Autonomous Systems

Localization using Extended Kalman Filter and a LRF

Group 15
Rúben Tadeia (75268) Manuel Moura (75756) Pedro Falcão (77063) Davi Mello (89126)
Instituto Superior Técnico

***Abstract*—The goal of this project is to allow a mobile-robot (Pioneer P3-DX) to localize himself conditioned to the use of Extended Kalman Filter and a Laser Range Finder. The Laser Range Finder will detect the distances to nearby objects (scan the environment) allowing us to acquire a map and then to estimate the localization of the robot pose in that map. It's known that the Kalman Filter (KF) is based on a method of linear Gaussian systems and can be used as a localization method in real systems. Although since some systems are not linear, we need to go a bit further and use the Extended Kalman Filter (EKF) that applies linearizations on non-linear equations. In short, the EKF used is an approximation to the original Kalman Filter. This method gave us the good results, meaning we had small error variation.**

***Index Terms*—Localization, Extended Kalman Filter, Laser Range Finder, Mapping, Mobile-Robot**

I. Introduction

The problem we had to deal with involved a recurring issue in everyday's life: the localization problem. Whether they are smart-phones, our computers or even mobile robots all of them have a method of self-localization, usually by triangulation (GPS). It is therefore a key topic today.

Our project consists of locating a mobile robot (Pioneer P3-DX). To accomplish this task we used the Extended Kalman Filter (EKF) with the help of a Laser Range Finder that scans the environment and allows us to acquire a map. The communication platform between the algorithm and the robot is the ROS, which is not only the basis implementation of all parts to a good functioning of the program, but also allows the integration of new packages developed by the user.

So as said before we have a global map for the robot. This map is uploaded as an jpeg image and with the help of the laser range finder we identify all of its boundaries, walls, into line segments, with the occupancy grids method, that divides the environment in cells. In other words, we match a position of the pixel of the image to a distance travelled by the robot. After, we use a method called Least Squares to match a line between the obtained points.

The Extended Kalman Filter provides optimal estimates for non-linear systems and is composed by 2 steps the Prediction Step and the Update Step. The EKF helps us find the main objective that is to correct the pose of the robot by subtracting the parameters of matching lines from local and global maps. We have a global map (the uploaded one) and the one that the robot "sees" called the local map, composed by information of the Odometry and the Laser.

The remainder of the paper is organized as follows. In Section II, we give a brief explanation of the methods used and ROS packages and algorithms used. Section III, divided in sections, describes how the implemented localization algorithm works **e.g.**, navigation, mapping and task coordination. In Section IV we show the most relevant results to illustrate the merits. Finally, we state our conclusions and lessons learned in Section V.

II. Methods and Algorithms

Obtaining the Global Map:
In order to obtain the parameters of the lines of global map we transformed the image of the map in a matrix by the pixel color and we created clusters of points for different lines that are on the map. The lines' equation for each line j is:

$$X_G * cos(\alpha_j) + Y_G * sin(\alpha_j) = \rho_j \tag{1}$$

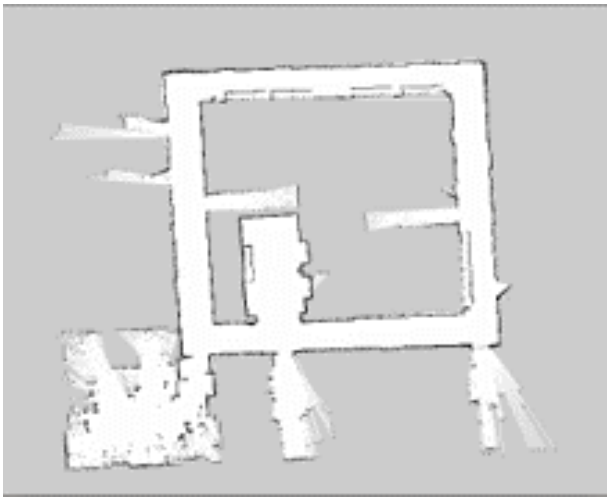

Fig. 1. Global Map

Obtaining the Local Map:
When the robot moves a new scan of Laser Range Finder is performed and the local map is constructed which consists in a set of line segments described by the equation for each line i is :

$$X_R * cos(\psi_i) + Y_R * sin(\psi_i) = r_i \tag{2}$$

We use Least Square Method to find this parameters.

