16-833 SLAM HW2: Extended Kalman Filter

Ryan Wu (Andrew ID: weihuanw)

March 15, 2024

In this report, we investigated a practical implementation of the extended Kalman filter (EKF) with a given dataset. The report consists of 3 sections: theory, implementation and evaluation, and discussion. The theory section covers the mathematical derivations of the EKF, and the written work is included at the end of the report. The implementation and evaluation section discusses the EKF code implementation and its performance. Lastly, the discussion section goes over some intuitive questions regarding EKFs. Overall, the EKF implementation results are satisfactory in our use case.

1 Theory

1.1

Assuming there is no noise or error in the control system, the predicted next pose P_{t+1} as a nonlinear function of the current pose P_t and the control inputs d_t and α_t is given by:

$$p_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} x_t + d_t \cdot \cos(\theta_t) \\ y_t + d_t \cdot \sin(\theta_t) \\ \theta_t + \alpha_t \end{bmatrix}$$

1.2

Assuming the errors follow Gaussian distributions, the predicted uncertainty of the robot, in the global frame, at time t+1 as a Gaussian distribution with zero mean is given by:

$$\Sigma_{t+1}^{\text{robot}} = G_t \Sigma_t G_t^\top + A_t \Sigma_t A_t^\top$$

where

$$G_t = \begin{bmatrix} 1 & 0 & -(d_t + e_x)\sin(\theta_t) - e_y \cos(\theta_t) \\ 0 & 1 & (d_t + e_x)\cos(\theta_t) - e_y \sin(\theta_t) \\ 0 & 0 & 1 \end{bmatrix}$$

$$G_t^\top = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -(d_t + e_x)\sin(\theta_t) - e_y \cos(\theta_t) & (d_t + e_x)\cos(\theta_t) - e_y \sin(\theta_t) & 1 \end{bmatrix}$$

$$A_t = \begin{bmatrix} \cos(\theta_t) & -\sin(\theta_t) & 0 \\ \sin(\theta_t) & \cos(\theta_t) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A_t^\top = \begin{bmatrix} \cos(\theta_t) & \sin(\theta_t) & 0 \\ -\sin(\theta_t) & \cos(\theta_t) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\Sigma_t = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\alpha^2 \end{bmatrix}$$

1.3

Consider a landmark l observed by laser sensor with measurement of the bearing angle β and range r, with noise $n_\beta \sim \mathcal{N}(0, \sigma_\beta^2)$ and $n_r \sim \mathcal{N}(0, \sigma_r^2)$. The estimated position (l_x, l_y) of landmark l in global coordinates as a function of p_t, β, r, and the noise terms is given by:

$$\begin{bmatrix} l_x \\ l_y \end{bmatrix} = \begin{bmatrix} p_t(x_t) + (r + n_r) \cdot \cos(p_t(\theta_t) + \beta + n_b) \\ p_t(y_t) + (r + n_r) \cdot \sin(p_t(\theta_t) + \beta + n_b) \end{bmatrix}$$

1.4

The predicted measurement of β and r based on l_x, l_y, p_t, and the noise terms is given by:

$$\begin{bmatrix} \hat{\beta} \\ \hat{r} \end{bmatrix} = \begin{bmatrix} warp2pi(np.arctan2(l_y - p_t(y_t), l_x - p_t(x_t) - p_t(\theta_t) - n_\beta) \\ \sqrt{(l_x - p_t(x_t))^2 + (l_y - p_t(y_t))^2} - n_r \end{bmatrix}$$

1.5

The derived analytical form of the measurement Jacobian H_p with respect to the robot pose is given by:

$$H_p = \begin{bmatrix} \frac{-l_y + p_t(y_t)}{(l_x - p_t(x_t))^2 + (l_y - p_t(y_t))^2} & \frac{l_x - p_t(x_t)}{(l_x - p_t(x_t))^2 + (l_y - p_t(y_t))^2} & -1 \\ \frac{-l_x + p_t(x_t)}{\sqrt{(l_x - p_t(x_t))^2 + (l_y - p_t(y_t))^2}} & \frac{-l_y + p_t(y_t)}{\sqrt{(l_x - p_t(x_t))^2 + (l_y - p_t(y_t))^2}} & 0 \end{bmatrix}$$

