EKF SLAM Matlab-Lab Sessions and Theory University West - Service Robotics Lab Reports

Department of Engineering, M.Sc.Eng Robotics and Automation
University West, Trollhättan, Sweden
*Solomon Gugsa †Email:solomon.mislu-gugsa@student.hv.se ‡SER700:Service Robotics, Department of Engineering Science
§Unversity West, Trollhättan, Sweden

I. Introduction

Mobile robot has the ability to localize itself using a method called EKF Localization, but this is only true when a map of the environment is known [1]. Where as, a method to construct a map of the environment and localise relative to the map called SLAM (Simultaneously localisation Mapping) and it will be based on EKF. The Kalman filter based approaches are not very popular today anymore for solving the slam problem because they have some disadvantages especially with respect to computational complexity and expensiveness but they are still often used for if you only have a shorter map of the environment [2]. In this paper, We will see experimental SLAM we've built in the lab, based on algorithm defined in the next sections. The result and conclusion will be discussed in the last section.

II. experimentation

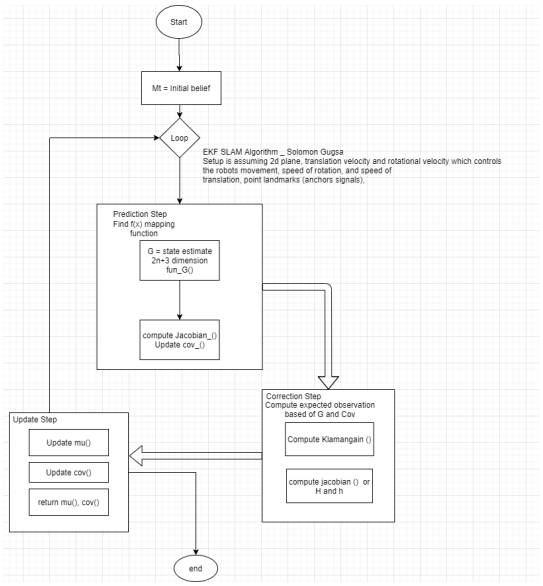
In the EKF slam experimental lab, we want to tackle the problem of estimating the location of a mobile robot as well as the map of the environment in an online approach. For that, we used the extended kalman filter to solve the simultaneous localization mapping problem. This is assumed that a localization means building a map of the environment with a mobile sensor robot and at the same time localize the robot within the map that is built and using the recursive base filter called Extended Kalman filter [3]. We're given with the following setup during the course of the project. The assumption is on 2d plane, translation velocity and rotational velocity which controls the robots movement, speed of rotation, and speed of translation, point landmarks (anchors signals) were given. We are required to come up with a map of the environment and current pose of the robot. In the following algorithm, I'm only interested finding the current pose of the robot but not the trajectory.

III. The Algo

The first step is to specify the state vector that's basically what's is being estimated. For that, motion transition model and observation function need to be defined as follows:-

$$X_t = \begin{pmatrix} x, y, , m_{1,X} , m_{1,y} ...., m_{n,x} , m_{n,y} \end{pmatrix}^T Pose, Landmark$$

$$State representation \begin{pmatrix} x \\ m \end{pmatrix} \begin{pmatrix} \sum_x x & \sum_x m \\ \sum_x m & \sum_m m \end{pmatrix}$$



(a) text1

Then the Extended Kalman Filter cycle proceeded below in the bullet points.This can also be seen on Figure 1.

- State prediction
- Predicted Measurement
- Getting Real Measurement
- Find difference between actual and expected observation
- Update Mean and Covariance

In the state prediction step, only the pose of the robot will be updated assuming that the anchors pose stay exactly similar throughout the prediction. However, while the robot at a moving pose, the uncertainty will be updated. This will represent the firs row and the first column of co-variance metrics in mat-lab implementation. The next step is to predict the pose of the robot without considering signal data (measurement data) and then update or correct the prediction based on the measurement received from signal data (Measurement landmark information). Then finding the difference between actual observation and expected observation, is preceding steps which is basically the mapping steps from state prediction steps in to observations steps. And at a final steps, mean and co-variance is updated and the robot will get more certain

about the map of the environment and correct its assumption of landmark position.

