TTK4250 Graded assignment 3:
EKFSLAM

Group 19
Andreas Haugland
Emil Martens

NTNU | Department of Engineering Cybernetics

Task 2 - EKFSLAM on simulated data

Tuning

The first parameters tuned were the R values. To do this we ran the simulation for only a couple time steps so we knew the estimated pose did not drift far away from ground truth. We tuned R such that the the estimated covariance of the estimated landmarks roughly corresponded with the distance from the estimated landmarks to their nearest true landmark. The initial guess (eq. 1) had a variance of measured angles that was to high, which resulted in long and narrow covariance ellipses for newly detected landmarks, especially when the landmarks were far away from the vehicle. This can be seen in figure 1a, where the ellipse corresponds to a 95% confidence interval. This was also reflected in NIS values for the full state, that generally were below the confidence interval.

After we had found values for R which worked and seamed reasonable, we ran the simulation for up to 200 time steps and tuned the Q values. The NEES values for the position was often lower than the confidence interval, which meant the filter was not confident enough in its state estimates, so the values for Q were decreased. Then, because there were a couple estimated landmarks that were duplicates of the same true landmark, we decreased the Alpha values such that the JCBB algorithm was more lenient towards associating new measurements with estimated landmarks. The alpha values had to be changed by orders of magnitude to get the desired effect.

Finally we ran through the full dataset multiple times and tuned Q and R to get the NEES and NIS values within their confidence bound and to minimize the error. As the performance of the filter got better, the alpha values could be increased, back to the initial guess, without getting duplicate landmarks. The increased Alpha parameters resulted in faster simulations as fewer association combinations were tested.

Results

Our final parameters gave good results, in terms of low error and good estimates of uncertainty. Notably we observe in figure 1c, that the estimated standard deviations corresponds well with the pose errors. Also, the estimated landmarks are precise and accurate, and there were no false detections. An even better tuning could probably have been achieved by using an automatic tuner and trying more parameter combinations, but the performance of the final parameters were deemed satisfactory. The values for the simulated data can be seen in equation 1 and 2.

$$Q = diag([0.2, 0.2, 0.001]), \qquad R = diag([0.0025, 0.01]), JCCBalphas = [1e-4, 1e-6] \tag{1}$$

$$Q = diag([0.0012, 0.0012, 0.00007]), \qquad R = diag([0.002, 0.005]), JCCBalphas = [1e-4, 1e-6] \tag{2}$$

Notes on the alpha parameters

When the filter had bad parameters it did not only produce an bad trajectory estimate, the computation time also increased a lot. At times it appear to freeze completely.

If the filter does not associate measurements with previous landmarks it assumes the measurement is a new landmark. This might result in a large number of landmarks, which just keeps increasing as new measurements are added. A large number of landmarks makes the association process slower as there are more landmarks to choose from as well as computations related to the covariance matrix. To avoid this the alpha values cannot be to high, as the JCBB algorithm then will disregard associations that are correct.

But even with a relative small number of estimated landmarks, the JCBB algorithm might be really slow. If the filter assumes all the new measurements should associated with a previous landmark, and tries all the possible combinations it is equivalent to a k-combination of the set of landmarks. This has the complexity $O(n!)$. To avoid this, the alpha values cannot be to low as well, as none of the possible associations will be gated.

For this simulated data set, finding alpha parameters that worked was not to difficult, as we could use the ground truth as reference. However without the true landmarks it would be much more difficult to know if the estimated landmarks are actually true landmarks. This was the case on the VP dataset.