1.6

The derived analytical form of the measurement Jacobian H_l with respect to its corresponding landmark l is given by:

$$H_l = \begin{bmatrix} \frac{-l_y + p_t(y_t)}{(l_x - p_t(x_t))^2 + (l_y - p_t(y_t))^2} & \frac{l_x + p_t(x_t)}{(l_x - p_t(x_t))^2 + (l_y - p_t(y_t))^2} \\ \frac{-l_x + p_t(x_t)}{\sqrt{(l_x - p_t(x_t))^2 + (l_y - p_t(y_t))^2}} & \frac{-l_y + p_t(y_t)}{\sqrt{(l_x - p_t(x_t))^2 + (l_y - p_t(y_t))^2}} \end{bmatrix}$$

Due to the independence assumption in EKF-SLAM, we do not need to calculate the measurement Jacobian with respect to other landmarks except for itself. Measurements from different landmarks are conditionally independent from each other.

2 Implementation and Evaluation

2.1

From the data file, there are 6 pairs (12 individuals) of measurements, which indicates 6 landmarks are being observed over the entire sequence.

2.2

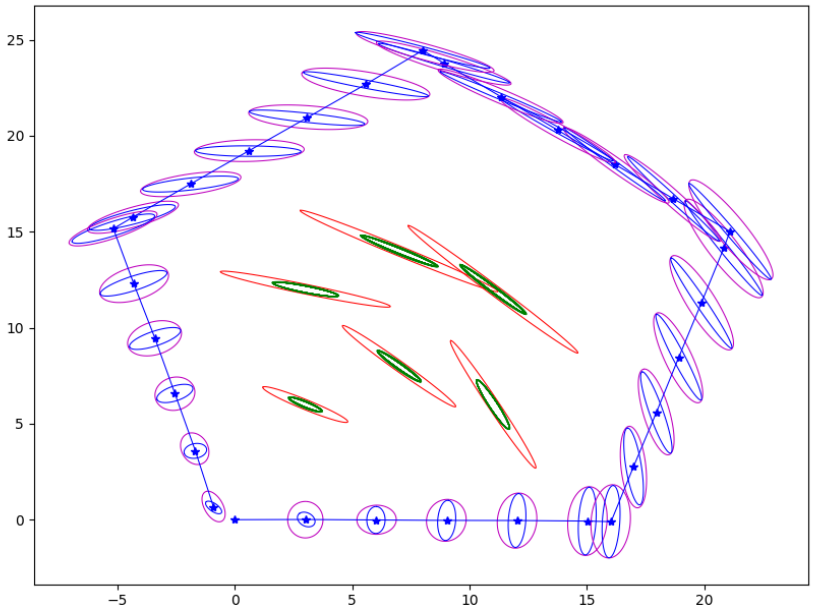

Figure 1: The output visualization of the EKF implementation with all the steps completed.

2.3

The EKF-SLAM implementation updates the robot's pose and landmark positions at each time step from sensor measurements. It reduces the robot's uncertainty and eventually leads to proper localization. From Figure 1. we can observe that as the robot moves farther away from the initial starting point, the predicted and updated uncertainties increase, indicated by larger ellipses. Similarly, as the robot moves back closer to the initial position, the uncertainty decreases with smaller ellipses, and the landmark uncertainty also decreases as the robot gains more information about the map.

2.4

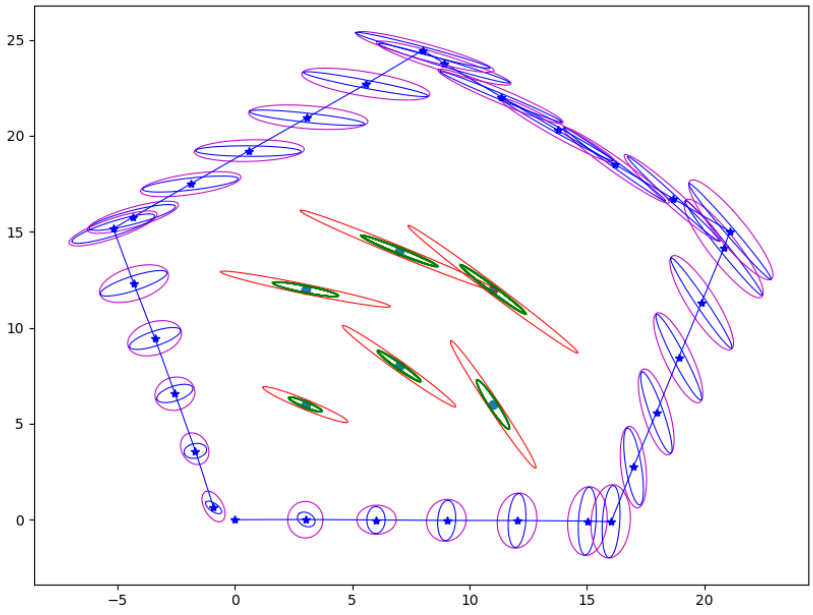

Figure 2: The output visualization of the EKF implementation with the ground truth positions of the landmarks

