Graded assignment 3: EKF-SLAM

Jacob H. Dahl, Viktor Korsnes

***Abstract*—With the emergence of autonomous systems operating in complex environments have lead to the simultaneous localization and mapping (SLAM) algorithm. This report explores the recursive EKFSLAM algorithm with JCBB for data association and its performance on a simulated and the famous Victoria Park dataset. An initial assessment of parameters for the simulated dataset is argued for, and is tuned for consistency. For the Victoria Park dataset, a compromise between computational feasibility and consistency is made, as the tuning either resulted in duplicate landmarks or a JCBB algorithm that is unable to terminate in a feasible time.**

I. Introduction

This report investigates the use of en extended kalman filter (EKF) to solve the simultaneous localization and mapping problem (SLAM). An EKF-SLAM algorithm has been implemented as in [1] with joint compatibility branch and bound (JCBB) data association and with odometry measurements as control input. The filter is tested on a simulated dataset and a real dataset of a car driving around a park. The tuning parameters used are similar to a standard EKF-algorithm, process noise $\boldsymbol{Q}$ and measurement noise $\boldsymbol{R}$. There is also a need to tune the JCBB-algorithm's individual- and joint compatibility boundaries α_i and α_j which set the boundary of possible landmarks to consider when associating measurements to landmarks. The report is divided into four sections, one for the simulated dataset section II, one for the Victoria park dataset section III, one for a discussion of some findings in the previous sections section IV and a conclusion section V.

II. Simulated dataset

Initial estimates on the tuning parameters were found from inspecting the simulated data. The JCBB alpha's set the boundaries for which landmarks the measurements should be compatibility tested with. These have a large impact on the run-time of the algorithm and were initially set to a radius of 3σ using χ^2 with two degrees of freedom. The individual compatibility $\alpha_i = 0.011$. α_j was set to the same value even though it has 3 DoF, meaning it does not represent a 3σ radius. As the simulated data is quite accurate, the inital estimates for $\boldsymbol{Q}$ and $\boldsymbol{R}$ were on the order of centimeters. Setting $\boldsymbol{R} = diag(\sigma_r^2, \sigma_\theta^2) = diag(10^{-4}, 10^{-4})$ and $\boldsymbol{Q} = diag(\sigma_x^2, \sigma_y^2, \sigma_\phi^2) = diag(10^{-4}, 10^{-4}, 10^{-4})$ with the mentioned alphas lead to a lack of convergence within a reasonable amount of time. From this initial tuning it was difficult to give the reasoning for the lack of convergence, but it was hypothesized that the measurement covariance $\boldsymbol{R}$ was too low, based on the low amount of associations done. Increasing this two orders of magnitude gave adequate results, but the filter was largely underconfident. Decreasing the process covariance $\boldsymbol{Q}$ an order of magnitude became too much of an over-compensation and again led to divergence. Fine-tuning the results using normalized estimation error squared (NEES) and normalized innovation squared (NIS). The general strategy was to look at the NIS distribution within its confidence interval. If this is too close to the lower bound it means the filter is too reliant on measurements, and if it is too close to the upper bound it is not relying on the measurements enough. From this a final tuning of $\boldsymbol{Q}$ and $\boldsymbol{R}$ was found. The alphas were tested in different orders of magnitude as well, and it was observed that once the tuning of the filter was adequate, then reducing the alphas had little impact on runtime. This meant that they could be chosen to be as small as ≈ 0, but no difference in performance was seen by reducing them that low either. This is because the map in the simulated data is so small that such small alphas on the order of 10^{-5} makes the JCBB-algorithm compare more than all landmarks which are even slightly plausible. The correct landmark is never outside the boundary. Alphas between 10^{-3} to 10^{-5} gave an increasing amount of associations. From 10^{-5} and lower gave the exact same amount of associations, 9719. The resulting tuning was:

$$\boldsymbol{Q}_{sim} = \begin{bmatrix} 0.022^2 & 0 & 0 \\ 0 & 0.022^2 & 0 \\ 0 & 0 & 0.0063^2 \end{bmatrix}$$

$$\boldsymbol{R}_{sim} = \begin{bmatrix} 0.057^2 & 0 \\ 0 & 0.028^2 \end{bmatrix}$$

$$\begin{array}{cc} \alpha_j & 10^{-5} \\ \hline \alpha_i & 10^{-5} \end{array}$$