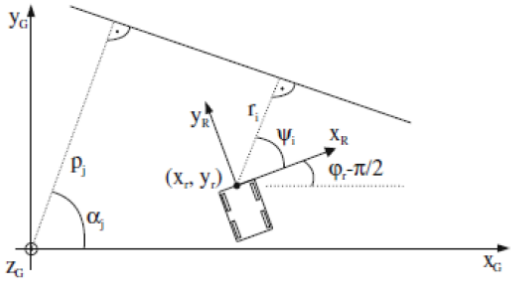

Fig. 2. Global and Local Map Coordinates.

Lines Segments Extraction:
For each cycle of data received from de LRF we get a set of distances d between the laser and the obstacle for a set of angles between -90º to 90º. The data collected by the LRF was first converted into Cartesian coordinates and stored in an array according to the equations:

$$x_i = d_i * sin(-\theta_i) \tag{3}$$

$$y_i = d_i * cos(\theta_i) \tag{4}$$

We want to cluster points so we need to analyze if the distance between two consecutive points is bellow a threshold and if the difference between the angle of those two points and the angle of the line formed by the cluster is bellow a threshold. If this two conditions are verified the point is clustered. If not, the point is excluded from the cluster and the cluster is finalized.

Least Square Method:
We use this method to get ψ and r. If a cluster belongs to a vertical line we will not be able to apply this method since the value of the slope will be infinite, so we calculate the approximated value of the scope using the first and the last points of the cluster. If the slope is bigger than 1 we need to apply a rotation of -90º to all points. In the end of the method we need to apply a rotation of 90º to ψ to obtain the correct line parameters. We use the following equations:

$$\hat{\theta} = [\hat{m}_l, \hat{b}_l]^T = (\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{y}, \quad \mathbf{U} = \begin{bmatrix} \mathbf{x}(1) & \cdots & \mathbf{x}(n) \\ 1 & \cdots & 1 \end{bmatrix}^T,$$
$$\mathbf{y} = [\mathbf{y}(1), \cdots, \mathbf{y}(n)]^T, \tag{5}$$

where n is the number of points of the cluster; $\hat{m}_l$ and $\hat{b}_l$ correspond to the line parameters of the line equation in the form $y = m_l x + b_l$. Now we obtain ψ and *r* with

$$r(\hat{m}_l, \hat{b}_l) = \frac{\hat{b}_l}{\sqrt{\hat{m}_l^2 + 1}} sign(\hat{b}_l), \tag{6}$$

$$\psi(\hat{m}_l) = \arctan 2 \left(\frac{sign(\hat{b}_l)}{\sqrt{\hat{m}_l^2 + 1}}, \frac{-\hat{m}_l}{\sqrt{\hat{m}_l^2 + 1}} sign(\hat{b}_l) \right), \tag{7}$$

Estimation of Line Parameters covariances:
For the update in EKF we need to compute the covariance matrix R_i that is composed by the variance of the line parameters, r_i and ψ_i, and the covariances between them.

$$\mathbf{Ce} = var(\mathbf{y}(j))(\mathbf{U}^T\mathbf{U})^{-1} = \begin{bmatrix} var(\hat{m}_l) & cov(\hat{m}_l, \hat{b}_l) \\ cov(\hat{m}_l, \hat{b}_l) & var(\hat{b}_l) \end{bmatrix}, \tag{8}$$

$$var(\mathbf{y}(j)) = \frac{\sum_{j=1}^{n} (\mathbf{y}(j) - \hat{\mathbf{y}}(j))^2}{n-1}, \hat{\mathbf{y}}(j) = \hat{m}_l \cdot \mathbf{x}(j) + \hat{b}_l \tag{9}$$

where $var(\mathbf{y}(i))$ is the vertical error variance of the line-segment points (**x**(j),**y**(j)) (*j*=1,...,*n*) according to the estimated line with the parameters $\hat{m}_l$ and $\hat{b}_l$. We can now calculate the variances and covariances between the parameters *r* and ψ as follows.