From Figure 2. we can observe that each of the ground truth positions of the landmarks (blue dots inside the green ellipses) is inside the smallest corresponding ellipse. This means that the EKF-SLAM implementation is providing accurate estimates and correctly representing the variability of the landmarks' ground truth positions.

```
Landmark 1:
        Euclidean distance: 0.0024196996063128889
        Mahalanobis distance: 0.05452482478160100 6
Landmark 2:
        Euclidean distance: 0.0058248448941840055
        Mahalanobis distance: 0.06477739336729885
Landmark 3:
        Euclidean distance: 0.0014075712224324325
        Mahalanobis distance: 0.035075010512631115
Landmark 4:
        Euclidean distance: 0.0032567657755233855
        Mahalanobis distance: 0.064104054644076 22
Landmark 5:
        Euclidean distance: 0.0017959422214237544
        Mahalanobis distance: 0.021952036127813 17
Landmark 6:
        Euclidean distance: 0.005583788669758309
        Mahalanobis distance: 0.0943598067458898
```

Figure 3: The calculated Euclidean and Mahalanobis distances of each landmark estimation with respect to the ground truth.

The Euclidean distance calculates the straight-line distance between the estimation and the ground truth. The Mahalanobis distance further takes into account the covariance uncertainties during calculation. Both the Euclidean and the Mahalanobis distances provide a benchmark in determining the accuracy of your EKF implementation. In our case, both numbers are small which indicates that the landmark estimation is close to the ground truth.

3 Discussion

3.1

At the initial robot state, the robot has no information about the landmark's location, which results in zero terms in the initial landmark covariance matrix. However, as the robot moves and updates its landmark measurements at each time step, the robot's landmark position uncertainty increases/decreases. This is why there are non-zero terms in the final state covariance matrix P. Furthermore, when setting the initial values for the full covariance matrix P, we assumed that the robot's pose and the landmarks' positions are independent. The cross-covariance between the two is set to zero, which is not necessarily correct in real-world applications.

```
P [[ 0.01319169 -0.00835486  0.00036886  0.00953206 -0.00574329  0.00972867 -0.00559654  0.00949868 -0.00592257  0.0098233  -0.00586156  0.00930263 -0.00617066  0.00968282 -0.00614015]
 [-0.00835486  0.01199743 -0.00262558  0.00623836 -0.00044252  0.01979214 -0.00058474  0.01083125 -0.0094197   0.02426721 -0.00948894  0.00642815 -0.01834281  0.01980774 -0.01838637]
 [ 0.00036886 -0.00262558  0.00164374 -0.00840243  0.00393319 -0.01791358  0.00396044 -0.0115822   0.01026312 -0.02107457  0.01027865 -0.00842825  0.01658952 -0.01790961  0.01659953]
 [ 0.00953206  0.00623836 -0.00840243  0.0542668  -0.02657389  0.10459163 -0.02659059  0.07082578 -0.06029786  0.1214129  -0.06031145  0.05392312 -0.09403636  0.10450306 -0.09405939]
 [-0.00574329 -0.00044252  0.00393319 -0.02657389  0.01511837 -0.05025781  0.01483078 -0.03444864  0.03066574 -0.05814429  0.03063858 -0.02650517  0.04649687 -0.05022012  0.04647452]
 [ 0.00972867  0.01979214 -0.01791358  0.10459163 -0.05025781  0.21270578 -0.05044635  0.14054808 -0.12205175  0.24830301 -0.12220027  0.10468835 -0.19388249  0.21235277 -0.19399095]
 [-0.00559654 -0.00058474  0.00396044 -0.02659059  0.01483078 -0.05044635  0.01519956 -0.0345391   0.03075553 -0.05838742  0.03073568 -0.02654454  0.04668839 -0.05041505  0.04666888]
 [ 0.00949868  0.01083125 -0.0115822   0.07082578 -0.03444864  0.14054808 -0.0345391   0.0943495  -0.0809067   0.16373709 -0.08096537  0.07082226 -0.12736513  0.14047126 -0.1274176 ]
 [-0.00592257 -0.0094197   0.01026312 -0.06029786  0.03066574 -0.12205175  0.03075553 -0.0809067   0.07210554 -0.14261201  0.07186923 -0.06035319  0.11297607 -0.12202611  0.11301219]
 [ 0.0098233   0.02426721 -0.02107457  0.1214129  -0.05814429  0.24830301 -0.05838742  0.16373709 -0.14261201  0.29082438 -0.14281236  0.12156587 -0.22713189  0.24824511 -0.22727121]
 [-0.00586156 -0.00948894  0.01027865 -0.06031145  0.03063858 -0.12220027  0.03073568 -0.08096537  0.07186923 -0.14281236  0.07223577 -0.0603738   0.11312875 -0.12218481  0.11317165]
 [ 0.00930263  0.00642815 -0.00842825  0.05392312 -0.02650517  0.10468835 -0.02654454  0.07082226 -0.06035319  0.12156587 -0.0603738   0.05415044 -0.09418897  0.10460875 -0.09422113]
 [-0.00617066 -0.01834281  0.01658952 -0.09403636  0.04649687 -0.19388249  0.04668839 -0.12736513  0.11297607 -0.22713189  0.11312875 -0.09418897  0.17980312 -0.19387097  0.17961318]
 [ 0.00968282  0.01980774 -0.01790961  0.10450306 -0.05022012  0.21235277 -0.05041505  0.14047126 -0.12202611  0.24824511 -0.12218481  0.10460875 -0.19387097  0.21258543 -0.19398588]
 [-0.00614015 -0.01838637  0.01659953 -0.09405939  0.04647452 -0.19399095  0.04666888 -0.1274176   0.11301219 -0.22727121  0.11317165 -0.09422113  0.17961318 -0.19398588  0.18000587]]
```

