SSY345 Sensor fusion and non linear filtering

HA3 Analysis

Isak Åslund (isakas)

May 5, 2020

Discussions

I have discussed the assignment with Osvald Lindholm who usually is my LAB partner in other courses.

Code

The used matlab code for solving the problems can be found in the submitted files on Canvas.

1 - Approximations of mean and covariance

(a)

In figure (1) and (2) the Monte Carlo approximations done with 10000 samples.

(b/c)

The different approximation methods for the two different state densities are shown in figure (1) and (2).In the first state density all approximation methods produce very similar result as the sample mean/covariance which can be seen by the overlap of mean circles and 3 sigma level curves. In the second case the UKF and CKF produce similar result as the sample mean/covariance while the EKF creates a very different Gaussian approximation

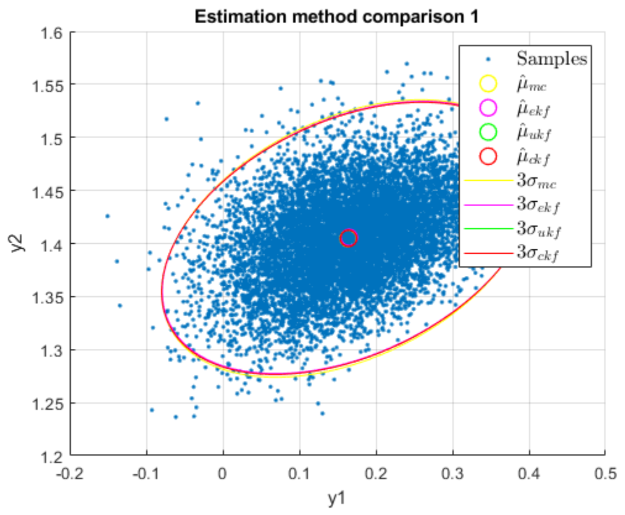

Figure 1: Scenario 1 showing samples that look Gaussianand similar approximations using EKF,UKF and CKF.

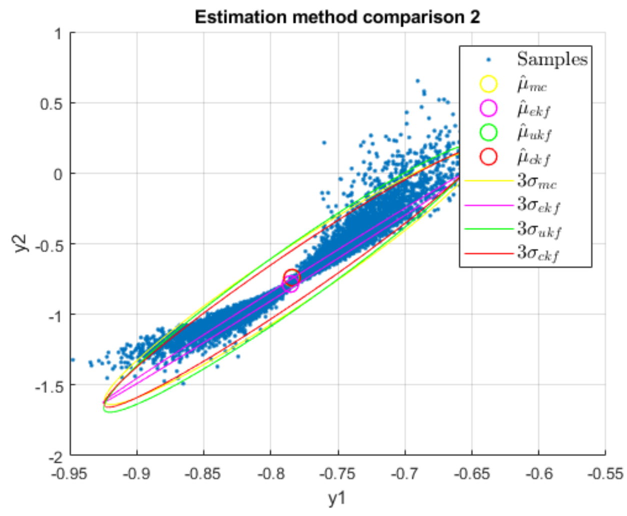

Figure 2: Scenario 2 showing samples that does not look Gaussian and different approximations EKF,UKF and CKF.

(d)

In the first density both the mean and covariance is almost indistinguishable and the samples seems to be taken from a Gaussian distribution. In the second density we see a different behavior where the distribution does not look Gaussian and the approximations are different. The UKF,CKF and MC (Monte Carlo) produce very similar mean and covariances while the EKF is drastically different in covariance and a bit different in mean. The EKF has a very small 3 sigma level curve that does not represent the samples well while the others approximations captures most of the points. The reason for the different behavior is that in the EKF we linearize the model around the working point and in the second density the model is highly non linear around that point. This is because the initial point is very close to sensor 2 and far away from sensor 1. This is different from when the sensors have an equal distance to the object as in density one and hence is not as non linear around that point.

(e)

The EKF performed well when the model was mildly non linear around the linearization point and poorly when it was highly non linear and therefore it would not be a good method for Gaussian approximation. However the UKF and CKF performed well in both scenarios and gave Gaussian approximations that captured most of the samples even if they weren't Gaussian. I think using the UKF or CKF for approximating $p(y)$ as a Gaussian distribution is the best approximation we can come up and that it would be sufficient.