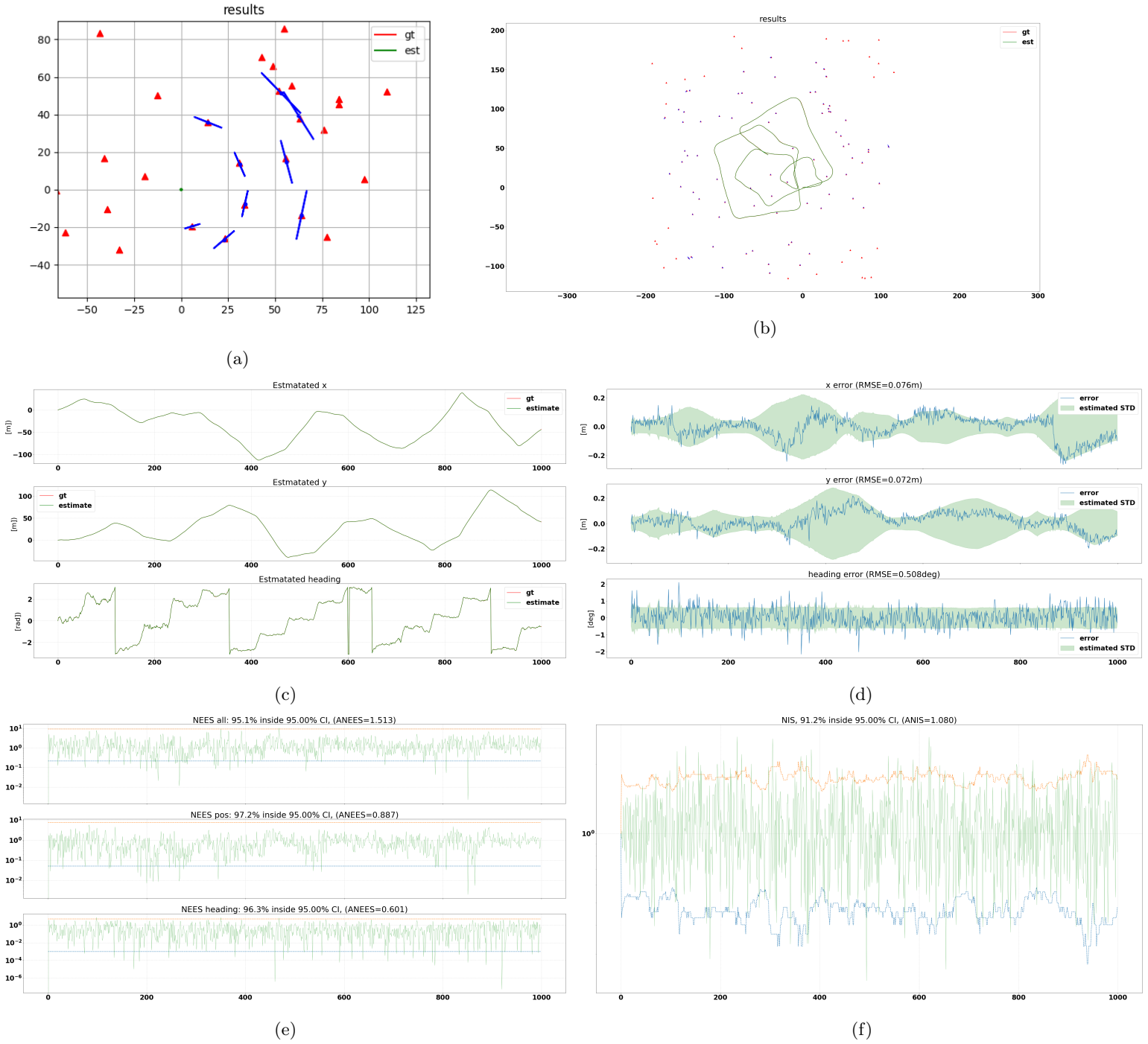


Figure 1: Resulting plots for task 2 - EKFSLAM simulation: **(a)** - Covariance in initial guess, **(b)** - pose estimate vs ground truth, **(c)** - pose error vs est STD, **(d)** - NEES all, NEES pose and ANEES, **(e)** - NIS vs

Task 3 - EKFSLAM on real data

Tuning

As there were no ground truth, some assumptions had to be made when tuning the VP dataset. For instance, we assumed none of the landmarks to be very close. We also assumed detected landmarks should generally be detected multiple times. The tuning for the VP dataset followed roughly the same method as seen in the tuning section for task 2. We started by tuning the R values, in the same way as previously, by only running a couple of timesteps and comparing the covariance of the estimated landmarks to the new measurements as can be seen in fig 2e. When the R values were to high, the confidence circles appeared to be larger than they needed and the NIS values were far below the confidence interval. When the R values were to low, new landmarks were created instead of measurements being associated to previous landmarks. This meant several landmarks would only be detected once, which were assumed to not be plausible.