$$\begin{aligned} var(\psi) &= K_\psi^2 var(\hat{m}_l), \\ var(r) &= K_{rm}^2 var(\hat{m}_l) + K_{rb}^2 var(\hat{b}_l) + 2K_{rm}K_{rb} \cdot cov(\hat{m}_l, \hat{b}_l), \\ cov(r, \psi) &= K_{rm}K_{\psi m} var(\hat{m}_l) + K_{rb}K_{\psi m} \cdot cov(\hat{m}_l, \hat{b}_l), \\ cov(\psi, r) &= cov(r, \psi), \end{aligned} \tag{10}$$

where

$$K_{rm} = \frac{-\hat{b}_l \hat{m}_l}{\sqrt{\hat{m}_l^2+1}(\hat{m}_l^2+1)} sign(\hat{b}_l), \quad K_{rb} = \frac{sign(\hat{b}_l)}{\sqrt{\hat{m}_l^2+1}}, \quad K_{\psi m} = \frac{1}{\hat{b}_l^2+1}. \tag{11}$$

Local and Global Map Matching:

III. Implementation

State Definition:
Our state is defined by a vector with the position, the angle, the linear and the angular velocities. Because of that we need to take in consideration one issue The initial state is defined with the first values of our measurements and we put them in a measurements vector that we called X.

$$x = \begin{bmatrix} x \\ y \\ \Psi \\ v \\ \dot{\Psi} \end{bmatrix}$$

where:
x = Position X
y = Position Y
Ψ = Heading
v = Velocity
$\dot{\Psi}$ = Yaw Rate

Prediction Step (Time update):
The dynamic model represents the behaviour of the robot over time. The behaviour is described as a set of states that occur in sequence. We implement a dynamic function G with the equations below to calculate how the state is evolving from step to step.

$$G = \begin{bmatrix} x + \frac{v}{\dot{\Psi}}(-sin(\Psi) + sin(T\dot{\Psi} + \Psi)) \\ y + \frac{v}{\dot{\Psi}}(cos(\Psi) - cos(T\dot{\Psi} + \Psi)) \\ T\dot{\Psi} + \Psi \\ v \\ \dot{\Psi} \end{bmatrix}$$

The model is non-linear so we have to linearize it as the extended Kalman filter is only efficient to linear models. We implement the Jacobian of the dynamic matrix J_A regarding the state vector x by calculating the entries of the matrix and then inserting them into it.

The disturbances of the states are described in a process noise covariance matrix Q. Finally, we project the error covariance ahead P.

$$P_{k+1} = J_A * P_k * {J_A}^T + Q \tag{12}$$

Measurement Update (Correction):

So now we have our measurement function H with the laser measured values of the position, orientation, linear and angular velocities.

This function can be used to compute the predicted measurement from the predicted state. We linearize it by calculating the Jacobian of this matrix J_H in order to apply the filter.

The measurement noise covariance R entries are defined by attributing variances to the measurements of the laser range finder.

Then we are set to build our Kalman Gain from the measurement Jacobian matrix, the projected error covariance matrix P and the measurement noise covariance matrix R.

$$K_k = P_k {J_H}^T (J_H P_k {J_H}^T + R)^{-1} \tag{13}$$

where:

K_k = Kalman Gain
P_k = Projected error covariance matrix
J_H = Jacobian of the Measurement function
R = Measurement noise covariance matrix

We update the estimate of the state by summing the previous state to the Kalman gain times the residual (innovation) that is the difference between time measurements.

$$x_{k+1} = x_k + K_k(z_k - h(x_k)) \tag{14}$$

where:

x_{k+1} = Current state estimate
x_k = Previous state estimate
$z_k - h(x_k)$ = Residual

Finally, we update the error covariance which tells the Extended Kalman Filter how much to trust the measurements and the predictions.

$$P_k = (I - K_k J_H) P_k \tag{15}$$

We stored all the variables and plot them. As you can see in the next section.

