ME 580 Homework 1: Line Fitting

Ahmed Elsaharti
Mechanical Engineering Department
University of North Dakota
ND, United States of America
ahmed.elsaharti@und.edu

***Abstract*—This report demonstrates the development of a line extraction algorithm based on the split and merge algorithm and the line fitting algorithm provided in [1]. In addition, results are demonstrated showing the performance of the developed algorithm on multiple data sets with multiple error thresholds. Finally, it briefly shows the implementation of the developed algorithm onto images to extract lines.**

I. Introduction

In order for robots to perform tasks, their location with respect to the surrounding environment needs to be known as the robot moves around [2][3]. Almost all robots perform this task using sensors that take readings of the surrounding environment.

To perform the localization task more efficiently and quickly, robots use SLAM systems that depend on matching features/lines detected by sensors. Since sensor data is never perfect, line extraction and fitting techniques need to be utilized to extract useful data from the readings.

This report will be discussing and demonstrating a line extraction and fitting algorithm based on the split and merge algorithm [4]. The report starts with a review of currently used line extraction algorithms in Section II. Followed by an explanation of the split and merge algorithm developed and its implementation in Sections III, IV, V and VI. Lastly, the report concludes by showing different results of the algorithm and an overall conclusion in Sections VII, VIII and IX.

II. Line extraction algorithms

There are numerous line extraction algorithms that are proven to be useful depending on the application. The authors in [2] discuss in detail multiple algorithms that exist and would perform well in our application such as:

1. Split and merge
2. Incremental
3. Hough-Transform
4. Line-Regression
5. RANSAC
6. EM

The authors also compare the performance of each of those similar to what is done in this report but on a larger scale.

III. Split and merge algorithm

The split and merge algorithm works as follows:

Algorithm 1: *Split-and-Merge*

1 Initial: set s_1 consists of N points. Put s_1 in a list $\mathcal{L}$
2 Fit a line to the next set s_i in $\mathcal{L}$
3 Detect point P with maximum distance d_P to the line
4 If d_P is less than a threshold, continue (go to 2)
5 Otherwise, split s_i at P into s_{i1} and s_{i2}, replace s_i in $\mathcal{L}$ by s_{i1} and s_{i2}, continue (go to 2)
6 When all sets (segments) in $\mathcal{L}$ have been checked, merge collinear segments.

This algorithm was implemented on MATLAB as shown in the m-file attached with this submission. The script starts by importing the dataset, removing the zero data and removing outliers using a manually written algorithm.

The outlier detection section works by measuring the distance between a point and its successor and predecessor. If the distances between the point AND both the previous and the next point are high, the point is counted as an outlier and is removed.

The following sections will cover the main steps of the algorithm.

IV. Line Fitting

The line fitting algorithm used is taken from the m-file provided with the assignment which utilizes the line fitting equations in [1]:

$$\alpha = \frac{1}{2}\operatorname{atan}\left(\frac{\sum w_i\rho_i^2\sin 2\theta_i - \frac{2}{\sum w_i}\sum\sum w_i w_j \rho_i \rho_j \cos\theta_i \sin\theta_j}{\sum w_i\rho_i^2\cos 2\theta_i - \frac{1}{\sum w_i}\sum\sum w_i w_j \rho_i \rho_j \cos(\theta_i + \theta_j)}\right).$$

$$r = \frac{\sum w_i \rho_i \cos(\theta_i - \alpha)}{\sum w_i}.$$

where α is the angle of a line perpendicular to the fit line and r is the radius of the perpendicular line.

Fig 1 shows how the line fit would look like for the sample dataset in [1] shown in Table 1.

