Lab5 - feature-based EKF SLAM

Probabilistic Robotics

Songyou Peng
psy920710@gmail.com

May 14th , 2016

1 Introduction

Simultaneous Localization and Mapping (SLAM) is one of the fundamental problems in the robotics field. The main idea is: a robot localizes itself and builds a map at the same time in a brand new environment. Because of its practical guiding significance, SLAM has already been applied in many aspects in our daily lives.

The feature-based EKF SLAM algorithm has almost the same framework and idea like EKF, but an essential improvement is inserting and updating new map features in the state vector. Since two algorithms share many similarities and differences, we will not only discuss the implementation details but also have a great interest to compare both methods through the report. Moreover, some observations about the EKF SLAM and problems we faced will also be mentioned.

2 Implementation

The algorithm is like EKF mainly composed of 3 main steps: prediction, data association, and updating. We will respectively discuss each step in details. First of all, some notations will be used throughout this report so they should be given in advance. n is the number of associated map lines in the state vector. m counts for new observed non-associated line in a unit of observation time.

2.1 Prediction

After getting an odometry, the robot ought to be able to estimate the next state of itself by compounding previous state with the odometry. There also always exists some noise during the period of prediction. Therefore, we can see both the estimated state and noise as a zero-mean Gaussian distribution, whose mean and variance (uncertainty) can be written as:

$$\hat{x}_k^- = f(\hat{x}_{k-1}^-, u_k, 0) \tag{1}$$

$$P_k^- = F_k P_{k-1} F_k^T + G_k Q_k G_k^T \tag{2}$$

Since the state vector in SLAM also contains the map features, its length should be $3+2n$. Correspondingly, P_k^- has the size of $(3+2n)\times(3+2n)$. Unlike EKF, the F_k and G_k in this case can be fomulated as follows:

$$F_k = \begin{bmatrix} J_{1\oplus} & 0 & \cdots & 0 \\ 0 & I & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & I \end{bmatrix}_{(3+2n)\times(3+2n)} , G_k = \begin{bmatrix} J_{2\oplus} \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{(3+2n)\times 3}$$

where $J_{1\oplus}$ and $J_{2\oplus}$ are the Jacobian matrix with respect to robot pose and noise, which have been used as A_k and W_k in the lab4 EKF.

2.2 Data Association

The thought of data association is the same as EKF localization. Every time when sensing a line, we calculate the Mahalanobis distance between all the map features in the state vector and the sensed line. As long as the minimum distance is smaller than a chi-square threshold, we will update the state vector and uncertainty matrix, which will be mentioned in Sec. 2.3. If the distance is large, we can say the sensed line is a new map feature and augment it into the state vector.

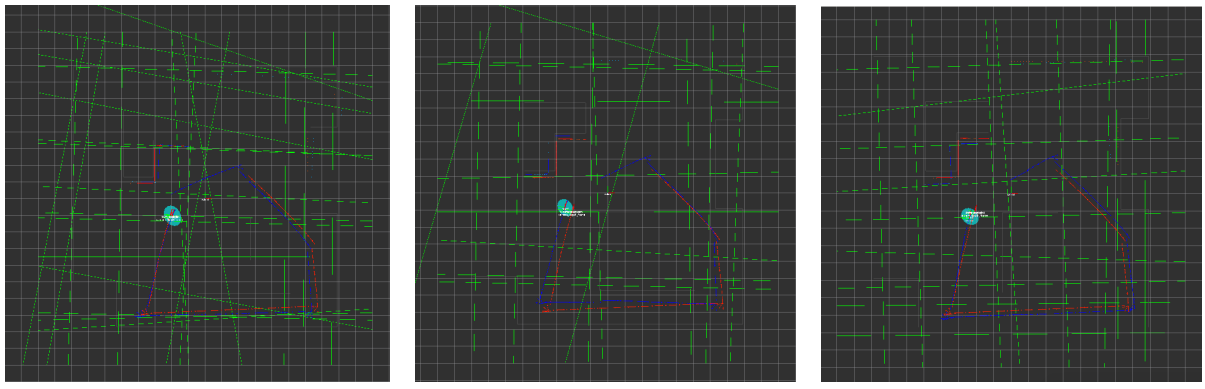
Along with the state vector, we also need to renew the uncertainty matrix P_k^- to $(3+2n+2m)\times(3+2n+2m)$, and the form should be the same as what we have already obtained in the pre-lab. There was a tricky problem we met here. In order to acquire the new P_k^-, we need to place the Jacobian matrices with respect to the current robot position and new sensed line in F_k and G_k respectively. The first step is to get the polar form of the new lines under the robot frame. The function `get_polar_line` can be used for the purpose. We should input the robot origin under the robot frame $x = 0, y = 0, \theta = 0$. Instead, my previous input was $x = self.xk[0], y = self.xk[1], \theta = self.xk[2]$, which was actually the robot position under the world frame. This input brought back the polar lines under the world frame, which led to a problem.

