SSY191 - Sensor Fusion and Nonlinear Filtering
Solution to analysis in Home Assignment 03

Lucas Rath

Analysis

In this report I will present my independent analysis of the questions related to home assignment 1. I have discussed the solution with Josip, Vide and Carl-Johan but I swear that the analysis written here are my own.

1 Approximations of mean and covariance

A, B, C, D) The samples of y, the sample mean/covariance, and the approximated means/covariances that used in the three different Kalman filters can be seen in Figures 1, 2 and 3.

Since we have used a very high number of samples: N=10000, and due to the law of large numbers we can expect that the sample mean/covariance are a very good approximation and we can consider them as the "true" values. Once this assumption was made, we can compare the performance of the approximations used in the three different Kalman filters.

It is clear that all the three filters perform very well approximating the mean and covariance of the y distribution for the first case when $x \sim p_1(x)$. The reason is that y has a distribution, which we know is not Gaussian due to the non-linear transformation, but it is very close from being Gaussian. What happens in this case is that the nonlinear transformation h(x) behaves very well and is fairly linear around the region of largest mass of $p_1(x)$. This makes the job much easier for all three filters. Even the EKF is able to approximate very well the mean and covariance due to the explained linearity around the desired operating point, which is $E[x] = \hat{x}_1$.

On the other hand, when we consider x distributed as $p_2(x)$, we notice a big difference between the three filters. Indeed, the function h(x) is very non-linear around the new operating point $\hat{x}_2$ and that is why the EKF performs so bad while estimating the mean and covariance. Considering now the sigma-point methods, the UKF and CKF filters perform much better approximating this non-linear transformation. No big difference between CKF and EKF can be seen in the plots.

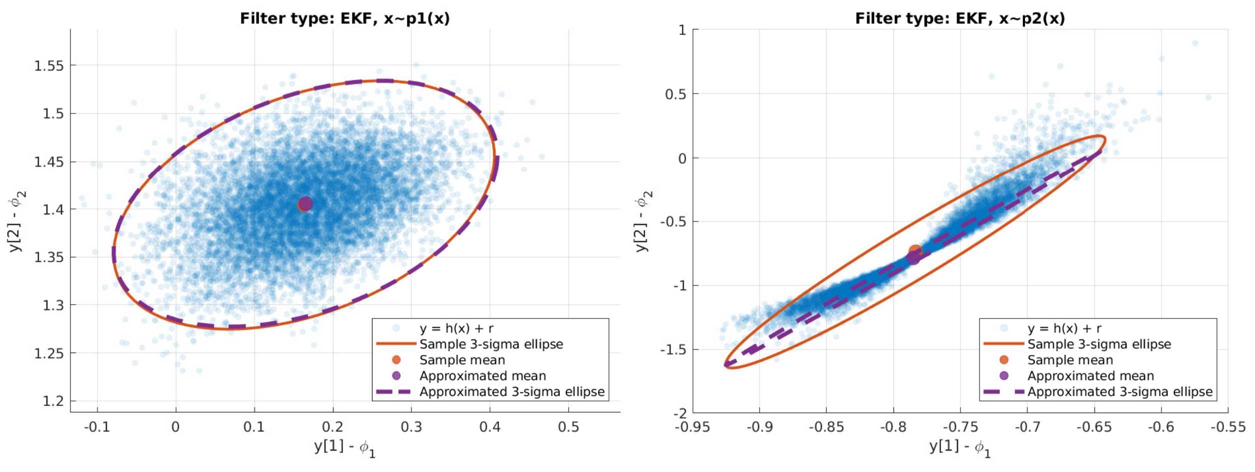

Figure 1: *Approximation of mean and covariance of the transformed distribution y=h(x)+r using the Extended Kalman filter method for two different Gaussian distributions of x.*