Figure 4: The final P matrix from the EKF-SLAM implementation.

3.2

From parameter tinning, we can observe that different parameter values have varying influences on the result. With an increased value, the model's uncertainties (ellipses) generally increased and vice versa. σ_x, σ_y, and σ_α changed the predicted and updated uncertainties of the robot's pose. σ_β and σ_r changed the predicted and updated uncertainties of the landmarks' position. More detailed changes in the model's behavior can be observed from the parameter tuning outputs attached below.

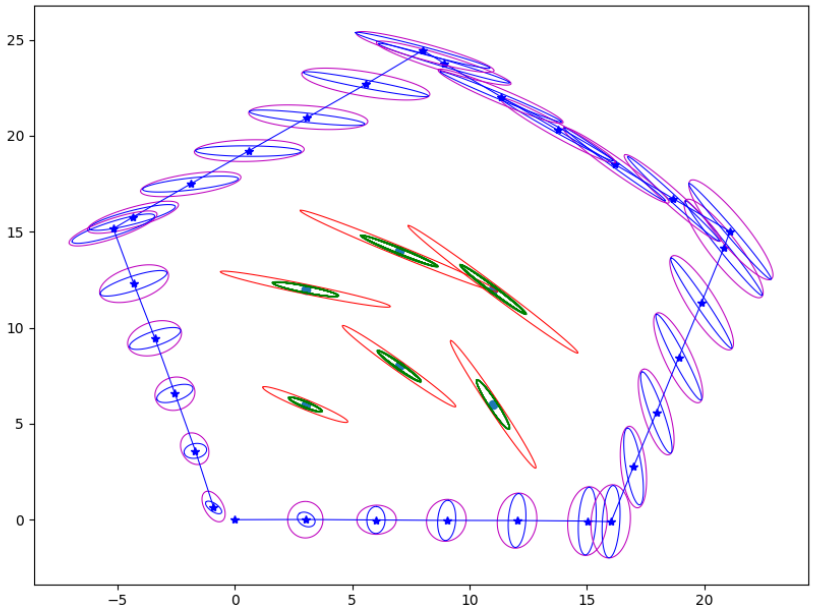

Figure 5: The output with the original given parameters.