Another problem happened in computing Mahalanobis distance in the function `lineDist`. The Jacobian of measurement function with respect to robot position we got was in the robot frame. Since all the calculation in the following updating step was under the world frame, the "wrong " Jacobian would bring mistakes unless it was transformed back to world frame. Therefore, based on the two faced problems, we should always be careful about which frame the current position is used for calculation.

2.3 Updating

When a certain measured line is associated with a map feature, we will use the innovation between them to help update the state. Compared to EKF, here, not only will robot position be refreshed, but also the map features information will become more accurate. The whole EKF process for one iteration finishes here and the new updated state vector and uncertainty matrix will be used as the input for the next iteration.

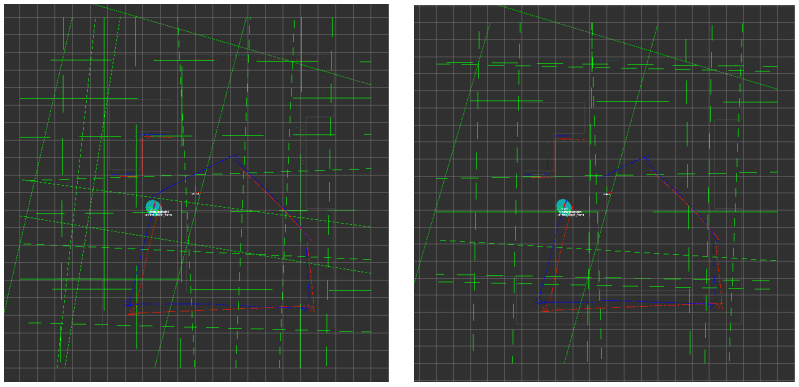
Here, we tune the chi-square threshold and the observation is interesting. That the threshold is high leads to a situation the robot trusts the sensor data, hence, less "bad" lines will be included. If it is low, the robot may have a higher probability of seeing the new sensed line as a new map feature. In this case, only a few accurate observations can be accepted and the state will not be updated often.



(a) Chi-square = 0.1 (b) Chi-square = 0.3 (c) Chi-square = 0.7

Figure 1: Illustrations for SLAM map with different chi-square thresholds

We have also done the optional part. Our idea is: use a list called `featureObservedN` is used for counting how many times map features have been associated. Only when a feature has been associated at least `min_observations`= 5 times, the map feature is ensured to be measured not because of measurement error and can be made use of updating. Every time a new sensed line is not associated and augmented to the state vector, we enlarge the list `featureObservedN`. It has been found out the change makes no significant differences in the results when the chi-square threshold is chosen properly (in our case should be about 0.7). However, as we can see in Fig. 2, adding the checking part can notably reduce the number of "bad" lines when the chi-square threshold is not chosen properly.



(a) No optional part (b) Add optional part

Figure 2: Illustrations for the benefit of the optional part with chi-square = 0.3.

3 Final remarks

In this lab, we have implemented the simultaneous localization and mapping based on map features under the EKF framework, where new state prediction, features and sensed lines association, state and features updating as well as new features augmentation are iteratively performed in order. Moreover, we count the number of appearance of every map feature in order to deal with some wrong data association situations and then the localization and mapping can become more accurately.

Meanwhile, we find out that the EKF feature-based SLAM algorithm depends on the chi-square threshold to a large extent, thus, the value should be chosen wisely. Also, the dimensions of different matrices like uncertainty and various Jacobians should be paid much attention to since every small mistake may make a huge difference in the results.