EKF-SLAM - performance and limitations

Kristian Stensgård, Leik Lima-Eriksen

November 25, 2020

1 Introduction

Simultaneous Localization and Mapping (SLAM) is a group of powerful techniques used for determining the position and orientation (pose) of itself, while at the same time creating a map of its nearby surroundings. Such techniques are used extensively in situations where global position data may not be available, or when it is important to create an accurate representation of its position relative to its surroundings. Use cases include robotic vacuum cleaners, lawnmowers and cars.

Being inherently complex and often non-linear in nature, the SLAM problem can be solved by the use of quite different methods depending on the use case and requirements. This report will discuss the performance of feature-based EKF-SLAM [1] on two different datasets. Landmark associations are performed through the Joint Compatibility Branch and Bound (JCBB) algorithm. First, we will take a look at a dataset generated through simulation. This will provide insight into how the algorithm actually performs compared with the ground truth. Next up is the Victoria Park dataset obtained from driving a car in the real world. Since this dataset contains real world values, the measurements in the dataset are influenced by disturbances and noises. Due to the lack of ground truth it is significantly more difficult to evaulate its performance. Together, these two datasets should provide useful insight into the performance, shortcomings and improvements of EKF-SLAM applied to the SLAM problem.

2 Simulated data

The SLAM problem has an $O(M^2)$ runtime complexity, where M represents the number of landmarks observed. Therefore, conservative parameters which possibly underestimates M makes a great basis for tuning. The major limiting factors in terms of performance are the JCBB parameters - individual compatibility significance level α_{ic} and the joint compatibility significance level α_{jc}. Setting α_{ic} low makes the gating ellipses for each landmark large, and will in turn make it more probable that a measurement is considered as an already existing landmark. On the other hand, the α_{jc} provides a means of specifying the treshold for how confident we should be that a set of associations could be the true association in order to consider it. Setting this small results in an increased likelihood that associations of larger subsets of the landmarks $\underline{m}$ are considered, which further decreases runtime. The values $\alpha_{ic} = 10^{-10}$ and $\alpha_{jc} = 10^{-10}$ was used as a conservative basis.

With the JCBB alphas out of the way, we start off by providing an initial guess based on common sense for the process noise covariance $Q = [\omega_u^2, \omega_v^2, \omega_\phi]^T I_3$. Since odometry is used as control input, the Q matrix specifies the uncertainty in such data. Furthermore, the covariances for the landmarks R need to be given an initial guess. These factors also greatly affects performance in a positive manner when underestimated, because it allows more uncertainty in both the landmarks and pose, and so it makes it more likely that the innovations are relatively large. So $\omega_u = \omega_v = 10$cm and $\omega_\phi = 0.01$rad and $R = 0.1^2 I_2$ were used.

With these parameters, we actually get an acceptable runtime and consistency with 95% of the heading, position and total NEESes within a 95% confidence interval. But the NIS 95% CI stays at 0.5%, and the $RMSE_{pos} = 0.513$m while $RMSE_{heading} = 1.228^o$. There are definitely improvements to be made with regards to the NIS.

Insight into why this happens can be found by inspecting a realtime plot of the trajectory together with the real landmarks, the measured landmarks and their covariance ellipses. This shows that the covariances start off quite large before slowly decreasing. Since the landmarks are moving in and out of sight all the time, we are dependent on quickly getting an accurate estimate of their positions in order for the filter to consider these in estimating the pose. Decreasing R to $R = [5.2 \cdot 10^{-2}, 2.15 \cdot 10^{-2}]^T I_2$ lets the covariance ellipses converge sufficiently fast. With this updated R, the NEES CIs stay the same, while the NIS CI becomes a more acceptable 89.8%.

But the RMSEs are still large at $RMSE_{pos} = 0.458m$ while $RMSE_{heading} = 0.543^o$. Specifically, the position NEES and RMSE have large peeks around the timesteps 400, 700, and 900. By looking at an association plot of the landmarks versus the measurements, one can get an intuition as to why this happens; around these timesteps, the measured landmarks are all packed tightly together. This means that when taking the uncertainty into account, they give us less information about the pose, which in turn results in larger errors. To fix this, the uncertainty of the odometry has to be reduced as well. By choosing $\omega_u = \omega_v = 4.5cm$ and $\omega_\phi = 5.7 \cdot 10^{-3}$rad, the $RMSE_{pos}$ is reduced to 13cm, which is fairly good. Now, the NEESes do not drift upwards when the landmarks give less information about pose, which is a good indication of stability of the system.