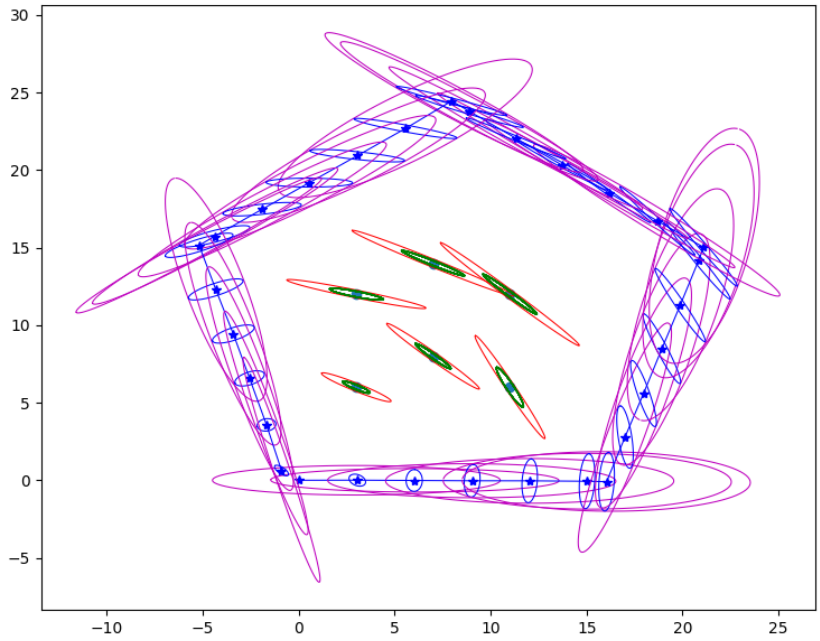

Figure 6: The output with σ_x increased by 10.

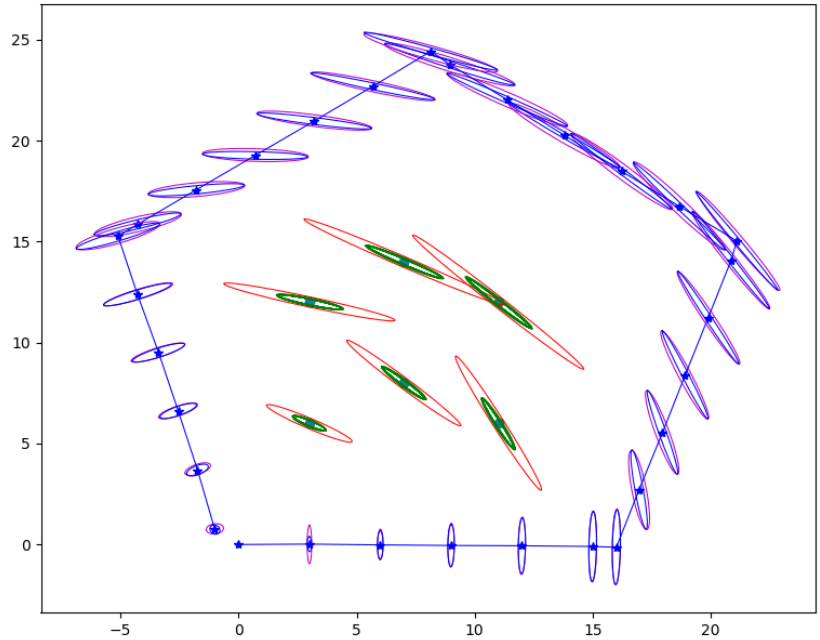

Figure 7: The output with σ_x decreased by 10.

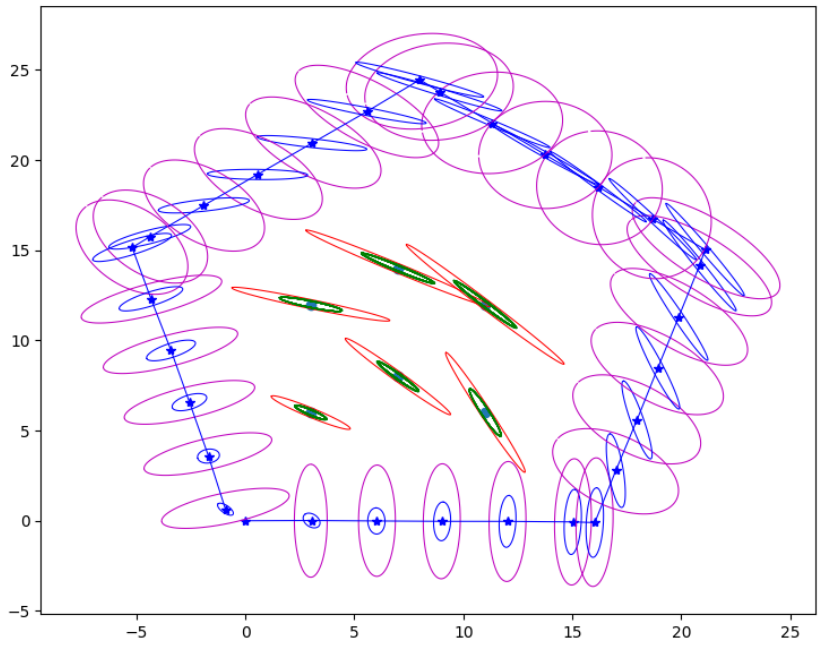

Figure 8: The output with σ_y increased by 10.