2 - Non-linear Kalman filtering

(a)

The first case and the three different non linear Kalman filters EKF, UKF and CKF are shown in figure (3), (4) and (5) respectively. This choosen state sequence is a relatively straight path followed by a somewhat constant turn and then straight path again. All three methods produces similar results in this and most of the tested scenarios. However one would expect the EKF to behave a bit different from the UKF and CKF at points where the model is highly non linear, which is close to the sensors. The estimates are generally quite bad since the 3 sigma level curves does not always contain the true state and sometimes loses track of the true state. In this particular example the filters perform decent and is able to keep the true state within the 3 sigma level curves since they are very large (high uncertainty/under confident).

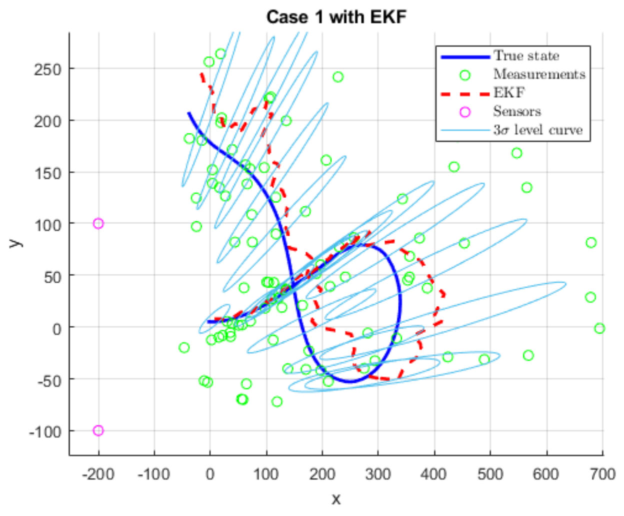

Figure 3: Case 1 with EKF

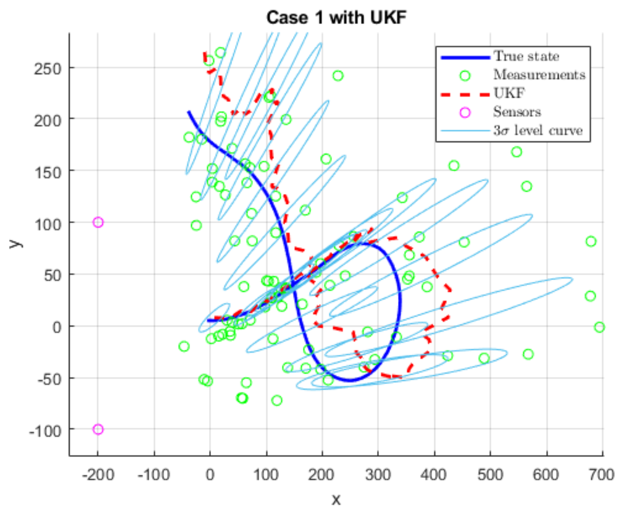

Figure 4: Case 1 with UKF

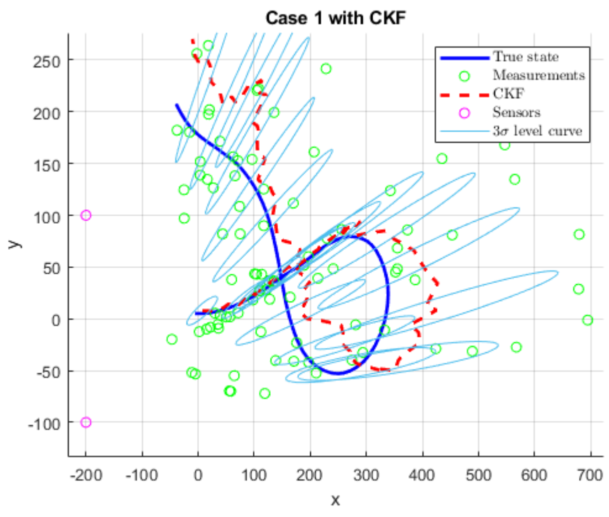

Figure 5: Case 1 with CKF

(b)

When the noise was reduced as in case 2 the filters become a lot better and was able to track the state with good confidence (small 3 level sigma curves always contains the true state) for all filters and almost all tried state sequences. The CKF filter of the previously shown state sequence can be seen in figure (6) where tracking is quite good compared to case 1.

