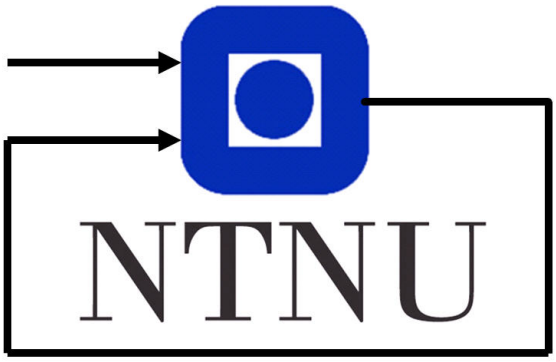
TTK4250 - Sensor Fusion Report

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 [2, p. 100-101, 120 – 122].

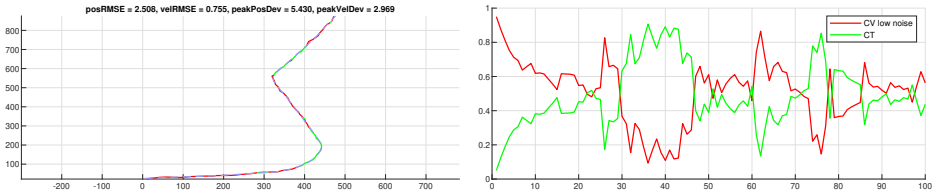
2.1 IMM-PDAF for simulated data set

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. 1a, 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. 1a, 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. The probabilities over time can then be seen from fig. 1b.

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

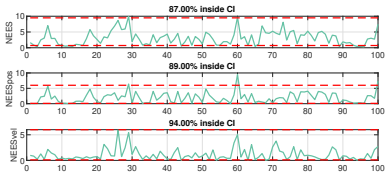


(a) Trajectory for simulated data (b) Probabilities simulated data

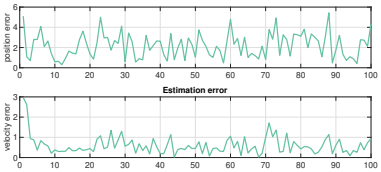
Figure 1: The same colour code was utilised for visualising the active modes in the trajectory as the probability. For the trajectory, ground truth is represented as a dashed line.

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 data set. 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. 2a, and the average NEES were for position 1.9248, for velocity 1.1427 and overall 3.0887. The errors seem generally ok, and limited in amplitude, as seen from fig. 2b.



(a) NEES for simulated data



(b) Errors for simulated data

Figure 2: The NEES and errors from the simulated data.

2.2 IMM-PDAF for joyride data set

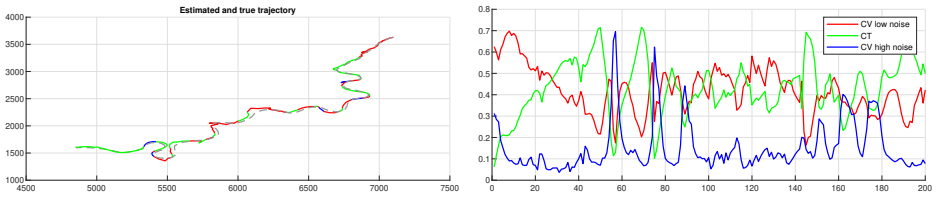
Initially, the IMM-PDAF for the Joyride data set was tuned similarly to the tracker for the simulated data set, and especially for the low noise CV model and the CT model, similar values were used. Furthermore, it should be noted that the Joyride data set also was noisier, such that more lenient tuning was necessary. For this data set, 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 data set.

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 the positional NEES too high. 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. 3a.

As for the IMM used in the simulated data set, the transition matrix π was tuned to prefer to stay in the CT model, so that the tracker should prefer to continue a turn if it was already turning. 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). The resulting changes to the probabilities can be seen from fig. 3b.

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

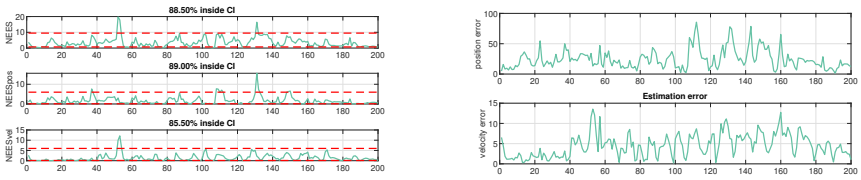


(a) Trajectory for joyride data (b) Probabilities joyride data

Figure 3: The same colour code was utilised for visualising the active modes in the trajectory as the probability. For the trajectory, ground truth is represented as a dashed line.

