Lab4 Report - EKF Map Based Localization

Mohammad Rami Koujan

1 Introduction

Map-Based Localization approaches use a local map of the sensed environment, the extracted lines coming from the Split and Merge algorithm in our case, that is matched against a previously stored map to correct the robot localization in the world. In many cases these methods are based on a probabilistic representation of the spatial uncertainty and use the Kalman Filter (KF) or the Extended Kalman Filter (EKF) to update the robot's location estimation. So, the aim of this lab assignment is to get familiar with the EKF Map Based Localization by programming the correct algorithm in order to carry out this type of localization. Basically, the algorithm should work with data taken from a real Turtlebot. The main schematic of such kind of filter is shown in figure 1 where there are three main blocks (predict, data association, and update).

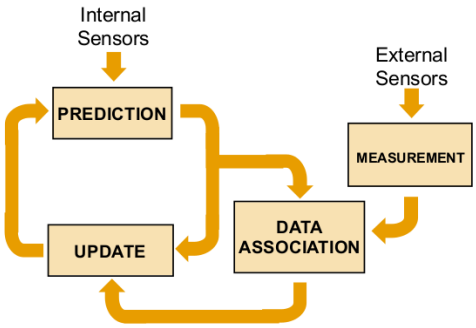

Figure 1: EKF map based localization diagram

Hence, the algorithm that was programmed in this lab has the following steps:

► ***Pose initialization***

$x_0 = \hat{x}_0$

$P_0 = \hat{P}_0$

for k = 1 **to** *steps* **do:**

$\left[u_k^{R_{k-1}}, Q_k\right] = get\_odometry()$

► ***EKF prediction***

$\left[x_{k|k-1}, P_{k|k-1}\right] = move\_vehicle(x_{k-1}, P_{k-1}, u_k, Q_k)$

$[z_k, R_k] = get\_measurements()$

► ***Data association***

$\mathcal{H}_k = data\_association(x_{k|k-1}, P_{k|k-1}, z_k, R_k)$

► ***EKF update***

$[x_k, P_k] = update\_position(x_{k|k-1}, P_{k|k-1}, z_k, R_k)$

end for

2 EKF prediction

This section explains the "prediction" part of the implemented algorithm. In fact, the main idea of this step is to compute (predict) the new position of the turtlebot from the odometry $\vec{u}_k$ and the previous

position. This is done by implementing the following equations:

► ***Prediction***

$$x_{k|k-1} = f(x_{k-1}, u_k, w_k)$$
$$P_{k|k-1} = A_k P_{k-1} A_k^T + W_k Q_k W_k^T$$

Where the first one predicts the mean of the new state pose vector and the second one predicts the covariance matrix of this random Gaussian vector.

In order to do that, a "predict" function inside the "EKF" class was defined. This function takes mainly a numpy array of shape(3,) ,which represents (δx, δy, δ θ), as an input and then transforms the odometry measurements from the turtlebot frame to the world frame by using the "comp" function so as to modify the mean value of the current state vector, which ws initialized in the class constructor to zeros. The next step is to define the Ak matrix, which describes how the state evolves without controls or noise, and Wk, which represents the odometry noise, in the following way:

$$\mathbf{A_k} = \frac{\mathbf{f}\left(\mathbf{x}^{\mathbf{B}}_{\mathbf{k-1}}, \mathbf{u}^{\mathbf{R_{k-1}}}_{\mathbf{k}}, \mathbf{w}_{\mathbf{k}}\right)}{\partial\left(\mathbf{x}^{\mathbf{B}}_{\mathbf{R_{k-1}}}\right)} = \begin{pmatrix} \frac{\partial f_x}{\partial x^B_{R_{k-1}}} & \frac{\partial f_x}{\partial y^B_{R_{k-1}}} & \frac{\partial f_x}{\partial \theta^B_{R_{k-1}}} \\ \frac{\partial f_y}{\partial x^B_{R_{k-1}}} & \frac{\partial f_y}{\partial y^B_{R_{k-1}}} & \frac{\partial f_y}{\partial \theta^B_{R_{k-1}}} \\ \frac{\partial f_\theta}{\partial x^B_{R_{k-1}}} & \frac{\partial f_\theta}{\partial y^B_{R_{k-1}}} & \frac{\partial f_\theta}{\partial \theta^B_{R_{k-1}}} \end{pmatrix} = \begin{pmatrix} 1 & 0 & -s\theta^B_{R_{k-1}} \cdot x^{R_{k-1}}_{R_k} - c\theta^B_{R_{k-1}} \cdot y^{R_{k-1}}_{R_k} \\ 0 & 1 & c\theta^B_{R_{k-1}} \cdot x^{R_{k-1}}_{R_k} - s\theta^B_{R_{k-1}} \cdot y^{R_{k-1}}_{R_k} \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{W_k} = \frac{\mathbf{f}\left(\mathbf{x}^{\mathbf{B}}_{\mathbf{k-1}}, \mathbf{u}^{\mathbf{R_{k-1}}}_{\mathbf{k}}, \mathbf{w}_{\mathbf{k}}\right)}{\partial\left(\mathbf{w}_{\mathbf{k}}\right)} = \begin{pmatrix} \frac{\partial f_x}{\partial w_x} & \frac{\partial f_x}{\partial w_y} & \frac{\partial f_x}{\partial w_\theta} \\ \frac{\partial f_y}{\partial w_x} & \frac{\partial f_y}{\partial w_y} & \frac{\partial f_y}{\partial w_\theta} \\ \frac{\partial f_\theta}{\partial w_x} & \frac{\partial f_\theta}{\partial w_y} & \frac{\partial f_\theta}{\partial w_\theta} \end{pmatrix} = \begin{pmatrix} c\theta^B_{R_{k-1}} & -s\theta^B_{R_{k-1}} & 0 \\ s\theta^B_{R_{k-1}} & c\theta^B_{R_{k-1}} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

