TTK4250 Sensor Fusion
Graded Assignment 3

Group 63
Torbjørn Smith 521669
Isak Hammer 257390

November 26, 2021

Department of Engineering Cybernetics

1 Introduction

This report presents the application and tuning of an extended Kalman filter (EKF) applied to a simultaneous localization and mapping problem (SLAM). The EKF-SLAM [1] is implemented using odometry as the control input, and joint compatibility branch and bound (JCBB) data association. The filter is tested on both a simulated and a real case (Victoria Park dataset), as a vehicle is simulated to drive around a park. As with more familiar Kalman filters, the tuning parameters are the matrices belonging to process and measurement noise: $\mathbf{Q}$ and $\mathbf{R}$ respectively. The JCBB is also tuned, as the algorithm's compatibility boundaries: α_i (individual) and α_j (joint) are set. This report first presents the tuning and results thereof for the simulated data set in section 2, then for the real dataset in section 3, before discussing the findings of the project and concluding in section 4.

2 Simulated Dataset

A set of initial tuning parameters were provided. These were decent, but an investigation into possible improvements and changes was conducted to explore the dynamics of tuning the filter. The normalized NIS was exchanged for actual NIS values.

Looking at the NIS distribution, A trend towards the bottom of the confidence interval was observed. This was interpreted as the filter being under-confident, with the innovation covariance being high. The measurement noise $\mathbf{R}_{sim}$ was decreased to improve this. Improvement in both the NIS and NEES distributions was observed. The MRSE however increased for both position and heading. By decreasing the values of the process covariance $\mathbf{Q}_{sim}$, trying to improve the measurement weighting and the filters confidence. The initial value for position and heading RMSE was regained while keeping (and slightly improving) the new NIS and NEES values. The compatibility boundaries were increased and decreased by several orders of magnitude, without much change observed. Both boundaries were decreased an order of magnitude for slight improvement in RMSE and decreased runtime. The resulting tuning, producing a slight improvement on the initial tuning parameters are presented below:

$$\mathbf{Q}_{sim} = \begin{bmatrix} 0.05^2 & 0 & 0 \\ 0 & 0.05^2 & 0 \\ 0 & 0 & 0.0087^2 \end{bmatrix}$$

$$\mathbf{R}_{sim} = \begin{bmatrix} 0.07^2 & 0 \\ 0 & 0.0157^2 \end{bmatrix}$$

$$\alpha_j = 10^{-4} \quad \alpha_i = 10^{-5}$$

The resulting dynamics and characteristics of the tuned filter are presented in figures 1, 2, 3, and 4, and table 1.

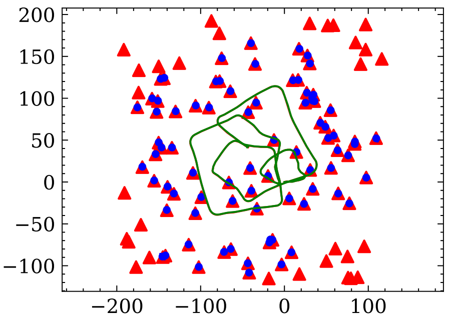

Figure 1: Result for improved tuning on the simulated dataset

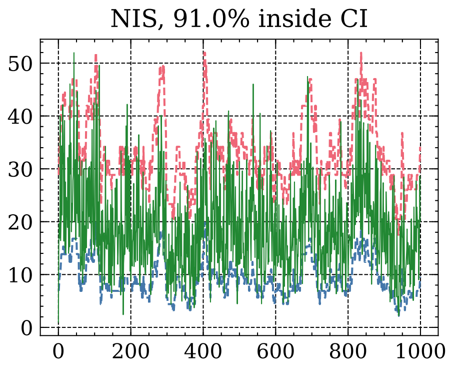

Figure 2: NIS with 95% upper and lower CI for improved tuning on the simulated dataset