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. 4a, and the average NEESes were found as 1.6653 for position, 1.3966 for velocity and 3.4346 overall. Both 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 handle 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. The errors also seemed to be acceptable, see fig. 4b.



(a) NEES for joyride data (b) Errors for joyride data

Figure 4: The NEES and errors from the joyride data.

2.3 IMM-PDAF discussion

The IMM-PDAF works under the single-target assumptions, as seen from [2, p. 105], which all seem reasonable considering both the simulated and real data sets. In the simulated data set, there is initially good reason to suspect that these assumptions were considered when the data set was generated, and it may therefore be more interesting to consider the real data set. There are a smaller number of measurements for each time step than there seemed to be for the simulated data set, and therefore, it may seem like the job of the tracker for this data set 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, see fig. 5 to see the joyride measurements used. 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 that 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 computationally 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 a measurement is gated for one mode, it is gated for all modes. This 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 have been considered, like only applying the measurements to the specific modes with NISes within the gate size. For these data sets however, there haven't been any specific cases where this assumption has caused problems.

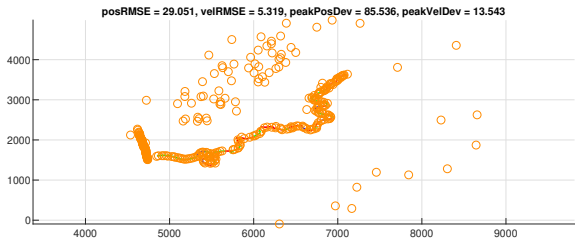


Figure 5: Measurements and estimated trajectory for the joyride data set.

Using the EKF, with the CV and CT models, is also a simplification, especially for the joyride data set. For the simulated data set, these models were probably used to generate the data set. It should therefore be possible to get very good result using these models, as seen from fig. 1a. The joyride data set, however, is different, as there is not *really* an underlying model which generated this data set. Still, by using several simpler models, including a high-noise mode for robustness, decent tracking was achieved for the joyride data set as well, as can be seen from fig. 3a.

3 Graded Assignment 2

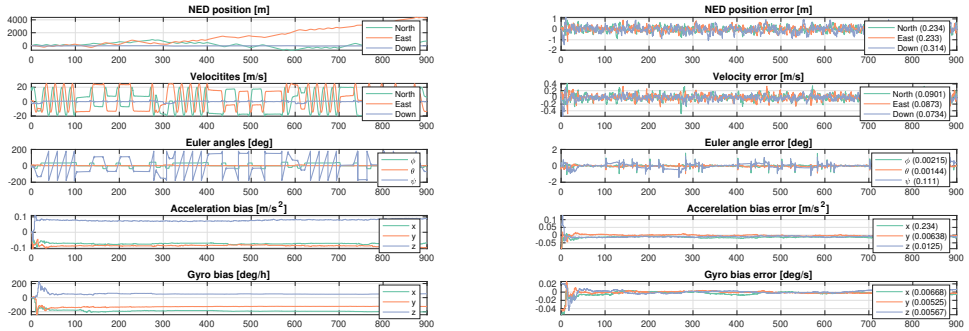
An error-state Kalman filter (ESKF) for a GNSS-aided fixed-wing UAV was implemented in MATLAB. The implementation is based on the standard formulation in [3]. But most notably, IMU sensor correction matrices has been added to counteract any mounting errors, scaling errors and orthogonality errors in the accelerometer and rate gyro. Furthermore, lever arm compensation for the GNSS receiver is also implemented.

3.1 INS for simulated fixed-wing UAV

The ESKF was first tuned to a simulated data set. The GNSS measurement standard deviation was tuned to 0.4 in each degree of freedom. The measurement noise covariance is therefore $R = 0.16I^2$. For the accelerometer the measurement noise covariance and bias driving noise covariance was tuned to be $q_a = (4 \times 10^{-2})^2$ and $q_{ab} = (1 \times 10^{-3})^2$ respectively. Similarly for the rate gyro the measurement noise covariance and bias driving noise covariance was tuned to be $q_\omega = (8 \times 10^{-4})^2$ and $q_{\omega b} = (1 \times 10^{-6})^2$ respectively. Finally the time constants in the Gauss-Markov bias processes were both tuned to be $T_b = 1 \times 10^8$ s.