A. Initializing

Given, $M_u, Cov, myanchorsunit$

mbar$_t$ $= g(u_t, m_t\_1)$ Given the state vector, the control and approximate vector in EKF, Time update function can be obtained traditionally [4] and P represent the landmarks. However, Ignore the P in the first loop on mat-lab implementation since it's initialized with some numbers.

$$\bar{\mathbf{x}} \leftarrow f(\bar{\mathbf{x}}, \mathbf{u}, 0)$$
$$\mathbf{P} \leftarrow \mathbf{F_x P F_x^\top} + \mathbf{F_n N F_n^\top}$$

As discussed above in the prediction step the above function will only update the pose of robot and the first row and column's of co-variance metrics . This will follow finding the joacobean of the motion model as follows:

$$\mathbf{F_x} = \begin{bmatrix} \frac{\partial f_R}{\partial R} & 0 \\ 0 & \mathbf{I} \end{bmatrix}$$

Observation function

$$\mathbf{y} = h(\mathbf{x}) + \mathbf{v}$$

Correction steps: The assumption here is that I've jacobian of Hx and the co-variance R

$$\begin{aligned} \bar{\mathbf{z}} &= \mathbf{y} - h(\bar{\mathbf{x}}) \\ \mathbf{Z} &= \mathbf{H_x P H_x^\top} + \mathbf{R} \\ \mathbf{K} &= \mathbf{P H_x^\top Z^{-1}} \\ \bar{\mathbf{x}} &\leftarrow \bar{\mathbf{x}} + \mathbf{K}\bar{\mathbf{z}} \\ \mathbf{P} &\leftarrow \mathbf{P} - \mathbf{KZK}^\top \end{aligned}$$

After Kalman gain full map is constructed, then correction step follows:

$$\bar{\mathbf{x}} \leftarrow \bar{\mathbf{x}} + \mathbf{K}\bar{\mathbf{z}}$$
$$\mathbf{P} \leftarrow \mathbf{P} - \mathbf{KZK}^\top$$

and Update step is final

$$\bar{\mathbf{x}} \leftarrow \bar{\mathbf{x}} + \mathbf{K}\bar{\mathbf{z}}$$
$$\mathbf{P} \leftarrow \mathbf{P} - \mathbf{KZK}^\top$$

Please see the attached slam.m file for full implementation.

IV. Discussion

During the implementation of the above algorithm, I found that my measurement update isn't intuitive and has computational complexity of O(n) in the way it was done in the implementation, doing every measurement update independently actually needs to requires a full update of belief at every point in time so I may want to combine those observations and can kind of combine three observations and three sensor observations at once and execute the update because only need to iterate over n(landmark) square elements. Being not to Normalize the angular components has also created a deadlock in the implementation as this might affect the calculation during obtaining jacobians.Overall, EKF SLAM isn't suitable when you've a large scale map and landmarks but it could be used in a situation where a number of landmark is limited.

Acknowledgment

I would like to thank Xiaoxio, and Morgan for their help and support during the process conducting this project.

References

[1] Thrun, Sebastian, Wolfram.Burgard, , and Dieter.Fox, "Probabilistic robotics," *Aaai/iaai 593598 (2002).*, 2005.

[2] R. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *Int. Journal of Robotics Research*, p. 56–68, 1987.

[3] Montemerlo and Michael, "Fastslam: A factored solution to the simultaneous localization and mapping problem," *Aaai/iaai 593598 (2002).*, 2002.

[4] Cyril.Roussillon, Aur´elien.Gonzalez, Joan.Sol'a, Jean.MarieCodol, Nicolas.Mansard, Simon.Lacroix, and Michel.Devy, "Rt-slam: a generic and real-time visual slam implementation," *Int. Conf. on Computer Vision Systems (ICVS)*, 2011.