16-833: Robot Localization and Mapping, Spring 2021
Homework 2 - SLAM using Extended Kalman Filter (EKF-SLAM)

Aditya Ramakrishnan

March 18, 2022

1 Theory

1. Predict the next pose $\mathbf{p}_{t+1}$ as a nonlinear function of the current pose $\mathbf{p}_t$ and the control inputs d_t and α_t.

   Solution:

   Pose matrix for current time step (t)

$$\mathbf{p}_t = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix}$$

   Pose matrix for time step (t+1) as a non-linear function of $\mathbf{p}_t$, d_t and α_t

$$\mathbf{p}_{t+1} = \begin{bmatrix} x_t + d_t.cos(\theta_t) \\ y_t + d_t.sin(\theta_t) \\ \theta_t + \alpha_t \end{bmatrix}$$

2. What is the predicted uncertainty of the robot at time $t+1$, if the uncertainty of the robot's pose at time t can be represented as a 3-dimensional Gaussian distribution?

   Solution: Predicted Uncertainty in Robot Frame

$$\sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\alpha^2 \end{bmatrix} \right)$$

   Predicted Uncertainty in Global Frame

$$\begin{bmatrix} cos(\theta_t) & -sin(\theta_t) & 0 \\ sin(\theta_t) & cos(\theta_t) & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\alpha^2 \end{bmatrix} . \begin{bmatrix} cos(\theta_t) & sin(\theta_t) & 0 \\ -sin(\theta_t) & cos(\theta_t) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3. Write down the estimated position (l_x, l_y) of landmark l in global coordinates as a function of $\mathbf{p}_t$, β, r, and the noise terms.

   Solution:

   The landmark position in global coordinates can be written as

$$\mathbf{l} = \begin{bmatrix} l_x \\ l_y \end{bmatrix} = \begin{bmatrix} P_{t_x} + (r + n_\beta).\mathbf{cos}(\beta + n_\beta + P_{t_\theta}) \\ P_{t_y} + (r + n_\beta).\mathbf{sin}(\beta + n_\beta + P_{t_\theta}) \end{bmatrix}$$

4. Predict the measurement of bearing and range based on l_x, l_y, $\mathbf{p}_t$, and the noise terms (use functions *np.arctan2*(.) and *warp2pi*(.) that warps an arbitrary angular value into the range $(-\pi, \pi]$ if needed).

   Solution:

   (i) Bearing β can be found using a arctan from slope calculation between landmark 2D coordinates and the robots current position. Then we add the noise term :

$$\theta_{\mathbf{p}-\mathbf{l}} = np.arctan2[(l_y - P_{t_y}), (l_x - P_{t_x})]$$

$$\beta = warp2pi(\theta_{\mathbf{p}-\mathbf{l}} - P_{t_\theta} + n_\beta)$$

   (ii) Range r can be found using a simple distance formula calculation between landmark 2D coordinates and the robots current position. Then we add the noise term:

$$\mathbf{r} = \sqrt{(l_y - P_{t_y})^2 + (l_x - P_{t_x})^2} + n_\beta$$

5. Find the analytical Jacobian H_p with respect to the robot pose.

   Solution:

$$\mathbf{H_p} = \begin{bmatrix} \frac{\partial \beta}{\partial P_{t_x}} & \frac{\partial \beta}{\partial P_{t_y}} & \frac{\partial \beta}{\partial P_{t_\theta}} \\ \frac{\partial r}{\partial P_{t_x}} & \frac{\partial r}{\partial P_{t_y}} & \frac{\partial r}{\partial P_{t_\theta}} \end{bmatrix} = \begin{bmatrix} \frac{(l_y - P_{t_y})}{(l_y - P_{t_y})^2 + (l_x - P_{t_x})^2} & \frac{(P_{t_x} - l_x)}{(l_y - P_{t_y})^2 + (l_x - P_{t_x})^2} & -1 \\ \frac{(P_{t_x} - l_x)}{\sqrt{(l_y - P_{t_y})^2 + (l_x - P_{t_x})^2}} & \frac{(P_{t_y} - l_y)}{\sqrt{(l_y - P_{t_y})^2 + (l_x - P_{t_x})^2}} & 0 \end{bmatrix}$$

6. Find the analytical Jacobian H_l with respect to its corresponding landmark l. Why do we not need to calculate the measurement Jacobian with respect to other landmarks except for itself ?