As there were no ground truth we did not have any NEES values we could use to tune Q. Here the GPS measurements could probably have been used as a substitute, but as mentioned later, constant offsets would

have been a problem. We found values that gave good performance through guesstimating and by comparing the shape of the trajectory with the shape of the GPS measurements, and comparing the NIS values to the confidence interval. Allthough the performance displayed by the EKFSLAM with the initial parameters is quite good, specially if one compares the estimated trajectory with the integrated odometry as seen in figure 2a, it had some notable errors. It has drifted between laps, as there are parallel lines that are not present in the final result (from fig 2a to fig 2b). There is also a constant offset between the GPS and the estimated trajectory due to drift. Constant offset is a challenge with SLAM method. If the initial estimate is wrong or there is some drift early on, new landmarks will inherit this error. EKFSLAM operates in a local frame, and the pose in NED frame is not measurable. To solve this, one could of course include the GPS measurements in the update step. The tuning parameters for the set of figures on the left hand side of figure 2 can be seen in eq. 3, and the parameters for the right hand column of figure 2 can be seen in eq. 4

$$Q = diag([0.01, 0.01, 0.0001]), \qquad R = diag([0.25, 0.0025]), JCCBalphas = [10e-4, 10e-6] \quad (3)$$

$$Q = diag([0.0012, 0.0012, 0.00007]), \qquad R = diag([0.002, 0.005]), JCCBalphas = [1e-6, 1e-8] \quad (4)$$

Results

As seen when comparing figure 2a and 2b, the offset is reduced by tuning the covariance matrices Q and R from the parameters in eq 3 to the parameters in eq 4. The parallel lines are more or less gone. However this tuning also led to an increased number of observed landmarks. In turn this increased the computational time, due to the amount of associations that had to be made with the increased state space. As the NIS values are more often within the confidence bound and there were less drift which resulted in a result closer to the GPS mesurements, we belive the final parameters are better. The parameters also seamed reasonable. However there are probably more false detections of landmarks, as there probably are some misdetections from clutter in the final run.

Problems with the algorithm

An important difference between the simulated dataset and the VP dataset is that the noise in the simulated one is Gaussian. Notably the detection of trees by the tree detector algorithm does appear to be non Gaussian and have several false positives.

The amount of associations that has to be made together with the denseness/sparseness property of the matrices is a problem with the EKFSLAM algorithm. Section 11.2 in the course curriculum [1] describes this in closer detail and concludes that the denseness property is inevitable due to the randomness of the measurements which affects the innovation and its covariance. $\mathbf{S_k}$ suffers from beeing a dense matrix even if $\mathbf{P}_{k|k-1}$ and $\mathbf{R}$ are block-diagonal or sparse. This denseness issue is a central aspect of the SLAM problem and requires large computational power when there are many landmarks present. This is something we experienced first hand when running the EKFSLAM on the VP dataset. When innovation covariance gets to high, the gate test (11.29)[1] will let more association combinations through. In both datsets there are quite a few landmarks, which for specially the VP in turn leads to a large state space and in turn to theese massive covariance matrices.

Room for improvement

Regarding improvement strategies for the consistency issues, there are several potential candidates.

A possible solution, which is supported by Brekke in section 11.2.2 [1], is to use the XKF by Johansen and Fossen and which we've briefly tried in other courses where a Nonlinear Passive Observer was used together with an EKF. Since the EKF believes that it has more information that it actually has, and updates the covariances as if the predictions were closer to the truth than it actually is, it becomes overconfident and produces inconsistencies and eventually diverges. This phenomena caused by lack of observable states, may be fixed through changing the linearization points to linearizing around the output of a globally stable observer such as an NPO.