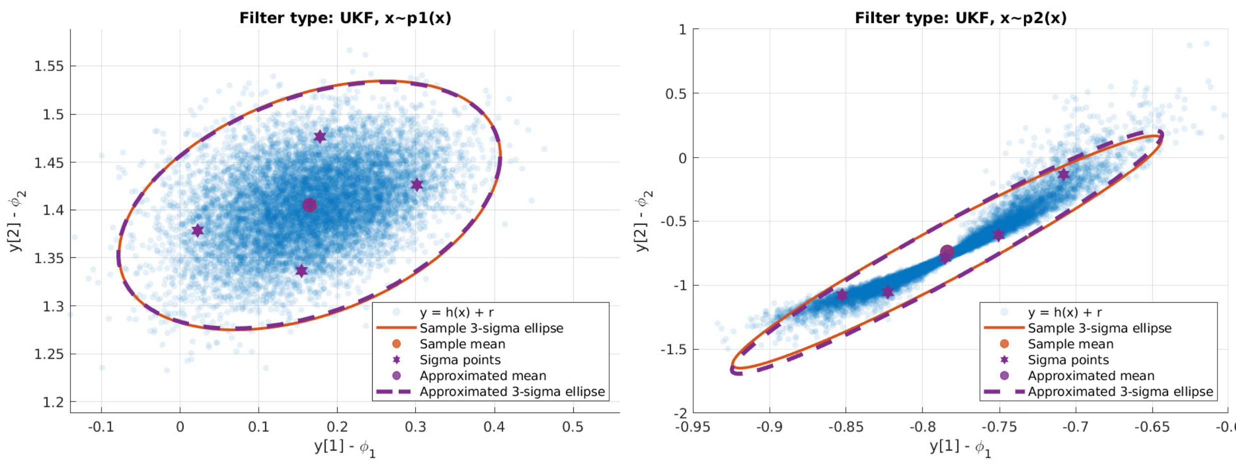

Figure 2: *Approximation of mean and covariance of the transformed distribution y=h(x)+r using the Unscented Kalman filter method for two different Gaussian distributions of x.*

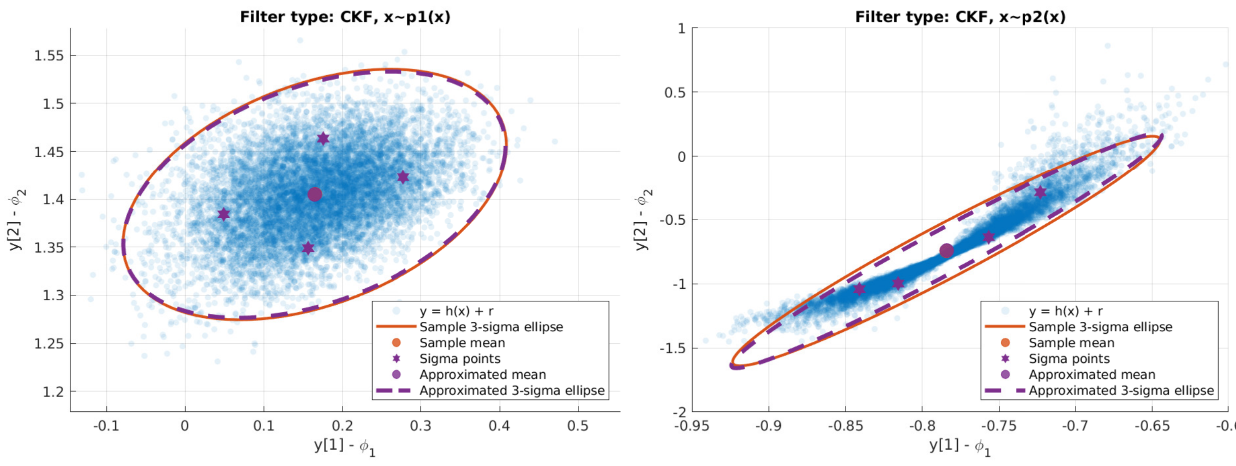

Figure 3: *Approximation of mean and covariance of the transformed distribution y=h(x)+r using the Cubature Kalman filter method for two different Gaussian distributions of x.*

E) In fact, it is only a good idea to approximate the transformed distribution as Gaussian, when the transformation function h(x) is linear around the region of large mass of the distribution $p(x)$. In this way, y=h(x) will be very similar to Gaussian if x is Gaussian. We can see this phenomena when analyzing the first distribution $x \sim p_1(x)$ for the three cases in Figures 1, 2 and 3. Even though h(x) is a non-linear transformation, it behaves well around $\hat{x}_1$ and therefore y can be well approximated as Gaussian.

On the other hand, for the second distribution $x \sim p_2(x)$, h(x) is very non-linear around $\hat{x}_2$ and therefore y does not look like Gaussian. We confirm this by looking at the samples, which do not look like being Gaussian distributed, since a good portion of the samples are distributed outside the 3-sigma ellipse and we can see a curved shape in the distribution. Therefore, it might not be a good idea to approximate p(y) as Gaussian when h is too non-linear around the region of large mass of the distribution x.

2 Non-linear Kalman filtering

A,B) In Figures 4 and 5 are presented the filtered sensor positions for the case 1 and 2 of noise parameters. Specially when the noise covariance of the measurements are high, Figure 4, it is possible to see easily clearly that the EKF has a much worse performance than the UKF and CKF.

