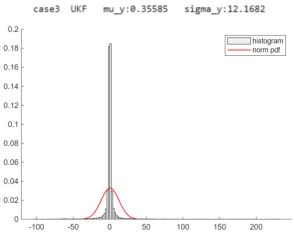

Figure 29: case 3 UKF y

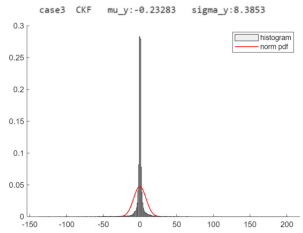

Figure 30: case 3 CKF y

3 Tuning non-linear filters

3.1 a

tuning σ_v large

The filtered data becomes very noisy, because our object is moving at a constant speed, and the only relevant changes occur when turning. Adding noise in the v direction makes our velocity change too much, and makes the filter tend to believe noisy measurements, making the final data very poor.

tuning σ_w large

The noise in the w direction is increased, so that our system can get better results when turning. But the object itself is not always turning, which deviates from the actual model, and the filter will not get good results.

tuning σ_v and σ_w large

In such a condition, the filter tend to trust measurement. However, since the measurement is noisy, we will get a strange and unrealistic result.

tuning σ_v small

Because we know in advance that the object is moving at a constant speed, the smaller the noise of v, the better the filtering results we can obtain.

tuning σ_w small

When an object leaves a straight line and enters a curve, the filter does not change the yaw rate as quickly as desired. A side effect of reducing the noise of w is that the filtered turning radius is much larger than it actually is.

tuning σ_v and σ_w small

If tuning σ_v and σ_v small, Filter tends to trust the motion model. However, such a model can not model the change of turning, which will deviate from our actual trajectory when turning, and simulate a larger turning radius.

3.2 b

As is shown in the figure 33, which is a good tuning result with $\sigma_v = 0.1$ and $\sigma_w = \pi/180$.We can see that the curve of filter fites the real states very well.

3.3 c

As is shown in figure 31-34, there are no changes scaling 100 for both, good tuning and scaling 0.1 for both respectively. Compared with figure 31, we can see the curve is quiet noisy, because the filter tend to trust the noisy measurement. From the figure 34, we can see that there is considerable delay when cornering, resulting in a much larger turning radius than it actually is while the process noise is low. For the well tuning, $\sigma_v = 0.1$, it performs better than all of them.

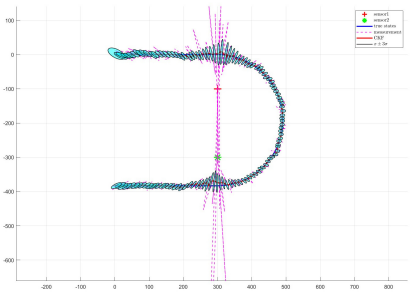

Figure 31: no change

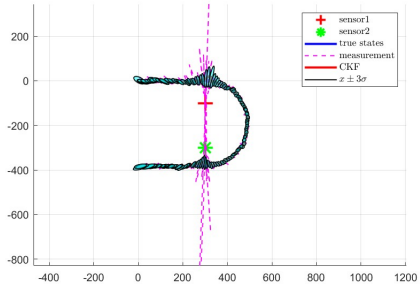

Figure 32: scaling=100 for both

From Figure 35, we can also observe that the well-tuned model is capable of maintaining low and largely stable errors in all regions.
Therefore, we can confidently conclude that the filter performs exceptionally well. It not only maintains low error during steady state on both straight lines and curves but also exhibits no lag during acceleration and state changes.

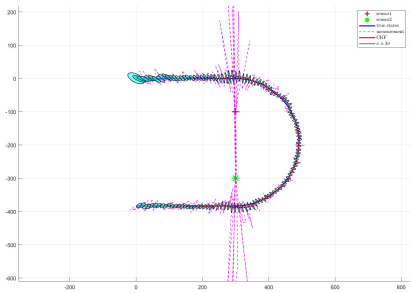

Figure 33: scaling=0.1 for v(good tuning)

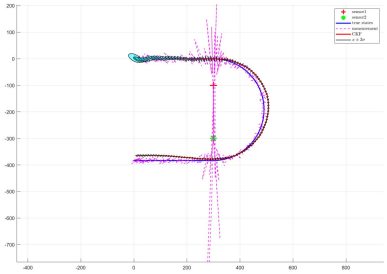

Figure 34: scaling=0.1 for both

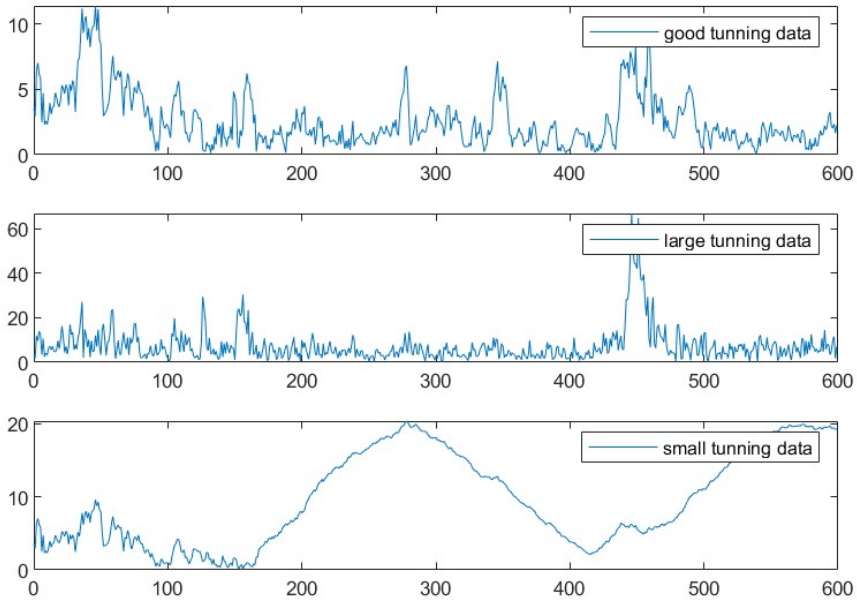

Figure 35: plot of $||P_k - \hat{P}_{K|K}||$

3.4 d

It is challenging to tune the filter to have accurate estimates of velocity, heading, and turn-rate for the entire sequence. The reason is that different parts of the true trajectory have different characteristics and require different tuning parameters.Or we can say it is not possible to do it with just tuning characters.

As what we mentioned above, linear constant velocity requires low σ_v and σ_w. However, when we turning, we want higher σ_v and σ_w to capture the dynamics and uncertainties associated with the turning motion. This is a conflict, Unless our filter can automatically adjust parameters according to different scenes, otherwise this is an impossible task.

In summary, tuning the filter for different parts of the true trajectory poses a challenge due to the varying dynamics and uncertainties involved. It is not feasible to have a single parameter setting that can provide accurate estimates for the entire sequence. The tuning parameters must be adjusted according to the specific characteristics of each segment, such as straight lines, turns, and transitions, in order to achieve accurate predictions.