Theoretical project 1

Theoretical project 1, Mobile Robotics

José António Filipe Santos, 98279

Departamento de Electrónica, Telecomunicações e Informática
Universidade de Aveiro
Portugal
May, 2023

Resumo Creation of a Extended Kalman Filter for application in a mobile robot in Matlab.

Keywords: Mobile Robotcis · Extended Kalman Filter

1 Introduction

This work is made with the objective of a virtual robot in an environment with multiple landmarks is able to localize itself using an Extended Kalman Filter. The robot is able to locate itslef in this environment thanks to the BeaconDetection() function. The other important function is the EKF or Exetended Kalman Filter, with was modified to meet the requirements of the teacher.

2 functions

In this part of the report i will explain the functions that were created and a brief explanation of how they work.

Plotting_path

This function has two objectives, one obtainin the steps needed to reach the goal, passing through the landmarks and the other is to use the function in Matlab named **pchip()** to obtain the smooth path that the robot will take.

For the calculation of steps, this is based on the distance between beacons, or between the initial position and the first beacon.

The equation $\left(\frac{\sqrt{((x2-x1)^2+(y2-y1)^2}}{\mathrm{Dt}*\mathrm{Vn}}\right)$ is used to obtain the number of steps taken between beacons, where the time step and velocity. With the number of steps aquired, we can obtain the total number of points, each with x and y variables.
After obtaining the total x and y values evenly spaced between the beacons, we use the **unique()** function to obtain the unique x values, and with these interpolated values or "xintep"values, we use the pchip function to obtain the interpolated y values,these being plotted with all the beacons and evenly spaced points. The results of this function the figure 1, is showed on the right.

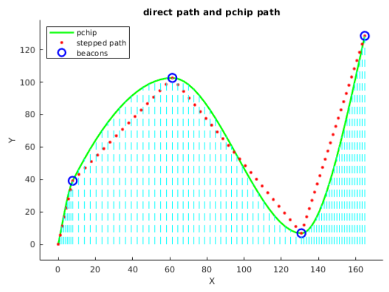

Figura 1. Plotting_path function results

Plotting_velocities

This function uses the **pchip()** interpolated coordinates to obtain the velocities of the robot. To calculate the linear and angular velocities we will use to generate the values for the Extended Kalman Filter, we use the following equations:
$distance = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$
$\theta = \sin(y2 - y1)/distance$
The velocity noises for the velocity are calculated by multiplying noise variable with a random sample from a standard normal distribution.

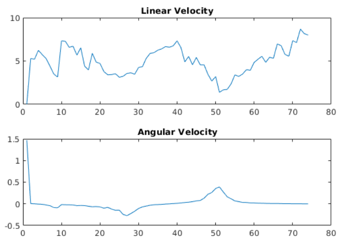

Figura 2. Plotting_path function results

The caltulations for the linear and angular velocities are:

$$\mathrm{Vn} = \frac{distance*(1+noise)}{\mathrm{Dt}} \quad ; \quad \mathrm{Vt} = \frac{\theta*(1+noise)}{\mathrm{Dt}}.$$

Generate_ekf_data

This function is used to generate the main data that will be used on the Extended Kalman Filter. Most of this variables are used to assist the prediction, such as **control_imput_mea** and **obs_range_bearing**, while others are used to verify the accuracy of the Extended Kalman Filter,for example **xstate_-true,** and **control_imput_true**. In this function other variables were made, mostly for ease of creating other functions, these being **landmarkxy,** and **obs_-landmark_ID**.

ekf_calculations

This function is used to prepare the values used in the extended Kalman Filter. Due to most of the variables having information about all states, its needed to obtain the true and estimated position, control input and the range and bearing of the beacons at that time.
After this, the function **ekf** will calculate the estimated position of the robot and with this we can see the estiamtions of the robot, by comparing it to the true values of the robot, these being in the variable **xstate_true**.
The image 3 ,listed below shows the result of the **ekf_calculations** function, showing in red the true values and in blue the estimated values calculated by the Extended Kalman Filter.