Firstly, the tuning was based on common sense. For instance the GNSS measurement covariance was initially based on a reasonable guess from a physical perspective and then more thoroughly tuned. This more thorough tuning was based on calculating and plotting the errors in position, velocity, attitude and bias. Furthermore the corresponding NEES in position, velocity, attitude and biases, as well as the overall NEES and NIS was calculated and analysed during the tuning process.

The resulting position estimate is plotted in fig. 7a, the state and state errors are presented in fig. 6a and fig. 6b and finally the consistency analysis is presented in fig. 7b. Note that the consistency figures are presented in logarithmic scale. Furthermore, the average NEES and NIS is approximately 18.57 and 2.27 respectively.

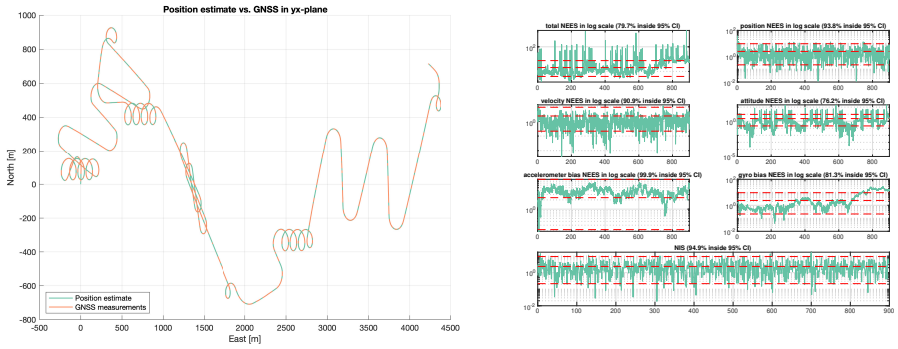


(a) UAV ESKF states

(b) UAV ESKF state errors

Figure 6

The ESKF was in particular tuned such that the bias states converged nicely, such that the errors propagated in the system from sensor drift were minimised. It should be emphasised



(a) Estimated UAV trajectory (b) UAV ESKF consistency analysis

Figure 7: Estimated UAV trajectory and consistency analysis

that one of the main features that makes the ESKF robust is the IMU bias estimation, and it was therefore also emphasised in the tuning process.

The consistency was also emphasised when tuning the ESKF. We see from fig. 7b that most of the different normalized square errors are all in the 80% or 90% range inside the calculated confidence intervals, which was found to be satisfactory. One should especially note that the attitude NEES is only about 76% inside the confidence interval. Studying the heading error in fig. 6b can give some insight into why this might be the lowest NEES. It is observed that the heading is at certain time intervals far off the ground truth, which is also confirmed by the RMSE of 0.111. This is about two magnitudes higher error than for pitch and roll. This is caused by the fact that pitch and roll are directly observable from the gravitational force acting on the accelerometer. Heading, on the other hand, is only observable when the system is exited by a manoeuvre. The heading error, therefore, has certain sections where it is unobservable and the update step is not able to correct the constant offset from the true value. This is further supported by studying the time intervals when the UAV is doing a spiral manoeuvre e.g. from about 300 to 400 seconds or from 600 to 700 seconds. In these intervals the heading is observable and as a consequence, we observe that the error drops to the same magnitude as the pitch and roll, before it starts acting unexpectedly again when the UAV starts flying straight. This issue might be why both the attitude NEES and total NEES is somewhat worse than the others. So the heading estimation would therefore not work if the UAV was standing still.

When letting the sensor correction matrices be identity matrices, and thereby removing any sensor correction, the ESKF estimation performance is noticeably worse. While the filter is still able to reliably estimate velocity and position with the aid of the GNSS, the estimates of heading and sensor biases are much worse. Most importantly, the rate gyro and accelerometer biases no longer converge to the true sensor biases. This means we are not able to remove the bias in the measurements, and as a consequence, they are integrated and propagated through the Kalman filter. This makes all the RMSEs significantly worse, and especially the heading now drifts quickly when the UAV is not turning.

Another consequence is that the NEES and NIS also worsens significantly. While the position and velocity NEES is still somewhat within the confidence intervals at 85.2% and 64.9% respectively, the attitude, accelerometer bias and gyro bias is at 3.09%, 14.6% and 2.89% respectively. This further supports the claim above about how important IMU bias compensation is when trying to design a robust attitude estimator. It is then evident that

correction for sensor misalignment, scaling errors and orthogonality errors are a critical part of designing such an estimator. Neglecting the GNSS lever arm, on the other hand, has no noticeable effect on the estimator performance.