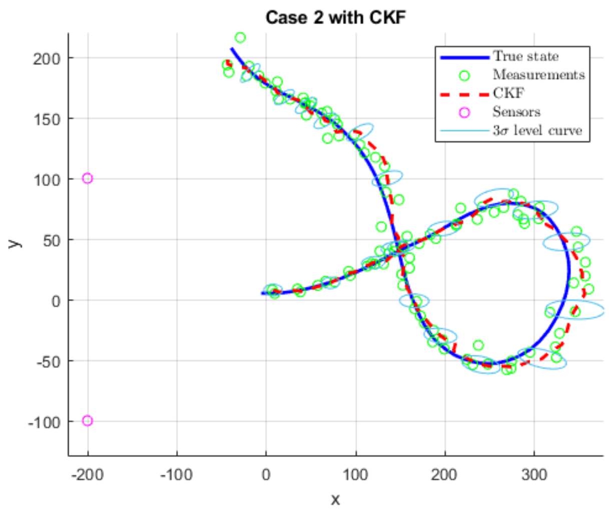

Figure 6: Case 2 with CKF

(c)

The histograms of estimation error for the different filters and positions can be seen in figure (7). The estimation errors appear Gaussian for all cases and filters due to their bell shaped curve. The only major differences between the filter types is that the EKF seems to have larger errors sometimes. The reason for the plots looking like spikes is that there are a couple state sequences where the filters behave very poorly and loses track of the state completely thus creating large errors. I was not able to draw any big conclusions about differences in the filter types from these plots but I would say that the UKF/CKF perform about the same and that the EKF is slightly worse. Also the EKF is more computationally expensive due to the linearization.

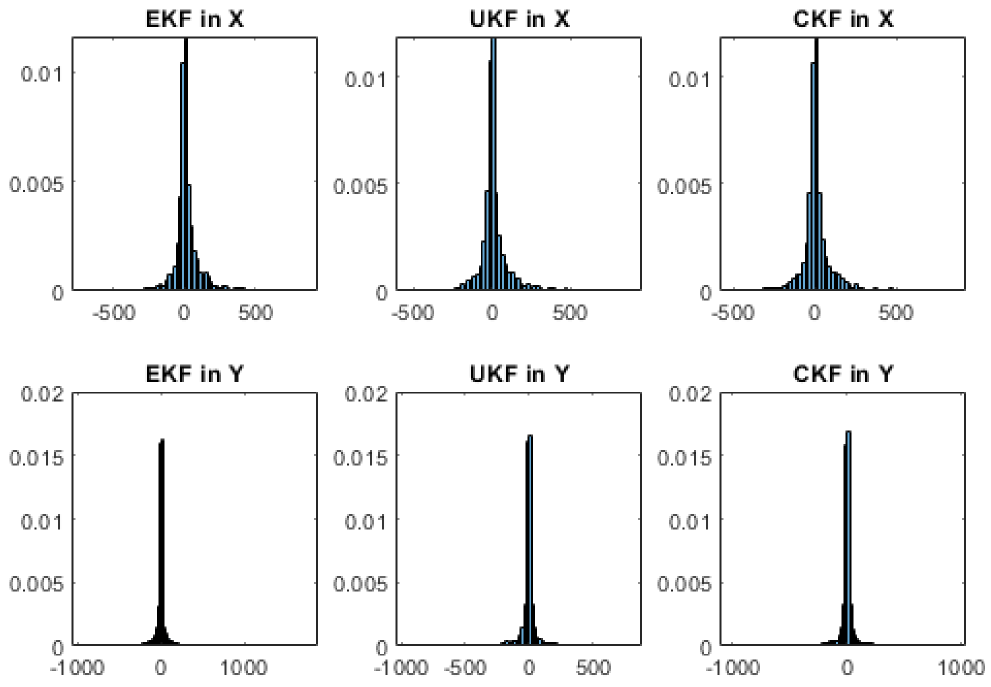

Figure 7: Estimation error histograms indicating a Gaussian distributions.

3 - Tuning non-linear filters

I chose the CKF as my favorite non-linear Kalman filter and will be used throughout this question.

