TTK4250 Sensor Fusion Assignment 5

Hand in: *Friday 11. October 23:59* on Blackboard.

Read Python setup guide on BB, it can be found under *Course work*.
This assignment should be handed in on Blackboard, as a PDF pluss **the zip created by running *create_handin.py***, before the deadline.
You are supposed to show how you got to each answer unless told otherwise.
If you struggle, we encourage you to ask for help from a classmate or come to the exercise class on Friday.

Task 1: *State dependent detection probability*

Let us generalize the single target assumption S4 in the book and say we have a variable detection probability in the state space, ie. $\Pr(\delta|x) = P_{\mathrm{D}}(x)$. Let the state distribution be given by $p(x)$. Will knowing that the target has been detected (not the location of the measurement) change the state distribution, ie. $p(x|\delta) \neq p(x)$? In what cases does it seem reasonable to use such a model?

Hint: Use Bayes rule and the total probability theorem for the denominator; $\Pr(\delta) = \int P_{\mathrm{D}}(x)p(x)\,\mathrm{d}x$.

Task 2: *The PDAF event probabilities*

A sensor has N cells. Each cell has the same measurement volume and the same probability, P_{FA}, of giving a false alarm independent of each other (and any correct detections for the sake of simplicity). In addition, we have a track on an object known to be in the sensor volume with the standard linear-Gaussian assumptions on its state estimate.

(a) What is the distribution of the number of false alarms, φ, for a portion of the sensor volume containing M_k sensor cells?

Hint: Simple reasoning and some knowledge from chapter 2 should give the answer without any calculations.

(b) We assume that this can be well approximated by a Poisson distribution with parameter $\lambda V_k = M_k P_{FA}$,

where V_k is the volume of M_k sensor cells. Find the more specific formula for the posterior event probabilities in theorem 7.3.1 under these assumptions. That is, prove corollary 7.3.3 starting with theorem 7.3.1.

Hint: You can take advantage of the proportionality sign to get rid of factors independent of a_k present in both cases or move them from one case to the other.

(c) Use the real distribution under our assumptions (part (a)) as the distribution $\mu(\varphi_k)$ in theorem 7.3.1 and show that the posterior event probabilities can be given as

$$\Pr(a_k|Z_{1:k}) \propto \begin{cases} (1 - P_{\mathrm{D}})\frac{P_{FA}M_k}{V_k}\frac{1-\frac{m_k-1}{M_k}}{(1-P_{FA})}, & a_k = 0, \\ P_{\mathrm{D}}\mathcal{N}(z_k^{a_k}; \hat{z}_{k|k-1}, S_k), & a_k > 0. \end{cases}$$

(d) Compare the formula for the event probabilities using the true distribution under our assumptions with corollary 7.3.3. Does the Poisson approximation seem good when M_k is large and P_{FA} is small?

Hint: Look at $\frac{m_k-1}{M_k} \approx 0$, and $P_{FA} \approx 0$.

Task 3: *IPDA vs PDAF*

(a) Briefly discuss why the posterior becomes a mixture in single target tracking. In particular, what is the interpretation of a component and its weight. What are the main complicating factors of this mixture?

(b) What problem does the IPDA try to solve that the PDA does not? Are there any problems that the PDA solves which the IPDA does not solve?

(c) What are some of the complicating factors in using an IPDA over a PDA?

Task 4: *Implement a parametric PDAF*

Implement a PDA filter in python.

(a) Finish the `filter.FilterPDAF.gate_zs` method.

(b) Finish the `filter.FilterPDAF.get_assoc_probs` method.

(c) Finish the `filter.FilterPDAF.get_estimates` method.

(d) Finish the `filter.FilterPDAF.step` method.

Task 5: *Tune your parametric PDAF*

All the parameters you are intended to change are located in `tuning.py`.

(a) The simulated data is based on the same IMM model as in assignment 4, while the filter is using a CV model as its dynamic model.

For this to work, both the estimated acceleration noise and the gate probability are quite high. Is this a good approach?

(b) What happens when you reduce the model noise and/or the gate probability?

Hint: At some point, there should be a significant behavior change.

(c) If ground truth is not available, would it be possible to detect this behavior change?

(d) Can you suggest a modification to the filter that would make it more robust?

(e) Find a set of parameters that gives some results you find interesting and explain why you think they are interesting. You can change the parameters used for the simulation as well as the parameters used by the filter.

This can for example be a set of filter parameters that yield a smoother trajectory estimate than the one in the handout or a set of simulation parameters that yield an even more challenging scenario for the filter.

Hint: Set the seed to 'None' to generate random paths on every run.