Finally, it must be said that finding the sensor correction matrices is not trivial, and the results will be empirically based and not absolutely correct. Therefore we will always have some sensor misalignment and miss-scaling in a physical system, especially since these matrices will, in reality, be time-varying.

3.2 INS for real fixed-wing UAV

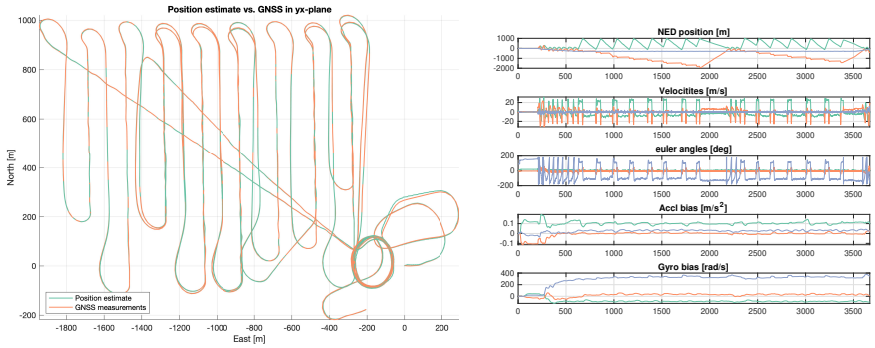
The ESKF was then tuned to a real data set from a fixed-wing UAV flight. The sensors on board was a STIM300 IMU and an u-blox M8 GNSS receiver. The GNSS receiver produces an accuracy estimate in standard deviation while operating. This was scaled by a gain parameter, squared and used as a time-varying GNSS measurement noise covariance signal. The gain was tuned to be $k_x = 0.5$.

The IMU measurement noise covariances and bias driving noise covariances were tuned from the Allan variances in the STIM300 datasheet [4]. Specifically, the gyro noise and bias noise root Allan variance is $0.15\,\mathrm{deg}/\sqrt{\mathrm{h}}$ and $0.5\,\mathrm{deg}/\mathrm{h}$ respectively. The accelerometer noise and bias noise root Allan variance is $0.06\,\mathrm{m/s}\sqrt{\mathrm{h}}$ and $0.05\,\mathrm{mg}$ respectively. These values were scaled to the correct units and used as initial parameters in the ESKF. The values were then tuned such that a satisfactory estimate and NIS were produced. The final tuning parameters were then $q_a = (9.8 \times 10^{-4})^2$, $q_{ab} = (4.9 \times 10^{-4})^2$, $q_\omega = (4.9 \times 10^{-6})^2$ and $q_{\omega b} = (2.4 \times 10^{-6})^2$. When comparing to the simulated case, one sees that the bias noise standard deviations are of about the same magnitude, while the measurement noise standard deviations are about two magnitudes larger in the simulated case.

The resulting UAV trajectory estimate is presented in fig. 8a and the estimated states are presented in fig. 8b. Finally the NIS is presented in fig. 9. The average NIS was approximately 3.01. Note that since this is a real data set, we do no longer have the ground truth, and can therefore not calculate the estimation errors and NEES. This certainly makes tuning more challenging, even though access to the IMU datasheet is useful. This shows how it is often helpful to test filter implementations on simulated data first, as has been done in all of the assignments discussed in this report. The Gauss-Markov bias time constants were both tuned to $T_b = 1 \times 10^{16}\,\mathrm{s}$, effectively making the bias processes random walks processes.

We observe how the estimated trajectory fits nicely with the GNSS data as one would expect. The estimates are also nicely behaved, while the state trajectories are naturally a lot less smooth in this case compared to the simulation, which is to be expected. In particular, the biases converge over time, with some corrections throughout the flight when the UAV turns and the system is excited. From the heading plot it seems like the bias estimates work well, as there is no noticeable drift in heading. For most of the flight, we see that the heading changes between two almost constant values. This is reflected by the UAV manoeuvres in fig. 8a, where the UAV flies straight north, then quickly does a U-turn, then flies straight south, then does a U-turn and so on.