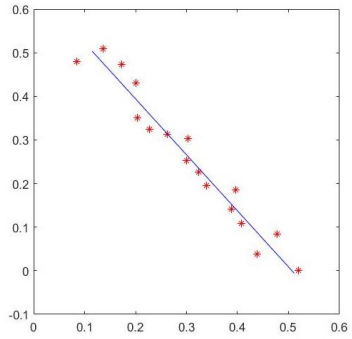

Fig. 1. Demonstration of the line fitting algorithm provided in [1]

pointing angle of sensor θ_i [deg]	range ρ_i [m]
0	0.5197
5	0.4404
10	0.4850
15	0.4222
20	0.4132
25	0.4371
30	0.3912
35	0.3949
40	0.3919
45	0.4276
50	0.4075
55	0.3956
60	0.4053
65	0.4752
70	0.5032
75	0.5273
80	0.4879

Table 1. Dataset from [1]

V. Set Segmentation

The MATLAB code segments the main set of points into subsets as described in Algorithm 1. Fig. 2 shows the points of splitting (green circles). This is where the main set was split into subsets.

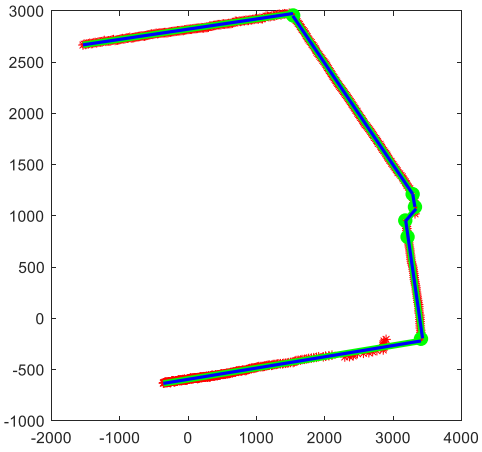

Fig. 2. Split and merge algorithm showin the points of splitting

VI. Set Merging

In this implementation of the algorithm the subsets were said to be similar if the α and r of 2 consecutive subsets were within a threshold. This means that after the splitting algorithm is complete, a line is fit to all subsets and the merging check was to be done (check m-file for annotations).

VII. Results

Fig. 3 shows the result of applying the developed algorithm to the first dataset provided with the homework. The green line is what would be generated before the merging process while the blue line is that generated when similar/close lines are merged. (a better resolution version of the figure is uploaded with the submission)

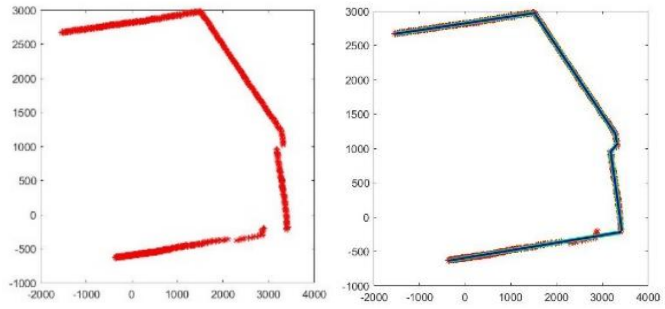

Fig. 3. First dataset (left) with the split and merge algorithm applied to extract lines (right). Showing the performance of the algorithm prior to merging (green line) and after merging similar lines (blue line)

Fig. 4 shows the performance of the algorithm on the same dataset but with different thresholds for the maximum distance allowed before the program splits the set into subsets.

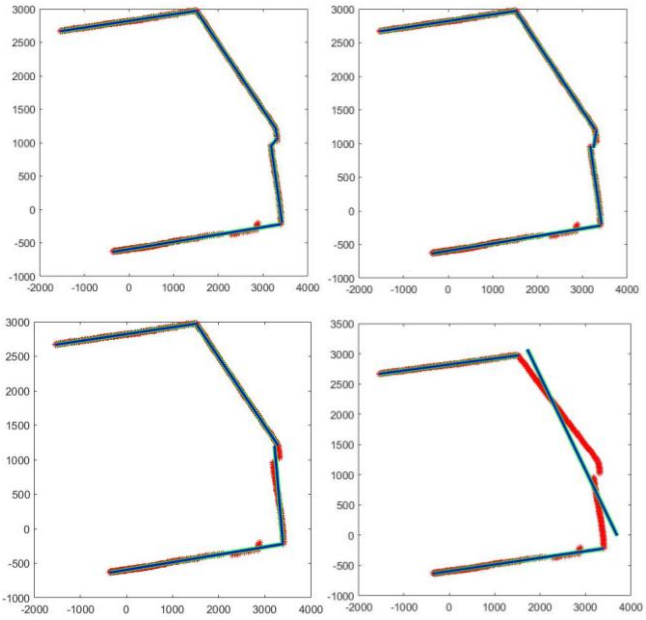

Fig. 4. First dataset with different maximum distance thresholds. 20mm threshold (top left), 50mm threshold (top right), 100mm threhshold (bottom left) and 500mm threshold (bottom right)

Lastly, Fig. 5 shows the performance of the algorithm on multiple datasets.

