TTK4250 Sensor Fusion

Assignment 2

Hand in: *Wednesday 11. September 16.00* on Blackboard or to teaching assistants in exercise class.

Tasks are to be solved on paper if you are not told otherwise, and you are supposed to show how you got to a particular answer. It is, however, encouraged to use MATLAB, Maple, etc. to verify your answers. Rottmann's mathematical formula collection is allowed at both the exercises and the exam.

Task 1: *Bayesian estimation of an existence variable*

We are back at tracking the number of boats in a region. However, you now know that there is at most one boat in the region, and that it is there with probability $r_k \in (0, 1)$ at time step k. The boat will stay in the region with probability P_S and leave with probability $1 - P_\text{S}$ if it is there. Otherwise, it may enter with probability P_E and not enter with probability $1 - P_E$ if it is not in the region. You have an unreliable measurement coming from a radar that detects a present boat with probability P_D, and may not detect a present boat with probability $1 - P_\text{D}$. If it did not detect a present boat, it may declare that there is a boat present anyway, termed a false alarm, with probability P_{FA}. The task is to apply a Bayesian filter to this problem.

Hint: It might ease proper book keeping if you start by denoting the events with variables and use their (possibly conditional) PMFs before you insert the given probabilities.

(a) Apply the Bayes prediction step to get the predicted probability $r_{k+1|k}$ in terms of r_k, P_S and P_E.

(b) Apply the Bayes update step to get posterior probability for the boat being in the region, r_{k+1}, in terms of $r_{k+1|k}$, P_D and P_{FA}. That is, condition the probability on the measurement. There are two cases that needs to be considered; receiving a detection and not receiving a detection.

Task 2: *KF initialization of CV model without prior knowledge*

The KF typically uses a prior for initializing the filter. However, in target tracking we often have no specific prior and would like to infere the initialization of the filter from the data. For the CV model with positional measurements, the position is observable with a single measurement, while the speed velocity needs two measurements to be observable (observable is here used in a statistical sense to mean that there is information about the state from the measurements). Since the KF is linear, we would like to use a linear initialization scheme that uses two measurements and the model parameters. That is

$$\hat{x}_1 = \begin{bmatrix} \hat{p}_1 \\ \hat{u}_1 \end{bmatrix} = \begin{bmatrix} K_{p_1} & K_{p_0} \\ K_{u_1} & K_{u_0} \end{bmatrix} \begin{bmatrix} z_1 \\ z_0 \end{bmatrix}, \tag{1}$$

where $z_k = \begin{bmatrix} I_2 & 0_2 \end{bmatrix} x_k + w_k = p_k + w_k$ with $x_k = \begin{bmatrix} p_k^T & u_k^T \end{bmatrix}^T$ and $w_k \sim \mathcal{N}(0, R)$. p_k is the position and u_k is the velocity at time step k.

(a) Write z_1 and z_0 as a function of the noises, true position and speed, p_1 and u_1, using the CV model with positional measurements. Use v_k to denote the process distrurbance, and w_k to denote the measurement noise at time step k.

Hint: A discrete time transition matrix is always invertible, and it is easy to find for the CV model (remove the process noise and think what you are left with).

(b) Show that to get an unbiased initial estimate, the initialization gain matrix must satisfy $K_{p_1} = I_2$, $K_{p_0} = 0_2$, $K_{u_1} = \frac{1}{T} I_2$ and $K_{u_0} = -\frac{1}{T} I_2$, where T is the sampling time. That is, find the $K_\times$ so that $E[\hat{x}_1] = x_1 = \begin{bmatrix} p_1 & u_1 \end{bmatrix}^T$

(c) What is the covariance of this estimate?

(d) You have used this initialization scheme for your estimator and found a mean and covariance. What distribution does the true state have after this initialization? What are its parameters.

Hint: From equations you have already used, you can write x_1 in terms of $\hat{x}$ and disturbances and noises. You should be able to see the result as a linear transformation of some random variables. Note that $\hat{x}$ is given since the measurements are given and thus can be treated as a constant.

(e) In theory, would you say that this initialization scheme is optimal or suboptimal, given that the model and two measurements is all we have? What would you say about its optimality in practice?