When the sensor correction matrices were set to identity for the real data, it was observed



(a) Estimated UAV trajectory (b) UAV ESKF states

Figure 8

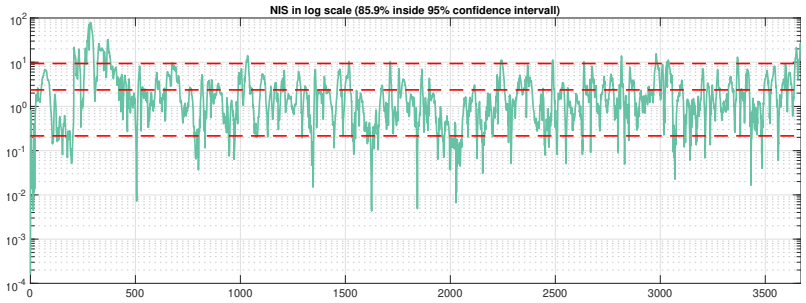


Figure 9: UAV GNSS logarithmic NIS

again that thanks to the GNSS measurements the position and velocity estimates are still viable. But the biases do not converge to the same values as before. Interestingly, the heading is now mirrored around zero, and the roll and pitch estimates are *swapped*, as can be seen in fig. 10. This is due to the fact that the sensor frame is no longer approximately aligned with the body frame. While in the simulated case the sensor correction matrices were close to identity, for this data set they are

$$S_g = \begin{bmatrix} -0.0243 & 0.9981 & 0.0454 \\ -0.9998 & -0.0242 & -0.0255 \\ -0.0272 & -0.0532 & 0.9949 \end{bmatrix}, \quad S_a = \begin{bmatrix} -0.0503 & 1.0021 & 0.0476 \\ -1.0243 & -0.0695 & -0.0197 \\ -0.0387 & -0.0483 & 1.0005 \end{bmatrix}. \tag{3}$$

Now notice that $S_g \approx R_z(-\frac{\pi}{2}), S_a \approx R_z(-\frac{\pi}{2})$. This means that in addition to small scaling, misalignment and orthogonality errors there is a 90° rotation that is needed in order to convert the measurements from sensor to body frame. When implementing an ESKF, and even more generally some sensor fusion algorithm in a physical application such as this, one obviously needs to consider that very rearily will sensors be mounted along the body axes, and a sensor to body transformation is therefore absolutely necessary for the system to work. Luckily, it is easy to see from the estimates or even the measurements themselves that they do not correspond to what common sense it telling you e.g. in this case that the heading estimate points southwards when the plane is flying north and vice-versa.

3.3 ESKF discussion

One could argue that the benefits of the ESKF are twofold. Firstly, since the error dynamics are approximately linear, the optimality of the linear Kalman filter, which is lost

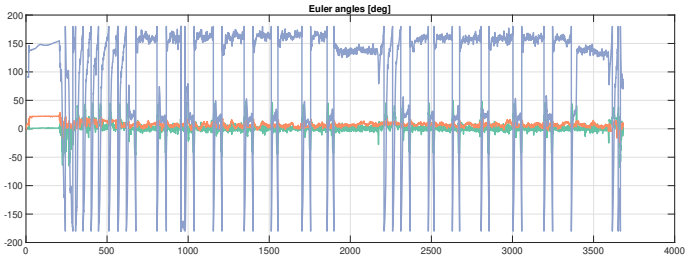


Figure 10: UAV estimated attitude without sensor error correction

when extending it to non-linear process and measurement models with the EKF, are (approximately) retained. Secondly, since the IMU is treated as a control input and the IMU biases are estimated, we achieve robustness, as has previously been discussed. In the test implementations in this assignment, with both simulated and real data sets, this has been observed. By masking modelling errors, sensor errors, uncertainties and actual sensor biases within these states, the filter system is able to operate in a dynamic environment where the plant performs various different manoeuvres, even with wind disturbance and temperature fluctuations.

One could argue that the main limitation of the ESKF for inertial navigation is its slow convergence for deficient initial attitude values. In these tests of the filter, this has not been an issue, as good initial values were provided. One should note that this is not always the case. In many real-time applications, you cannot expect to turn on your INS and have good initial estimates of the attitude. If one tries to give an arbitrary attitude initialization to the ESKF, one observes poor estimates that are not at all usable. Due to length limitations this cannot be further explored here, but nevertheless this is a problem that could be further analysed and extensions to the ESKF could be added to handle this e.g. with optimization methods.