For the case of Figure 4, the measurements are so noisy that it is visually impossible to imagine what could be the real trajectory given the measurements. However, the Kalman filter is so powerful that it is still able to estimate fairly well the true trajectory.

The 3-sigma level ellipses represent very well the uncertainty of the measurements. One can see in Figure 4, that the major axis of the ellipses is always more aligned with the location of the sensor with less noise covariance, which is in this case the sensor 2. This is a very reasonable behavior. Since the sensor 2 is giving a better estimate of its detection angle, we know that is more probable that the real position is somewhere on the line that is described by the sensor 2 location and its measured angle. Sensor 1 will then just give a bad estimate of the position and give a very uncertain contribution. Another curious thing to notice is that the more the true position is aligned with the location of both sensors, the larger is the ellipse, which seems also reasonable, because slight changes in one angle measurement would change completely the interception between the two lines described by the sensor locations and measured angles. Lastly, we see that for case 1 we have bigger ellipses than for case 2, which has more accurate sensors, which is a reasonable behaviour.

As intensively discussed in the previous question, the EKF filter has a worse performance than the UKF and CKF filters due to the way it approximates the prediction and the update distributions. The EKF linearizes the non-linear function of the motion and measurement models around the expected value for the prior and predicted value respectively. This works well if the nonlinear functions are fairly linear around the region where the correspondent distributions have the most of its mass. When this is not the case, the EKF performs bad, as can be seen in Figure 4.

The beauty of UKF and CKF rely on the way they use samples to estimate the transformed mean and covariance of the distributions of interest in the prediction and update step. This so-called sigma point methods have much superior performance in comparison to EKF.