Lastly, the JCBB alphas should be tuned in order to further increase the stability and performance. Since there are no false alarms in terms of landmark measurements, the individual compatibility α_{ic} can be kept quite low at $\alpha_{ic} = 10^{-10}$ resulting in large gating ellipses. This will positively impact the runtime of the filter.

However, the same cannot be said for the joint compatibility α_{jc}. By setting $\alpha_{jc} = 10^{-10}$, we get quite poor performance around the time steps 340-370 in terms of RMSE and NEES. This comes from the fact that a low α_{jc} will gate associations with large normalized innovations. When the best association is

[1] Brekke, Edmund. "Fundamentals of Sensor Fusion".

chosen by comparing their mahalanobis distances, it could quite well happen that the wrong association is chosen because of incorrect gating. Empirically it was found that $\alpha_{jc} = 10^{-3}$ yielded the best performance while not compromising on runtime.

With the filter tuned to the given dataset, we get a great compromise between consistency in NEES as seen in Figure 1 while having small errors in position and velocity. $RMSE_{pos} = 8.52$cm and $RMSE_{heading} = 0.568^{o}$. Having both NEES and NIS above 90% for a 95% CI should be considered good, especially taking the many nonlinearities into account. Also, notice that the heading error as shown in Figure 3 is quite stable. According to [2] , heading inconsistency is a major concern with regards to the stability of EKF-SLAM, which further supports our claim that the filter performs well.

However, what is not ideal is the large errors in $RMSE_{pos}$ around timesteps 700 and 900. It is a bit unclear to us what causes these errors which are over 2 times larger than the average. The errors may be due to the fact that only 7 landmarks on average are associated during these periods compared to the global average of 10 associations. If one assumes that some of them always are new, then the associations would result in less information which gives larger errors. However, this has not been analyzed any further.

From this analysis, we see that there are indeed some shortcomings for the algorithm on this dataset. First of all, it is dependent on associating a significant number of landmarks in order to perform well. Having 10 landmarks nearby at all times is certainly not possible when navigating in some parts of the map, and so the filter performs poorly. Taking this to the real world, it might be the case that this number of landmarks is not even possible to obtain. A better LIDAR with lower noise covariance would certainly help in that situation. Another improvement would be to account for the overall shape of the landmarks in order to ease and improve the association step.

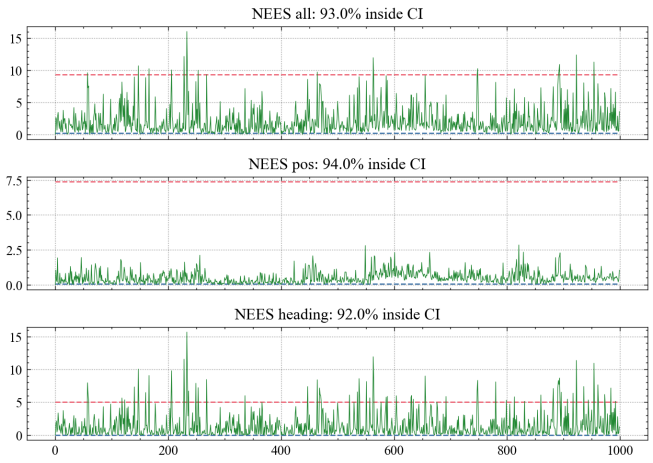

Figure 1: NEESes together with a 95% confidence interval for the simulated dataset.

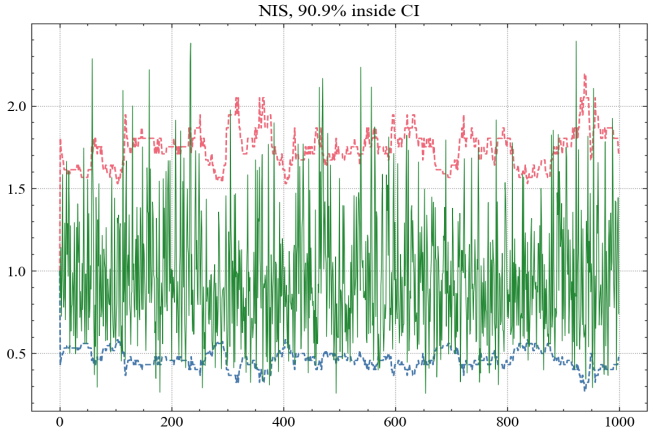

Figure 2: NIS together with a 95% confidence interval for the simulated dataset.