   Solution:

   Each landmark is assumed to be independent of other landmarks, hence we only calculate the Jacobian H_l with respect to its corresponding landmark l.

$$\mathbf{H_l} = \begin{bmatrix} \frac{\partial \beta}{\partial l_x} & \frac{\partial \beta}{\partial l_y} \\ \frac{\partial r}{\partial l_x} & \frac{\partial r}{\partial l_y} \end{bmatrix} = \begin{bmatrix} \frac{(P_{t_y} - l_y)}{(l_y - P_{t_y})^2 + (l_x - P_{t_x})^2} & \frac{(P_{t_x} - l_x)}{(l_y - P_{t_y})^2 + (l_x - P_{t_x})^2} \\ \frac{(l_x - P_{t_x})}{\sqrt{(l_y - P_{t_y})^2 + (l_x - P_{t_x})^2}} & \frac{(P_{t_y} - l_y)}{\sqrt{(l_y - P_{t_y})^2 + (l_x - P_{t_x})^2}} \end{bmatrix}$$

2 Implementation and Evaluation

1. What is the fixed number of landmarks being observed over the entire sequence?

   Solution:

   There are a total of 6 landmarks in the *data.txt* file provided.

2. Code and Visualization of the system:

   Solution:

   Refer Figure 1 - Output Plot for Visualization

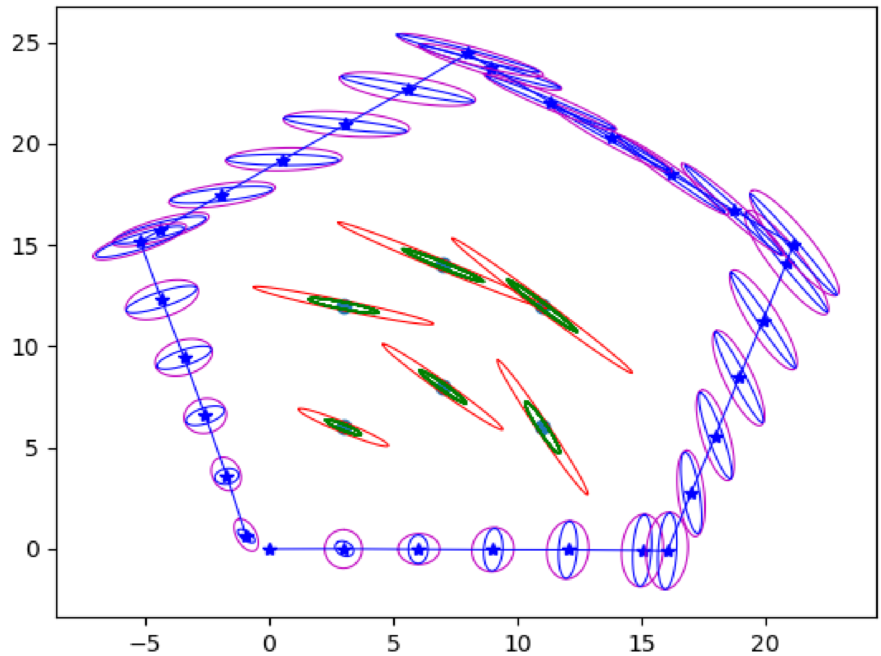

Figure 1: Output Plot

3. Describe how EKF-SLAM improves the estimation of both the trajectory and the map by comparing the uncertainty ellipses.

   Solution:

   As mentioned, the magenta ellipses represent the predicted instantaneous uncertainties of the robot's position - this is w.r.t the robot having made a prediction about the future state of the system without any observations, hence the uncertainty of the platform location increases.
   The blue ellipses represent the updated instantaneous uncertainties of the robot's position, where a real observation is made, and the discrepancies between actual observation and predicted observation describe our updation step. Here the uncertainty reduces by a margin determined by the gain factors describing the system.

