Connected Components for a Fast and Robust 2D Lidar Data Segmentation

Conference Paper · July 2013

DOI: 10.1109/AMS.2013.31

3 authors:

Daniel Oñoro
University of Alcalá
15 PUBLICATIONS 1,167 CITATIONS
SEE PROFILE

Artem Lensky
Australian National University
30 PUBLICATIONS 306 CITATIONS
SEE PROFILE

Jee-Hwan Ryu
Korea Advanced Institute of Science and Technology
220 PUBLICATIONS 5,932 CITATIONS
SEE PROFILE

Connected Components for a Fast and Robust 2D LIDAR Data Segmentation

Daniel Oñoro Rubio, Artem Lenskiy
School of Electrical, Electronics & Communication Engineering
Korea University of Technology and Education
Cheonan, Korea
daniel.onoro@gmail.com, lensky@koreatech.ac.kr

Jee-Hwan Ryu
School of Mechanical Engineering
Korea University of Technology and Education
Cheonan, Korea
jhryu@koreatech.ac.kr

***Abstract*—The paper presents a novel segmentation approach applied to a two-dimensional point-cloud extracted by a LIDAR device. The most common approaches perform well in outdoor environments where usually furniture and other objects are rather big and are composed of smooth surfaces. However, these methods fail to segment uneven, rough surfaces. In this paper we propose a novel range data segmentation algorithm that is based on the popular one-pass version of the Connected Components algorithm. Our algorithm outperforms most commonly used approaches, while keeping the low computational complexity. The algorithm is used as a part of control and perception system in our unmanned ground vehicle where real-time response time is required. We presented experimental results obtained indoors and outdoors. The latter experiment was conducted in the real test field while the vehicle was autonomously driven.**

Keywords- 2D Lidar Segmentation; Connected Component; Intelligent Vehicles; Point-cloud analysis; Unmanned Ground Vehicle;

I. Introduction

In the past decade or so the rapid growth of robotics and particularly increasing popularity of autonomous vehicles opened up new areas of applications of LIght Detecting And Ranging (LIDAR) devices. LIDAR is a remote sensing technology that allows measuring distances by emitting a laser beam and analyzing light reflected by the object. LIDAR technology is known to be accurate and the measurements can be obtained at a high frequency. These are the reasons why LIDAR technology is found to be useful in various robotics fields. Nevertheless, the field of unmanned ground vehicles (UGV) that is a sub-field of robotics brings its specific requirements to range data analysis. These requirements include fast processing, accurate and reliable performance. In this paper we focus on the segmentation task. LIDAR data segmentation is a necessary part in many tasks such as obstacle detection and object detection/recognition.

A number algorithms have been reported in the area of range data segmentation [1] [2] [3]. However, most of them focus on 3D data analysis. Nevertheless, some of the reported works employ 2D data segmentation for car detection and tracking. The authors of [4] implemented a general approach based on the idea that beams reflected from the same object have similar distances. We investigated this approach by calculating distances between consecutive points. This approach generates artifacts that make this technique unacceptable in some UGV applications.

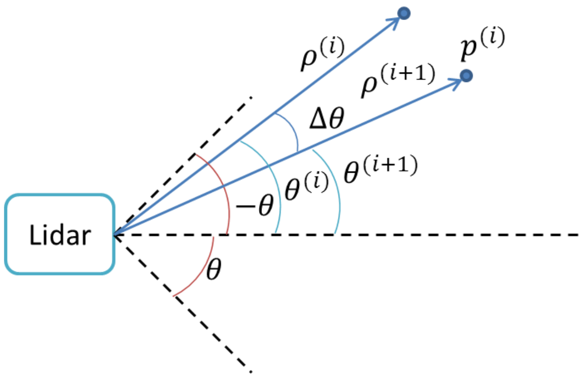


Figure 1: This image illustrates a single layer LIDAR. In this image, each point $p_l^{(i)}$ is defined by the distance to the device $\rho^{(i)}$ and an angle $\theta^{(i)}$.

The contribution of this paper consists proposing a simple yet fast and accurate segmentation algorithm. We also overviewing the most common approaches for 2D data segmentation and compare them with the proposed algorithm.

The remainder of this paper is organized as follows: Section II gives an introduction to the problem and discusses the most common approaches and their limitations. Section III describes the proposed algorithm and its implementation. Section IV presents the experimental results and comparison analysis. In section V we give concluding remarks and briefly describe future directions of our work.

II. Literature Overview

Before we proceed with the description of the proposed algorithm it is important to review the functional principles of the sensor and show some of the issues of commonly used segmentation approaches.

