Lab4 - EKF Map Based Localization

1 Introduction

The goal of this Lab exercise is to program an EKF Map Based Localization algorithm to work with data taken from a real Turtlebot.

Map based localization is a concept that in real life each one of us uses whenever we go to a new place. When a human being goes to a new environment it tries to know where he is. In order to do that, what he does is to read a map of the place and try to relate key parts of the environment he has seen with features of the map so he can localize himself in the map while he is moving around. This can be applied also to robots using an EKF map based localization algorithm:

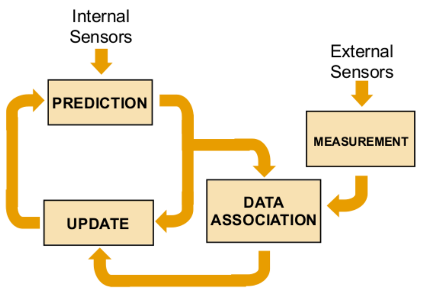

Figure 1: Schematic of EKF map based localization

The EKF map based localization is based on two different steps, prediction and update. The prediction is based on the internal odometry of the Turtlebot, while the update is based on lines sensed by its Kinect RGB-D sensor.

2 Pre-lab

Read and understand all the guide for this Lab and look about the EKF algorithm in your lecture slides. The pre-lab is the equations for the update step where you need to transform world referenced lines (ρ_w, φ_w) of the known map to robot referenced lines (ρ_r, φ_r) (look at Fig. 2).

Define the expected measurement equations $h(x_{k|k-1})$ and calculate the jacobian H of the expected measurement with respect to the robot state $x_{k|k-1}$.

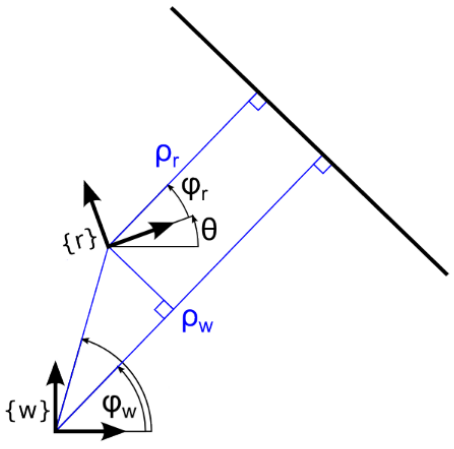

Figure 2: Schematic to deduce the measurement function $h(x_{k|k-1})$

You can write them in paper and attach a picture in the online form submission.

3 Lab work

The ROS package named `lab4_ekf` contains the skeleton code for running this lab. The input messages are `nav_msgs/Odometry` and `sensor_msgs/LaserScan`. You can test the code by running:

```
roslaunch lab4_ekf ekf.launch
```

As you can see in Rviz that the robot doesn't move at all. You will need to complete the code inside the `ekf_localization.py` file in order to complete properly the EKF Localization algorithm. No changes are needed in the `node.py`. Remember that you have useful functions in the library `functions.py`. If you think you need to modify any of these files ask your lab assistant first. The EKF algorithm goes as follows:

```
▶ 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
```

3.1 EKF prediction

You will need to complete the function `predict` inside `ekf_localization.py`. This part of the algorithm is where, from the odometry $\vec{u}_k$ and the previous position the new one is computed.

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

These measurements have an uncertainty associated. This uncertainty has been estimated to be 0.025 m as linear noise and 2 degrees as angular noise, so the Q_k matrix can be estimated for all the measurements as:

$$Q_k = \begin{pmatrix} 0.025 & 0 & 0 \\ 0 & 0.025 & 0 \\ 0 & 0 & \text{deg2rad}(2) \end{pmatrix} \quad (1)$$

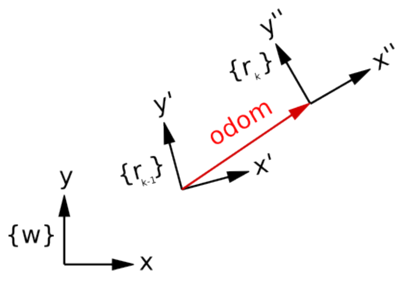

Figure 3: Schematic of the odometry measurement. Remember that robot position is defined in the world frame (W) while odometry is measured in the robot frame (R).

Once implemented correctly, when you test the script you should see the robot (blue arrow) moving around the room, and an ellipse representing uncertainty increasing its size when the robot moves.