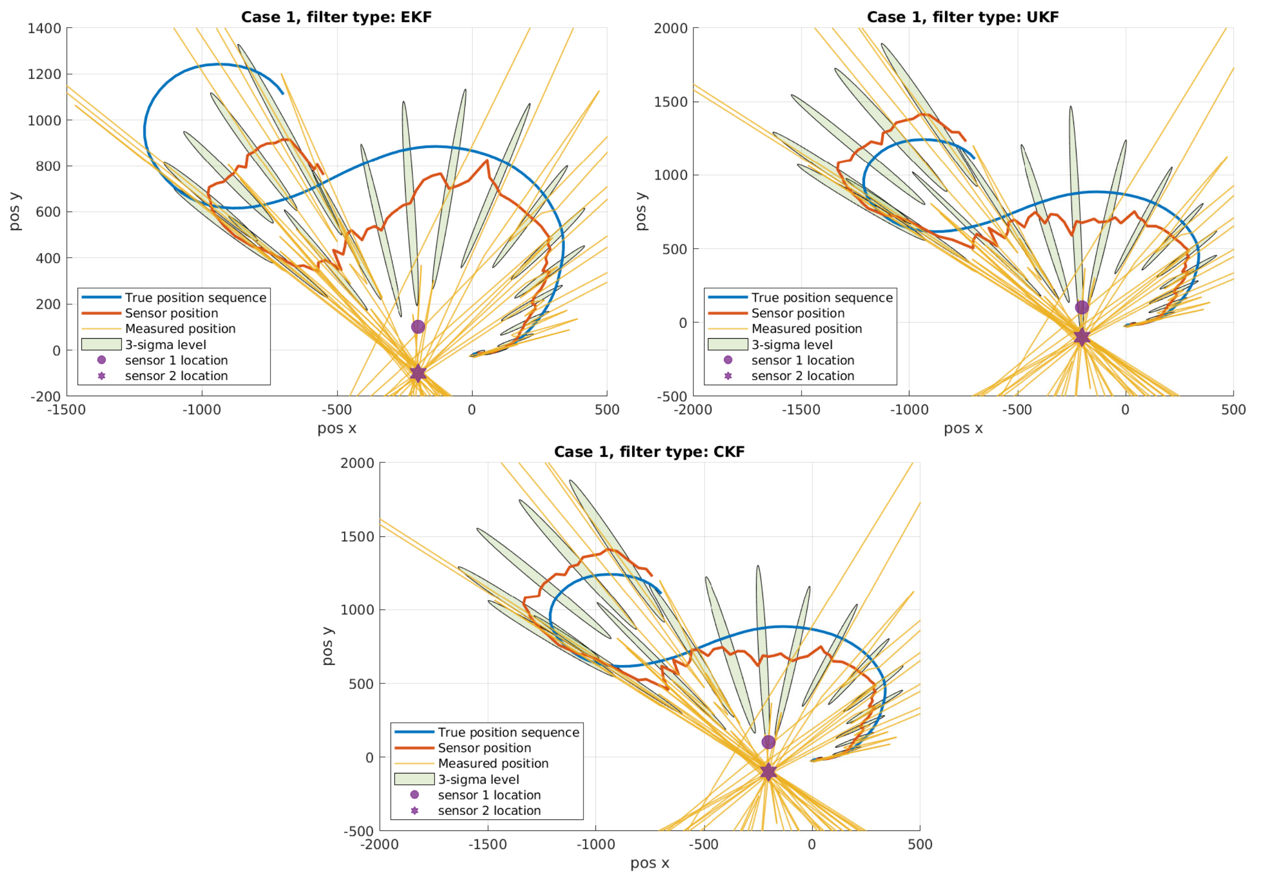

Figure 4: *Filtered sensor positions, measurements and the true position sequence for the case 1 of noise parameters. A CT motion model with the range/bearing measurement model was used to generate and filter the data.*

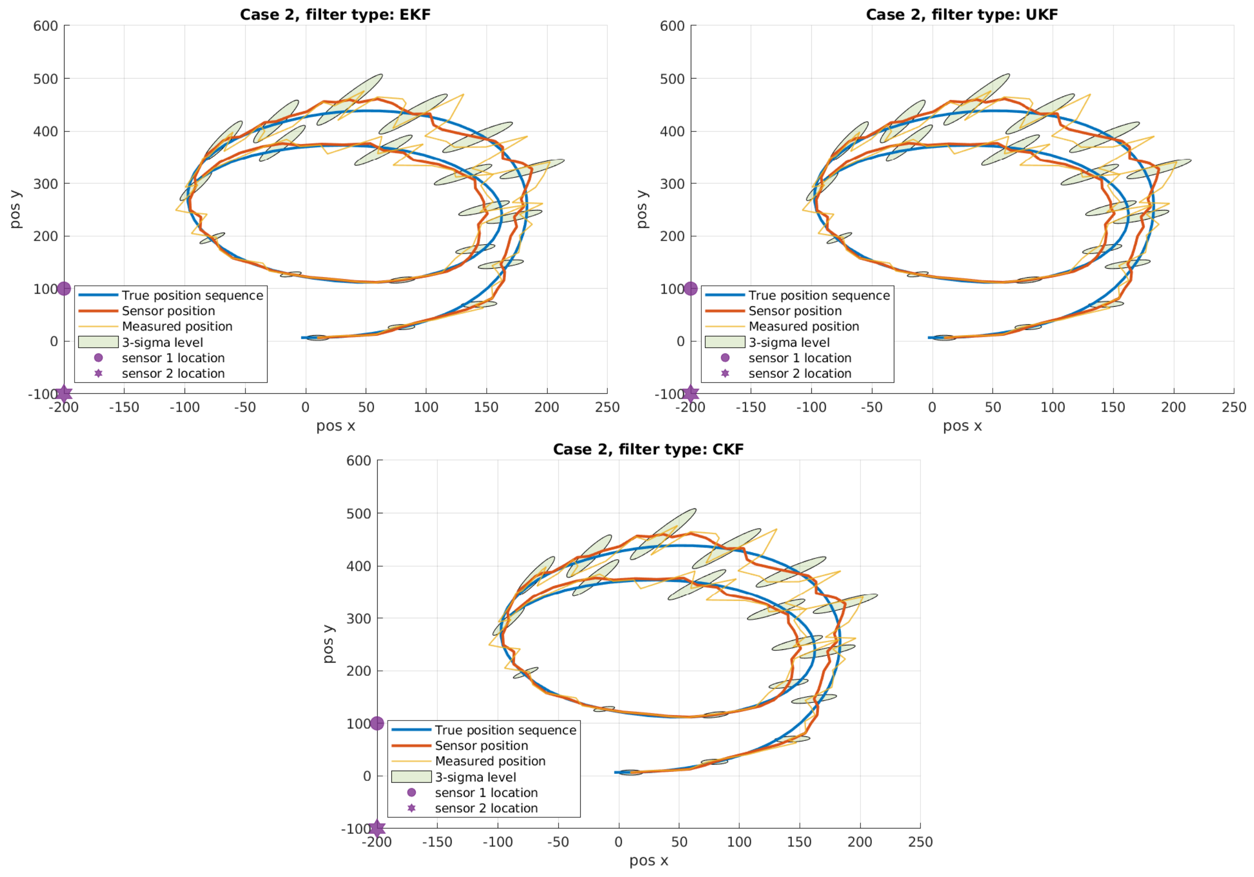

Figure 5: *Filtered sensor positions, measurements and the true position sequence for the case 2 of noise parameters. A CT motion model with the range/bearing measurement model was used to generate and filter the data.*