where ""s" stands for sin() function and "c for cos() function. Those two matrices are used to modify the covariance matrix (P) of the current state Gaussian vector as follows:

$$P_k^- = A_k P_{k-1} A_k^T + Q_k$$

where Qk is defined as shown below:

$$Int = \begin{bmatrix} odom\_lin\_sigma^2 & 0 & 0 \\ 0 & odom\_lin\_sigma^2 & 0 \\ 0 & 0 & odom\_ang\_sigma^2 \end{bmatrix}$$

The following two images shows the results of only running the prediction step where the robot (blue arrow) moving around the room, and an ellipse representing uncertainty increasing its size when the robot moves.

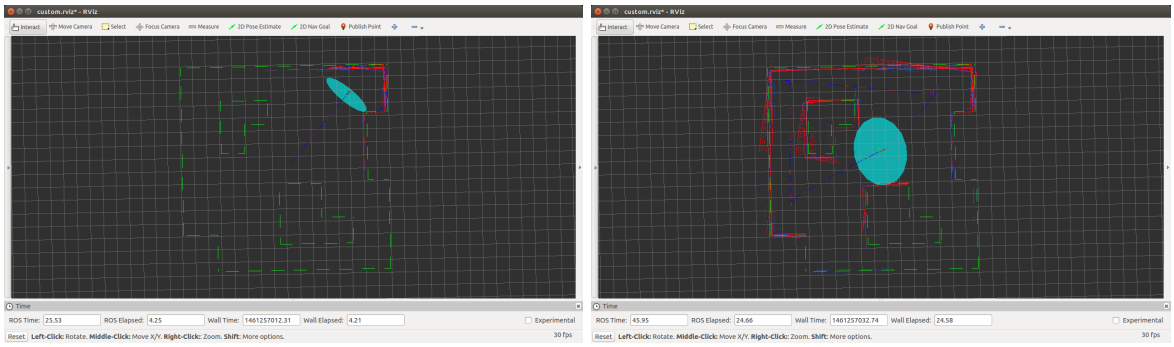

Figure 2: Result of running the prediction step only

3 Data association

In this section, the implementation of the data association part of the EKF filter is explained. The basic idea of this part is to match the observed lines by the turtlebot, which are obtained from the split and merge algorithm, with the already known map. A function called "data_association" was defined . This function works in the following way:

- It starts by iterating through each observed line in a "for" loop inside which it does the following:
  1. getting the polar coordinates of each observed line from its Cartesian coordinates.
  2. creating another "for" loop to iterate through each map line.
  3. inside this loop, it first obtains the polar coordinates of each map line with respect to the robot frame, calculates the Jacobian matrix, in a separate function with the same relation derived in the prelab, from the polar coordinates of each map line in the world frame, computes the innovation by subtracting the polar coordinates of the current map line, referenced with respect to the robot frame, from the current observed line, also in polar coordinates and in robot frame, computes the innovation uncertainty using the following equation:

$$S_{ij} = H_j P^B_{k|k-1} H_j^T + R$$

     where H is the Jacobian matrix and R is the covariance matrix for the observations:

$$R_k = \begin{bmatrix} 0.2 & 0 \\ 0 & \text{deg2rad}(10) \end{bmatrix}$$

     And finally, computes the Mahalanobis distance from the innovation (V) and the innovation uncertainty (S). Actually, the last step in this loop is to keep the previously calculated parameters (z, j, h, H, V, S, D) of the line with the smallest Mahalanobis distance.

- The next step is to check the minimum Mahalanobis distance whether it is smaller than chi_threshold or not. If yes, the line is considered associated and can be used in the update . In fact, running this step with the prediction one will not now change the previously shown results since the update step has not taken place.

4 Update

The update part of the EKF algorithm is discussed in this section. First, the function begins by Composing the list of matrices that are computed in the "data_association" function as single matrices in a loop. Those matrices are: H, V, S, R. Then, the following equations are used to update the position of

the robot and its uncertainty:

$$
\begin{aligned}
K_k &= P_{k|k-1} H_k^T S_k^{-1} \\
x_k &= x_{k|k-1} + K_k v_k \\
P_k &= (I - K_k H_k) P_{k|k-1} (I - K_k H_k)^T + K_k R_k K_k^T
\end{aligned}
$$

where all the required parameters for the preceding three equations have been calculated previously. The following images show the results of running the entire algorithm.

