16-833: Robot Localization and Mapping, Fall 2020
Homework 2 - SLAM using Extended Kalman Filter (EKF-SLAM)

Due: Friday Oct 23, 11:59pm, 2020

Your homework should be submitted as a **typeset PDF file** along with a folder including **code only (no data)**. The PDF must be submitted on **Gradescope**, and code submitted on **Canvas**. In your implementation, please only write your code under the sections marked as `TODO`, `TEST`, or `EVAL`, and do not change any other line. We have two versions: MATLAB and Python, choose whichever you feel most comfortable with. If you have questions, post them on Piazza or come to office hours. Please do not post solutions or codes on Piazza.

This homework must be done **individually**, and plagiarism will be taken seriously. You are free to discuss and troubleshoot with others, but the code and writeup must be your own. Note that you should list the name and Andrew ID of each student you have discussed with on the first page of your PDF file.

1. Related Math and Equations (40 points)

In this part we are going to guide you through some steps of EKF-SLAM in math. This will be helpful in your implementation in the next problem.

A robot is moving on the 2D ground plane. In each time step t, the robot is controlled to move forward (the x-direction of the robot's coordinates) d_t meters, and then rotate α_t radian. The pose of the robot in the global coordinates at time t is written as a vector $\mathbf{p}_t = \begin{bmatrix} x_t & y_t & \theta_t \end{bmatrix}^\top$, where x_t and y_t are the 2D coordinates of the robot's position, and θ_t is the robot's orientation.

(a) Based on the assumption that there is no noise or error in the control system, 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. (5 points)

(b) However, in reality there are some errors when the robot moves due to the mechanism and the terrain. Assume the errors follow Gaussian distributions: $e_x \sim \mathcal{N}\left(0, \sigma_x^2\right)$ in x-direction, $e_y \sim \mathcal{N}\left(0, \sigma_y^2\right)$ in y-direction, and $e_\alpha \sim \mathcal{N}\left(0, \sigma_\alpha^2\right)$ in rotation respectively (all in robot's coordinates). Now if the uncertainty of the robot's pose at time t can be represented as a 3-dimensional Gaussian distribution $\mathcal{N}\left(0, \Sigma_t\right)$, what is the predicted uncertainty of the robot at time $t+1$? Please express it as a Gaussian distribution with zero mean. (5 points)

(c) Consider a landmark l being observed by the robot at time t with a laser sensor which gives a measurement of the bearing angle β (in the interval $(-\pi, \pi]$) and the range r, with noise $n_\beta \sim \mathcal{N}\left(0, \sigma_\beta^2\right)$ and $n_r \sim \mathcal{N}\left(0, \sigma_r^2\right)$ respectively. 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. (5 points)

(d) According to (c), if we now know that l is at (l_x, l_y) in the global coordinates, please predict the measurement of bearing and range based on l_x, l_y, $\mathbf{p}_t$, and the noise terms (use functions atan2(.) and warpToPi(.) that warps an arbitrary angular value into the range $(-\pi, \pi]$ if needed). (5 points)

(e) An important step in EKF-SLAM is to find the measurement Jacobian H_p with respect to the robot pose. Please apply the results in (d) to derive the analytical form of H_p (Note: H_p should be a 2×3 matrix represented by $\mathbf{p}_t$ and l). (10 points)

(f) For each measurement of bearing and range, we also need a measurement Jacobian H_l with respect to its corresponding landmark l. Please again derive the analytical form of H_l (Note: H_l should be a 2×2 matrix represented by $\mathbf{p}_t$ and l). Why do we not need to calculate the measurement Jacobian with respect to other landmarks except for itself (Hint: based on what assumption)? (10 points)

2. Implementation and Evaluation (45 points)

In this part you will implement your own 2D EKF-SLAM solver in Matlab/Python. The robot in problem 1 starts with observing the surrounding environment and measuring some landmarks, then executes a control input to move. The measurement and control steps are repeated several times. For simplicity, we assume the robot observes the same set of landmarks in the same order at each pose. We want to use EKF-SLAM to recover the trajectory of the robot and the positions of the landmarks from the control input and measurements.

(a) Find the data file `data/data.txt` that contains the control inputs and measurements. Open the file to take a look. The data is in the format of $control = \begin{bmatrix} d & \alpha \end{bmatrix}$ (refer to problem 1 for the notations) and $measurements = \begin{bmatrix} \beta_1 & r_1 & \beta_2 & r_2 & \cdots \end{bmatrix}$, where β_i, r_i correspond to landmark l_i. The lines are in sequential time order. Question: what is the fixed number of landmarks being observed over the entire sequence? (5 points)

(b) The `EKF_slam_matlab/` folder contains the Matlab code file `EKF_SLAM.m` (or `EKF_SLAM.py` in `EKF_slam_python/`). Follow the comments and fill in all the parts marked with `TODO`. Only add in your lines and do not change any existing lines. After finishing them properly, you will get an output figure with each blue "*" representing a position of the robot, and the blue edges showing the estimated trajectory. The uncertainties of the positions of both the robot and the landmarks are shown as 3-sigma error ellipses. Attach a clear figure of your result in the PDF file. (25 points)

(c) In the output figure, the magenta and blue ellipses represent the predicted and updated uncertainties of the robot's position (orientation not shown here) at each time respectively. Also, the red and green ellipses represent the initial and all the updated uncertainties of the landmarks, respectively. Describe how EKF-SLAM improves the estimation of both the trajectory and the map by comparing the uncertainty ellipses. (5 points)

(d) Now let's evaluate our output map. Suppose we have the ground truth positions of all the landmarks:

$$\begin{bmatrix} l_1^\top & l_2^\top & \cdots & l_k^\top \end{bmatrix} = \begin{bmatrix} 3 & 6 & 3 & 12 & 7 & 8 & 7 & 14 & 11 & 6 & 11 & 12 \end{bmatrix}.$$

Plot the ground truth positions of the landmarks in the output figure (write your code under `EVAL` comment in `EKF_SLAM.m`/`EKF_SLAM.py`) and attach it below. Is each of them inside the smallest corresponding ellipse? What does that mean? Write down the *Euclidean* and *Mahalanobis* distances of each landmark estimation with respect to the ground truth. What do the numbers tell you? (10 points)

3. Discussion (15 points)

(a) Explain why the zero terms in the initial landmark covariance matrix become non-zero in the final state covariance matrix (print out the final P matrix to check it). Additionally, when setting the initial value for the full covariance matrix P (line 51 in `EKF_SLAM.m` / line 118 in `EKF_SLAM.py`) an assumption is made regarding certain cross-covariances that is not necessarily correct. Can you point out what that is? (5 points)

(b) Play with the parameters (change the code under `TEST` comment in `EKF_SLAM.m`/`EKF_SLAM.py`)! Fix σ_x, σ_y, σ_α and try different σ_b, σ_r. What do you observe? What if you fix σ_b, σ_r and change σ_x, σ_y, σ_α instead? Attach figures below for better explanation. (10 points)

(c) 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)? (Bonus 5 points)

4. Code Submission

Upload all your code in the `EKF_slam/python/` or `EKF_slam/matlab/` folder. Please do not upload the `data/` folder or any other data or image.