Activity 3

Sensor characterization and line fitting

1. Connect to REALabs via RESTTHRU from MATLAB and your web browser

In order to do so be able to access to the real robots from MATLAB, follow the instructions in the document **doc/Tutorial Matlab + Restthru.pdf.**

Now you can use the real robot if you are in the local network (i.e. in the laboratory connected to the local network REALabs, connecting locally to the robots (If you are, ask for some help).

However, for this activity, you do not need to be connecting your computer to the local network, and should use the REALabs platform of remote access to the robots/ sensors of the laboratory. You will need to reserve first a slot to be able to access the laboratory hardware following the instructions of **doc/ REALabs How to access.pdf**

2. Establish the characteristics of laser sensors robots Pioneer P3-DX.

Tasks: Initially, based on technical data of the laser Sick LMS-100, raise the possible characteristics of the sensor based on the manufacturer's data (see **doc/lms100 specs.pdf**).

Then, connect to the experiment called **Percepção**, via the REALabs platform. With the robot stopped, get a number N of measurements (e.g. 100 measurements) of distances with the laser in a certain directions.

Task: Edit **code/common/vrep/GetRealLaserDistances.m** and add your MATALB code.

You can use measures -100: 100: 20 (measures from -100° to 100° every 20°), which results in 11 sets of measures. With these measures you should compute the average and standard deviation, and with this, the accuracy of the sensors.

To get laser reading within the suggested range edit the global variable optionStr like that:
optionStr= '?range=-100:100:20';

To get one reading of distances use
dist = Pioneer_p3dx_getLaserData(connection,'distances');

Additionally, using the Shapiro-Wilk test (**swtest.zip**), verify that the N measurements in each direction correspond to a normal distribution for different significance levels (0.05, 0.1 ...).

Regardless of the test, plot the N measurements in a given direction, and see if this approaches a normal distribution.

3. Line fitting and extraction for robot localization

3.1. Introduction

For a lot of application in robotics, knowledge of the position and orientation of the platform is essential. This activity could be motivated by an autonomous vehicle hauling goods across the corridors of a warehouse. In order to navigate from one place to another, the vehicle would need to know its position in the warehouse as well as its heading. On its way, it might come across walls, doorways, and racks, all of which would be perceived as measurements located along lines by a laser scanner mounted in a way that its scanning plane is parallel to the ground.

Activity 3 and 4 explore these line features for localization against a known map. While Activity 3 will show how to extract lines from laser scans using the split-and-merge approach, Activity 4 will demonstrate how to employ these measurements in combination with a map for robot localization in a Kalman filter framework.

3.2. Line extraction

A range scan describes a 2D slice of the environment. Points in a range scan are specified in a polar coordinate system with the origin at the location of the sensor. It is common in literature to assume that the noise on measurements

follows a Gaussian distribution with zero mean, some range variance and negligible angular uncertainty.

We choose to express a line in polar parameters (r, α) as defined by the line equation (1) for the Cartesian coordinates (x, y) of the points lying on the line

$$x \cos \alpha + y \sin \alpha = r \quad (1)$$

where $-\pi < \alpha \leq \pi$ is the angle between the x-axis and the shortest connection between the origin and the line. This connection's length is $r \geq 0$ (see Figure 1).

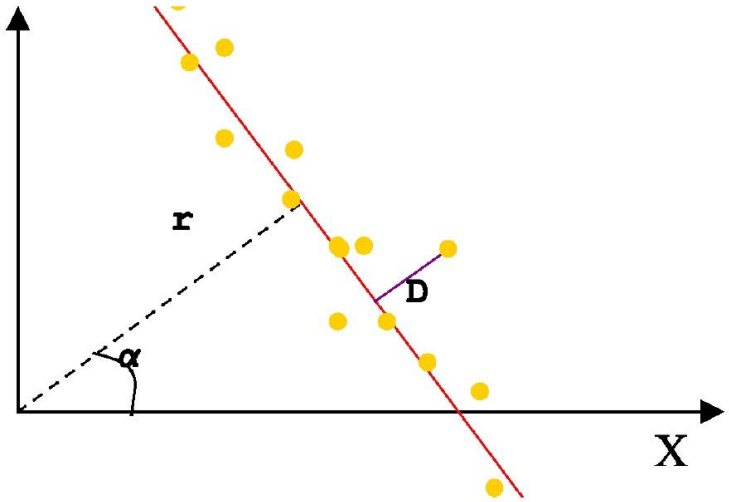

Fig. 1: Fitting line parameters: D is the fitting error we aim to minimize expressing a line with polar parameter (r, α)

3.3. Split-and-Merge

We employ the popular "Split-and-Merge" [1, p.249-250] line extraction algorithm to divide the obtained range measurements (points) into segments of points lying roughly on a common line.

Algorithm 1: Split-and-Merge

Data: Set S consisting of all N points, a distance threshold $d > 0$
Result: L, a list of sets of points each resembling a line
$L \leftarrow (S), i \leftarrow 1;$
while $i \leq len(L)$ **do**
  fit a line (r, α) to the set L_i;
  detect the point $P \in L_i$ with the maximum distance D to the line (r, α);
  if $D < d$ **then**
    $i \leftarrow i + 1$
  else
    split L_i at P into S_1 and S_2;
    $L_i \leftarrow S_1; L_{i+1} \leftarrow S_2;$
  end
end
Merge collinear sets in L;

3.4. Algorithm in Matlab/Octave

The Split-and-Merge algorithm is implemented inside the function $[a^i, r^i, \ldots] = extractLines(x^i, y^i)$. A crucial part of this function is the line fitting step.

Task: Edit **code/Ex3_LineExtraction/fitLine.m** and follow the instructions to complete the mathematical formula for computing line regression (line fitting) using a set of points in Cartesian coordinates after reading the following theory. The aim of the function is to minimize the mean squared error of:

$$S(r, \alpha) := \sum_i (\underbrace{r - x^i \cos\alpha - y^i \sin\alpha}_{=D((\alpha,r),(x^i,y^i))})^2 \tag{2}$$

where (x^i, y^i) are the input points in Cartesian coordinates. The solution of (r, α) can be found imposing: grad$S = 0$. The solution for α is then

$$\alpha = \frac{\operatorname{atan2}(\text{nom}, \text{denom})}{2} \tag{3}$$

$$\text{nom} := -2 \sum_i (x^i - x_c)(y^i - y_c) \tag{4}$$

$$\text{denom} := \sum (y^i - y_c)^2 - (x^i - x_c)^2 \tag{5}$$

where (x_c, y_c) are the Cartesian coordinate of the (x^i, y^i)'s centroid (see instructions in **code/Ex3_LineExtraction/fitLine.m**).
In order to solve for *r*, consider the equation (1) and a point that will surely lie on the line (which one is it?). Please find additional information on [1, pp. 244] including a solution for polar input on [1, p. 246]. As soon as the lines are correctly fitted, the algorithm performs Split-and-Merge and extracts the endpoints of each segment.

Validation: Run **code/Ex3_LineExtraction/test/testLineExtraction.m** to check if the code is correctly completed. If not it will show a figure with the measured points and expected lines (in yellow) as together with the found lines and segments (in red, green and blue). To test only the fitLine function on artifical data use **code/Ex3_LineExtraction/test/testLineFitting.m**.

References

[1] Roland Siegwart, Illah Nourbakhsh, and Davide Scaramuzza. "*Introduction to Autonomous Mobile Robots*." MIT Press, 2nd edition, 2011.