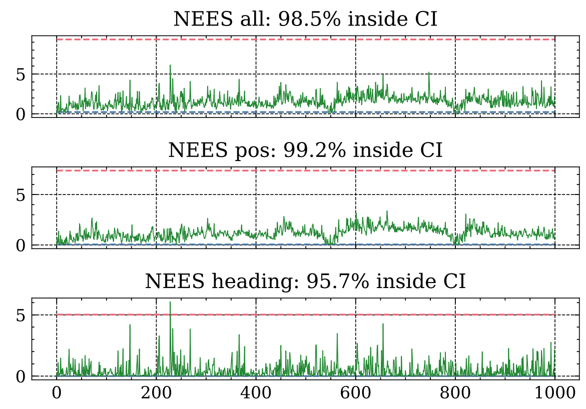

Figure 3: NEES for improved tuning on the simulated dataset

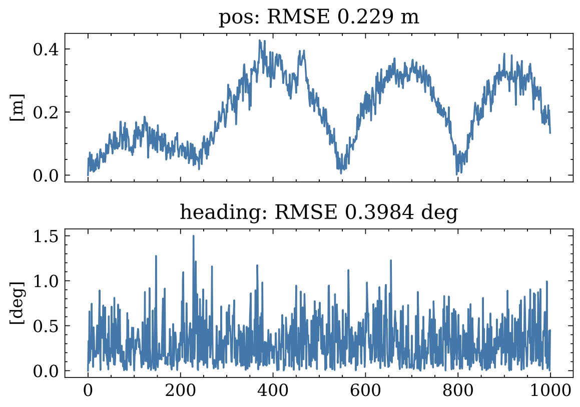

Figure 4: RMSE for improved tuning on the simulated dataset

Descriptor	CI	Value
ANEES all	[2.8501 3.1537]	1.573
ANEES pos	[1.8779 2.1258]	1.184
ANEES heading	[0.9143 1.0895]	0.440
ANIS	[19.0705 19.8442]	19.06

Table 1: Filter characteristics for improved tuning on the simulated dataset

Resorting to the initial tuning parameters, and by increasing the values of both the convariance matrices, a very under-confident filter was produced, with a similar error, but much worse consistency. Figures 5 and 6 shows the confidence and error for the following covariance matrices.

$$\mathbf{Q}_{sim} = \begin{bmatrix} 0.3^2 & 0 & 0 \\ 0 & 0.3^2 & 0 \\ 0 & 0 & 0.0262^2 \end{bmatrix}$$

$$\mathbf{R}_{sim} = \begin{bmatrix} 0.11^2 & 0 \\ 0 & 0.0192^2 \end{bmatrix}$$

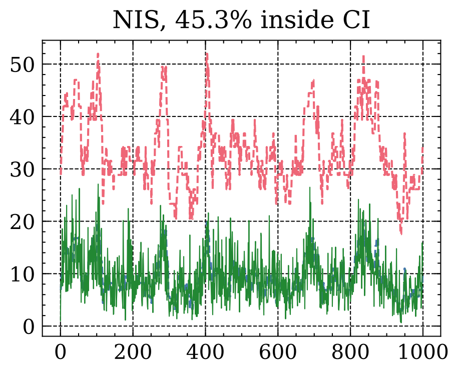

Figure 5: NIS with 95% upper and lower CI showing worse confidence for tuning on the simulated dataset

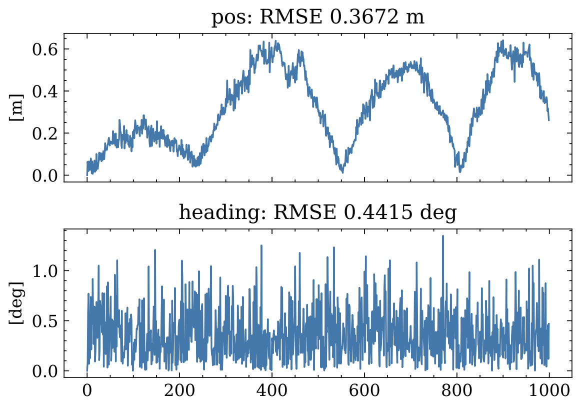

Figure 6: RMSE for tuning giving worse confidence on the simulated dataset

Going back to the provided initial tuning for the process covariance, but setting the measurement covariance very high, an interesting effect is observed on the final landmark estimates. The measurement covariance is set to:

$$\mathbf{R}_{sim} = \begin{bmatrix} 1.0^2 & 0 \\ 0 & 0.1047^2 \end{bmatrix}$$

giving the result shown in figure 7, where the ellipses showing the uncertainty of each landmark estimate is much larger than for the initial tuning parameters. This is in keeping with the theory of the filter as the covariance matrix $\hat{\mathbf{P}}$ increases.

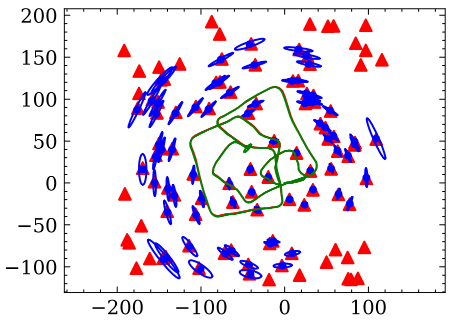

Figure 7: Result for high measurement noise covariance tuning on the simulated dataset

3 Real Dataset - Victoria Park

As for the simulated dataset, initial tuning parameters were provided for the real dataset. The normalized NIS was kept for this task, and to make up for the lack of ground truth, the estimated trajectory was plotted against the GPS data provided in the dataset. This is not ground truth, but an approximation deemed "close enough" when used in conjunction with the other plots. Only the first 20% of the data were used and are presented (about 13000 iterations), as it was deemed sufficient to show the wanted results, while still balancing the runtime of the filter for efficient iteration while tuning.

