SSY345 – Sensor Fusion and Non-Linear Filtering
Home Assignment 3 - Analysis

Basic information

Please upload your reports to ping-pong no later than **7 May 2019 18:00**.

You are encouraged to discuss the assignments with classmates, but you must write up your own solutions (including Matlab implementations). Also, when writing up the solutions, you should write the names of people with whom you have discussed the assignments. It is important that you make an effort to present/illustrate your solutions nicely and reflect on your results!

This home assignment is related to the material in lecture 6 and 7. The main focus on the design, implementation, tuning, and evaluation of non-linear Kalman filters. Specifically, the Extended Kalman Filter (EKF), the Unscented Kalman Filter (UKF), and the Cubature Kalman Filter (CKF). To pass the home assignment you need to get a pass on all implementation assignments and correctly complete the accompanying quiz on ping-pong. To be able to get a higher grade in the course, you need to collect POE by completing this analysis assignment.

In the analysis part we want you to use the toolbox that you have developed and apply it to a practical scenario. Associated with each scenario is a set of tasks that we would like you to perform, and a set of questions on ping-pong that you should answer. The tasks are designed such that they will help you get the insights needed to answer the questions on ping-pong.

The result of the tasks should in general be visualized and compiled together in a report (pdf-file). A template for the report can also be found on course homepage. The report should also be uploaded to ping-pong before the deadline. Please make sure that your analysis text is not unnecessarily lengthy, and that all figures are properly labelled, as well as captioned or referenced in your text.

1 Approximations of mean and covariance

The non-linear Kalman filters all use the same type of update for the state estimate, with a Kalman gain $\mathbf{K} = \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}$; here x denotes the state and y denotes the measurement. Depending on the type of non-linear Kalman filter, different approximations are used to compute the cross covariance $\mathbf{P}_{xy}$ and the covariance $\mathbf{P}_{yy} = \mathbf{S}$.

Consider a 2D state vector $\mathbf{x}$ that consists of x-position and y-position. We will consider two different state densities,

$$p_1(\mathbf{x}) = \mathcal{N}\left(\mathbf{x}; \hat{\mathbf{x}}_1, \mathbf{P}_1\right) = \mathcal{N}\left(\mathbf{x}; \begin{bmatrix}120\\120\end{bmatrix}, \begin{bmatrix}5^2 & 0\\0 & 10^2\end{bmatrix}\right), \tag{1}$$

$$p_2(\mathbf{x}) = \mathcal{N}\left(\mathbf{x}; \hat{\mathbf{x}}_2, \mathbf{P}_2\right) = \mathcal{N}\left(\mathbf{x}; \begin{bmatrix}120\\-20\end{bmatrix}, \begin{bmatrix}5^2 & 0\\0 & 10^2\end{bmatrix}\right). \tag{2}$$

and the dual bearing measurement model from the implementation part of HA3, with the bearing sensors located in $\mathbf{s}_1 = [0, 100]^T$ and $\mathbf{s}_2 = [100, 0]^T$, each with Gaussian measurement noise with standard deviation $\sigma_\varphi = 0.1\pi/180$rad. In this task, we will focus on the approximation of the mean $E[\mathbf{y}]$ and the covariance $\mathrm{Cov}(\mathbf{y}) = \mathbf{P}_{yy}$.

Task:

a For each state density, generate samples of $\mathbf{y} = \mathbf{h}(\mathbf{x}) + \mathbf{r}$ by first sampling the state density, computing the dual bearing measurement, and then adding random sample noise. Use the samples to approximate the mean and covariance of $\mathbf{y}$. Remember to use a sufficient number of samples, so that your sample mean and sample covariance are as accurate as possible.

b Compute three approximations of the mean and the covariance, using the type of density approximations that are used in EKF, UKF, and CKF, respectively.

c Plot the samples of $\mathbf{y}$, the sample mean/covariance, as well as the three different approximated means/covariances. Compare the approximate means/covariances to the sample mean/covariance.

d What differences can you see? Is any approximation method (EKF, UKF or CKF) better than the other two? The results from the two state densities should show some different results in this respect. Explain what causes the differences.

e The Kalman filter, as well as the non-linear Kalman filters, propagates the mean and the covariance of the state estimate. We have seen earlier in the course that the mean and the covariance are sufficient statistics for the Gaussian distribution.[1] Given the approximations of the mean and

[1] The mean and the covariance are the parameters that we need for the probability density function.

