Lab5 - EKF Simultaneous Localization and Mapping

1 Introduction

The goal of this Lab exercise is to program a Simultaneous Localization and Mapping (SLAM) algorithm using an Extended Kalman Filter (EKF). The work will be done on a dataset gathered with a real Turtlebot and programmed in python.

Simultaneous Localization and mapping is a concept that in real life is used whenever a new place is visited. What a robot does while executing a SLAM algorithm is composed of two parts: build a map and localize itself into it. A map can contain multiple types of information. However in the implementation of this Lab exercise only lines sensed will be used as map features.

This EKF-SLAM is based in two different sensors. The first one are the encoders of the wheels of the Turtlebot which will give us information of the movement of the robot (odometry). The second one is the Kineckt sensor which allows the robot to sense the environment (walls, obstacles, doors...) and map the features.

2 Pre-lab

- Read and understand the guide for this Lab and look into the course slides for the feature-based EKF-SLAM.
- Find the expression of F_k and G_k for the prediction equation as functions of the jacobian of the composition with respect to the state vector $J_{1\oplus}$ and the odometry readings $J_{2\oplus}$.
- Given a feature in the robot frame z_k with uncertainty R_k. Find the equation to increment the state vector with this feature. Please, look at the function `self.tfPolarLine(tf,feature)` where the conversion of a feature between frames related by `tf` is programmed and differenciate it in order to get the equation for adding features. Is it possible to do the state augmentation with just one equation for n features? How would the matrices involved in the equations be (matrice size)?
- Find the size of the matrices H, S, v and R in order to be able to update the filter with n features observations without using a for loop.

Submit a short pdf with the equations asked in this section.

3 Lab work

For this lab work it is possible to use code previously done in the Lab4 - EKF Map Based Localization.

3.1 Running the code

In this Lab session (and thinking of the following one where one of the Lab exercises has to be tested in a real robot) the launch file has several arguments in order to work with a gathered dataset or a simulation of the robot. The different arguments and its default values are:

- bagfile [true]: if true uses the data from the bagfile
- pause [false]: if true sets the rosbag play node to pause so the messages can be posted step by step by pressing `s`.
- rviz [true]: if true displays the robot and the map in rviz.
- sim [false]: if true launches gazebo simulator with a simple room. If this option is used, run the teleoperation node for the turtlebot in a different terminal so you can drive arround the turtlebot.

3.2 Prediction

In the `predict` function implement the equations:

$$\widehat{x}^B_{k|k-1} = f\left(\widehat{x}^B_{k-1}, \widehat{u}^{k-1}_k\right) \tag{1}$$

$$P^B_{k|k-1} = F_k P^B_{k-1} F_k^T + G_k Q_k G_k^T \tag{2}$$

Use the measurement uncertainty of the previous lab (0.025m for the linear movement noise and 2 degrees for the angular). Once implemented please check that the uncertainty grows without boundaries as shown in the `prediciton.ogv` video. Note that F_k ang G_k are the ones asked in the second point of the pre-lab.

3.3 Data association

Compared to the previous lab, now the lines of the map are not saved as initial and ending point, so the equations for computing distance and transform lines between frames have to change.

First, you need to complete the function `tfPolarLine` in order to return the jacobians asked in the per-lab. Once this function is implemented use it in `lineDist` to compute the distance between a given observation z and a map feature given by its index in the map. With this two functions implemented compute the data association. Finally you also need to define the chi-square threshold in `slef.chi_thres` in the class constructor. Note that in this case the data association also returns the indexes of the non associated features in order to be able to add them to the map. Note that while the state augmentation is not fulfilled you will not be able to associate any data since the state vector will only contain the robot position.

3.4 Update

With the associations done, and taking into account the pre-lab work, update the filter without using a for loop. Note that while the state augmentation is not fulfilled you will not be able to update the robot position since no data will be associated.

3.5 State augmentation

This part is totally new from the previous lab. In this function the state vector grows with the newly observed features in order to build the stochastic map. Use the non associated observations to enlarge the state vector. Use the equations of the 3rd point of the pre-lab work in this step.

4 Optional

Since some time the line extraction algorithm is not really robust and lines which are not in the environment appear in the scan, trace each line before using it as a map feature in order to make sure this outliers do not take part into the process. Use the variables `featureObservedN` (a vector containing how many times a feature has been observed) and `min_observations` in order to assess a minimum number of observations for a feature before adding it to the map.

5 Lab report

Write a brief report explaining your solution and problems faced. Include the final code.