TTK4250 Sensor Fusion

Assignment 4

Hand in: *Friday 4. October 16.00* on Blackboard (as single PDF-file) or to teaching assistants in the 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: *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 the 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. In addition, we have a track with the standard linear-Gaussian assumptions.

(a) What is the distribution of the number of false alarms, φ, for a portion of the sensor 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 to the other.

(c) Use the real distribution under our assumtions (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*

You are to implement a parametric PDAF class in MATLAB using the skeleton PDAF.m found on Blackboard. This class builds on the EKF class you made in exercise 2, and uses an instance of it as an initialization parameter. In addition to the EKF input it also takes clutterRate, PD and gateSize – corresponding to λ, P_{D} and g^2 respectively – as parameters.

Note: the structure is on purpose made general, so that it should be fairly simple to expand to, IMM-(I)PDA and/or their multitarget counterparts (IMM-)J(I)PDA.

(a) Implement the prediction step in the function `[xp, Pp] = predict(obj, x, P, Ts)`.

(b) Implement gating in the function `gated = gate(obj, Z, x, P)`.

(c) Implement the log likelihood ratios (logarithm of corollary 7.3.3 in the book) in the function `ll = loglikelihoodRatios(obj, Z, x, P)`.

(d) Implement the calculation of the association probabilities in the function `beta = associationProbabilities(obj, Z, x, P)`.

(e) Implement the update the state for all possible associations (equations (7.20) - (7.21) in the book) in the function `[xupd, Pupd] = conditionalUpdate(obj, Z, x, P)`.

(f) Implement the mixture reduction step in the function `[xred, Pred] = reduceMixture(obj, beta, x, P)`. Note that for simplicitys sake, we are here neglecting the speedupds that can be achieved in the PDAF using (7.23) and (7.25) in the book.

(g) Implement the combination of the earlier steps to get an overall update function, `[xupd, Pupd] = update(obj, Z, x, P)`.

Task 5: *Tune your parametric PDAF*

Tune your parametric PDAF using a CV model with position measurements on the data given in the .mat file on Blackboard. The trajectory is the same as the one in assignment 2, so you should have some idea of where to start tuning r and q. You have to tune the process noise, the measurement noise, the detection probability, the false alarm rate and the gate size. We recomend looking at position and velocity RMSE, as well as total, position and velocity NEES.

You can initialize the position using the first measurement and the velocity as zero. The covariance for position can then be set to, say, $2R = 2rI_2$ for the sake not making anything uncertain, while the velocity covariance can be set using that the maximum velocity is limited to 20.

Note: We have not enforced the computation of NIS here as there is a complicating factor in using NIS with are several measurements. However, one can, for instance, use the averaged innovation in the standard NIS calculation, $\bar{v}_k = \sum_{a_k=1}^{m_k} \beta_k^{a_k}(z_k^{a_k} - \hat{z}_{k|k-1}) = \sum_{a_k=1}^{m_k} \beta_k^{a_k} v_k^{a_k}$, if one wishes.