We can clearly identify this by the blue ellipses being shown within the magenta ellipses.

4. Assumed ground truth positions of all the landmarks, plot the ground truth positions of the landmarks in the output figure and attach it below. Is each of them inside the smallest corresponding ellipse? What does that mean?

   Compute and report the *Euclidean* and *Mahalanobis* distances of each landmark estimation with respect to the ground truth. What do the numbers tell you?

   Solution:

   Each of the ground truth landmarks are within the smallest corresponding ellipse which implies the positioning is accurate.

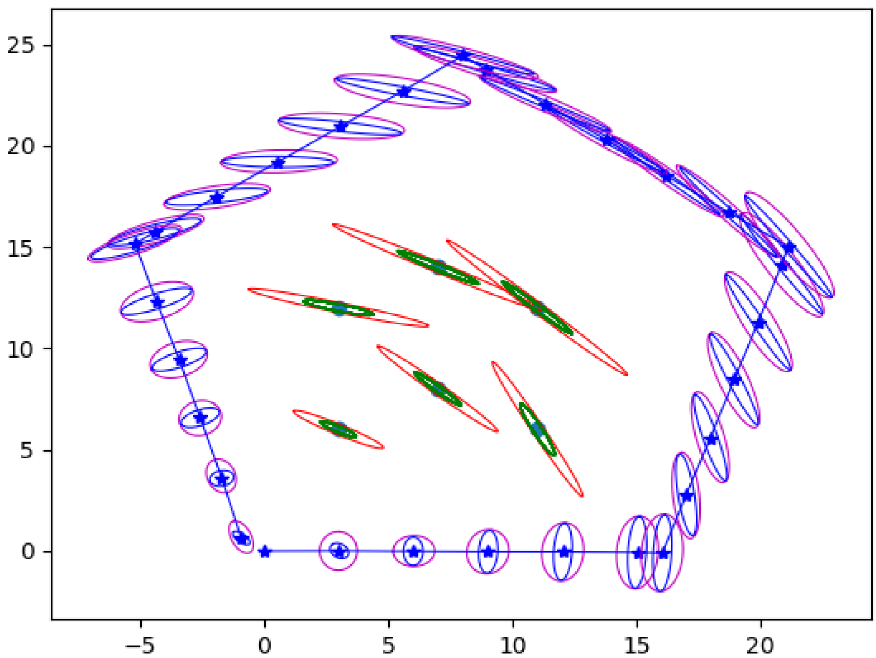

Figure 2: Output Plot

The Euclidian and Mahalanobis distances are quite small and indicate the distance between actual position of and its estimate and distribution respectively. Since the values seen below are quite small, it is indicative of the good preciseness and accuracy of the model.

```
l =  1
Eucliean Distance =  [0.00309959]
Mahalonobis Distance =  [0.00051219]
-------------------------------------------------------------------------------------
l =  2
Eucliean Distance =  [0.00741726]
Mahalonobis Distance =  [0.00281787]
-------------------------------------------------------------------------------------
l =  3
Eucliean Distance =  [0.00206045]
Mahalonobis Distance =  [0.00053611]
-------------------------------------------------------------------------------------
l =  4
Eucliean Distance =  [0.00501295]
Mahalonobis Distance =  [0.00231418]
-------------------------------------------------------------------------------------
l =  5
Eucliean Distance =  [0.0034668]
Mahalonobis Distance =  [0.00134018]
-------------------------------------------------------------------------------------
l =  6
Eucliean Distance =  [0.00760238]
Mahalonobis Distance =  [0.003674]
-------------------------------------------------------------------------------------
```

Figure 3: Distances

3 Discussion

1. Explain why the zero terms in the initial landmark covariance matrix become non-zero in the final state covariance matrix. When setting the initial value for the full covariance matrix an assumption is made regarding certain cross-covariances that is not necessarily correct. Can you point out what that is?