C) Clearly, the EKF filter as a much worse performance compared to the UKF and CKF filters. This is confirmed when analyzing the histogram distribution of the position error for the case 1 and 2 of noise parameters, as seen in Figures 6 and 7. The mean of the EKF is far from zero and the standard deviation is very high for both cases. When comparing the UKF and CKF filters, we do not see such a stark difference between them. However, the CKF filter performs slightly better than the UKF in terms of mean and variance of the position error.

At first sight, the distributions might look Gaussian. However, the Gaussian distribution generated using the two first two moments of the position error does not match with its normalized histogram. The problem is that there are too many samples distributed far away from what seems to be the 3-sigma region in the histogram diagram.

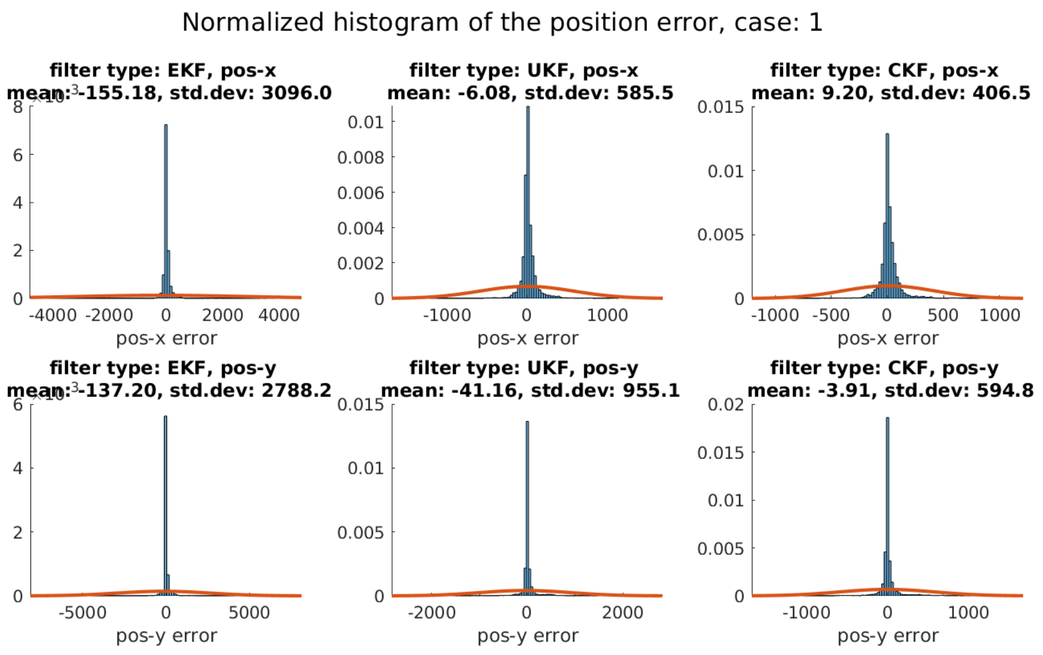

Figure 6: *Histogram of the position error for different Kalman filter types with data from 100 state/measurement sequences. In orange, the Gaussian distribution generated using the estimated mean and covariance. Noise parameters from case 1.*