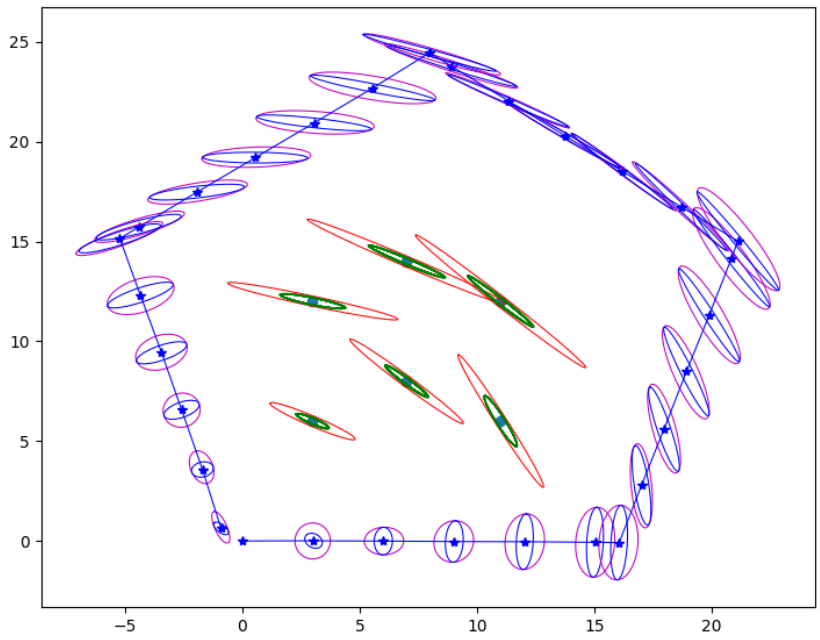

Figure 9: The output with σ_y decreased by 10.

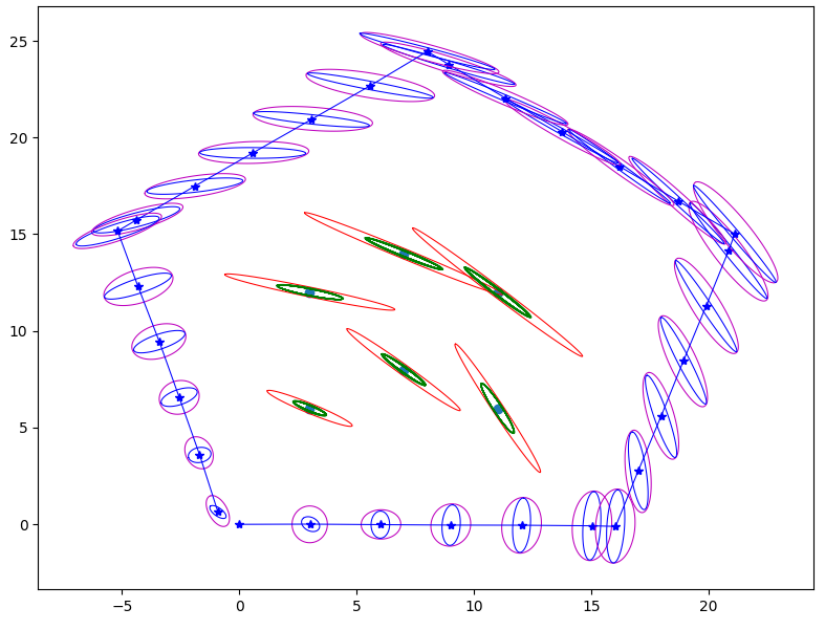

Figure 10: The output with σ_α increased by 10.