4 Graded Assignment 3

An extended Kalman filter was implemented for solving the SLAM problem in MATLAB. Specifically, the EKF-SLAM formulation in this report considers only the 2D case, with odometer measurements acting as control input. Furthermore, data association is done with the JCBB algorithm. For the relevant theory behind this EKF-SLAM formulation and JCBB, consult [2, p. 185 - 196].

4.1 EKF-SLAM on simulated vehicle data set

The EKF-SLAM was first tuned using a simulated vehicle data set. The odometer measurement noise was tuned according to $Q = \text{diag}(\ [(\sigma_u^2 \quad \sigma_v^2 \quad \sigma_\varphi^2]^\top)$, where $\sigma_u = \sigma_v = 1 \times 10^{-1}\,\text{m}\,\text{s}^{-1}$ and $\sigma_\varphi = 1 \times 10^{-2}\,\text{rad}$. The range-bearing measurement noise covariance was tuned according to $R = \text{diag}(\ [\sigma_r^2 \quad \sigma_\theta^2]^\top)$, where $\sigma_r = 4 \times 10^{-2}\,\text{m}$ and $\sigma_\theta = 2 \times 10^{-2}\,\text{rad}$. The JCBB algorithm was tuned by letting the individual compatability significance level be $\alpha_{ic} = 1 \times 10^{-5}$ and the joint compatabiltiy significance level be $\alpha_{jc} = 1 \times 10^{-10}$.

The system was tuned by trying to minimize the errors in pose and landmark estimates, while also looking at the calculated total NIS and NEES in pose. Note that in the NIS case the confidence interval was calculated at each time step, using the size of the current innovation as the number of degrees of freedom in the χ^2 distribution. The estimated pose trajectory and landmark positions are presented in fig. 11, and the consistency analysis plots are presented in fig. 12. We observe that 90% of the NIS is inside the calculated the confidence interval, while 94.6% of the NEES is inside the confidence intervals.

When tuning the α-values for the JCBB, the interpretation of these values are important. The individual compatibility α-value is used as the validation gate for the landmarks. Any measurements outside the validation gate would not be considered. For the individual compatibility found for the simulated data, $\alpha = 1 \times 10^{-3}$, which gives a validation gate $\approx 3.7^2$. The joint compatibility α-value, on the other hand, is used for gating the NIS-value for

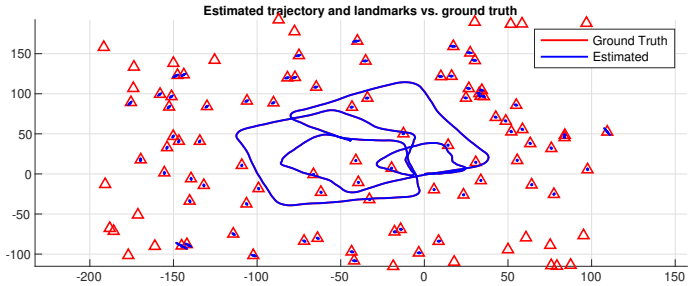


Figure 11: Estimated and ground truth pose trajectory and landmarks for simulated vehicle data

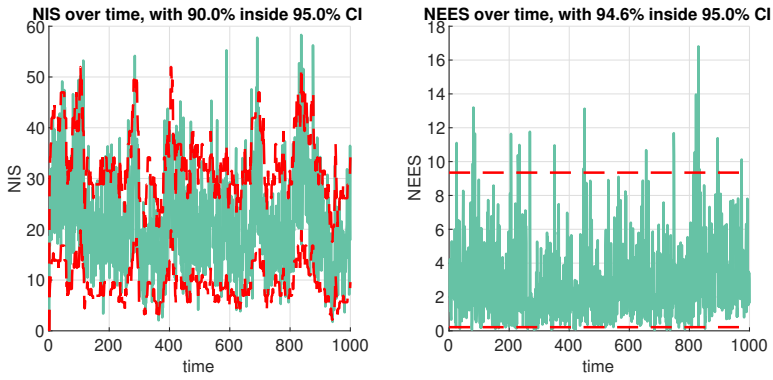


Figure 12: Consistency for simulated vehicle data set with confidence intervals over time

Initially, the noise covariance matrices were set higher than necessary, to get a conservative result, and decreased until sufficient results were produced. At the same time, the compatibility alphas in the JCBB algorithm were reduced, as we could assume most of the measurements were from real landmarks.