The resulting plots of the tuned algorithm can be seen in fig 1, fig 2, fig 3, fig 4 and in table I.

Parameter	CI	Value
ANEES all	[2.85 3.15]	2.80
ANEES pos	[1.88 2.13]	1.22
ANEES heading	[0.91 1.09]	1.39
ANIS	[19.0 19.8]	16.7

Table I: Resulting average NEES (ANEES) values with corresponding confidence intervals from simulated dataset.

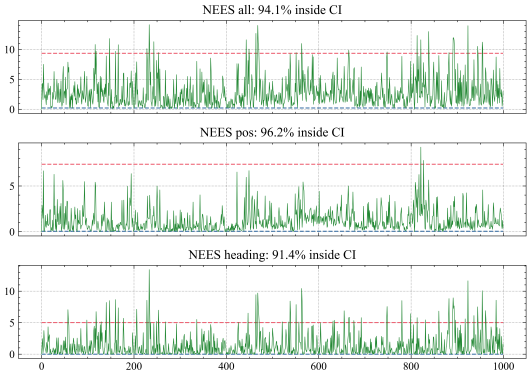

Figure 1: NEES values for simulated data set.

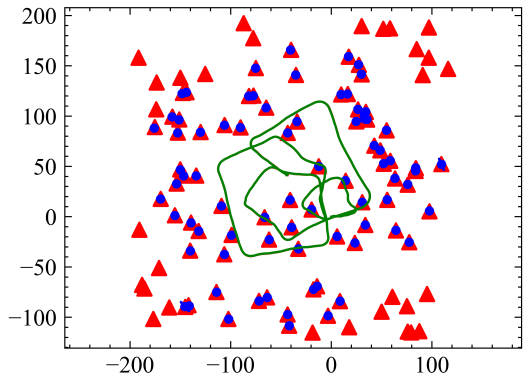

Figure 2: Resulting map of landmarks, their covariance and path taken by the robot.

III. Victoria Park Dataset

The simulated dataset tuning was considered to be a good starting point for tuning the real dataset. Using those values gave good initial estimates, but diverged after ≈ 3000 iterations. The previously used alphas were set too low, and especially with $\alpha_j = 10^{-5}$ caused issues. Increasing this two orders of magnitude gave better NIS results, which is odd and will be discussed later.

Seeing as the simulated tuning values gave good initial estimates, but later diverged, it was reasoned that the filter was overconfident and required an increase in the process model covariance. Increasing these an order of magnitude gave good results. The heading covariance σ_ψ was kept lower than σ_x and σ_y just like in the simulated case, which gave good results. When tuning the measurement covariance matrix $\boldsymbol{R}$ it was found that both runtime and the quality of the estimation was sensitive to differences in $\boldsymbol{R}$. Increasing σ_θ to 0.04, doubling from the simulated case, gave the best results. Looking at the sensor used in the dataset, it has 361 lasers spread over 180 degree area. This means we have an inherit uncertainty of 0.5 degrees from each measurement. Increasing the uncertainty further then compensates for clustering and

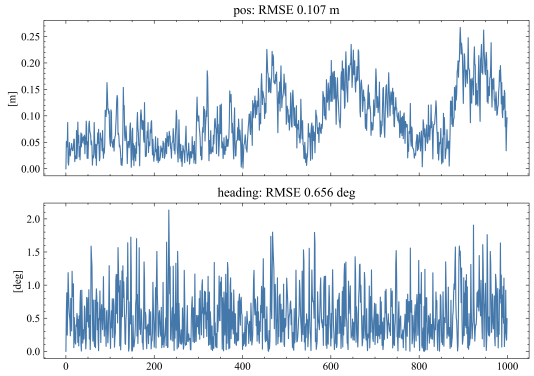

Figure 3: Position- and velocity error with corresponding RMSE values.