Initially a better performance was desired. Like for the simulated case, the tuning relied heavily on the NIS plot. A tendency towards the lower part of the confidence interval was observed, and in response the measurement noise matrix was decreased, looking at the percentage inside the confidence interval for the NIS plot. The process noise matrix value was increased to correct for the increase in error between the estimated trajectory and GPS. Ones an acceptable improvement was achieved, the values for the compatibility boundaries were increased, balanced with runtime. An increased percentage inside the NIS confidence interval was observed increasing these, but as the values got large the computation time became unacceptable. The resulting tuning, producing a slight improvement on the initial tuning parameters are presented below.

$$\mathbf{Q}_{real} = \begin{bmatrix} 0.0000055^2 & 0 & 0 \\ 0 & 0.00000275^2 & 0.0000882^2 \\ 0 & 0.0000882^2 & 0.0031416^2 \end{bmatrix}$$

$$\mathbf{R}_{real} = \begin{bmatrix} 0.080^2 & 0 \\ 0 & 0.0150^2 \end{bmatrix}$$

$$\alpha_j = 10^{-2} \quad \alpha_i = 10^{-3}$$

The resulting dynamics and characteristics of the tuned filter are presented in table 2, and figures 8, 9, and 10.

Descriptor	CI	Value
ANIS	[15.9289 16.5157]	13.804

Table 2: Filter characteristics for improved tuning on the real dataset

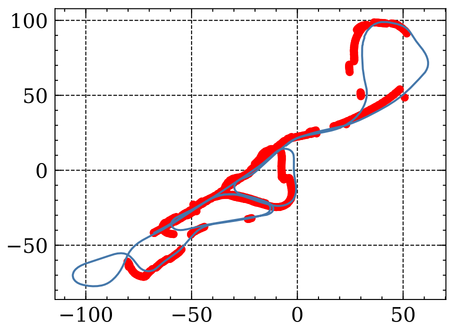

Figure 8: Estimated trajectory (blue) vs GPS (red) for improved tuning on the real dataset

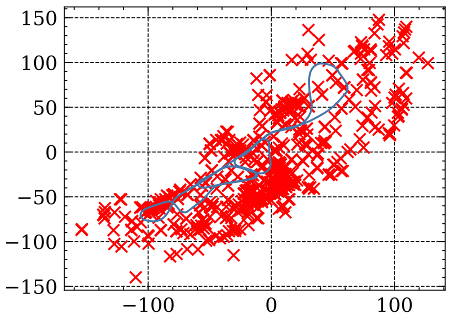

Figure 9: Estimated trajectory (blue) vs landmarks (red) for improved tuning on the real dataset. Number of steps: 12388, laser scans: 1450, landmarks: 594, measurements: 10, new: 0

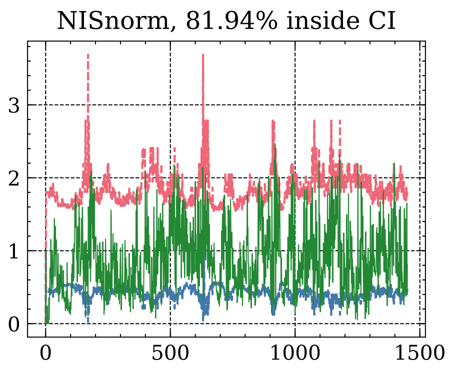

Figure 10: Normalized NIS with 95% upper and lower CI for improved tuning on the real dataset

When tuning for an improved performance, the sigmas that went into the process noise matrix were tuned proportionally. This gave a good NIS plot, but led to a larger error between the estimated trajectory and the GPS. The tuning parameters that yielded this result are given below. The increased error is shown in figure 11 and the corresponding NIS plot in figure 12. Notice the difference in the values corresponding to the heading.