3 Real dataset

The real dataset is way trickier to tune since it contains false alarms. Also, an important distinction between simulated data and real world data is that there are no underlying mathematical distributions for the noises. It may very well happen that the noises are far from gaussians, which complicates the task even further.

The odometry obtained in the dataset is from a LIDAR scanner mounted on a car driving around in the Victoria Park. Any mounting offset is taken into consideration when using the LIDAR measurements. A GPS receiver was also used in the data acquisition, but unfortunately this sensor has a lot of missing datapoints and is not very accurate. Thus it cannot be used as a ground truth. The only metric for performance is then the NIS. A visual inspection of the track for

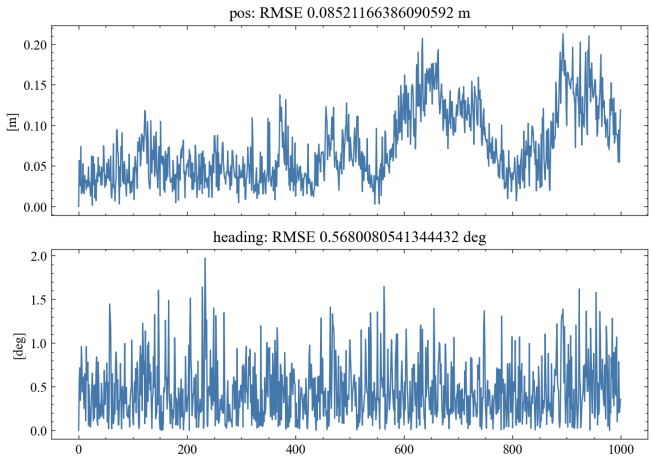

Figure 3: RMSEs for the simulated dataset.

[2] 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).

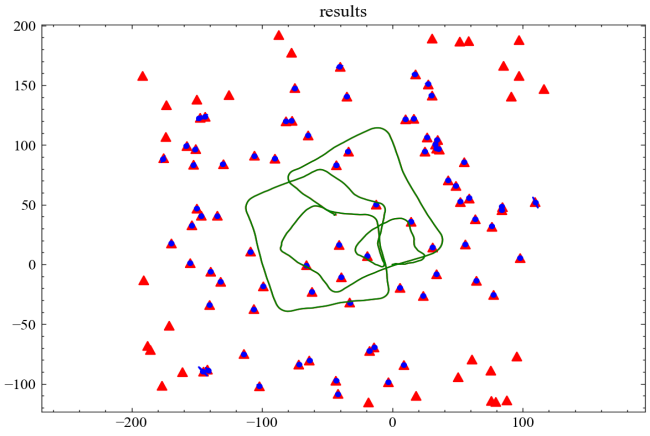

Figure 4: Trajectory for the simulated dataset. True position of the landmarks (red) have been plotted together with the estimated position of the landmarks (blue).

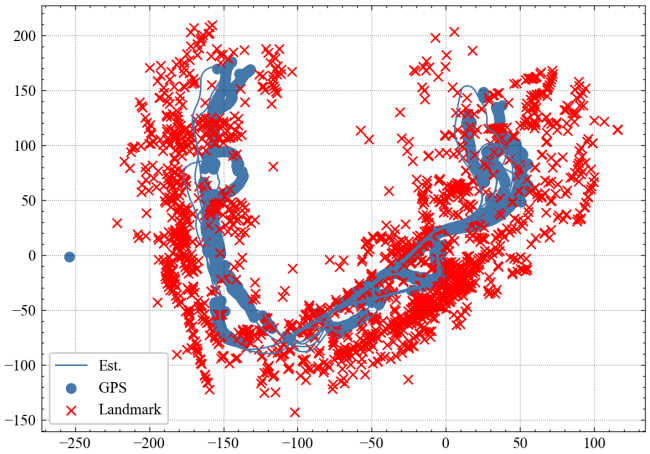

Figure 5: Trajectory for the real dataset together with GPS position measurements and measured landmarks.

smoothness and typical movement patterns (CV, CT) of a car was used under tuning. But this is a metric that is impossible to quantify.