(a)

When increasing the process noise in velocity the filtered position becomes very jumpy due to the fact that we trust the measurements a lot and not the model. When the noise in the angular velocity is increased the reaction isn't as noticeable but a worse performance can be observed. Increasing both a lot at the same time creates a very noisy estimate that is very fast at reacting to changes but way to noisy for a reasonable estimate of an actual object.

Decreasing the noise in velocity makes the estimate a little bit more smooth and decreasing the angular velocity noise makes the filter smooth but slow to react to changes as in the transition from straight to curve. This is because we trust our model more then the noisy measurements. The model is a constant turn model and without any initial angular velocity the model tells us that we would expect to remain that way and hence the measurement update is too slow to adjust the estimate when we transition from straight to curve. Decreasing both a lot at the same time creates a smooth estimate that is trusting too much in the model and fails to track the curve properly.

(b)

Analyzing the true track we see that the velocity is constant meaning we should trust our constant velocity model and hence $\sigma_v = 0$ was chosen. The turn rate changes two times, when we transition from straight to turn and turn to straight and hence we want to trust the model during these different regions but not in the transition. By visual inspection the initial $\sigma_\omega = \frac{\pi}{180}$ was found to give the best results.

(c)

Figure (8) shows a filter where we trust the model too much and hence the filter fails to track the curve properly in the turn and becomes too over confident.

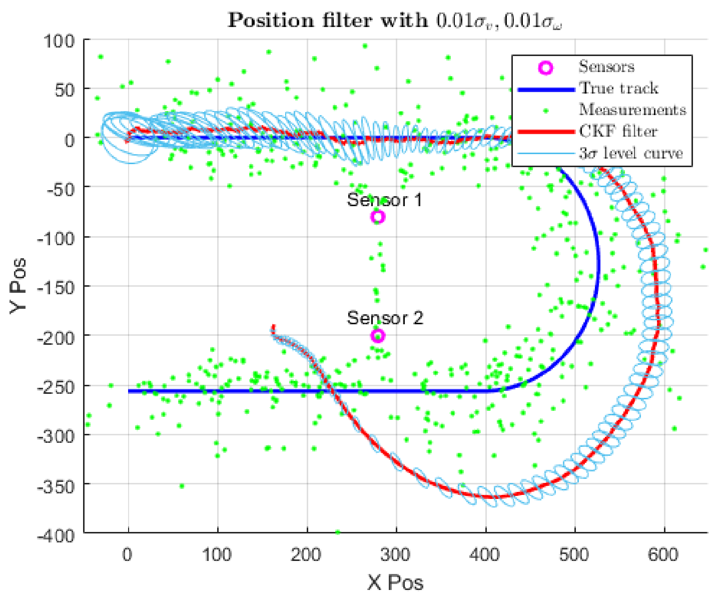

Figure 8: CKF with a too small process noise.

Figure (9) shows a filter where we trust the measurements too much and the filter becomes very noisy and under confident.

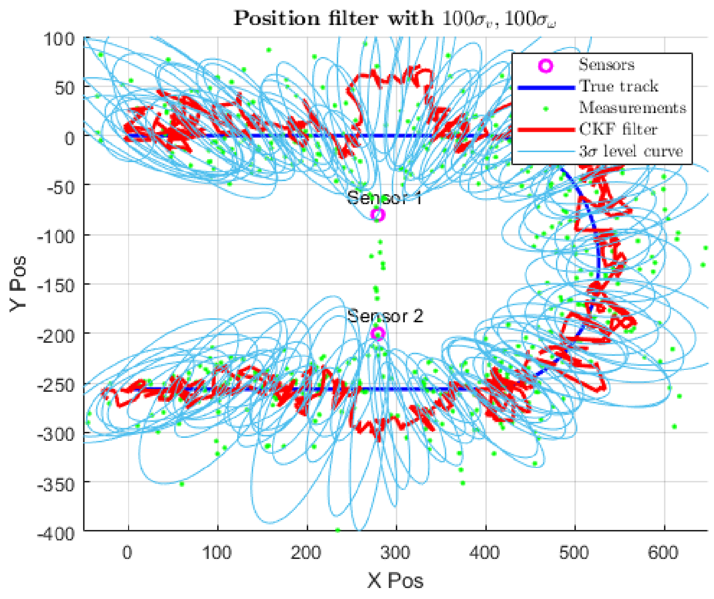

Figure 9: CKF with a too large process noise.