From this image we can see that the estimated values are very close to the true values, however the estimations orientation when passing the beacons is not as accurate as the positions further away from the beacons. To be able to see more clearly the errors, the table 1 was made to be able to see the average,largest, smallest errors, and the standard deviation of the errors.

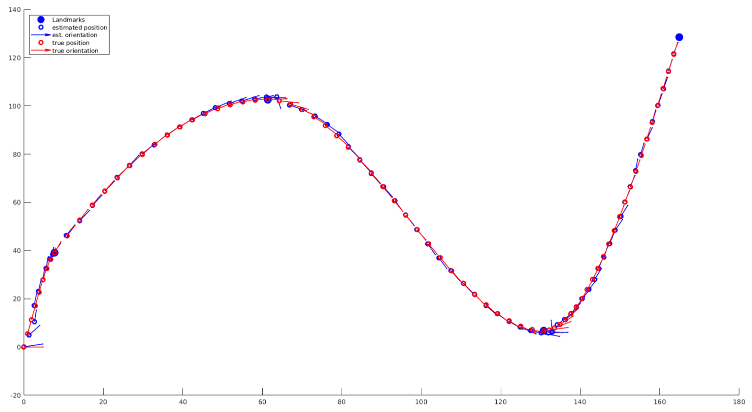

Figura 3. ekf_calculations function results

Tabela 1. Errors for x, y, and theta values

	Average error	Smallest error	Maximum error	deviation
x	0.200152	0.000000	0.754331	0.159424
y	0.274439	0.000000	1.852798	0.331390
θ	0.079925	0.000000	1.160790	0.175788

EKF

This function is used to calculate a estimated position and orientation of the robot.

The first step is calculating the initial estimation of the robot. This is made by using the function **motionmodel(xstate,Vin,Dt,t)**

$x_{\text{state}} = x_{\text{state}} + (V_{\text{in}}(1) + D_n(1))t\cos(theta_{\text{state}});$

$y_{\text{state}} = y_{\text{state}} + (V_{\text{in}}(1) + D_n(1))t\sin(theta_{\text{state}});$

$theta_{\text{state}} = theta_{\text{state}} + (V_{\text{in}}(2) + D_n(2))t$

This function estimates the x,y and orientation on the step being estimated on the EKF.

The next step is to calculate the uncertainty of the robot, this is done by calculating the matrix of the predicted covariance,or uncertainty, and noise. The uncertainty Matrix is calculated by the following equations:

$$\text{Jfx} = \begin{bmatrix} 1 & 0 & -\delta_t \cdot \text{controlinput} \cdot \sin(\text{est}_\theta) \\ 0 & 1 & -\delta_t \cdot \text{controlinput} \cdot \cos(\text{est}_\theta) \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{Jfw} = \begin{bmatrix} \delta_t \cdot \text{controlinput} \cdot \cos(\text{est}_\theta) & 0 \\ \delta_t \cdot \text{controlinput} \cdot \sin(\text{est}_\theta) & 0 \\ 0 & \delta_t \end{bmatrix}$$

$$\text{uncertainty} = \text{Jfx} \cdot P_t \cdot \text{Jfx'} + \text{Jfw} \cdot Q \cdot \text{Jfw'} \tag{1}$$

The next step is to use the **sensormodel(xym,xstate1,ndnphi)**function to get a predicted range and bearing with noise from the position(xstate1) of the robot.
$bearing = \sqrt{(x - xstate)^2 + (y - ystate)^2} + \text{ndnphi}$
$angle = \arctan 2(y - ystate, x - xstate) - \theta + \text{ndnphi}$
With the predicted range and bearing we can subtract both of them and obtain the error,this being the variable **innov** that will be used for the calculations later.