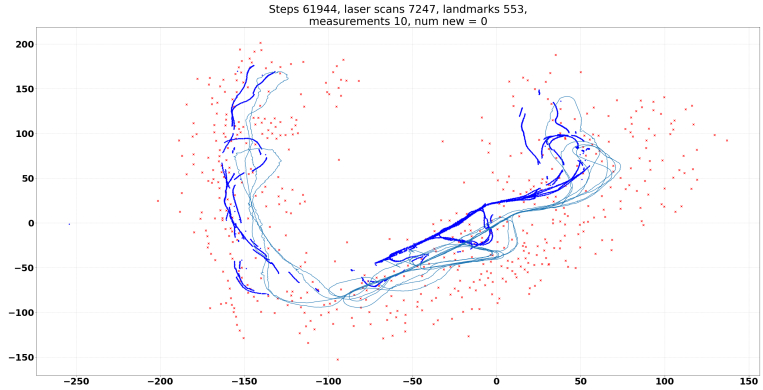
Brekke states in section 12.2.1 [1] that the Newton-Gaussian optimization and Laplace approximation can be used to express $\eta_{\mathbf{k},\hat{(\mathbf{i+1})}}$ through the Jacobian and Hessian of $f(\eta)$, and the inverse Hessian of η_k is used to estimate the covariance matrix. It is worth noting that the Hessian has to be positive definite to ensure that the Newton optimization works, thus this method should only be used in cases where we can guarantee this, which is probably not possible for this dataset. Another drawback with this method is the computational load that comes with evaluating the Hessians.

Descriptions of landmarks could possibly help in the association process. For the VP dataset, a possible descriptor could be the diameter of the detected trees. Otherwise a camera could have been used to get visual descriptors using for example the BRIEF descriptor[2].

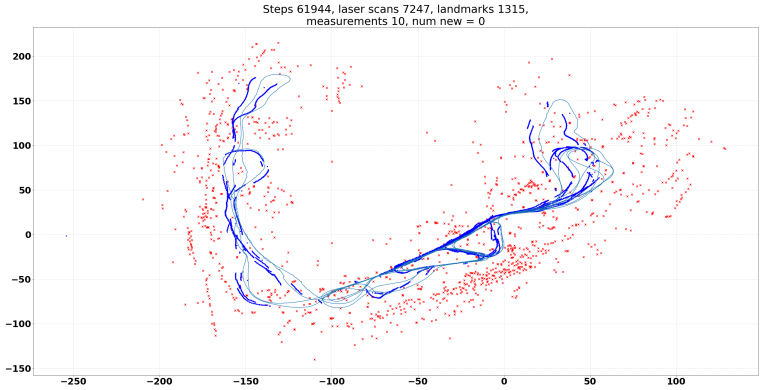
Another possibility would be to remove landmark that are false positives. If the filter predicts there is a tree at a given position, but the tree is never detected in proceeding time steps, even if it is within the range of the sensor, one can assume the tree is not actually there. This is similar to JIPDA from chapter 8 in the course curriculum, where an existence state for every landmark could be included in the state, and landmarks with low probabilities could be removed in each update. Removing landmark that are not detected when they should, would probably filter faster and more robust.

Closing remarks

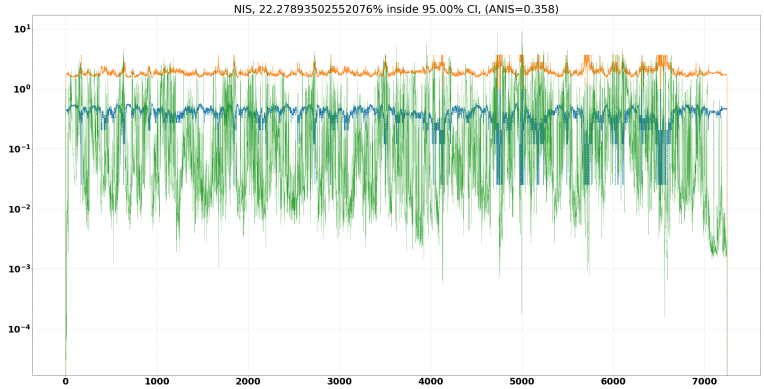
EKFSLAM appears to be a viable algorithm for GPS denied positioning and can, under the right cicumstances, be used as an alternative to GPS based navigation.