The LIDAR device consists of a laser and a mirror or a set or mirrors that are constantly rotating within the device. The mirror reflects the laser beam and then the sensor measures how far photons have travelled round-trip. The device returns an array of distances $\rho^{(i)}$ from the laser to the object that reflected the beam. The array index corresponds to the angle of the mirror that varies in the range of $[-\theta, \theta]$. The range of angles is divide into intervals of $\Delta\theta$. A general representation of one of these devices in shown in fig. 1.

Due to the nature of these devices, the absolute value of the differences of consecutive ρ distances $|\Delta d_\rho(p_l^{(i)}, p_l^{(i+1)})|$ between a LIDAR point $p_l^{(i)}$ with a distance $\rho^{(i)}$ for an angle $\theta^{(i)}$ and the consecutive point $p_l^{(i+1)}$ with a distance $\rho^{(i+1)}$ for an angle $\theta^{(i+1)} = \theta^{(i)} + \Delta\theta$ should be close to 0 for all the points that lie on the same object [4].

If the absolute value of the difference of the distances between two consecutive laser points is less than some certain threshold Δd_{th}, these two points are assigned to the same object. Otherwise, the point is associated with a new object.

We have implemented and experimented with the above-described approach. Figure 2 shows the differences of distances calculated for consecutive points. The blue line represents the differences for the ρ distances between two conductive points in polar coordinates (i.e $\rho^{(i)}$ and $\rho^{(i+1)}$). The back line shows the differences of the Euclidian distances from one point $p_C^{(i)}$ to next one $p_C^{(i+1)}$. The points first were converted from polar coordinate system to Cartesian coordinates The green line shows the differences of the Euclidian distances of one point $p_C^{(i)}$ to its nearest neighbor in the Cartesian plane. Lastly, the red line shows the ρ values of each laser beam. These values have been normalized in order to fit them in the same graph. Hence, in this graph we can see a high response of the differences of the distances every time there is a gab between objects. in ρ (red line).

As it can be seen from the figure 5a the segmentation based on comparing the differences in ρ can fail to segment any object with an uneven surface. For example this approach will fail to segment a fence made of bars. The bars will reflect some of the beams while other beams will penetrate through the fence making each bar labeled separately as a different object.

The solution that could resolve this problem consists is applying clustering algorithms. However, to apply clustering algorithms we have to know the number of clusters in advance. K-means clustering algorithm is an example of such algorithm. In reality the number of cluster is not given. Nevertheless there are clustering algorithms that do not require knowing the number of clusters in advance. One of them is Fast-RNN [5]. Unfortunately, Fast-RNN has a complexity of $O(n^2)$. Moreover, as it is seen in figure 6a Fast-RNN does not provide good clustering.

III. Proposed Model

One of the most essential algorithms in pattern recognition and image processing is the Connected Components (CC) algorithm [6]. The purpose of the CC algorithm is to transform the input binary image, into a symbolic image

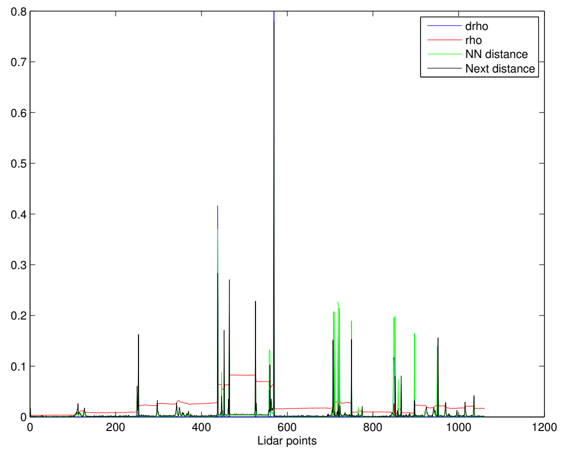


Figure 2: This graph shows the responses of the differences of the distances to consecutive points. The blue line represent the difference operator applied directly to the ρ's values from the laser. The green line shows the difference of the Euclidian distances from a point $p_C^{(i)}$ to its nearest neighbor. The black line shows the difference of the Euclidian distances from the point $p_C^{(i)}$ to the point $p_C^{(i+1)}$. Lastly the red line shows the ρ values from the laser.