   Solution:

   The zero terms in the initial landmark covariance matrix change due to uncertainties introduced from robot motion which we account for in the correction step (control inputs and noise factors contribute to the uncertainty as well). The cross covariances between landmarks are assumed to be zero during initialization in our problem, which may not be the case in reality.

2. Play with the parameters (line 163-167). Modify each parameter σ_x, σ_y, σ_α,σ_b, σ_r at a time and fix the others. In particular, you should set each parameter to be 10 times larger/smaller to the original parameters to discuss how each parameters influence the results. Attach figures for better explanation.

   Solution:

   It is observed that changing the values of σ_x and σ_y have a significant impact on the ellipses and uncertainty distributions, however, the effect of changing σ_α is relatively negligible. This is accounted to the fact that more weightage is given to the prediction of the robot pose and its associated uncertainty.

   With regard to the σ_b, the effect is not as pronounced on the system, however changing the value of σ_r imacts the system in a large manner. Here, the impact can be observed in the updation step and the Kalman gains as uncertainty is introduced in all the states - both pose as well as landmarks.

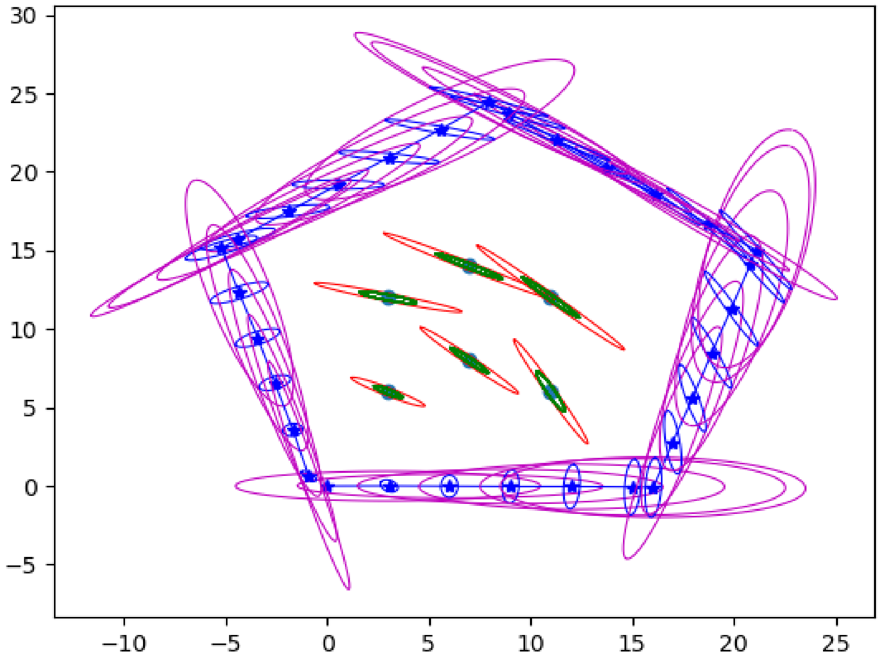

Figure 4: 10 X σ_X

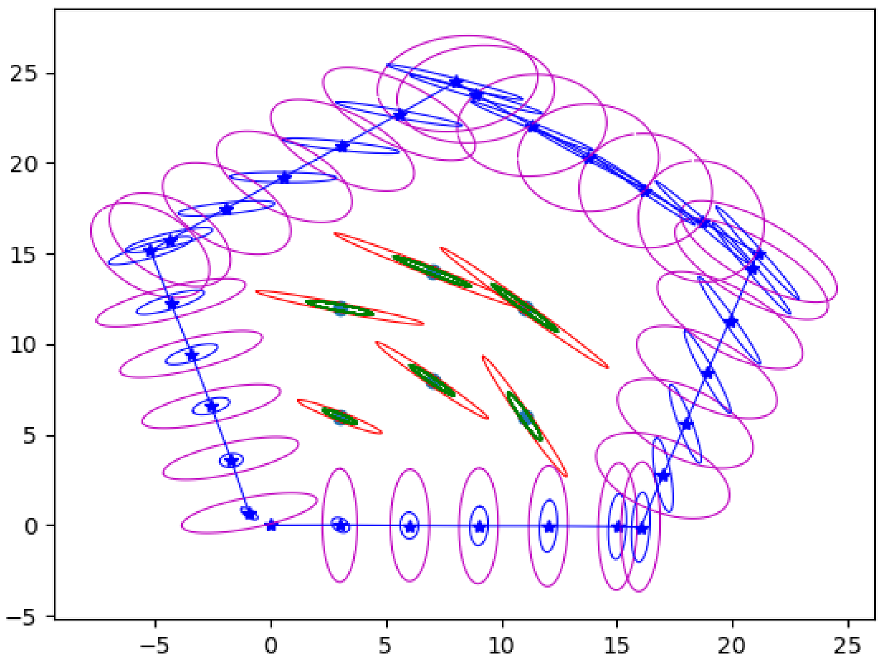

Figure 5: 10 x σ_y

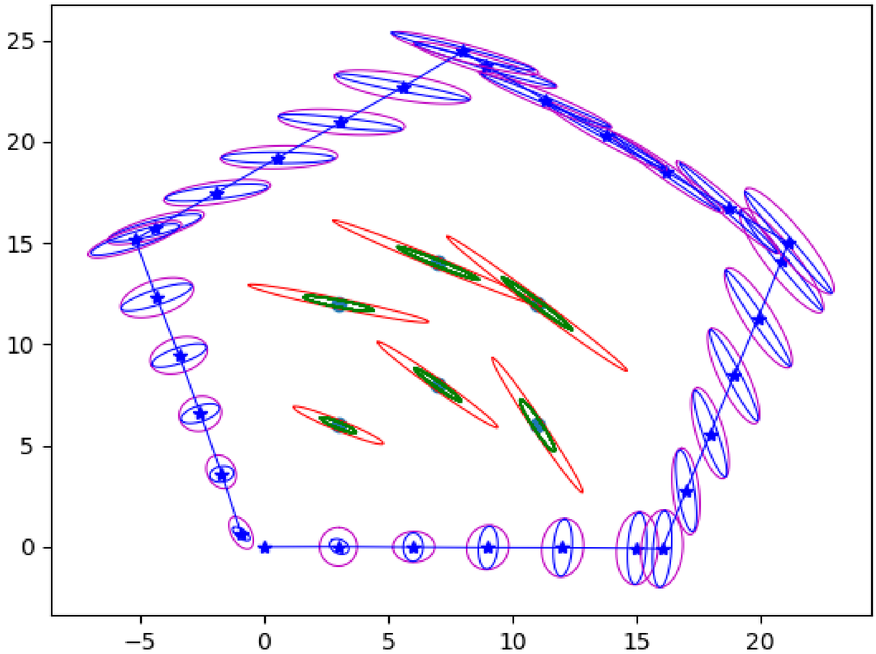

Figure 6: 10 x σ_α

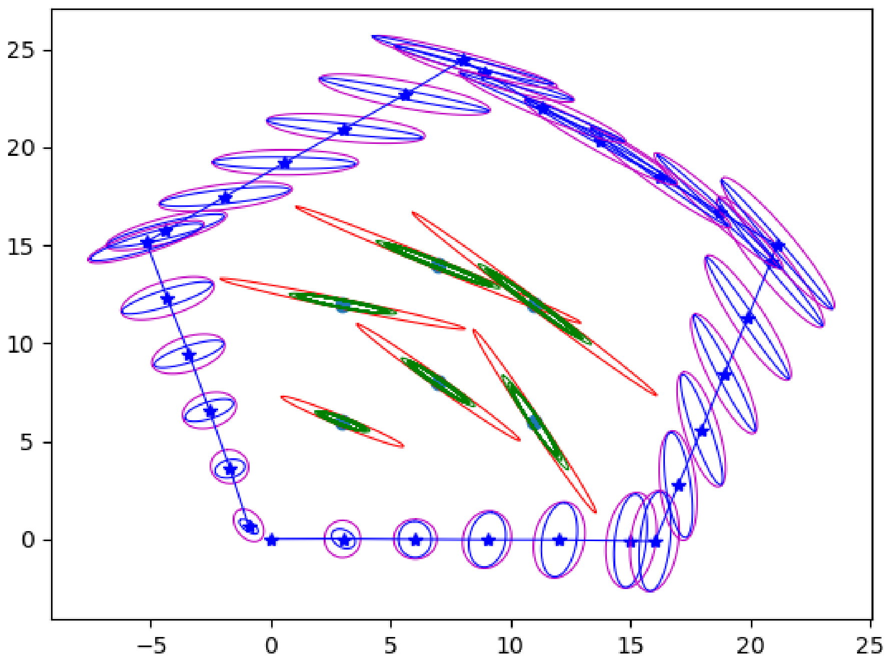

Figure 7: 10 x σ_β

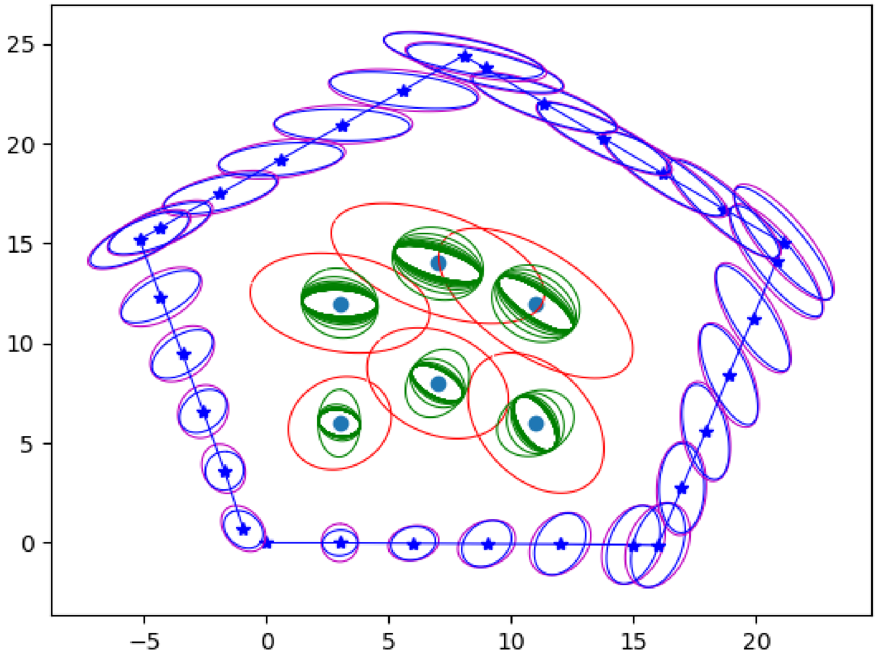

Figure 8: 10 x σ_r

3. With the same set of landmarks being observed all the time, the EKF-SLAM system runs in constant time in each iteration. However, for real-world problems, the total number of landmarks can grow unbounded if the robot keeps exploring new environments. This will slow down the EKF-SLAM system a lot and become a crucial problem for real-time applications. What changes can we make to the EKF-SLAM framework to achieve constant computation time in each cycle or at least speed up the process (list as many possible solutions as you can think of)?

   Solution:

   As mentioned in the book - Probabilistic Robotics by Sebastian Thrun, we can approach an unbounded problem, by removing the need to compute a joint posterior over the whole path of the robot and the map (full SLAM). Instead we can implement online SLAM to estimate a posterior over the current robot pose along with the map - which saves a tremendous amount of computational time and resources. Additionally, as seen in our previous assignment, we can vectorize our input data and variables in order to remove computation downtime from multiple for-loops. Another approach would involve using Reinforcement Learning Algorithms to better optimize the robots ability to localize and map the environment effectively by adding a cost factor to the estimation process.