The tuned filter is shown in figure (10) where the estimates stay close to the true track at all times while trying to smooth it out.

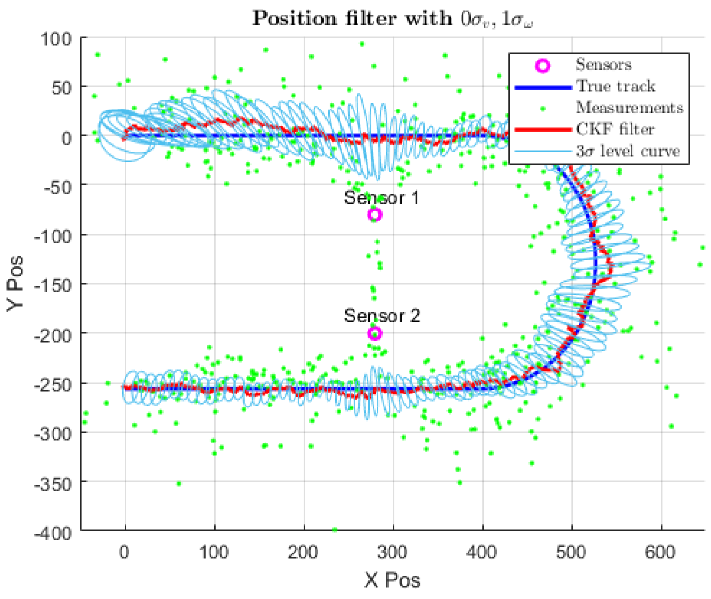

Figure 10: CKF with a tuned process noise to achieve lower overall error.

The position error over time for the three different filters are shown in figure (11) and indicate the the tuned filter almost always has a smaller error than the others.

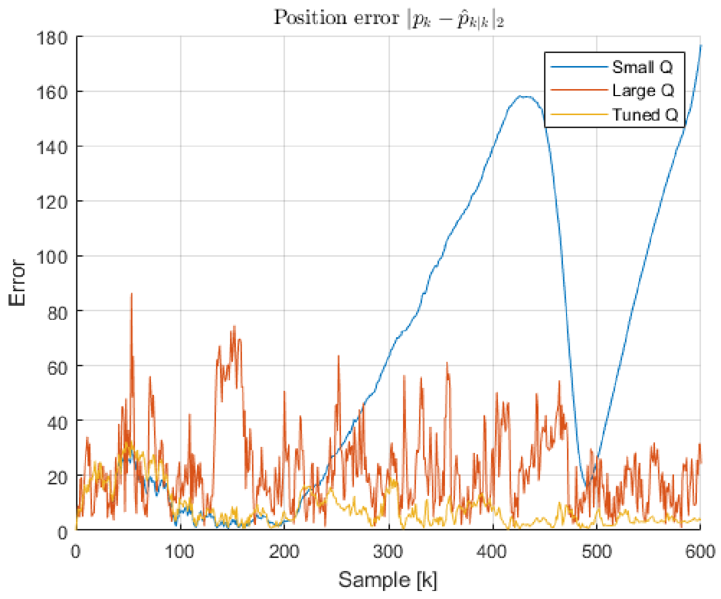

Figure 11: Position error indicating that the best filter is the tuned one.

A better way to tune the filter could have been to integrate all the errors up for different process noises and choose the one with the lowest accumulated error,

(d)

As previously explained the true track consists of three main parts where the velocity is constant at all times and on the straight segments the turn rate is zero while in the turn beeing $\omega = \frac{-\pi}{201 \cdot T}$. For the velocity we could choose $\sigma_v = 0$ since we know the true track. If this was a real world scenario we would probably not choose zero but instead a small value to make the filter still adapt reasonably fast to speed changes. For the angular velocity we would like to trust our model in the three segments and hence reduce our process noise but the problem is that during the transition from straight to turn and vice versa we would like the process noise to be higher in order to quickly adapt. This creates a conflict since we can't have it both low and high at the same time. My suggestion is therefor to keep the process noise quite low to do good predictions but not too low so that we are too slow to adapt when we realize we have started to turn.