CSE-483: Mobile Robotics **Monsoon 2019**

Assignment-4

Released: October 12th *Deadline: October 26th*

Instructions

- This assignment is designed to get you familiar with localization using an Extended Kalman filter.
- Code can be written in Python or C++ only. Make sure your code is modular since you might be reusing them for future assignments.
- Submit your code files and a report (or as a single Jupyter notebook if you wish) as a zip file, named as ⟨team_id⟩_assignment4.zip.
- The report should include all outputs and results and a description of your approach. It should also briefly describe what each member worked on. The scoring will be primarily based on the report.
- Refer to the late days policy and plagiarism policy in the course information document.

Q) Localization using sensor fusion

A ground robot is driving [video] amongst a set of known landmarks. The robot has a wheel odometer that measures its translational and rotational speeds, and a laser rangefinder that measures the range and bearing to the landmarks. Both the sensors are noisy. Your task in this assignment is to estimate the 2D pose of the robot throughout its traverse using these sensor measurements by applying an Extended Kalman filter.

In this problem, the motion model of the robot and the observation model are non-linear and are as follows,

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + dt \begin{bmatrix} \cos\theta_{k-1} & 0 \\ \sin\theta_{k-1} & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} v_k \\ \omega_k \end{bmatrix} + \mathbf{w}_k \right)$$

$$\begin{bmatrix} r_k^l \\ \phi_k^l \end{bmatrix} = \begin{bmatrix} \sqrt{(x_l - x_k - d\cos\theta_k)^2 + (y_l - y_k - d\sin\theta_k)^2} \\ \text{atan2}(y_l - y_k - d\sin\theta_k, x_l - x_k - d\cos\theta_k) - \theta_k \end{bmatrix} + \mathbf{n}_k$$

(x_k, y_k, θ_k) is the robot's 2D pose that we want to estimate, (v_k, ω_k) is the robot's translational and rotational speed measured using its odometer, dt is the sampling period, $\mathbf{w}_k$ is the process noise, (r_k^l, ϕ_k^l) is the range and bearing to the landmark l, (x^l, y^l) is the position of the landmark l, $\mathbf{n}_k$ is the sensor noise, and d is the distance between the robot center and the laser rangefinder. Note that at each timestep, k, there can be many laser scan observations.

Dataset:

The data is provided to you in the form of a numpy archive file (`dataset.npz`). It contains the following variables,

t: a 12609×1 array containing the data timestamps [s].

x_true: a 12609×1 array containing the true x-position, x_k, of the robot [m].

y_true: a 12609×1 array containing the true y-position, y_k, of the robot [m].

th_true: a 12609×1 array containing the true heading, θ_k, of the robot [m].

l: a 17×2 array containing the true xy-positions, (x^l, y^l), of all the 17 landmarks [m].

r: a 12609×17 array containing the range, r_k^l, between the robot's laser rangefinder and each landmark as measured by the laser rangefinder sensor [m] (a range of 0 indicates the landmark was not visible at that timestep).

r_var: the variance of the range readings (based on groundtruth) [m^2].

b: a 12609×17 array containing the bearing, ϕ_k^l, to each landmark in a frame attached to the laser rangefinder, as measured by the laser rangefinder sensor [rad] (if the range is 0 it indicates the landmark was not visible at that timestep and thus the bearing is meaningless). The measurements are spread over a 240° horizontal-FoV, and the bearing is 0° when the landmark is straight ahead of the rangefinder.

b_var: the variance of the bearing readings (based on groundtruth) [rad^2].

v: a 12609×1 array containing the translational speed, v_k, of the robot as measured by the robot's odometers [m/s].

v_var: the variance of the translational speed readings (based on groundtruth) [m^2/s^2].

om: a 12609×1 array containing the rotational speed, ω_k, of the robot as measured by the robot's odometers [rad/s].

om_var: the variance of the rotational speed readings (based on groundtruth) [rad^2/s^2].

d: the distance, d, between the center of the robot and the laser rangefinder [m].

Use `dataset = np.load('dataset.npz')` to load the archive, and `dataset['array_name']` to access each individual array contained within it. The groudtruth data, including the locations of the landmarks, were captured using a ten-camera motion capture system [video]. Use the groundtruth pose of the robot as $\hat{\mathrm{x}}_0$ to initialize your filter and choose $\hat{\mathrm{P}}_0 = \mathrm{diag}\{1, 1, 0.1\}$. Also assume a uniform 0.1 second sampling period.

Acknowledgement: This is from the "*Lost in the woods*" dataset created by Prof. Tim Barfoot.

Deliverables

Code aside, your report must include the following,

- A description of your filter design including expressions for all the Jacobians involved.
- A plot of the trajectory estimated only using the wheel odometer measurements, along with the groundtruth trajectory and the landmark locations. This is just to appreciate why we cannot rely only on wheel odometry.
- A plot of your EKF-estimated robot trajectory, along with the groundtruth trajectory and the landmark locations.
- [Bonus] A video showing your EKF estimate as the robot drives around, as a dot on a map, along with the true robot position and the landmark locations. Also show the estimated 3-sigma covariance ellipse associated with every position.