TTK4250 - Sensor Fusion

Group 42
Martin Albertsen Brandt
Martin Eek Gerhardsen

November 27th 2019

Department of Engineering Cybernetics

Abstract

This report will highlight and discuss the results of the three graded assignments for the course TTK4250 Sensor Fusion. The code implemented was tested and tuned for both simulated and real datasets.

Contents

1 Introduction

2 Graded Assignment 1

The IMM-PDAF combines Interactive Multiple Models (IMM) with Probabilistic Data Association Filter (PDAF). Relevant theory can be found in [1, p. 100-101, 120 – 122].

Results

Simulated dataset

Tuning the full system for the simulated data required tuning of, in total, four systems. These were the Extended Kalman Filters, for the Constant Velocity and Constant Turn-rate modes, the IMM, and the IMM-PDAF.

For the CV-model, the measurement noise covariance parameter used was 4, while the acceleration noise covariance parameter used was 8×10^{-2}. Similarly, the same measurement noise covariance parameter was used for the CT-model, as the measurement noises should be similar. Furthermore, the acceleration noise covariance parameter used was 5×10^{-3} and the turn-rate noise parameter used was 3×10^{-4}. Tuning these parameters would have some impact on the consistency, higher noise would lower the NEES, while lower noise increases the NEES, which is to be expected as we are essentially changing how much the filter should trust the measurements contra the estimates. From fig. 1c, it can be seen that both models handle their trajectories well.

The IMM was tuned using the transition matrix π, which was tuned as eq. (1). With the CV-model as the first model and the CT-model as the second, the argument is that if the IMM is in a given mode, it should most likely stay. Still, it may be beneficial to be more likely to change to, and stay as, the CT-model. When observing the trajectory, see fig. 1c, it seems to primarily use the CV-model, and then increasing the transition probability for changing to the CT-model would speed up this transition.

$$\pi = \begin{bmatrix} 0.92 & 0.05 \\ 0.08 & 0.95 \end{bmatrix} \tag{1}$$

Finally, for the IMM-PDAF, the clutter rate λ, detection probability P_D and the validation gate were tuned. The clutter rate was tuned to $\lambda = 1 \times 10^{-4}$, and as there were very few false alarms, this could be chosen as a low value. Furthermore, as the only place the clutter rate is used in the IMM-PDAF is to scale the probability of no detection, there was really no use further decreasing λ, as it already would put the association likelihood ratio for no detection very low. The detection probability is tuned to be rather high, with $P_D = 0.95$, as it is expected that there is a good chance of detection for this dataset. A validation gate size of 5^2 seemed also to give decent results. Increasing it would generally not make the tracker worse overall, but would increase the magnitude of certain errors, which would then spike. Decreasing it would of course make it more difficult for any measurements to be gated, such that this value seemed to be the best compromise between gating all and no measurements. The resulting consistency plot can be seen as fig. 1a, and the average NEESes were for position 1.9248, for velocity 1.1427 and overall 3.0887.

Joyride dataset

Initially, the IMM-PDAF for the Joyride dataset was tuned similarly to the tracker for the simulated dataset, and especially for the low noise CV model and the CT model, similar values were used. Furthermore, it should be noted that the Joyride dataset also was noisier, such that more lenient tuning was necessary. For this dataset, a high noise CV model was also included, which could *take over* if both the low noise CV model and the CT model were uncertain. Otherwise, the overall structure of the tracker is the same as for the simulated dataset.

The measurement noise covariance parameter was again the same for the CT and two CV models, and was set to 25^2. Though this seemed rather higher than necessary, however lowering it would especially push higher the positional NEES. Similar responses were generated from tuning the acceleration noise covariance parameters and turn-rate noise covariance parameter for the different models. Then, for the low noise CV model, the acceleration noise covariance parameter was tuned to 0.5, for the high noise CV model, the acceleration noise covariance parameter was tuned to 20 and for the CT model, the acceleration noise covariance parameter was tuned to 1×10^{-4} and the turn-rate noise covariance parameter was tuned to 2×10^{-4}. Similarly to the results from the simulated data, these models handle their trajectories well, as can be seen from fig. 1d.

As for the IMM used in the simulated dataset, the transition matrix π was tuned to prefer to stay in the CT-mode, so that the tracker should prefer to continue a turn if it has begun. Since the high noise CV model was included primarily to handle high noise areas where it could be unambiguous whether to use the low noise CV or CT model, this transition probability was set lower than the others, as those should be preferred. From observing the track and measurements, there also seemed to be unambiguous which of the other modes each should transition to, and as such these transitions were set to be equally likely, see eq. (2).

$$\pi = \begin{bmatrix} 0.900 & 0.025 & 0.075 \\ 0.050 & 0.950 & 0.075 \\ 0.050 & 0.025 & 0.850 \end{bmatrix} \tag{2}$$

The IMM-PDAF was tuned to have a lower clutter rate and higher detection probability than for the simulated data, while keeping the gate size. The clutter rate was tuned to be $\lambda = 5 \times 10^{-5}$. Higher values would inevitably make the tracker loose track and converge to one mode, rendering the tracker useless. As the clutter rate affects the likelihood ratio for no detection, a higher clutter rate implies a higher likelihood compared to a lower clutter rate, and a higher value would be given to the association probability for no detection. The detection probability was tuned to $P_D = 0.97$, which is a rather high value. It was possible to reduce P_D and, specifically, improve the percentage where the NEES is inside CI, but at a cost of the NEES for the velocity, and a higher P_D was chosen so that the NEESes would be more similar.