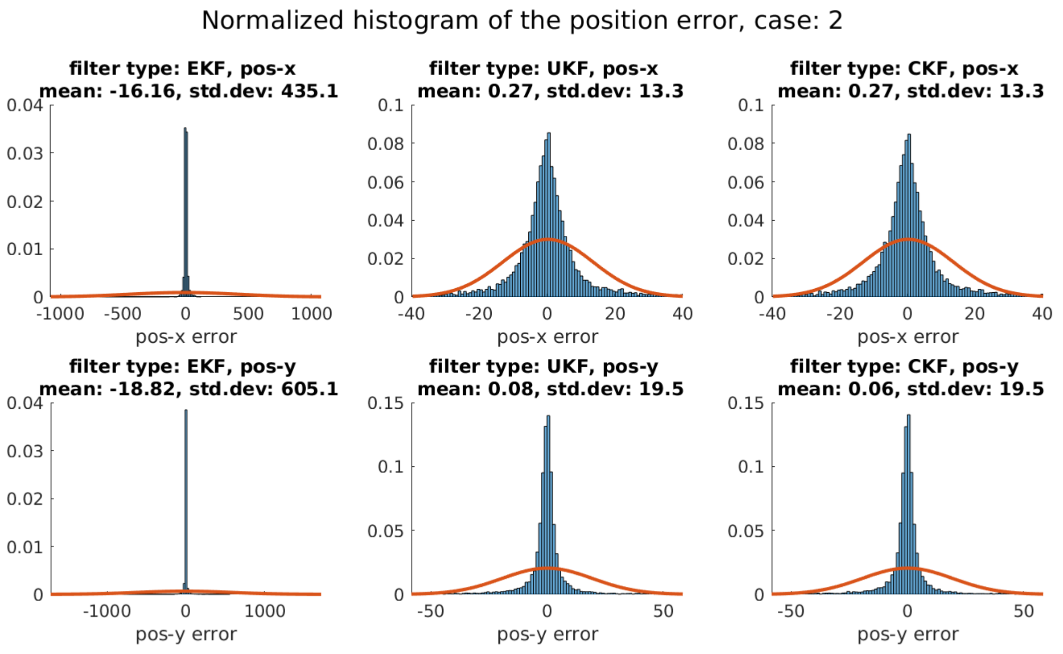

Figure 7: *Histogram of the position error for different Kalman filter types with data from 100 state/measurement sequences. In orange, the Gaussian distribution generated using the estimated mean and covariance. Noise parameters from case 2.*

3 Tuning non-linear filters

A) The CKF has been chosen to be used in the following exercises due to its superior performance in comparison to the other two filters. The filtering effects of changing the model noise covariance for speed and yaw rate are presented in Table 1. The parameters were increased/decreased by 4 orders of magnitude.

Table 1: *Effect of changing the model noise covariance for speed and yaw rate in the CT motion model.*

Parameter	**Effect**
$\sigma_v \uparrow$	Filtered position becomes very noisy. Speed oscillates too much (which is a very unrealistic behaviour for a vehicle driving in a road). If we keep increasing σ_v, it gets so bad that the filter starts to output negative speeds quite often.
$\sigma_v \downarrow$	The more we decrease, the better is our filter output up to a certain limit. This happens because the data was generated with constant velocity. Therefore, if we know beforehand that the vehicle always maintains constant speed and we have a good prior distribution, we will get better results setting the speed variance to very low values.
$\sigma_w \uparrow$	The yaw of the vehicle changes a lot all the time at high frequency resulting in an unrealistic driving behaviour. The filter is however able to adapt itself better to higher yaw accelerations of the true sequence.
$\sigma_w \downarrow$	The filter is not able to change the yaw rate as fast as desired, when the vehicle leave the straight line and enters in the curve. The side-effect is that the vehicle does a turning radius much larger than the real one.
$\sigma_v, \sigma_w \uparrow$	In this case, the filter tends to trust more the measurements, which are very noisy, resulting in a very unrealistic and noisy vehicle trajectory.
$\sigma_v, \sigma_w \downarrow$	Effect similar when decreasing only w, i.e. the vehicle is not able to adapt to the abrupt change of yaw. The vehicle takes a long time to change the yaw rate and as a result the vehicle does a turn radius much larger than it should.

B) Taking into account the true trajectory, we observe that the speed is constant all the time. Since we have a good prior distribution, we do not even need to have a speed covariance in the motion model because a good prior distribution and the update step based on the measurements will be enough to drive the system to the desired constant speed. In sum, the speed noise covariance is needed when the system is subjected to accelerations, which is only the case at the beginning of the simulation when we are not sure about the true vehicle and we need to correct this value to the true one.

Moreover, the yaw rate noise covariance was tuned in such a way that the output was not too noisy and could follow well the trajectory mainly during the transition straight-curve. The good parameters found were $\sigma_v = 0.01$ and $\sigma_w = pi/200$. It is also important to notice that we need to discretize the noise before applying to the Kalman filter formulas, such that $Q_{k-1} = diag([0, 0, T.\sigma_v^2, 0, T.\sigma_w^2])$.

C) We now proceed to present typical results by generating a measurement sequence, and filtering with The sensor positions, the positions corresponding to the measurements, the true position sequence, and the estimated position sequence for three different process noise settings: too small, too large, and well-tuned are shown in Figures 8, 9, and 10 respectively.