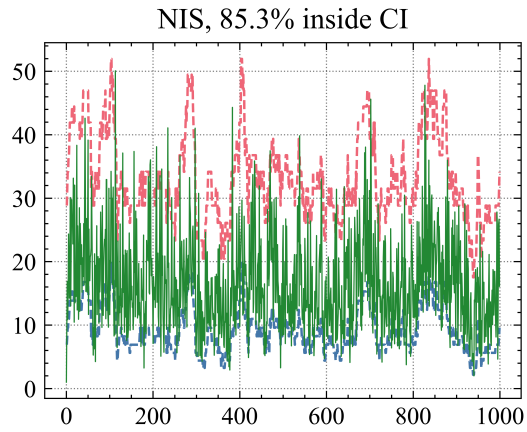

Figure 4: NIS plotted with a 95% upper- and lower confidence interval on the simulated dataset.

other disturbances. Decreasing it further gave higher runtimes and lower NIS. Increasing it had a similar effect. The only explanation the authors can come up with is that with $\boldsymbol{R}$ higher then the JCBB association takes much longer to find the optimal association since the innovation matrix $\boldsymbol{S}$ becomes larger. With a smaller $\boldsymbol{R}$ the filter becomes underconfident and the association process becomes more difficult because of possible wrong estimates in heading and JCBB is stricter with the associations. The first issue can be countered by decreasing α_j and shows that the values of $\boldsymbol{R}$ and the alpha's are closely related. The final tuning is given as follows:

$$\boldsymbol{Q}_{real} = \begin{bmatrix} 0.09^2 & 0 & 0 \\ 0 & 0.09^2 & 0.0125^2 \\ 0 & 0.0125^2 & 0.00194^2 \end{bmatrix}$$

$$\boldsymbol{R}_{real} = \begin{bmatrix} 0.04^2 & 0 \\ 0 & 0.04^2 \end{bmatrix}$$

α_j	10^{-3}
α_i	10^{-5}

The results are shown in fig 5, fig 6 and fig 7 with an average NIS (ANIS) of 18.5 with confidence interval [15.9 16.5]. For these results the dataset was ran on 13000 iterations, which is less than 1/4 of the total dataset, but after 13000 iterations, the state vector had 1131 elements which will increase more throughout the dataset and be increasingly expensive to run, but 13000 iterations was considered to be enough to get an accurate description of the algorithms performance since testing out later sections of the dataset on the same tuning parameters gave similar results.

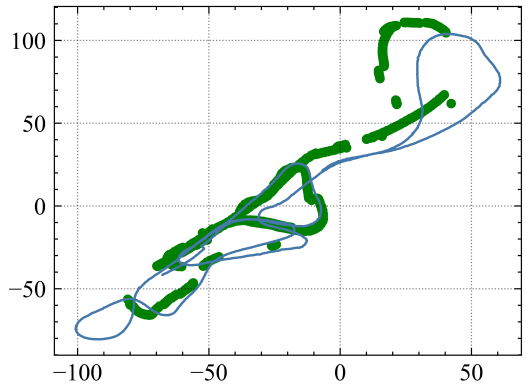

Figure 5: Estimated trajectory (blue) and GPS (green).