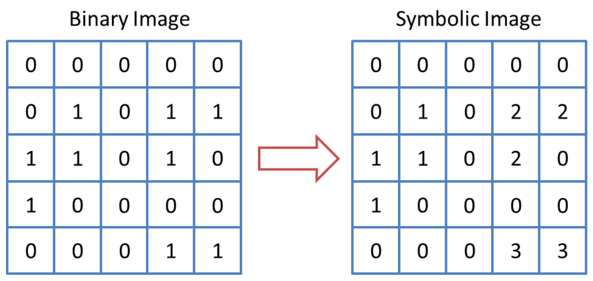


Figure 3: The left part of the figure shows an example of a binary image. The right side shows a result of the labeling process using CC.

where each connected component will have a unique label (see figure 3).

We extend the CC algorithm from labeling binary images to segmentation of 2D LIDAR data by firstly preprocessing LIDAR data. The data returned by LIDAR is represented as an array of k 2D points. The index of the array corresponds to the angle $\theta^{(i)}$ and the value is the distance to the object along the beam emitted in the direction of the angle $\theta^{(i)}$. Before applying the CC algorithm, the points should be converted to Cartesian coordinate system. Then points are projected onto occupancy grid. This grid is treated as a binary image and used for labeling. The whole process is summarized as follows:

1) Convert all the input points p_l (originally in polar coordinates) to points p_C in Cartesian coordinates.
2) Discretize the obtained Cartesian values

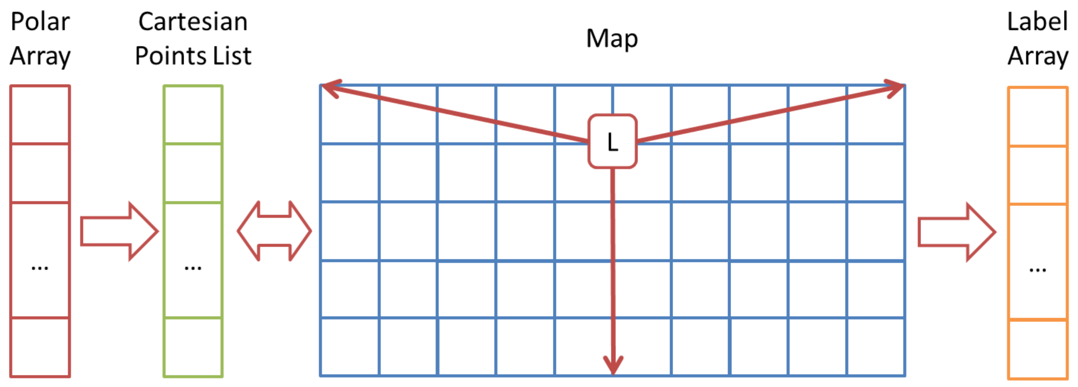


Figure 4: The diagram shows the segmentation flow of the range data obtained by the LIDAR. *Polar Array* contains the raw data obtained by the laser and is shown in red. *Cartesian Point List* is a linked list that is a structure that holds all the unlabeled points in Cartesian coordinates. *Map* is a $m * n$ occupancy grid which is used as an input data for connected component analysis. Finally, *Label Array* is an array which contains the labels of the input point-cloud.

3) Project all the 2D points onto an occupancy grid.
4) Find connected component over the grid.
5) Return labels.

The following formula is used to convert the raw data represented in polar coordinates, to a Cartesian coordinate system:

$$p_c^{(i)} = \rho^{(i)}[cos(\theta^{(i)})\ sin(\theta^{(i)})]^T. \tag{1}$$

The dimensions (m and n) of the grid are calculated as follows:

$$m = \frac{range\_max}{s}, \tag{2}$$

$$n = 2\frac{range\_max}{s}. \tag{3}$$

Where s is a scale factor that defines the size of each bin of the map. Therefore, we get a grid of $m * n$ bins. Thus, each bin can be thought as a portion of a real word with a constant size that can have 0 or k points of the scan. By putting all together, the grid is treated as a binary image, where we can consider each bin as a pixel that can have two values, 0 or 1 depending if it has points inside or not. During the labeling process a bin can have one of two states, activated and deactivated depending on whether that bin has been considered for labeling or not. The process of filling up of the occupancy grid given by

$$x = \frac{x^{(i)}}{s} + l_x, \tag{4}$$

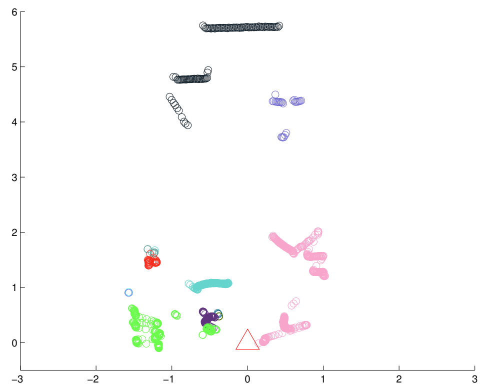
$$y = \frac{y^{(i)}}{s} + l_y. \tag{5}$$

Algorithm 1 Proposed Labeling Algorithm.