It is noticeable that the algorithm does not perform very well when there are multiple parallel walls far away from each other as seen in the last dataset. This implementation of the algorithm tends to create a line running from one of these walls to the other which is not always a good thing.

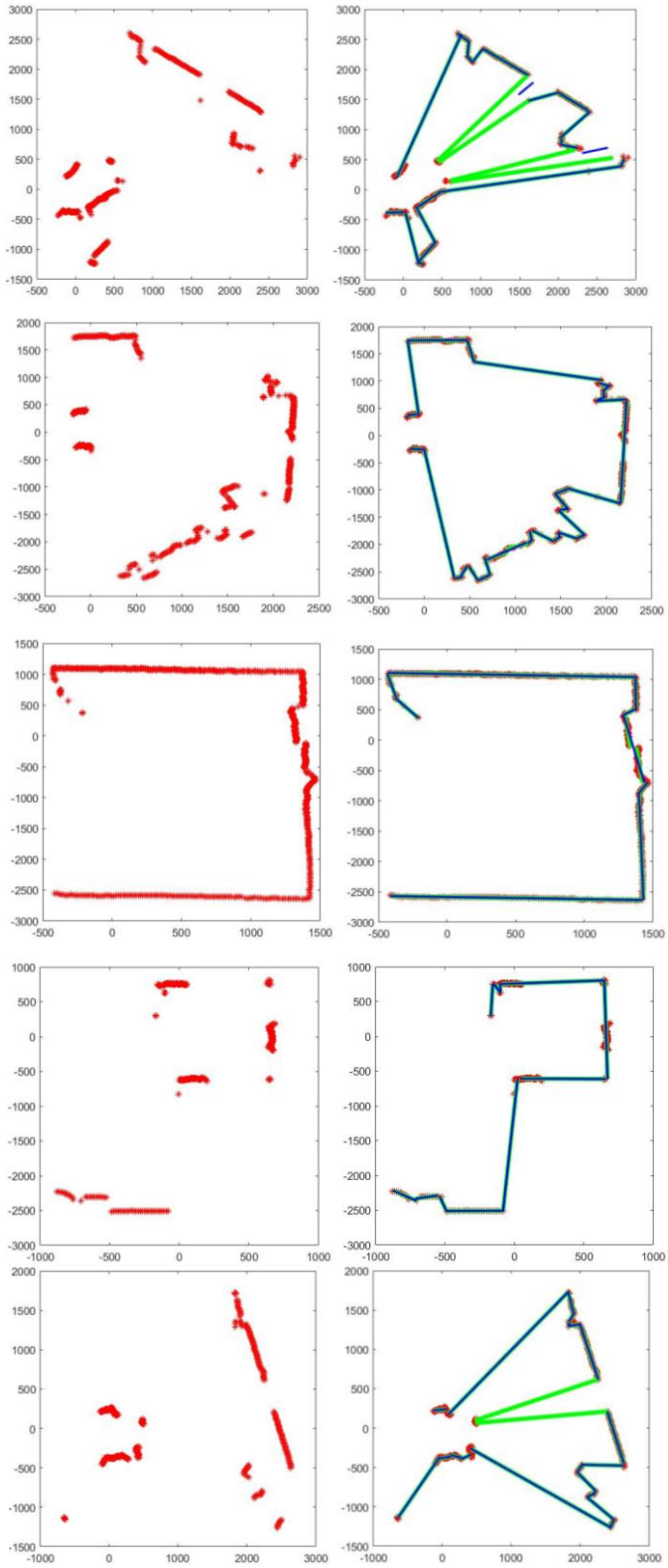

Fig. 5. Results (right) of applying the algorithm to multiple datasets (left) with a maximum distance threshold of 20mm

VIII. Split and Merge on Images (Extra Credit)

The algorithm was made to work with images by first thresholding the image to convert it into a binary image, detecting the edges within the image and then applying the split and merge algorithm to the detected edges. This works because as seen in Fig. 6, the edge points look exactly like points that would be gathered through a sensor.