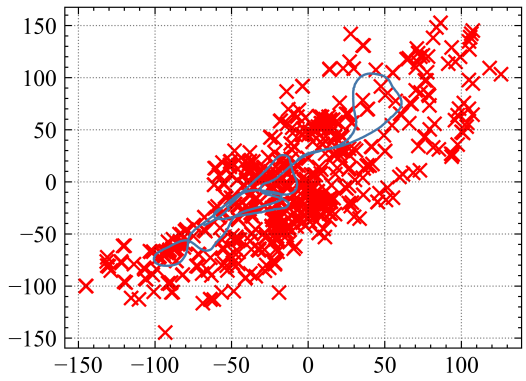

Figure 6: Estimated trajectory (blue) and landmarks

IV. Discussion

From the plot in fig 6 we can clearly see a lot of duplicates in landmarks which should have been merged together by the JCBB data association. This result was removed by increasing the JCBB alpha's, however this also came with a huge loss in NIS and increased run-time. The performance of the algorithm is evaluated mainly based on NIS, so a large drop in NIS is unwanted. Based on this it is necessary to make a priority of what is most important in the SLAM problem. If the main purpose of the algorithm is for a robot to find its

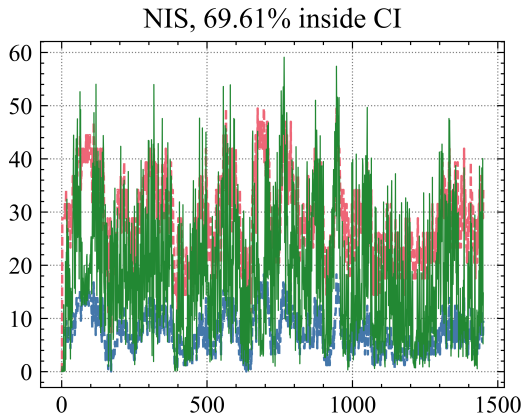

Figure 7: NIS plotted with a 95% upper- and lower confidence interval from the first 13000 iterations of the Victoria park dataset.

accurate location then the addition of extra landmarks does not ruin the robots sense of location. As we can see in fig 5, the estimated trajectory follows the GPS measurements quite accurately most of the run, but it also struggles when fewer landmarks are around, as seen in fig 5, and some drift occurs. However with so many extra landmarks the state vector and covariance matrix become huge after some iterations which is computationally expensive. If the main purpose of the algorithm is to map accurately, and the position of the robot is of less importance (although it is still important for accurate mapping) then extra landmarks are unwanted. Then the alpha's can be decreased further at the expense of the position estimate and run-time. Since SLAM is a "chicken-and-egg" problem both these are important and the results present a compromise between both.

As mentioned previously the joint alpha was higher than the individual alpha. This means we are more strict on the compatibility of the joint landmarks than the individual which is counter-intuitive. The reasoning behind why this gives the filter better results is thought to be related to the amount of landmarks in the map. As discussed, the amount of landmarks is too high, but this gives an easier job for the joint compatibility. Since there are more landmarks to pick from, it becomes easier to associate the joint landmarks.

From both the simulated- and the real dataset there was an apparent underconfidence in the simulated case and and overconfidence in the real case. This is seen from the ANIS being out of the confidence interval, but it proved difficult to tune the algorithm by looking at the averages alone. It was prioritized to tune with respect to the NIS and NEES values, when available, and let the averages be a second priority. However this is telling of a sub-optimal tuning, some more tuning would give a more consistent filter.

V. Conclusion

The EKF-SLAM is not a state-of-the-art solution to the SLAM problem and in this report multiple weaknesses of the implementation have been discussed. The main issues faced in this implementation have been the compromises needed to get decent consistency out of the filter. Getting good estimates of position and orientation has been shown to come at the expense of an accurate map with accurate landmarks. An accurate map has come at the expense of pose and also run-time. Run-time has been a large issue and with this implementation, it could not be run in real-time through the whole Victoria park dataset since the state vector and covariance matrix grow larger with each new landmark. However EKF-SLAM did show to give good results in a small map with few landmarks and accurate sensors.

References

[1] Edmund Brekke. *Fundamentals of sensor fusion, target tracking, navigation and SLAM*. 2020.