Table 1: Noise parameters

Case	σ_v	σ_ω	$\sigma_\varphi^{(1)}$	$\sigma_\varphi^{(2)}$
1	1	$\frac{\pi}{180}$	$\frac{10\pi}{180}$	$\frac{0.5\pi}{180}$
2	1	$\frac{\pi}{180}$	$\frac{0.5\pi}{180}$	$\frac{0.5\pi}{180}$

the covariance of $\mathbf{y}$, do you think it is a good idea to approximate the density $p(\mathbf{y})$ with a Gaussian distribution? *Hint:* compare the approximate mean/covariance with the samples. Do the samples looks Gaussian?

2 Non-linear Kalman filtering

In this scenario you will study the properties of the non-linear Kalman filters in a scenario where we combine the CT motion model with the range/bearing measurement model.

We will consider two different cases of the dual bearing measurement model, where one or both sensors are accurate. The process and measurement noise standard deviations are given in Table 1. The initial prior is

$$\mathbf{x}_0 = \begin{bmatrix} 0 & 0 & 14 & 0 & 0 \end{bmatrix}^T \tag{3}$$

$$\mathbf{P}_0 = \text{diag}\left(\begin{bmatrix} 10^2 & 10^2 & 2^2 & \left(\frac{\pi}{180}\right)^2 & \left(\frac{5\pi}{180}\right)^2 \end{bmatrix}\right) \tag{4}$$

Assume that the sensors are stationary, located in $\mathbf{s}_1 = [-200, 100]^T$ and $\mathbf{s}_2 = [-200, -100]^T$, and that the sampling time is $T = 1$.

Task: To get a feeling for how the different non-linear Kalman filters perform we would like you to do the following tasks and reflect on what you observe.

a Start with Case 1 in Table 1. Generate a state sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots$ and a measurement sequence $\mathbf{y}_1, \mathbf{y}_2, \ldots,$; a suitable length of the sequence is $N = 100$ time steps. Filter the measurement sequence using the three different non-linear Kalman filters that you have implemented. We now have to assess the performance of the different non-linear Kalman filters.

- One way to evaluate the results is to compare the estimated positions to the true positions. Plot the sensor positions. Plot the measurements[2] and the true position sequence. For each of the filters, plot the estimated position sequence, together with 3σ-contours at every

[2] For the plot of the measurements to make sense, you have to compute positions from the measurements as the intersection points between the lines formed by each sensor's position and measured angle.

Table 2: Monte Carlo simulation

```
for imc = 1:MC
    % Simulate state sequence
    X = genNonLinearStateSequence(x_0, P_0, f, Q, N);
    % Simulate measurements
    Y = genNonLinearMeasurementSequence(X, h, R);
    % Run Kalman filter (you need to run all three, for comparison)
    [xf,Pf,xp,Pp] = nonLinearKalmanFilter(Y,x_0,P_0,f,Q,h,R,type);
    % Save the estimation errors and the prediction errors!
end
```

5^{th} estimate. If your figure becomes very cluttered, you can have three subplots, one for each filter.

- To be able to draw reasonable conclusions, you will need to simulate a several different state/measurement sequences. In your report, include figures from a single state/measurement sequence that you think is representative of the general performance.

b Now do the same for Case 2 in Table 1. You do not have to include plots from the Case 2 in your report. For the two cases, can you see any significant differences between the EKF, the UKF and the CKF? Do the filters behave well? Do the error covariances represent the uncertainty well? Can you explain the differences, or the absence thereof, in terms of how the different filters approximate the prediction and the update?

c You should have noticed from filtering different state/measurement sequences that the results can vary quite a lot between different sequences. It is important to not "cherry-pick" sequences for performance evaluation. Instead, one typically filters a larger number of state/measurement sequences, and then studies the average performance. Generate at least 100 state/measurement sequences, and for each sequence, compute the EKF, UKF and CKF estimates and compare the estimates to the true state values.