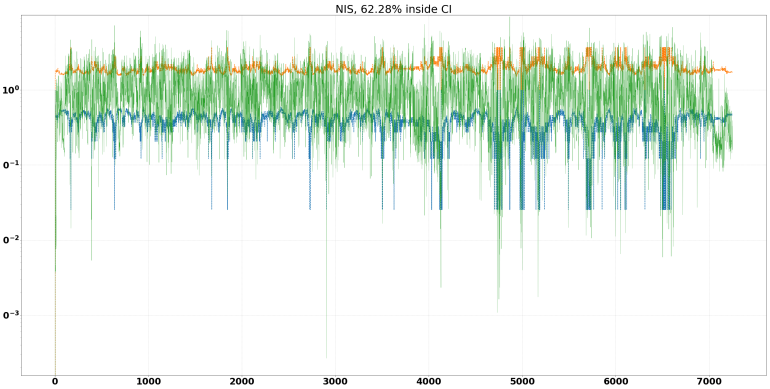
(a)



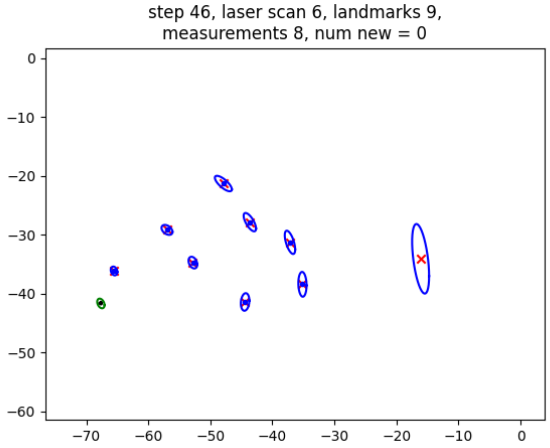
(b)



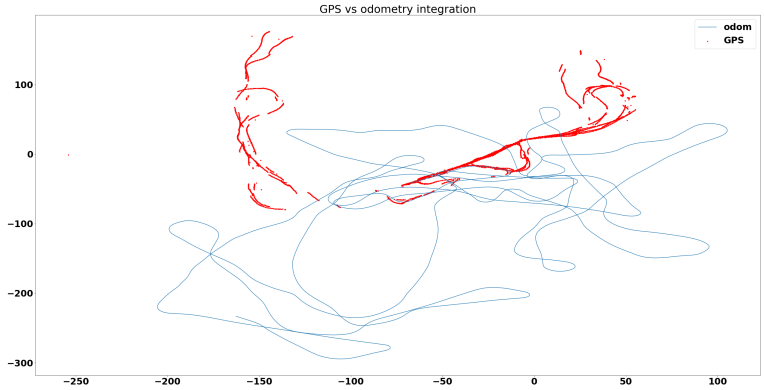
(c)



(d)



(e)



(f)

Figure 2: Resulting plots for task 3 - EKFSLAM on VP data: **(a), (b)** - Estimated trajectory vs gps and landmarks, **(c), (d)** - NIS, **(e)** - covariance of landmarks after first update with initial parameters, **(f) - NIS**. Left column: Attempt with data from eq. 3, Right column: Attempt with data from eq. 4

References

[1] Brekke, Edmund (2020): *Fundamentals of Sensor Fusion*

[2] Michael Calonder, Vincent Lepetit, Christoph Strecha, Pascal Fua (2010): *BRIEF: Binary Robust Independent Elementary Features, https* : *//link.springer.com/chapter/*10.1007/978 − 3 − 642 − 15561 − $1_5$6