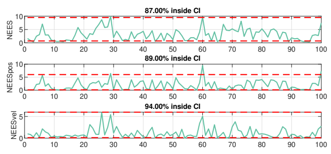
The resulting consistency plot can be seen fig. 1b, and the average NEESes were found as 1.6653 for position, 1.3966 for velocity and 3.4346 overall. Both from the plots and the averages tend to be closer to the lower bound of the confidence interval, though as can be noted, there were certain spikes in the plot. These were difficult to as they seemed to originate from especially bad measurements, and a compromise was made where the consistency would tend to be low, in exchange for limiting the spikes.

Assumptions

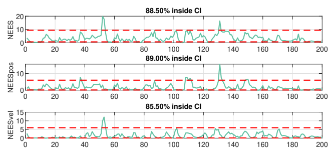
The IMM-PDAF works under the single-target assumptions, as seen from [1, p. 105], which all seem reasonable considering both the simulated and real datasets. In the simulated dataset, there is initially good reason to suspect that these assumptions were considered when the dataset was generated, and it may therefore be more interesting to consider the real dataset. As there are a smaller number of measurements for each time step than there seemed to be for the simulated dataset. Therefore, it may seem like the job of the tracker for this dataset is, primarily, to consider whether there is a detection or not. In certain cases, the measurements are also pretty far away from each other and the track, which could also make it easier for the tracker. Since it is likely that several raw measurements may originate from the sensor, like a RADAR, it is common to apply some form of clustering to reduce the measurements from the target to one single measurement. Depending on the method for clustering, this is very likely to make the tracker work, and it is computationally a good idea to use the single target assumption, though this clustering may also cluster bad measurements as well, which may make the tracking more difficult. This was not further explored here, as there doesn't seem to be a need for it.

There are certain simplifications done in this algorithm to make it more viable, and as mentioned above, assuming only one measurement may originate from the target allows for a more computationally efficient algorithm.

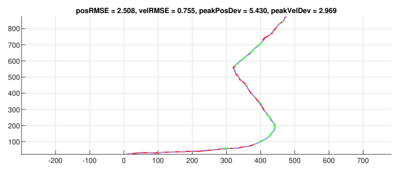
Additionally, as the associations (and modes) are seen as mixtures, then as to not end up with an extremely large number of values, the mixtures can be reduced by moment matching. This also makes the tracker a more com-



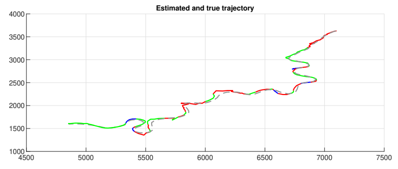
(a) NEES for simulated data



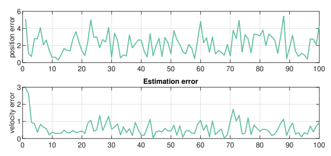
(b) NEES for Joyride



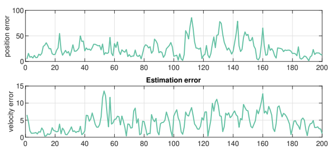
(c) Trajectory for simulated data



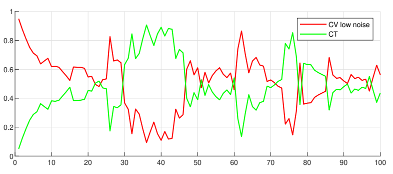
(d) Trajectory for Joyride



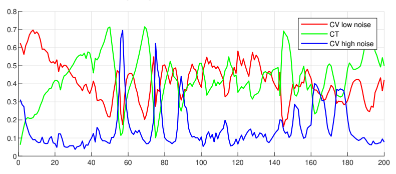
(e) Errors for simulated data



(f) Errors for JoyriThe results of this tuning gave de



(g) Probabilities simulated data



(h) Probabilities for Joyride

Figure 1: The same colour code was utilised for visualising the active modes in the trajectories as the probabilities. For the trajectories, ground truth is represented as a dashed line.

putationally efficient algorithm, and keeps the memory usage manageable.

Gating of the measurements, is done, for each measurement, by checking the NISes for all modes given this measurement. If one is gated, it is gated for all. The assumption seems to be good, as it then seems like at least one of the models can *accept* this measurement. There may still be cases where a stricter gating rule could've been considered, but for these datasets, there haven't been any specific cases where this has caused problems.

Certain simplifications were also made to make the code easier to implement, which probably slows the tracker slightly down, as when calculating the log-likelihood ratios, the *update*-method from the IMM was utilised, and most of the results were discarded. If there was a need to save a some small amount of time, or reduce the number of operations done, there are some places in the code where one could save time. This could be necessary if there was a need to run the tracker online.

Using the EKF, with the CV and CT models, is also a simplification, especially for the joyride dataset. For the simulated dataset, these models were probably used to generate the dataset, it is possible to get very good result using these models, as seen from fig. 1c. The joyride, however, is different, as there is not *really* an underlying model which generated this dataset. Still, by using several simpler models, decent tracking was achieved for the joyride dataset as well, as can be seen from fig. 1d.

3 Graded Assignment 2

An error-state Kalman filter was implemented in MATLAB. For the relevant theory behind the implementation, see [2] and [1].

4 Graded Assignment 3

5 Conclusion

References

[1] E. Brekke. *Fundamentals of Sensor Fusion*. 2019.

[2] J. Solà. *Quaternion kinematics for the error-state KF*. `http://www.iri.upc.edu/people/jsola/JoanSola/objectes/notes/kinematics.pdf`. 2017.