As discussed in the last questions, setting the motion model noise covariance too low will result in a model that is not able to accelerate in terms of position and yaw angle as desired when changing from straight to curve and vice-versa. This can be seen in the position error plot, which starts to increase at t=20s when the vehicle leaves the straight and faces a abrupt yaw-acceleration to enter in the curve and at t=50s when it goes back to the straight line needing again a yaw-acceleration.

On the other hand, if the variances are set too high, the filter tends to trust more the measurements, which are very noisy, leading to very noisy and unrealistic outputs. In this case, the position errors are higher when the true position is aligned with the two sensor locations. This is reasonable, since small changes in the measured angles by the sensors will cause a drastic change in the measured object position. Therefore, since in this case we rely much on the measurements, the position error will be higher on those regions (t=14s and t=46s).

The well-tuned model is then able to maintain a fairly low and mostly constant error for all the regions. Therefore, we confirm that the filter performs well because it is not only able to maintain low errors in the steady state during the straight line and curve but also it is able to accelerate and change the state without lag.

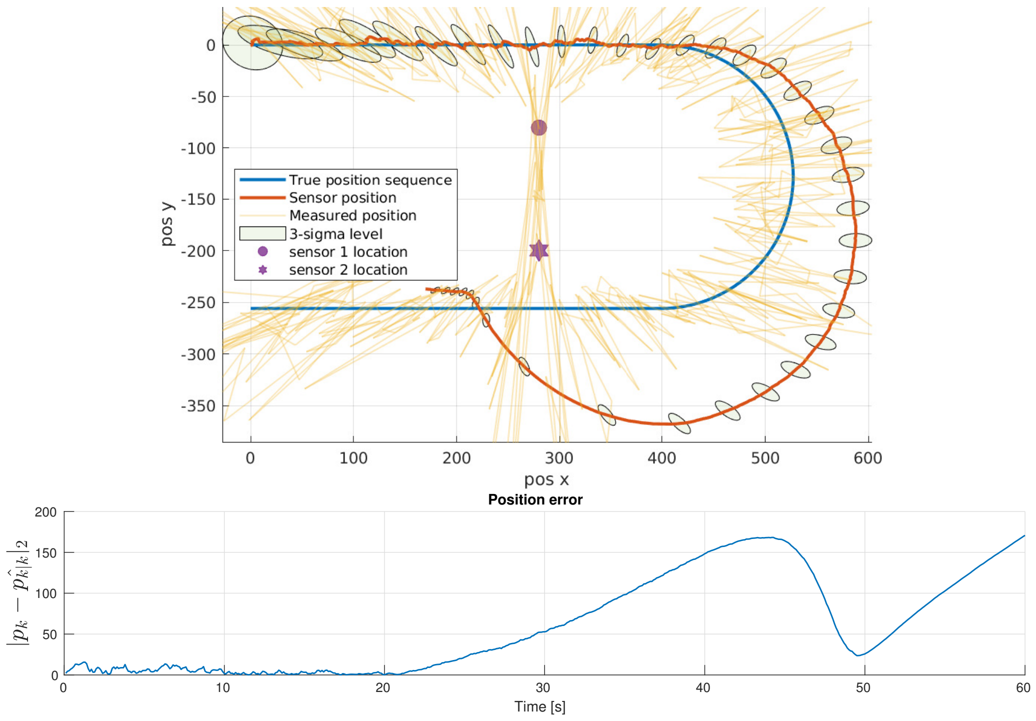

Figure 8: *Process noise settings: too small*

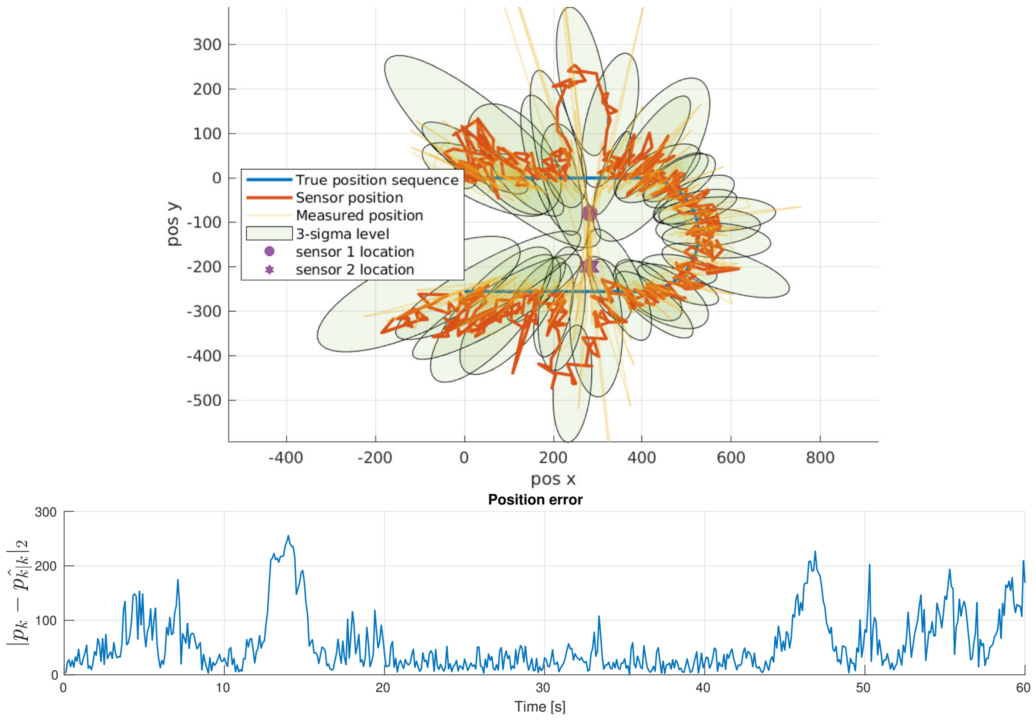

Figure 9: *Process noise settings: too large*

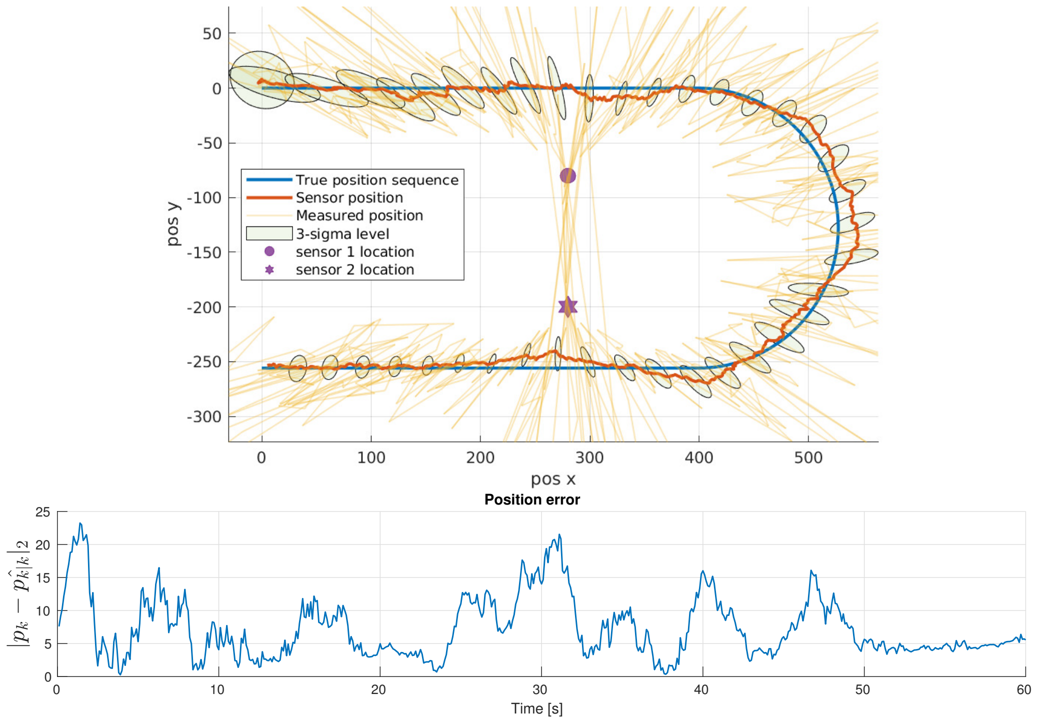

Figure 10: *Process noise settings: well tuned*

D) It is not possible to tune the filter to have a good accuracy all the time for this sequence. The problem is that there are optimal tuning parameters for the straight line and curve and for the transition.

In the straight line and during the curve, the best option is to set both σ_v and σ_w very low, since there is no change in speed and angular velocity. Indeed, this requires a good prior distribution, so the filter does not need to accelerate too much its position and yaw in order to correct its values.

However, once tuned in that way, the model is not able to follow the transition between straight line and curve properly, as shown in Figure 8. For scenarios with large accelerations in position and yaw, we want to set σ_v and σ_w to larger values.

This conflict of effects makes difficult to tune the filter so we get a very precise and accurate outputs. Despite all this, the parameters chosen as in Figure 10 are reasonably good given the very noise measurements.