Lab2 - Line features with Split & Merge

MENG Di

March 2017

1 Introduction

The objective of this lab is to implement a way to extract line features from the point data. Basically, the robot will sensor a set of 2d points when moving around to perceive the environment. Line features could be extracted from messy points for better building the map of the environment and also reducing some noise. In this lab assignment, we mainly use the method of Split and Merge.

2 Split and Merge

2.1 Split

A set of points are given in the form of Cartesian coordinate $\big(\mathrm{x},\ \mathrm{y}\big)$ in 2d plane. The idea of this method is that we assume there's one straight line started at the first point of the data set and ended at the last point . The distances are computed between each point in the dataset to the line. The maximum distance is compared with a threshold. If the maximum distance is larger than the threshold, that means the points are not following the linear least square approximation. In other words, this group of points are not converged to one line. Under this circumstance, the line we assumed before should be split into two lines of which the first line starts from the first point and ends at the point having the maximum distance with the assumed line, and the second line starts from the end of the first line and ends at the last point of dataset.

The above process is for splitting the assumed lines that do not include all the points information. In other words, it is mainly for building the lines from the points. Of course we can not consider all the points in the dataset because there are many noises. The way of filtering the noise is to set a minimum number of points to build up one line so that some sparse points will be ignored.

Then based on these extracted lines, the next step is to split these lines which are not qualified. The interval distances between each point in the line are computed and compared with a threshold. If the interval distance between

two points is greater than the threshold, that means this pair of points are belonging to two different lines and should not be connected. Then this line would be split between this two points.

2.2 Merge

After the process of splitting the lines, we get a stack of lines. There is a possibility that some of this lines are belonging to one single line. So, the section of merging is to check if two lines could be connected and considered as one line and merge them if could.

There are two conditions for checking if two lines can merge. One is the angle difference between this two lines is smaller than a threshold. The other one is that the distance difference is smaller than a specific threshold.

As we know, the lines are saved in order. The angle difference is computed between every line and its next line in the container. The distance difference is computed between every line's last points and its next line's first point. If both of these differences are in the tolerance, two lines are merged.

3 Result

3.1 Testing on dataset3

Figure 1: Extracted lines and original points perceived

Figure 1 shows the original points that robot perceived(in white) and the extracted lines from the algorithm(in red). The lines appears in sequence and

finally are shown as above. As we can see, the red lines mostly fit the dataset and the noise is not considered.

Figure 2: Extracted lines only

3.2 Testing on dataset2

Another dataset is also tested by this package with the same parameter values. The results are shown as follows.

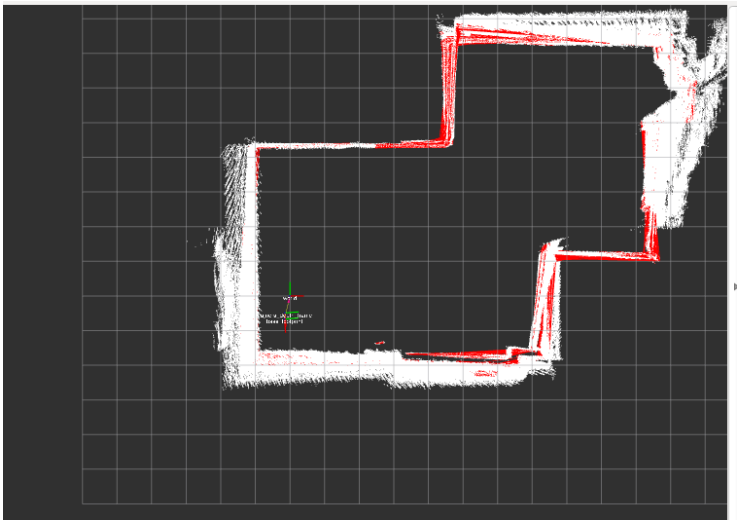

Figure 3: Extracted lines and original points perceived

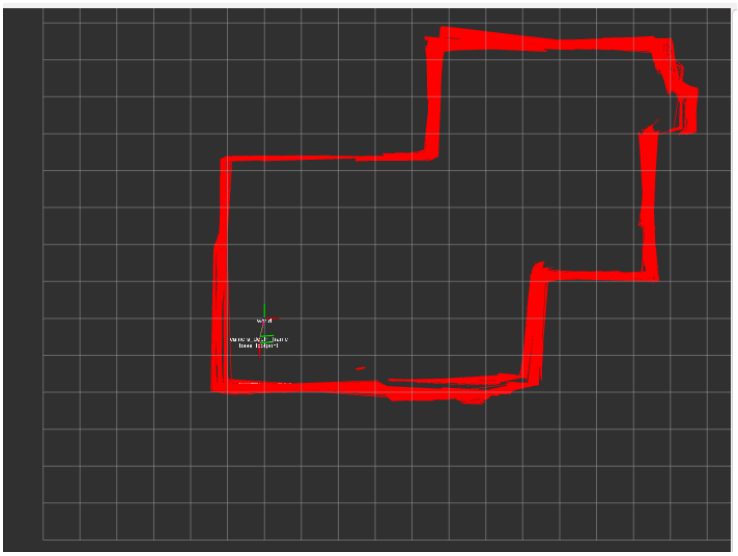

Figure 4: Extracted lines only

4 Additional information

The code(changes in splitandmergenode.py) for representing all the lines permanently is shown as follows:

Inside the constructor "def __init__(self):", initialize the variable "self.counter = 0".

Then inside the function "def laser_callback(self, msg):", replace the function publish_lines as following codes:

```
self.counter += 1
new_scan = 'scan_line' + str(self.counter)

publish_lines(lines, self.pub_line, frame=msg.header.frame_id,
              time=msg.header.stamp, ns=new_scan, color=(1,0,0))
```

Figure 5: Changes in node.py