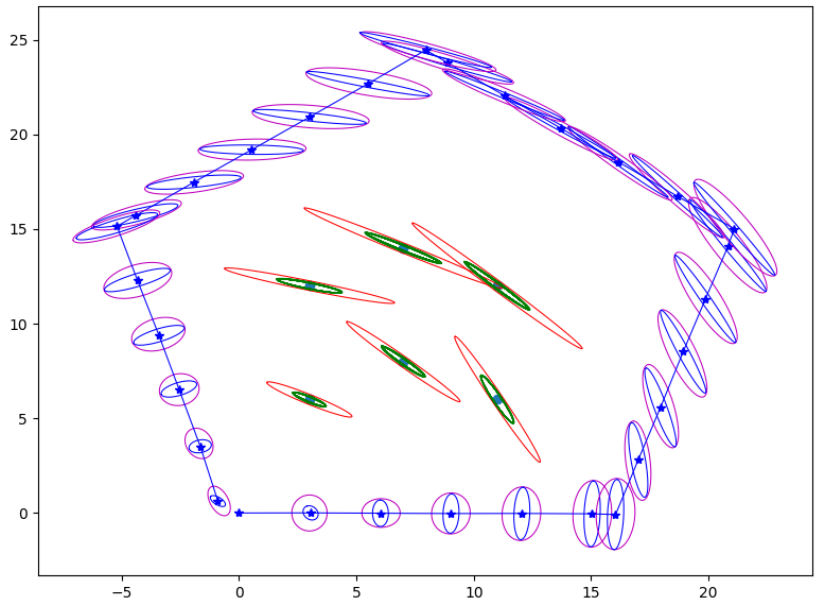

Figure 11: The output with σ_α decreased by 10.

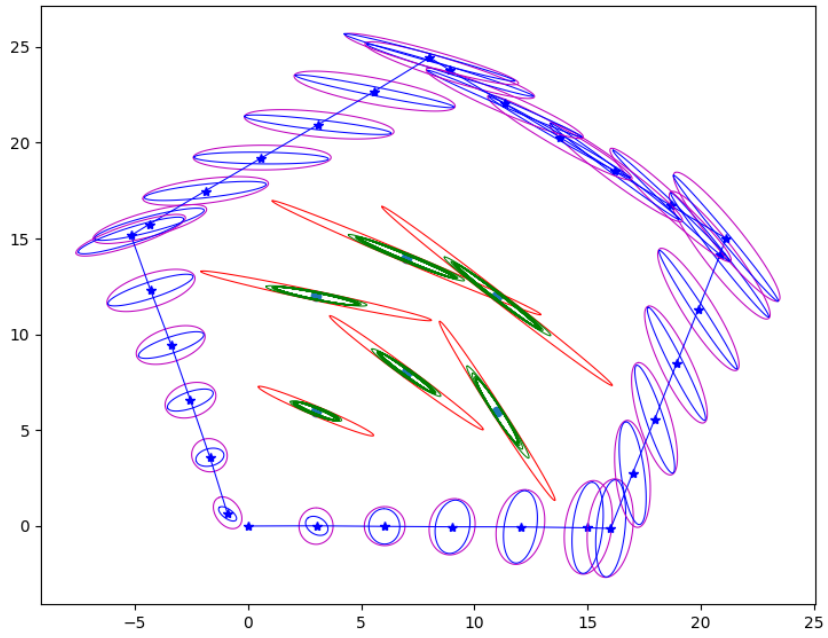

Figure 12: The output with σ_β increased by 10.

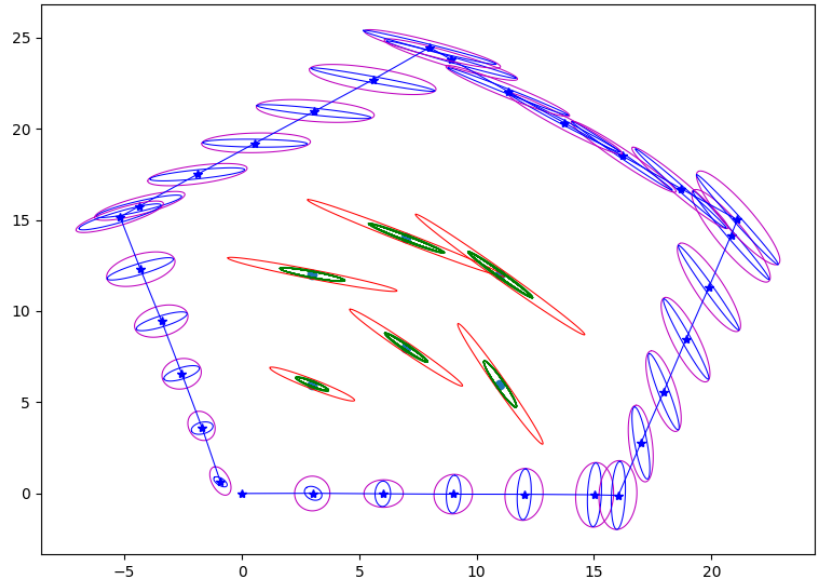

Figure 13: The output with σ_β decreased by 10.