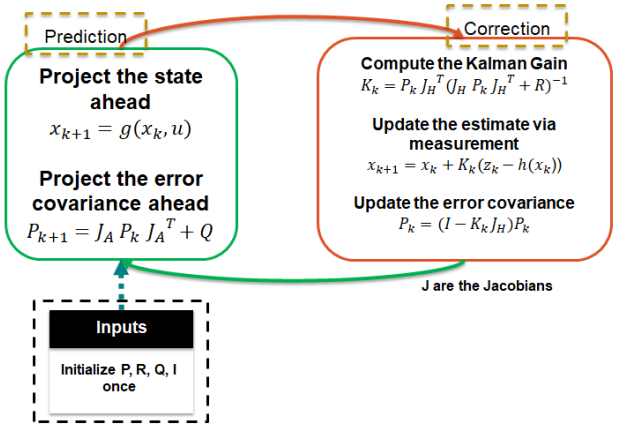

Fig. 3. Extended Kalman Filter diagram

IV. Experimental Results

In this section we present the results obtained. Firstly, once all modules have been functioning individually, meaning, that we obtained results as expected.

One thing to notice is that since our EKF is velocity based, we had to come up with a solution for a bug with the Laser, because it became sensitive the the angular velocities. These solutions are to first **ignore values from Laser** when we have quick variances in the distances. For instance, this can happen when an object or person crosses the robot's path. The other solution was to stop reading Laser data when the robot was **rotation or just changing the orientation**.

For the kidnapping solution, we used the Global and local mapping matching method and run it in it parallel. Then we obtained a prediction of $> K\%$, comparing it with the EKF, we get an error of $> \ epsilon\%$. Now we just have to restart the EKF, with the starting position given by the Global and local mapping matching method.

Solving the problem of kidnapping by re-calibrating the position of the robot over time.

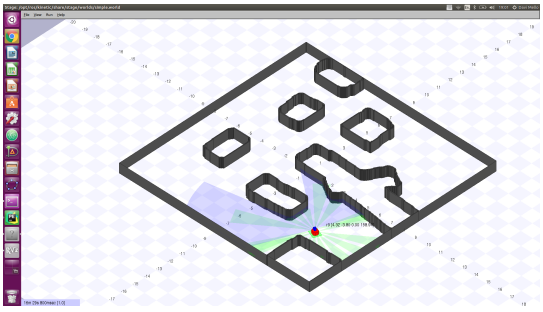

Fig. 4. Testing EKF and LRF in simulation mode

So as we can see from the image above, Figure 4, we run a simulation using the tele-operation method and observed the result in Figure 5. In which the red arrows are the robot orientation, the red dots are laser data readings and the green dots are EKF predictions.