Another calculation step is to obtain the Jacobian of the sensor model, this is done by follwoing these equations:

$$a = \frac{x - xym(1)}{\sqrt{(x - xym(1))^2 + (y - xym(2))^2}}, \quad b = \frac{y - xym(2)}{\sqrt{(x - xym(1))^2 + (y - xym(2))^2}}, \quad c = 0; \tag{2}$$

$$d = -\frac{(y - xym(2))}{\left(\frac{(y-xym(2))^2}{(x-xym(1))^2} + 1\right)(x - xym(1))^2}, \quad e = \frac{1}{\left(\frac{(y-xym(2))^2}{(x-xym(1))^2} + 1\right)(x - xym(1))}, \quad f = -1; \tag{3}$$

$$J = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \tag{4}$$

The final jacobian matrix will be used in the next step, this being the Kalman Gain.

One of the last steps of the Extended Kalman Filter is to calculate the Kalman Gain and the new estimated position and orientation of the robot. In the equations below, we calculate **S**, the measurement error covariance and **K**, the Kalman Gain.

$$S = J \cdot P_t \cdot J' + R \quad ; \quad K = P_t \cdot J' \cdot S^{-1}$$

The final step of the EKF is to calculate the new estimated position and orientation of the robot, this is done by the following equations:

$$x_{t+1} = x_t + K \cdot (innov) \quad ; \quad P_{t+1} = (I - K \cdot J) \cdot P_t$$

plotting_robot_atributes

In this function, we calculate and plot the values requested by the teacher, regarding the virtual use of the Direct Drive robot and the Tricicle robot. The results of these equations are below, where we can see the slight variations between the left and right Direct Drive robot, and the similarities between those and the Tricicle robots' rear velocity.

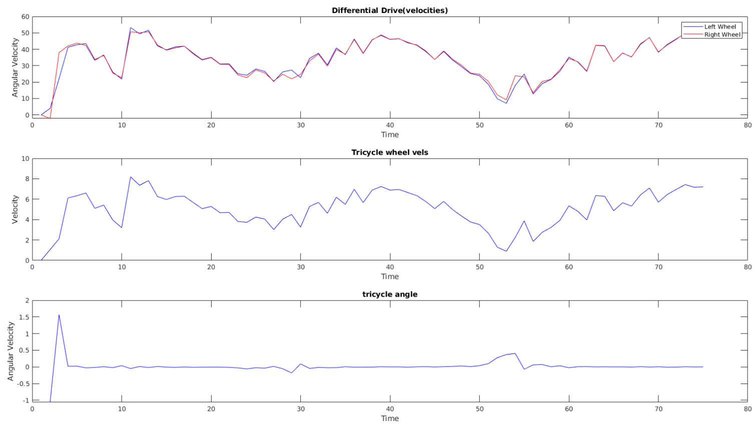

Figura 4. Plotting_path function results

Direct Drive Robot The calculations for the Direct Drive velocities are based on the equations of the classes. The equations are the following:

$$\mathrm{DD}_{left} = \frac{v - w\frac{L}{2}}{r}; \quad \mathrm{DD}_{right} = \frac{v + w\frac{L}{2}}{r}$$

Tricicle Robot The calculations for the Tricicle velocities are based on the equations of the classes. These were the equations used:

$$\mathrm{TRI}_{velocity} = v - w \cdot L; \quad TRI_{\alpha} = \arcsin(\frac{w \cdot L}{\mathrm{TRI}_{velocity}})$$

3 conclusion

In conclusion, this project was a great way to understand the concepts of the Extended Kalman Filter and how to use it in a practical way. EKF is a very good system that predicts the position of a robot, but it is not perfect, as we can see in the results, the estimated path is not the same as the real path, this is because of the noise of the sensors and the noise of the robot's movement, which is not taken into account.
This is a very good way to estimate the position of the robot, but it is not perfect as it has it's limitations. The results of the Direct Drive and the Tricicle robots are very similar because the equations used to calculate the velocities are very similar being the only difference it's length of the Tricicle robot as well as the third wheel