TTK4250 Sensor Fusion

Graded Assignment 1 (15% of total grade)

Code and results hand in (5%): *Friday 18. October 16.00* on Blackboard as a single .zip file.
Report hand in (10%): *End of semester (TBA)* on Blackboard as a single PDF file.

Pairs: This assignement is to be solved in pairs. These pairs will be the same for all the graded assignements.

Code hand in: Functioning code that produces your results for task 2, along with a one paragraph describing why you ended up tuning as you did in task 2, should be handed in by 18. October 16.00 as a zip file on Blackboard. This will account for one third of the possiple points for this assignment (5% of total grade).

Report hand in: At the end of the semester you will hand in a report for all the graded assignements combined. The whole report should be maximum 10 pages long, including abstract, introduction and conclusion. This leaves approximately 3 pages per assignment, although this is *up to you*. You should, therefore consider carefully what to have in it. We recomend starting to write this now, when you actually are given some time to do it.

For task 2 and 3, we want to se a plot of the total, position and velocity NEES along with your chosen condfidence bounds for these, and estimation error (Euclidian distance) in position and velocity for the tracker run with your chosen parameters. In addition we want the numbers of the averaged NEESes along with their confidence bounds, the number of times the NEESes fall within the confidence regions and the positional and velocity RMSE (mean taken over time). We then expect you to base your choise of parameters on these values, and compare to other sets of parameters. To show plots of other parameter settings is *up to you*.

Task 1: *Implement IMM-PDAF*

This task is worth up to 3% for the code hand in. Full score is given if the classes are capable of doing the job of the EKF, IMM and IMM-PDA.

You are to implement an IMM-PDA class in MATLAB. This should be similar structured to the PDA class you made in assignment 4, but should take an IMM class instance as initialization input instead of EKF. Another difference is the number and sizes of the different inputs in the different functions. The PDA mixture reduction stage is probably the biggest change where you need to calculate the marginal mode probabilities and the mode conditioned association probabilities before you reduce the mixture for each mode.

The class skeleton to use for this assignment can be found on Blackboard. In addition we provide some notes to how the functions should be implemented

- The predict, associationProbabilities, conditionalUpdate and update functions are very similar to the PDA class.
- The gate function is a bit more complicated than for the PDA class due to the different modes. We shall adopt the rule that the track gates a measurement if at least one of the modes gates this measurement.

  Hint: IMM.NIS should provide some of what you need. Also, the function `any` might come in handy.

- loglikelihoodRatios is also very similar to the PDA class. However, the IMM class does not provide a loglokelihood function like the EKF class, so you would either need to implement a loglikelihood function for the IMM class or use its update function and simply discard some of the outputs. The latter is probably easiest.
- reduceMixture has probably the biggest complications compared to the PDA class. We have the probabilities $\Pr(s_k = i|a_k = j, Z_{1:k})$ and $\Pr(a_k = j|Z_{1:k})$, as well as the densities $p(x_k|a_k = j, s_k = i, Z_{1:k})$. What we want is the probabilities

$$\Pr(s_k = i|Z_{1:k}) = \sum_{j=0}^{m_k} \Pr(s_k = i|a_k = j, Z_{1:k}) \Pr(a_k = j|Z_{1:k}) \tag{1}$$

(the total probability theorem), as well as the densities

$$p(x_k|s_k = i, Z_{1:k}) = \sum_{j=0}^{m_k} p(x_k|a_k = j, s_k = i, Z_{1:k}) \Pr(a_k = j|s_k = i, Z_{1:k}) \tag{2}$$

$$= \sum_{j=0}^{m_k} p(x_k|a_k = j, s_k = i, Z_{1:k}) \frac{\Pr(s_k = i|a_k = j, Z_{1:k}) \Pr(a_k = j|Z_{1:k})}{\Pr(s_k = i|Z_{1:k})} \tag{3}$$

(first the total probability theorem and then Bayes on the probabilities). As usuall we approximate $p(x_k|s_k = i, Z_{1:k})$ by a single Gaussian that matches the first two moments.

Task 2: *Tune the IMM to the given data*

This task is worth up to 2% for the code hand in, and up to 4% in the report. Full score is given in the code/result hand in for a reasonable (not necessarily optimal) choice of parameters that works on the data set, along with one paragraph/10 lines describing why you chose these. In the report you should do a bit more analysis of what happens with different parameters settings, and base your choice on relevant plots and numbers. To achieve a full score we also want to see at least some minor reflections on the algorithm and the approximations it does in light of your results.

You are given a data set on Blackboard in the file "task2data.mat". This corresponds to the same scenario as in assignment 3, but with added false alarms and missed detections removed. It contains

- K (natural number): the number of time steps.
- Ts (positive scalar): the sampling time.
- Xgt (5 x K): the state ground truth, where the row order is x-y position, x-y velocities and turn rate
- Z (K x 1 cell): a cell of measurements. Z{k} (2 x m_k): the m_k measurements at time step k
- a (K x 1): the true association.

Tune you IMM-PDAF, using a CV model and a CT model with position measurements to this dataset. You need to tune the process disturbances of the models, the measurement noise, the detection probability, the false alarm intensity and the gate size.

Hint: Since this is the same scenario as in assignment 3, you should already have a good starting point for the process noises and measurement noises, as well as means to investigate the tuning of these parameters without the problems that false alarms and missed detections introduce.

Task 3: *IMM-PDAF on the real radar data set "Joyroide"*

This task is worth up to 6% in the report hand in. We want to see an analysis of what happens and why with some different parameter/model settings, and base your choice of model and parameters on relevant plots and numbers. To achieve a full score we also want to see at least some minor reflections on the algorithm and the approximations it does, in light of your results. Relating results in this tasks or assignments to the previous task can give additional points.

You are given a data set on Blackboard in the file "joyridedata.mat". This corresponds to the real world "Joyride" dataset partially described in the book and lecture slides. It is collected with a RADAR mounted on a moving boat, where a quite agile boat that we want to track is doing a "joyfull ride" inside the field of view of the RADAR. The RADAR measurements have been processed into single points in cartesian coordinates per RADAR scan. The dataset consists of

- K (natural number): The number of time steps in the data set
- time (1 x 200): the time when the measurements were captured
- Ts (1 x 199): the sampling time between the measurements (calculated using `diff(time)`)
- Xgt (4 x 200): "ground truth" as measured by GPS. x-y position followed by x-y speed in the rows. Note that this is not necessarily a perfect ground truth.
- ownship (4 x 200): the RADAR position and speed at each time step. Same layout as for Xgt.
- Z (200 x 1 cell array): a cell of measurements. Z{k} (2 x m_k): the m_k measurements at time step k

Tune your PDAF and/or IMM-PDAF to this data set. Experiment with using a EKF (CV and CT) and IMM (CV – CT and CV – CT – CVhigh) as your state estimators. This is a challenging data set, and we have not been able to get some of these model configurations to work satisfactory at all (read section 7.6.2).

Hint: A reasonable, not necessariliy optimal, value for the measurement noise covariance is found to be $r = 6^2$, but also $r = 10^2$ can give good results.