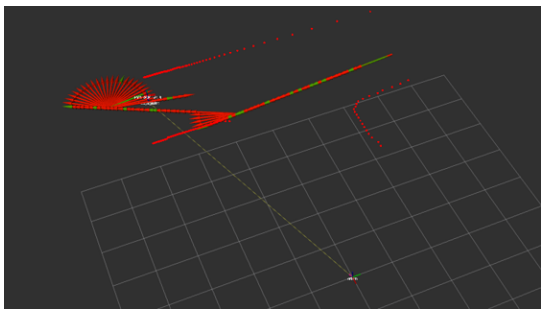

Fig. 5. Testing the EKF in a simulation environment

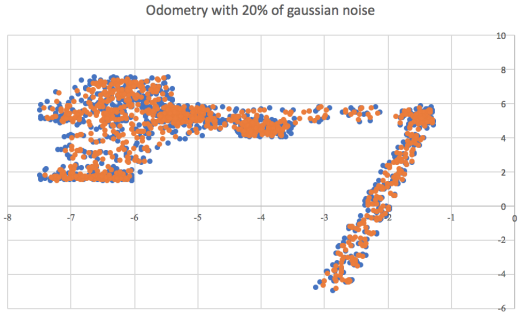

Fig. 6. Result with an Odometry with 20% of random noise

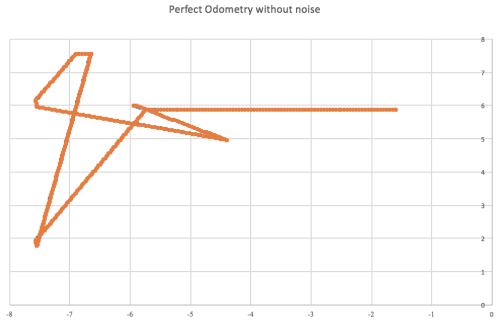

Fig. 7. Result with an Odometry without noise

Considering the images above, one thing to have in mind is that the frequency is 10Hz and generated 600 Points, meaning we run it for this image for 1 minute, although the code can run for undetermined time. The first image is Figure 6 which shows values from Odometry with 20% random noise,and Figure 7, without random noise.

We results are pretty clear. Without noise the results are very good, and the Extended Kalman Filter values and the Odometry ones are overlapped. Considering the 20% noise Odometry results, we can see that the algorithm is working acceptably well, although with some discrepancies on some points.

Finally the Figure 8 proves our result on EKF of error in the order of 10^{-3}. The way we plot this was with some points (fifty) from the perfect Odometry (without noise) and subtracted the values from the predictions resulting from EKF. And the output can be observed.

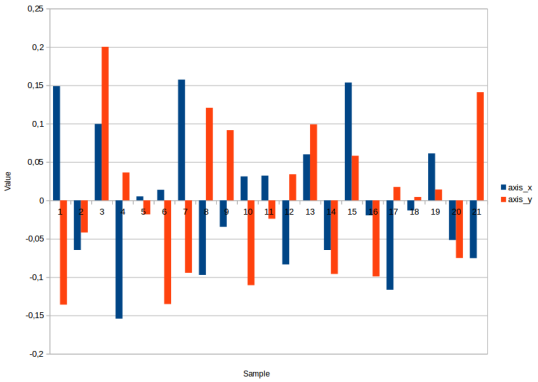

Fig. 8. Prove of EKF working (Subtraction of points)

V. Conclusions

In Autonomous Systems, for a mobile robot to function properly, we need to have proper readings and decide upon those to find the best possible action. So it's intended that these readings are as accurate as possible. Here the considered readings were the Laser Range Finder and Odometry ones. For that reason the EKF algorithm described above, is implemented in order to minimize the errors, for the sake of a more accurate robot localization. Both the EKF and the Laser data are more accurate than the result from Odometry. That's one of the reasons why the EKF is a "complement" widely used in autonomous systems. We note that it's important to do the readings as soon as possible so that the whole process can be fluid, in search of the robot determination of its location in real time. With the work developed, it was possible to understand that it is necessary to minimize all possible errors, not only because the accuracy of readings is a very crucial factor for good operation but to avoid the propagation of errors that can prove to be disastrous too.

In short, since the EKF uses a linearized model, the first thing to do is linearize the kinematic model to perform the prediction step. Then we perform the matching step extracting the parameters from the map of the 5^{th} floor the North Tower of Instituto Superior Técnico. From there we had to compare the results from the Laser Range Finder. Finally we go into the update step, that updates robot's position estimation. It's interesting to note that when using the Harris detection method, in order to achieve the maximum similarity between lines, the extracted points of each line were the first and last ones. This will be later the solution for the kidnapping problem. In this report, some results were not present *e.g.*, the values obtained for the covariance matrices and better comparisons between the errors obtained with EKF algorithm.

Nonetheless these results as well as the matching step from the EKF (consisting in a comparission with the lines between the local and the global map), and a self-explanatory video will be presented in the poster section. Some reasons for things that could had gone better were the group's lack of experience regarding ROS and Python, as a result the group spent more hours learning this basic tools than with the algorithm itself.

REFERENCES

[1] Pedro Lima, *Notas das Aulas Teóricas*, Lisboa - Instituto Superior Técnico, 2017

[2] Python Implementations *of the Kalman Filter*, https://balzer82.github.io/Kalman/