ELEC 3210 Introduction to Mobile Robotics
Hong Kong University of Science and Technology
Fall 2023, Huan Yin

Project 2: EKF-based SLAM

TA: Zhijian Qiao & Qiucan Huang

1 Project Work

1.1 Instruction

In Project 2, you'll be required to implement the extended Kalman filter for a 2D landmark-based SLAM system. You'll receive a ROS package, ekf_slam, which contains skeleton code for the filter. You'll need to follow the instructions detailed in the last page, step by step.

In the prediction phase, we provide two key parameters: the velocity of axis X and the angular velocity of axis Z. These are stored in the variable `ut` of the `EKFSLAM :: run` function. Given the differential wheeled robot model, only these two values should be non-zero in theory. We also assume that there are noises in both values, represented by σ_v and σ_ω. The interval time is denoted by `dt`.

In the update phase, you'll be given code for cylinder detection. This code generates a set of 2D coordinates for the centers of detected cylinders in the body frame. These are represented by `Eigen :: MatrixX2d cylinderPoints`. Your task is to associate cylinder observations with your landmarks in the state and to use observed cylinders to update your pose and landmarks. If a new observation occurs, you should augment your state and covariance matrix. Remember to keep the angle between -pi and pi.

Every section of the package that requires your implementation is marked by `TODO`. If you're unsure of your next steps, searching for `TODO` globally in the package could help guide you. If you encounter a bug, a useful debugging strategy is to print out as much information as possible.

1.2 Provided Materials

- Rosbag and the ground truth trajectory are the same as the last project.
- Run-able but incomplete code.

2 Running Example

```
cd catkin_elec3210/src/
catkin_init_workspace
cd .. && catkin_make
source devel/setup.bash
# ekf slam
```

```
roslaunch ekf_slam ekf_slam.launch
```

3 Scoring Rules

Notes:

1. Please title your submission "PROJECT2-YOUR-GROUP-ID". Your submission should include a maximum 2-page report and all necessary files to run your code, and a video, excluding the bag.
2. The report should contain at least a visualization of your final point cloud map, trajectory, and contribution of each group member. The code should save your best trajectory to a txt file, so we can compare it with your provided result in the report.
3. If your code is similar to others' or any open-source code on the internet, the score will be 0 unless the sources are properly referenced in your report. Refer to [1] for details.
4. Excessive white space, overly large images, or redundant descriptions in report will not contribute to a higher score. Simple but efficient code is preferred.
5. If any problem, please google first. If unsolved, create a discussion on Canvas.
6. If you have no experience of C++ coding, start this project as early as possible. Programming proficiency is not determined by the courses you've taken, but rather by ample practice and persistent debugging.
7. Submit your code and documents on Canvas before the deadline. Scores will be deducted for late submission.

Higher scores will be awarded if:

1. Your estimated trajectory is more accurate and your running time is less.
2. A concise and clear method description, along with ample ablation studies using the method of control variables.
3. Coding: object-oriented programming logic, initializing each variable in the constructor, standardized variable and function naming, and a clear code structure.

[1] https://registry.hkust.edu.hk/resource-library/academic-integrity

1. Prediction

State: $\mu = \begin{bmatrix} x & y & \theta & m_1 & \dots & m_n \end{bmatrix}^T \quad \Sigma = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xm} \\ \Sigma_{mx} & \Sigma_{mm} \end{bmatrix}$

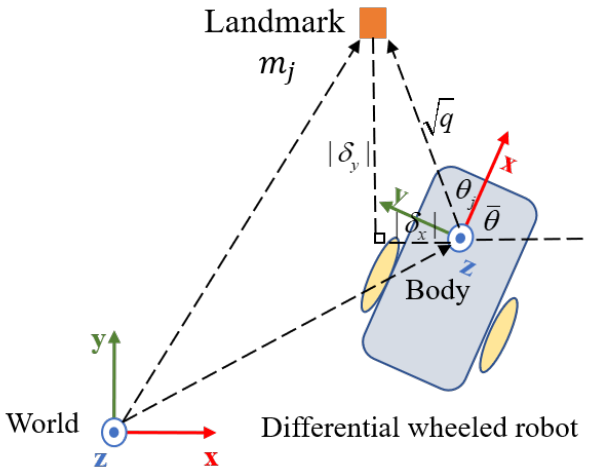
Robot control: $u_t = \begin{bmatrix} v_x \\ \omega_z \end{bmatrix}, n = \begin{bmatrix} \sigma_v \\ \sigma_\omega \end{bmatrix}, R_n = \mathrm{cov}(n)$

Motion function:

$$\bar{\mu}_t = \mu_{t-1} + Bu_t$$
$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + F_t R_n F_t^T$$

$$B = F_t = \begin{bmatrix} \Delta t \cos(\theta_{t-1}) & 0 \\ \Delta t \sin(\theta_{t-1}) & 0 \\ 0 & \Delta t \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix} \quad G_t = \begin{bmatrix} 1 & 0 & -v_x \Delta t \sin(\theta_{t-1}) & 0 & \cdots & 0 \\ 0 & 1 & v_x \Delta t \cos(\theta_{t-1}) & \vdots & \ddots & \vdots \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 \end{bmatrix}$$

2. Update



$$\delta_x = m_{j,x} - \bar{x}$$
$$\delta_y = m_{j,y} - \bar{y}$$
$$q = (m_{j,x} - \bar{x})^2 + (m_{j,y} - \bar{y})^2$$
$$\hat{z}_t = h(\bar{\mu}_t, n_m) \rightarrow \begin{bmatrix} \sqrt{q} \\ \arctan(\delta_y, \delta_x) - \bar{\theta} \end{bmatrix}$$

$$H_t = \frac{1}{q} \begin{bmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & \sqrt{q}\delta_x & \sqrt{q}\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \underbrace{0}_{2*j-2} & 0 & 1 & \underbrace{0}_{2n-2j} \end{bmatrix}$$

$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t (z_t - \hat{z}_t)$$
$$\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$$

Figure 1: The instruction for EKF-SLAM.