In fig. 13 the covariance matrix and the inverse covariance matrix i.e. the information matrix at the last time step is presented. One should firstly note the denseness of the covariance matrix. This is a result of the fact that for the SLAM problem, all the states are highly correlated. We are trying to estimate the pose of the vehicle and the position of the landmarks relative to it simultaneously, which means that when we get additional information about the pose or a landmark from a measurement, we indirectly get infor-

mation about all the other states as well. This means the covariance matrix will be very dense. Furthermore, we observe some additional structure in the covariance matrix, from the fact that landmarks that are close to each other will be more correlated than landmarks at different ends of the map. Finally, we also observe two outliers in the bottom right part of the diagonal. This might correspond to some of the landmarks in fig. 11 with the largest covariance ellipses. These landmarks are far away from the trajectory of the vehicle and have therefore not been measured many times during the simulation. It then follows that they will be more uncertain than the rest and we get a few spikes in the covariance matrix.

Now consider the inverse of the covariance matrix i.e. the information matrix. When the covariance matrix is dense, the inverse matrix will naturally be sparse. As seen in fig. 13 it is even approximately diagonal. This is a useful fact that could be used to reformulate the EKF equations in terms of the information matrix. This would be an interesting extension of the algorithm that might reduce computation time drastically. This reformulation is known in the literature as SEIF SLAM.

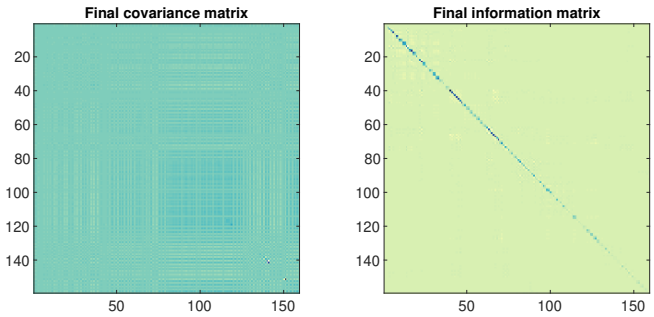


Figure 13: Covariance matrix and information matrix for final timestep

4.2 EKF-SLAM on Victoria Park data set

The EKF-SLAM code was then tested and tuned to the Victoria Park data set. The range-bearing measurement noise covariance was tuned with the same diagonal matrix, now with $\sigma_r = 5 \times 10^{-2}\,\text{m}$ and $\sigma_\theta = 5 \times 10^{-3}\,\text{rad}$. The odometer measurement noise covariance matrix was now tuned to be

$$Q = \begin{bmatrix} 0.25 & 0 & 0 \\ 0 & 0.25 & 0.0225 \\ 0 & 0.0225 & 0.0025 \end{bmatrix}. \tag{4}$$

Notice how for this data set a high correlation between the measured sideways displacement v and the measured heading φ has been added, which can be seen empirically from the data. The JCBB individual and joint compatability significance levels were now tuned to be $\alpha_{ic} = 1 \times 10^{-3}$ and $\alpha_{jc} = 1 \times 10^{-3}$.

The system was largely tuned by the same process as for the simulated data, but there are several differences worth mentioning. Firstly, the pose ground truth is naturally no longer available, only GPS position measurements. This was plotted in relation to the estimated pose trajectory and landmarks, as seen in fig. 14. But there is no easy way to properly compare the estimate to the GPS measurement, as the EKF-SLAM algorithm only produces a

local estimate in relation to how it was initialized. In fig. 14 the GPS measurements were therefore rotated until the two different frames looked somewhat aligned.

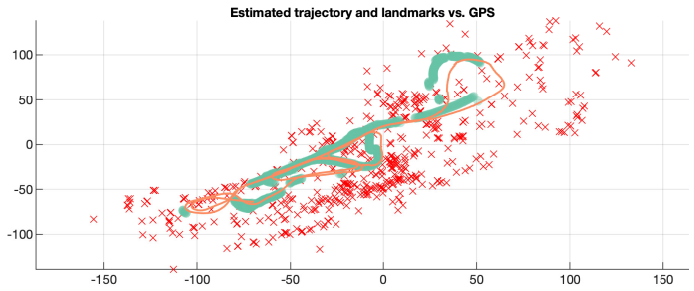


Figure 14: Estimated pose trajectory, landmarks and GNSS measurements for Victoria Park data set

Since we no longer have the pose ground truth, the pose NEES is no longer available which limits the available tools when tuning. But the NIS, again with a variable degree of freedom confidence interval, was actively used and is presented for the final tuning parameters in fig. 15. It was calculated that 67% of the NIS was inside the confidence intervals.