The tuning process for the real dataset follows the same methodology as for the simulated dataset. A smaller subset of the dataset was used initially to obtain values for the different parameters. The initial values for the parameters were based on the results from the simulated dataset as they provide a satisfactory solution to a more ideal situation. The dataset differ in many ways, but nevertheless, this proved to be a nice strategy.

For the process noise covariance matrix, $Q = Q_{diag}A_{corr}Q_{diag}$ where$Q_{diag} = [\sigma_u^2, \sigma_v^2, \sigma_\phi]^T I_3$, the values $\sigma_u^2 = 3.2 \cdot 10^{-2}$, $\sigma_v^2 = 1.2 \cdot 10^{-2}$ and $\sigma_\phi = 5 \cdot 10^{-3}$ was used. $\sigma_v < \sigma_u$ was chosen because the vehicle would be subject to smaller accelerations transverse to the direction of motion. The A_{corr} matrix is defined in Equation 1. Notice how the correction matrix induces a correlation between sideways displacement and rotation. Intuitively this can be understood through the fact that a rotation makes it more likely that sideways displacement occurs, which indeed is what happens when performing a turn.

The values are of the same order of magnitude as for the simulated dataset, except for the σ_ϕ^2 being significantly lower. This results in smoother tracks as the EKF has more confidence in the process model.

$$A_{corr} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0.9 \\ 0 & 0.9 & 1 \end{bmatrix} \tag{1}$$

For the measurement noise covariances we have $R = [\sigma_r^2, \sigma_\theta^2]^T I_2$. These covariances proved to be very important for the number of landmarks that the model creates. This is due to the fact that R directly influences the P matrix that holds the covariances for the landmarks, since it is used for the calculation of the initial covariance for new landmarks. Under tuning the values giving optimal performance was $R = [\sigma_r^2, \sigma_\theta^2] = [(6 \cdot 10^{-2})^2, (2 \cdot 10^{-6})^2]^T I_2$. This is significantly lower than for the simulated dataset.

The JCBB alphas were initially chosen the same as for the simulated dataset i.e $\alpha_{ic} = \alpha_{jc} = 10^{-10}$. The number of landmarks seemed to be too high based on aerial footage of the park used in the dataset found in [3] . Therefore, under tuning the values $\alpha_{ic} = 10^{-4}$ and $\alpha_{jc} = 2 \cdot 10^{-6}$ were found to be a better choice. This significant decrease in size for the gating of individual landmarks means one new measurement could be considered a possible association to many nearby landmarks. The α_{jc} were also increased to limit the number of possible combinations of associating a set of landmarks to a set of measurements.

For the complete dataset we get 1636 landmarks and from Figure 5, we see that a lot of them are very closely spaced together. This is likely an effect of the landmarks being quite large, e.g a building, fence or several trees, resulting in clutter. This influences the computational performance of the EKF-SLAM negatively, as described in the next section. By inspection, the track appears very smooth and resembles the results found in [3] . One of the results from that paper is shown on page 53. This track has great resemblance with the track produced in this report. One interesting thing to note is the number of landmarks, the paper [3] has significantly fewer landmarks. This is likely due to different implementations of landmark association, but One would need greater insight into the implementation in [3] to make any further assumptions.

Evaluating the performance of the filter on this particular dataset is a challenging task. As we do not have the ground truth available, the only numerical metrics readily available is the NIS with a chosen confidence interval of 95%. This has been plotted in Figure 6, and we got 64.28% within the CI for our parameters. Initially one may think that having a NIS CI as high as possible is an indication of a great performing filter. However, this is actually obtained by simply setting R very large. The filter would then trust only the odometry and be overconfident, resulting in a perfectly consistent NIS.

[3] http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/pdf/slam10-fastslam.pdf

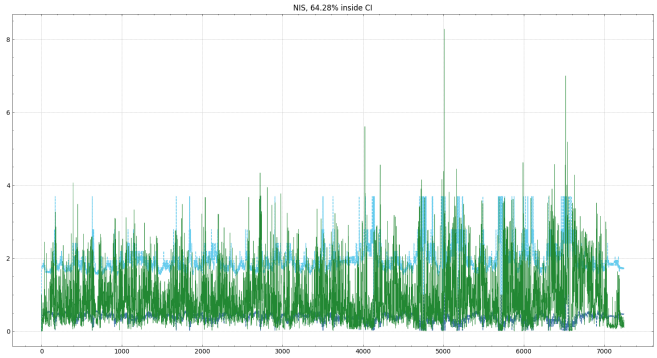

Figure 6: NIS together with a 95% confidence interval for the simulated dataset.