- Plot histograms of the estimation errors for the position states in the state vector. Using 2×3 subplots may be a good idea. Do this for both cases in Table 1.
- Can you see any differences between the EKF, UKF and CKF? Do the histograms look Gaussian? *Hint:* a template for the Monte Carlo simulation of prediction/estimation errors is given in Table 2.

3 Tuning non-linear filters

In this scenario we focus on the tuning of the filter parameters. It is a fairly reasonable assumption that the measurement noise covariance is known, because sensors can often be characterised using experiments. However, the process noise is more difficult, and can typically not be assumed to be known.

Generate a true state sequence

$$\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{600} \tag{5}$$

using the code in Table 3. If you plot the positions, you will see that the sequence consists of a straight line, followed by a turn, followed by another straight line. We are going to use the dual bearing measurement model and the true sequence to generate measurement sequences

$$\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{600}. \tag{6}$$

The initial prior is

$$\mathbf{x}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{7}$$

$$\mathbf{P}_0 = \operatorname{diag}\left(\begin{bmatrix} 10^2 & 10^2 & 10^2 & \left(\frac{5\pi}{180}\right)^2 & \left(\frac{1\pi}{180}\right)^2 \end{bmatrix} \right) \tag{8}$$

Let the sensor positions be stationary,

$$\mathbf{s}_1 = [280\ ,\ -80]^T, \quad \mathbf{s}_2 = [280\ ,\ -200]^T \tag{9}$$

The measurement noise standard deviations are known for both sensors,

$$\sigma_\varphi^{(1)} = 4\pi/180, \quad \sigma_\varphi^{(2)} = 4\pi/180. \tag{10}$$

As above, it will not be sufficient to consider just one measurement sequence; you have to try several ones.

Task: From the three alternatives, select your favourite non-linear Kalman filter. Use the CT motion model. The challenge in this task is the process noise covariance $\mathbf{Q}$, which is unkown. Tuning $\mathbf{Q}$ means to try different values and comparing the estimation results against each other.

a Start from the values $\sigma_v = 1$ and $\sigma_\omega = \pi/180$.

- What happens when you make the process noise very large? Try increasing just one of the two parameters, and also try increasing both.
- What happens when you make the process noise very small? Try decreasing just one of the two parameters, and also try decreasing both.

Hint: Do not be afraid to push the limits of your filter by changing the noise by several orders of magnitude. To really understand a filter, and the models that it is based on, one has to really excite the system.

b Try to tune the filter so that you get good position estimates for the whole sequence. *Hint:* the noise affects the velocity and the turn-rate. Think about the true track, and what the corresponding noise should be.

c Present typical results by generating a measurement sequence, and filtering with three different process noise settings: too large, too small, and well tuned.

- Plot the sensor positions, the positions corresponding to the measurements, the true position sequence, and, for each process noise setting, the estimated position sequence with corresponding covariance contours at every 5^{th} estimate. If your figure becomes too cluttered, use subplots.
- Complementary to plotting the positions is to compute the position error $\|\mathbf{p}_k - \hat{\mathbf{p}}_{k|k}\|_2$ and plot vs time. Compare the position errors for all three noise settings.

d In many applications it is necessary to use the estimate to predict several time steps into the future. In this task, this means that the more accurate estimates we have of velocity, heading and turn-rate, the more accurate our predictions will be. Is it possible to tune the filter so that you have accurate estimates of those three states for the whole sequence? Why, or why not? *Hint:* again, think about the true track. Is there any conflict between how one would like to tune the filter for different parts of the true track? Do we want the same for the straight line as for the turn? How about the transition from straight to turning?

Table 3: Generate true track

```
%% True track
% Sampling period
T  = 0.1;
% Length of time sequence
K = 600;
% Allocate memory
omega = zeros(1,K+1);
% Turn rate
omega(200:400) = -pi/201/T;
% Initial state
x0 = [0 0 20 0 omega(1)]';
% Allocate memory
X = zeros(length(x0),K+1);
X(:,1) = x0;
% Create true track
for i=2:K+1
    % Simulate
    X(:,i) = coordinatedTurnMotion(X(:,i-1), T);
    % Set turn-rate
    X(5,i) = omega(i);
end
```