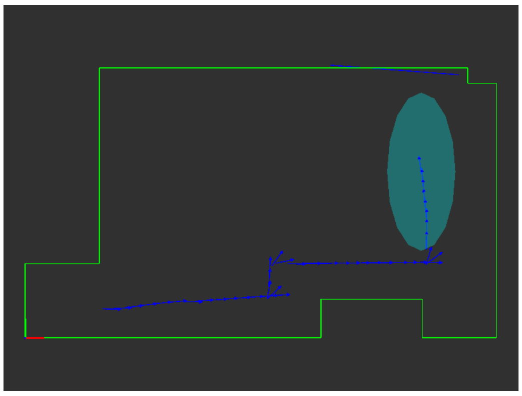

Figure 4: Prediction without updates.

3.2 Data association

Data association is the main parts in the algorithm. It tries to match the line measurements with the a priori known map of the environment. If data association is carried correctly the filter updates will be meaninfull and the robot will be correctly located.

To decide whether or not the distance between a feature and an observation is close enough both uncertainties have to be taken into account. Mahalanobis distance must be used.

Each measured line is compared to all lines in the known map. The smallest Mahalanobis distance is kept. If this distance is smaller than chi-square $\chi^2_{\rho,\varphi}$ test threshold, the line is considered associated and can be used in the update. You can choose this parameter.

▶ ***Data association***

$v_{ij} = z_i - h(x_{k|k-1})$

$S_{ij} = H_j P_{k|k-1} H_j^T + R_i$

$D_{ij}^2 = v_{ij}^T S_{ij}^{-1} v_{ij}$

Take the smallest D_{ij}^2 if $D_{ij}^2 < \chi^2_{\rho,\varphi}$

To obtain the expected measurement function $h(x_{k|k-1})$ that transforms a feature from the real map in polar coordinates in the world frame ${}^W x_f$ to a feature in polar coordinates in the robot frame ${}^R x_f$ you can use Fig. 5 for guidance.

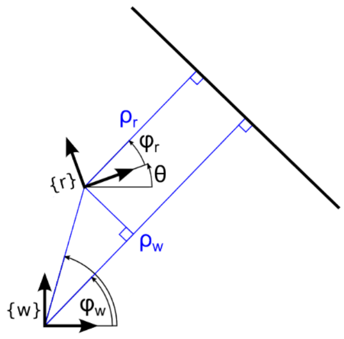

Figure 5: Schematic to deduce the measurement function $h(x_{k|k-1})$

Regarding R_k, the covariance matrix for the observations, the angle measurement noise has been estimated to 10 degrees and the range measurement to 0.2 m. R_k matrix is estimated as constant.

$$R_k = \begin{pmatrix} 0.2 & 0 \\ 0 & \text{deg2rad}(10) \end{pmatrix} \tag{2}$$

3.3 Update

In this step, the matches obtained on the data association are used to update the position of the robot and its uncertainty. The main algorithm is the following one. Starting by calculating the innovation v_k which is the difference between the observed features z_k and the expected observation $h(x_{k|k-1})$. Then the uncertainty of this innovation is calculated in S_k. Finally. the Kalman gain K_k is computed allowing to update the state vector x_k and the covariance matrix P_k.

```
▶ Update
$v_k = z_k - h(x_{k|k-1})$
$S_k = H_k P_{k|k-1} H_k^T + R_k$
$K_k = P_{k|k-1} H_k^T S_k^{-1}$
$x_k = x_{k|k-1} + K_k v_k$
$P_k = (I - K_k H_k) P_{k|k-1} (I - K_k H_k)^T + K_k R_k K_k^T$
```

Note that $h(x_{k|k-1})$ and H_k have already been computed for each feature in the data association. In this step the difference is that both can have more than one observation to update the robot position. In order to update with all the information at the same time $h(x_{k|k-1})$ and H_j computed for each observation associated to a feature are stacked vertically to obtain the final $h(x_{k|k-1})$ and H_k matrix and be able to run the update algorithm.

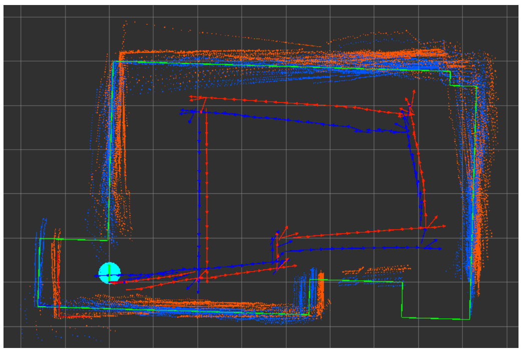

Figure 6: Comparison between raw data (red) with EKF (blue). Known map in green.

4 Optional

Using polar representation introduces a new challenge. Walls are represented by lines without ends and not by segments. These can introduce confusions in the data association. Try adding new constraints in the data association subsection to avoid matching the sensed lines with "ghost" walls.

5 Lab report

Write a brief report in less than 4 pages explaining your solution and problems faced. Include the final code and the report in a zip file.