Task 3: *Implement EKF i MATLAB*

In MATLAB, Finish the following functions in the EKF skeleton found on blackboard.

You are free to change the prewritten code (for instance to optimize), but the function definitions must be the same for evaluation purposes. Model has functions as members: .f(x, Ts), .F(x, Ts), .h(x), .H(x), .Q(x, Ts), R(x, z)

(a) `[xp, Pp] = EKF.predict(obj, x, P, model)`

(b) `[vk, Sk]=EKF.innovation(obj, z, x, P, model)`

(c) `[xupd, Pupd] = EKF.update(obj, z, x, P, model)`

(d) `NIS = EKF.NIS(obj, z, x, P, model)` (for parameter evaluation)

(e) `ll = EKF.loglikelihood(obj, z, x, P, model)` (for future use)

Task 4: *Make CV model to use with the EKF class*

In MATLAB: Make a model structure that implements an EKF for the CV model to use with the EKF class you made in the last task. You can use the skeleton makeCVmodel function found on Blackboard. You have the standard model for a KF

$$x_k = F x_{k-1} + v_{k-1}, \qquad v_{k-1} \sim \mathcal{N}(0; Q) \tag{4}$$

$$z_k = H x_k + w_k, \qquad w_k \sim \mathcal{N}(0, R) \tag{5}$$

(a) Implement the prediction of the mean, `model.f(x, Ts)`.

(b) Implement the Jacobian of f with respect to x, `model.F(x, Ts)`.

(c) Implement the measurement prediction, `model.h(x)`.

(d) Implement the Jacobian of h with respect to x, `model.H(x)`.

(e) Implement the process noise covariance matrix as a function of x, `model.Q(x, Ts)`.

(f) Implement the measurement noise covariance matrix, `model.R(x, z)`.

Task 5: *Tuning of KF*

Tune your CV KF implementation to fit the given data. The data is created by simulating a CT model with $\sigma_a = 0.25^2 = 0.0625$ and $\sigma_\omega = \left(\frac{\pi}{15}\right) = 0.0439$. You can also generate new data if you want to test more.

The CV continuous time acceleration covariance will be denoted by q, and the measurement noise covariance by r in this task and code.

Hint: Section 4.6 - 4.8 will probably be helpfull.

(a) Simply tune q and r until you are satisfied

(b) Make a grid for q and r and plot NIS using `surf`. Use the distribution of NIS to find confidence intervals and plot the contour lines of your evaluated NIS for these intervals using `contour`. Would you do some corrections to your tuning from A? Do you expect NIS to be good for tuning a CV model to a CT trajectory?

Hint: Write help ”the function” in MATLAB to get information on it. chi2inv might come in handy

(c) Do the same for NEES as you did with NIS. Would you tune differently now? Do you expect NEES to be good for tuning a CV model to a CT trajectory?

(d) Run your tuned filter on some new data. Are the results as expected? Why/why not?

Task 6: *Implement a SIR particle filter for a pendulum*

We want to estimate the position of a pendulum influenced by some random disturbance, with a suboptimal measuring device.

The pendulum dynamic equation is given by $\dot{\theta} = \frac{g}{l}\sin(\theta) + F$, where g is the gravitational acceleration and l is the lenght from the center of rotation to the pendulum. The pendulum is assumed to be a point mass attached to a massless rod. F represents other forces that acts on the pendulum, here modeled as a stochastic process. θ is 0 when the rod is at the bottom.

A measuring device is set up L_d below the rotation point, and L_l to the left so that $z_k = h(\theta) = \sqrt{(L_d - \cos(theta))^2 + (\sin(\theta) - L_l)^2)} + w_k$, where $w_k \sim \text{triangle}(E[w_k], a)$, a symmetric triangle distribution with width $2a$, height $\frac{1}{a}$ and peak at $E[w_k]$.

A skeleton that creates data and provides some plot aid and so on can be found on Blackboard.

(a) Finish the particle filter. To finish it you need to implement particle prediction, weight update and resampling (algorithm 3). How does the filter perform?

(b) Do not resample the simulated pendulum, but vary L_l and generate new measurements to see how it changes the estimate. Does it remove any problems in the estimation you found above?

(c) When would you choose to try a PF instead of a EKF and why?