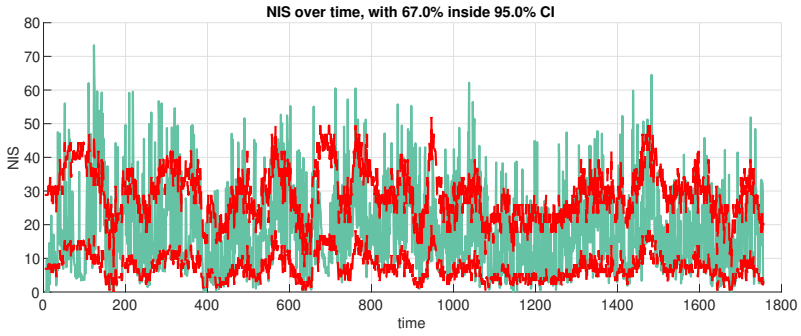


Figure 15: NIS for Victoria Park data set with confidence intervals over time

One should also note the much higher compatibility alphas compared to the simulated data set. It was quickly observed while testing the system that the spread in range-bearing measurements of the landmarks was larger compared to the simulated case. It was concluded that the JCBB gating that was previously used was too low for this data set, and therefore increased.

Another factor that influenced this tuning choice is the fact that each measurement that is not a part of the optimal choice with the most association matches from the JCBB algorithm are treated as new landmark states and added to the state vector. By increasing the individual and joint compatibility gate sizes we give JCBB more leeway, thereby reducing the number of new states added as the EKF-SLAM algorithm runs. Since, as previously discussed, the covariance matrix tends to be dense and very large, with a dimension in the hundreds, it is desirable to reduce the number of new states the algorithm adds to reduce computation time. It should be noted that JCBB will naturally use more time when the alphas are increased, as more association hypotheses must be considered. So how effective this strategy is in saving computation time is questionable, but it will slow down the growing state vector size that slows the algorithm down more and more over time, which was the primary concern here.

4.3 EKF-SLAM discussion

When considering the performance of EKF-SLAM in these tests, one could argue that the performance is satisfactory, but several limitations are evident from the results. The first limitation, which applies to SLAM in general, is the fact that the estimated pose and landmark positions are only local estimates. While it is most important for the robot or vehicle to know its location relative to the environment, one can imagine several applications where global estimates are needed and this algorithm not being sufficient. One could remedy this by adding additional sensors to the system like a magnetometer, or simply making sure to initialize the system in the global frame.

Another limitation is poor consistency, as a result of the fact that we are using an EKF which linearizes around the current state. For the simulated data set the NEES did not evolve to be too large over time, but never the less it is a valid concern that the EKF will be too confident over time[1]. One improvement to this is to formulate the EKF equations in a robocentric view. Alternatively, particle filter approaches or nonlinear observer theory can be used.

Perhaps the more important limitation of EKF-SLAM is the memory and processing power requirements for the algorithm in large scale applications. Observe that the number of calculations and needed memory will scale quadratically for this algorithm. With the relatively small scale test using the Victoria Park data set, the script used about 10 minutes to finish $N = 15000$ time steps, which corresponds to about 6,6 minutes of data, on a 2,7 GHz Intel Core i5. So for even this small scale example with only a few hundred landmarks the system was not able to operate in real-time. This demonstrates how infeasible it is to scale this algorithm to a more complex environment over a larger time horizon. For many applications, one can imagine that the number of landmarks could be several magnitudes larger than for this data set, and making it run online would not be feasible. This is not only because of the processing speed requirements but the memory requirements as well, when you have a dense covariance matrix consisting of millions of elements.

There are many possible improvements or alternatives that address these concerns. One idea is to simply separate the map into several independent parts to reduce the number of landmarks we are considering at the same time. This might allow us to better scale to a larger map. Reformulating the EKF-SLAM problem using the information matrix, as previously discussed, is also an alternative that could be explored. Graph-based methods are also based on the sparsity of the information matrix and could be used as an alternative to the EKF-SLAM.

5 Conclusion

References

[1] T. Bailey et al. "Consistency of the EKF-SLAM Algorithm." In: *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'06)* (2006).

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

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

[4] *STIM300 Inertia Measurement Unit*. Rev. 9. Sensonor. May 2013.