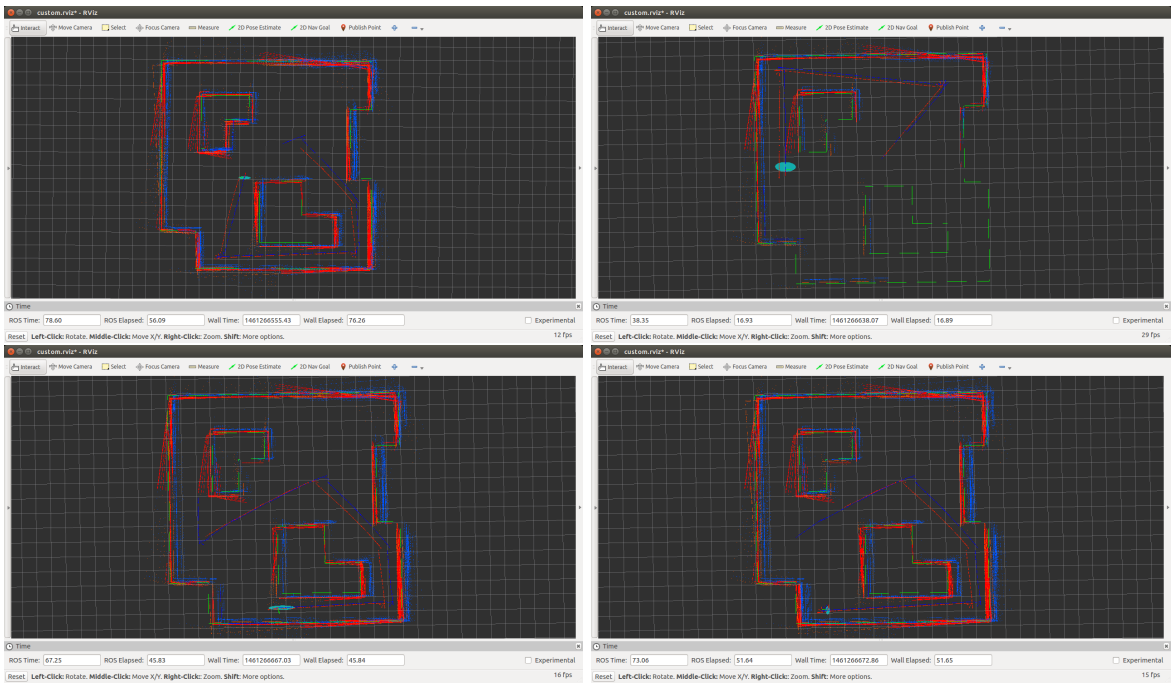

Figure 3: Results of running the entire algorithm (prediction, data association, update)

where the blue lines represent the EKF lines and the red ones are the raw data while the green lines are the original known map. As it is shown, the three lines are quite similar and lie near each other. Furthermore, the ellipse around the robot has got smaller after implementing the update step since we are incorporating data due to the observed lines and the association with the map lines which eventually decreases the uncertainty about the robot position.

5 Optional Part

The aim of this section is to improve some of the drawbacks of the implemented algorithm because of using polar representation. The problem that may arise is that walls are represented by lines without ends and not by segments which can introduce confusions in the data association. In order to compensate for this problem, an additional step was implemented in the data association function after checking the Mahalanobis distance. The aim of this step is to compare the length of the map line with the observed one to check if it is greater or not. If it is not, the parameters calculated with respect to this line are discarded, not used for the update. In this case, the effect of the ghost walls will be much less and the results are quite better.

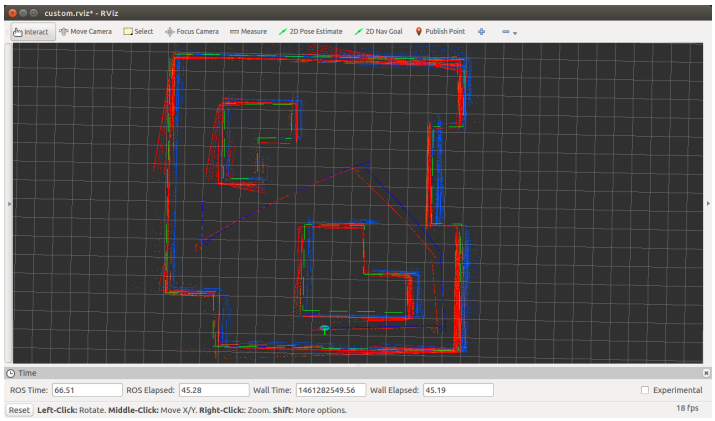

Figure 4: Results of running the entire algorithm including the optional part

6 Conclusion

In this lab assignment, a Map-Based Localization algorithm is designed, implemented and analyzed. Furthermore, it is tested on a given map with the aid of split and merge algorithm that extracts lines from the map based on a "bag" file which contains the measured data from the same map. The algorithm is further optimized in the optional part by taking into account the effect of the ghost walls.