What is more important with regards to evaluating the performance is to visually inspect the trajectory and the number of landmarks created. The trajectory seems quite smooth, which matches well with how a vehicle would navigate. However, it is also evident from Figure 5 that the trajectory is rather biased at the leftmost and rightmost parts of the map. This may indicate that a strong correlation between the landmarks and the pose combined with a possibly underestimated noise covariance of the landmarks makes errors in landmarks propagate through a biased estimation of the pose. Increasing the noise covariances did however not fix the issue, and so the error may in fact be that the posterior densities of the landmarks are non-Gaussians. A Laplacian approximation should lead to better estimates, however this has not been explored.

4 Shortcomings of EKF-SLAM and improvements

The performance of the EKF-SLAM is satisfactory for small datasets, but it has its shortcomings and limitations, both when it comes to large datasets, but also for small dataset. With few landmarks the information about the map may be too sparse and the localization may be difficult as the EKF-SLAM utilizes the correlation between landmarks to determine the pose. This was specifically observed around timesteps 650 and 900 for the simulated dataset as in Figure 3, where the positional error quickly grew large when there were quite few landmarks associated. On the other hand, large datasets with dense maps has this other problem with computational complexity. A large number of landmarks M leads to an even larger covariance matrix P of size M^2, and so the algorithm will just start to perform poorer and poorer as time goes by.

One solution to this could perhaps be to remove landmarks which are of a certain distance away from the vehicle so that we keep the P matrix small. A similar solution would be to partition the map into several independent maps, and switch between them in some way. This would indeed improve on runtime, but the algorithm would still suffer from all the problems inheriting from the linearizations.

Another significant limiting factor lies in the way the association algorithm works; when JCBB fails to associate a new measurement with an existing landmark, it will immediately assign it as a new landmark. This did however lead to a huge problem in the Victoria Park dataset, as the measurements did indeed include a lot of clutter. In return, a lot of landmarks were created which did not resemble to a real landmark. Adding such false alarms not only increases the computational complexity, but it also makes the filter perform worse. Since false positives now can be associated with real measurements, the JCBB can much more easily conclude with an incorrect association. Immediately one may think that an association algorithm along the lines of the one in a JPDA which considers posterior densities may be better suited. But such an algorithm is way too complex to evaluate, and so it is not suitable.

So when we are sort of limited to using the JCBB association algorithm, it is important to be aware of its sensitivity and limiting factors. It is quite evident that it is very hard to determine the true number of landmarks in the VP dataset. When tuning it became clear that for some sub optimal parameters there were duplicates of landmarks as the total number were high and the landmarks are situated very close together as previously mentioned. The JCBB alphas are obviously very important for landmark association and creation. But more importantly, the σ_θ seemed to influence the performance significantly in terms of number of landmarks created. Only adjusting it by a factor of 15% wrong would lead to the execution stalling. Since this is a parameter dependent on the particular landmark detector, e.g. the LIDAR sensor, such parameter should be chosen with great care.

From the discussion above, it seems that the two major concerns are the computational complexity and overconfidence. The first problem can be addressed by employing e.g. Fast-SLAM. This algorithm has a runtime of $O(K \cdot logM)$, where K denotes the number of particles, and thus it should perform great over long periods of time. As an added benefit, the algorithm actually solves the full SLAM problem as opposed to the EKF-SLAM's online solution, which means that the post trajectory will be updated upon each iteration. The solution does still suffer from errors with regards to linearization. However, these should be minor because of the extensive use of Rao-Blackwellization on the nonlinearities.

5 Conclusion

The feature based EKF-SLAM provides adequate solutions to the SLAM problem for medium size problems when gaussianity can be assumed. There are several shortcomings of the EKF-SLAM that originates linearization of nonlinearities as well as the quadratic runtime. Specific use-cases such as a self driving lawn mower on a restricted area may gain satisfactory results. However, in most cases the algorithm requires further optimizations and should be used with great caution.