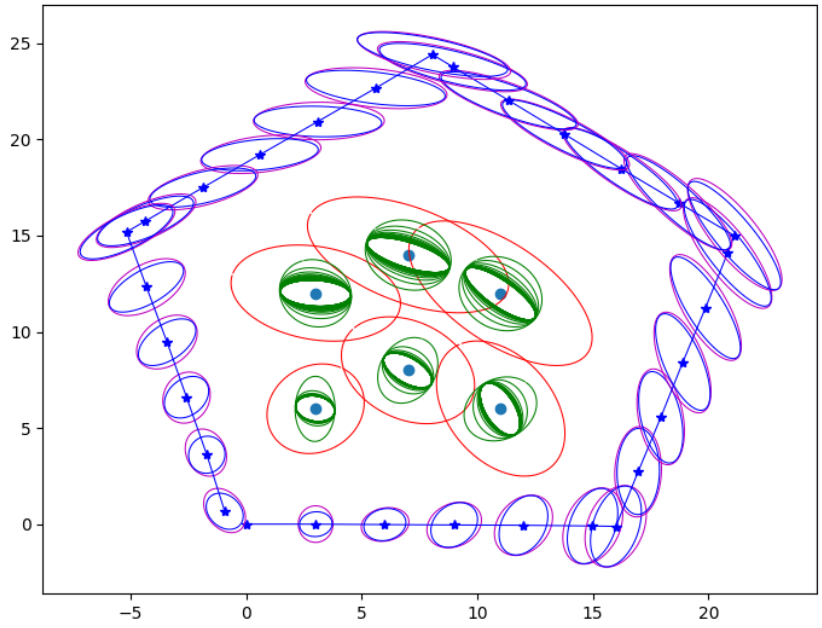

Figure 14: The output with σ_r increased by 10.

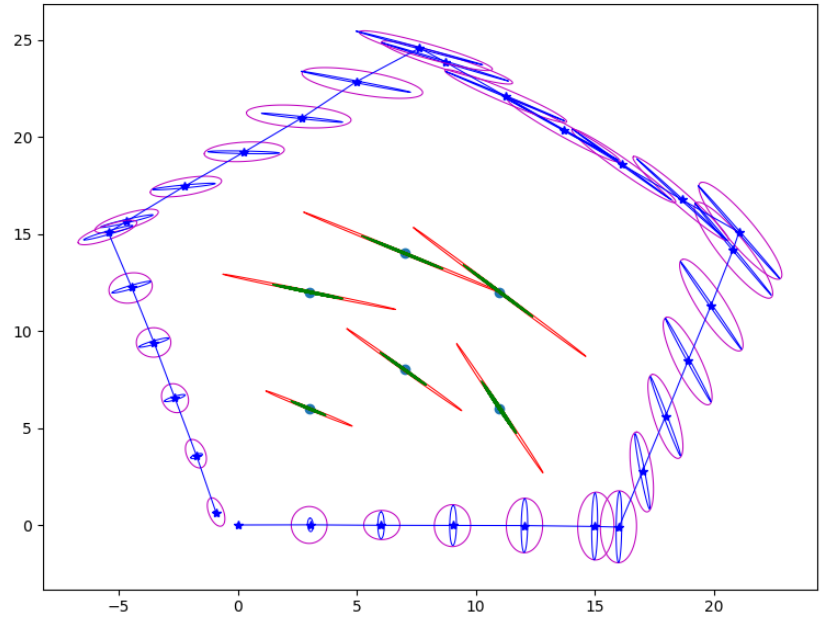

Figure 15: The output with σ_r decreased by 10.

3.3

In real-time applications, there are several possible solutions for achieving constant computation time or improving the process speed. We can consider incremental update techniques to prevent re-calculating the entire system at each time step. The use of sparse matrix representation for the covariance matrices can also reduce computational time. We can also incorporate more efficient data structures like Bayes trees or KD-trees to better organize and search for the landmarks' data. The use of incremental smoothing like iSAM2 can be another method that can cut down computation time. There are many more approaches one can consider when trying to improve implementation efficiency.

References

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., & Dellaert, F. (2012). iSAM2: Incremental smoothing and mapping using the Bayes tree. The International Journal of Robotics Research, 31(2), 216–235. https://doi.org/10.1177/0278364911430419