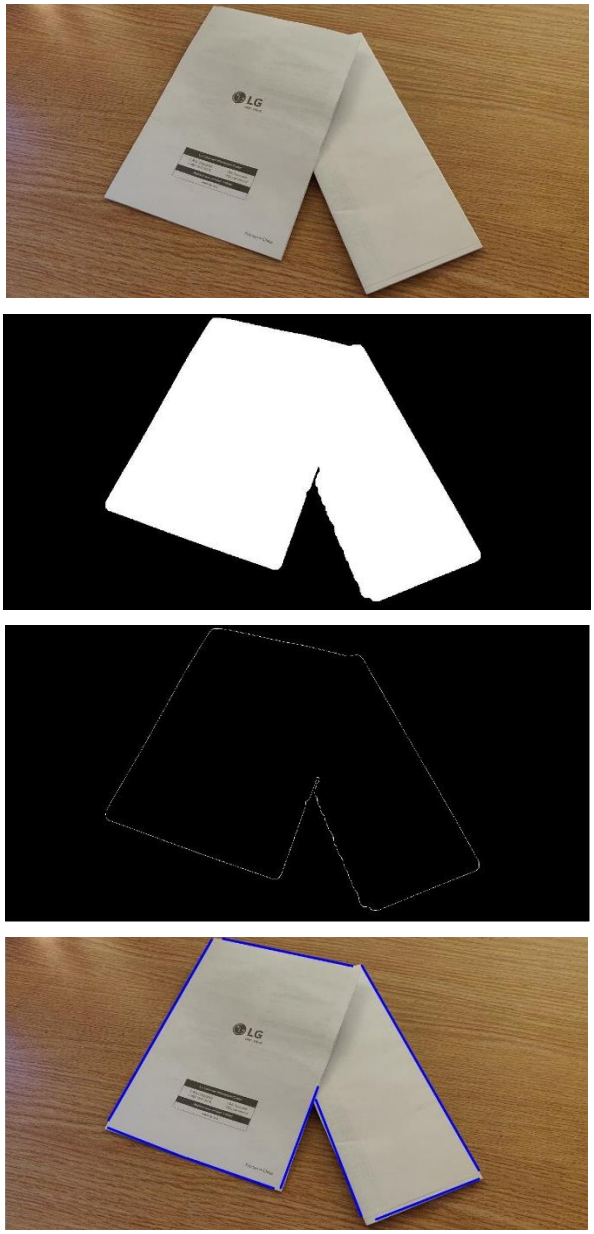

Fig. 6. Results of applying the algorithm to an image. Original Image (1st) Thresholded Image (2nd) Edged Image (3rd) Final Result (4th)

IX. Conclusion

To conclude, the split and merge algorithm was implemented successfully using the guides in the references, the results are promising, and the image results are what was expected from the algorithm.

References

[1] Siegwart, Roland, Illah R. Nourbakhsh, and Davide Scaramuzza, Introduction to Autonomous Mobile Robots, second edition

[2] V. Nguyen, S. Gächter, A. Martinelli, N. Tomatis, and R. Siegwart, “A comparison of line extraction algorithms using 2D range data for indoor mobile robotics,” Autonomous Robots, vol. 23, no. 2, pp. 97–111, Jun. 2007.

[3] H. Gao, X. Zhang, Y. Fang, and J. Yuan, “A line segment extraction algorithm using laser data based on seeded region growing,” International Journal of Advanced Robotic Systems, vol. 15, no. 1, p. 172988141875524, Jan. 2018.

[4] G. A. Borges and M.-J. Aldon, “A split-and-merge segmentation algorithm for line extraction in 2D range images,” in Proceedings 15th International Conference on Pattern Recognition. ICPR-2000.