$$\mathbf{Q}_{real} = \begin{bmatrix} 0.0000055^2 & 0 & 0 \\ 0 & 0.00000275^2 & 0.0001194^2 \\ 0 & 0.0001194^2 & 0.00576^2 \end{bmatrix}$$

$$\mathbf{R}_{real} = \begin{bmatrix} 0.065^2 & 0 \\ 0 & 0.0150^2 \end{bmatrix}$$

$$\alpha_j = 10^{-2} \quad \alpha_i = 10^{-3}$$

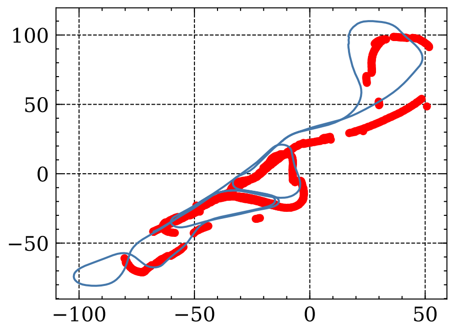

Figure 11: Estimated trajectory (blue) vs GPS (red) for tuning parameters giving increased error on the real dataset

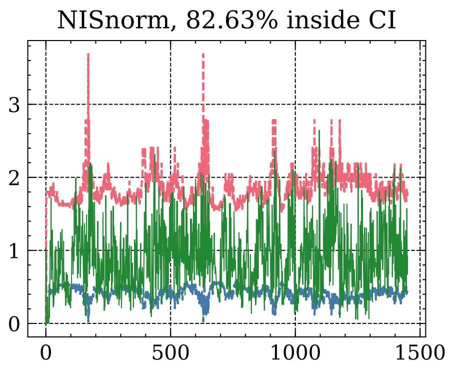

Figure 12: Normalized NIS with 95% upper and lower CI for tuning parameters giving increased error on the real dataset

Going back to the provided initial tuning, a set of tuning parameters are produced that give a significantly less consistent filter, but keep the error (at least the one we can observe) relatively similar. The values for the measurement noise are increase significantly as compared to the initial tuning parameters, in order to make the innovation covariance large, and the resulting NIS, lower. The new $\mathbf{R}_{real}$ matrix is given below.

$$\mathbf{R}_{real} = \begin{bmatrix} 0.3162^2 & 0 \\ 0 & 0.0552^2 \end{bmatrix}$$

The new plot comparing GPS and estimated trajectory is shown in figure 13, and the normalized NIS is shown in figure 14.

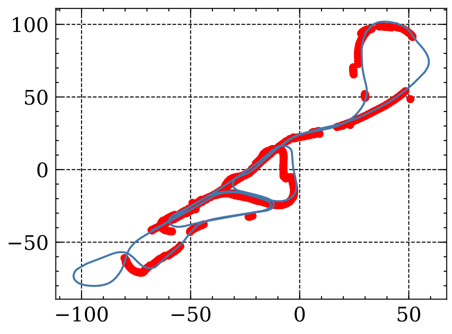

Figure 13: Estimated trajectory (blue) vs GPS (red) for tuning parameters giving bad consistency on the real dataset

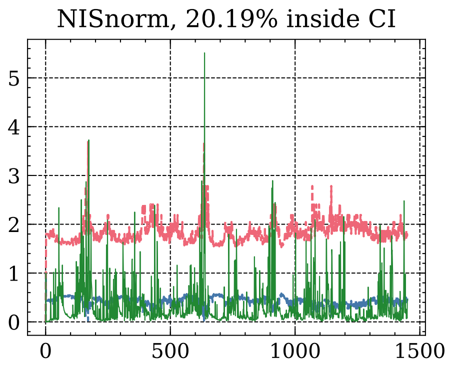

Figure 14: Normalized NIS with 95% upper and lower CI for tuning parameters giving bad consistency on the real dataset

4 Discussion

Looking at figure 9, it is clear that many duplicates exist in the landmark data. This could have been negated with an increase in the compatibility boundaries, but the increase in runtime and possibly diverging filter on the real dataset hindered this. However, as the filter's performance was primarily evaluated based on the NIS value and comparison to GPS, the result was deemed appropriate. The higher joint compatibility as compared to the individual was kept from the provided initial tuning. A possible reason this gives a better result, is that there is a large amount of landmarks to chose from, making it easier to associate potential joint landmarks.

It is noted that the ANIS value for both datasets, and the ANEES for the simulated, are out of the confidence interval. Some more than others, but all below the lower limit. This indicates under-confidence in the filter. However, looking at the NIS and NEES plots for both datasets, the results are deemed acceptable. Even more tuning could bring both satisfactory NIS and NEES plots, and ANIS and ANEES values.

The EKF-SLAM filter has shown several weaknesses in this report. The problem with increased computational cost with an increase size of the state vector is important to note. Also, the fact that getting a filter with good consistency for both mapping and pose estimation of the robot is important to note. It seemed like an accurate map was competing for the an accurate pose estimation of the robot. On the other hand, the EKF-SLAM is a simple algorithm to implement, and does give decent results.

References

[1] Edmund Brekke. *Fundamentals of Sensor Fusion.* Not published. Third edition, August 2, 2021.