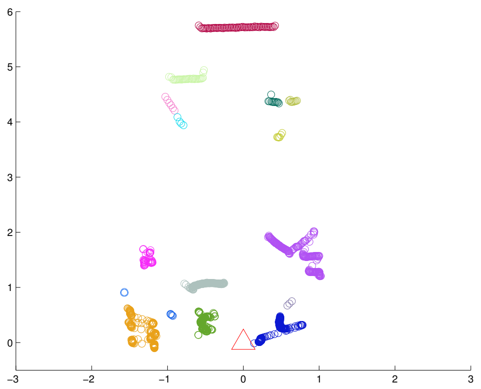
```
1: procedure CCLABELING($map, unlabelDataList$)
2:     $labelsVector \leftarrow zeros(length(unlabelDataList))$
3:     $currLabel \leftarrow 0$
4:     while $length(unlabelDataList) > 0$ do
5:         $point \leftarrow unlabelData.first()$
6:         $unlabelData.pop\_front()$
7:         $point2LabelList\ ADD\ point$
8:         repeat
9:             $point2Label \leftarrow point2LabelList.first()$
10:            $point2LabelList.pop\_front()$
11:            $nh \leftarrow getNeighbourhood(point2Label)$
12:            $deactivateBins(map, point2Label)$
13:            $deactivateBins(map, nh)$
14:            $point2LabelList\ ADD\ nh$
15:            $labelsVector \leftarrow labelPoint(point2Label)$
16:        until $length(point2LabelList) = 0$
17:        $currLabel \leftarrow currLabel + 1$
18:    end while
19:    return $labelsVector$
20: end procedure
```

Where $x^{(i)}$ and $y^{(i)}$ are the x and y of the point $p_C^{(i)}$ and l_x and l_y are the x and y coordinate where the LIDAR is placed on the map.

After we filled up the occupancy grid, we are able to apply CC to the obtained grid. There are several versions of the algorithm: one pass [7], two passes [8]. There are server ways to decrease the computational by using complex data structures such as binary trees (e.g. [9]). Such algorithms are able to can reach a constant execution time regardless of the number of pixels. However, for our specific model, we propose a new single pass version of this algorithm that

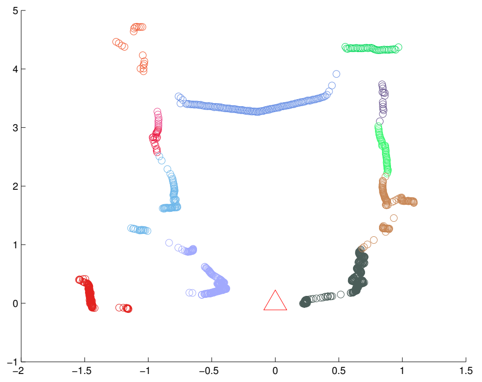


(a) This image show the result of a segmentation based on the most common approaches that rely on difference of ρ.

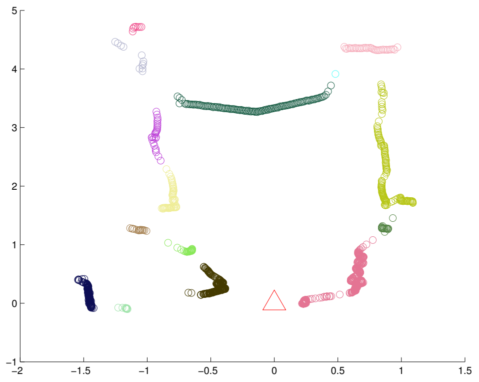


(b) This plot shows the results of the proposed algorithm.

Figure 5: This figure shows a comparison results between the approach based on the difference of the ρ distances and the proposed algorithm of this paper based on Connected Components algorithm. In both images, the laser is located in the position $(0, 0)$ which is represented by a triangle. As we can appreciate, (a) presents some artifacts like dividing some object that should not be divided, or by joining other that should be sparse. In contrast, we can see in (b) how our algorithm do not divide the object that are clearly just one object while it can separate in a better way those objects that are not connected.



(a) This figure shows the results of a data segmentation done by the algorithm Fast-RNN.



(b) This graph shows the results of the algorithm proposed in this paper.

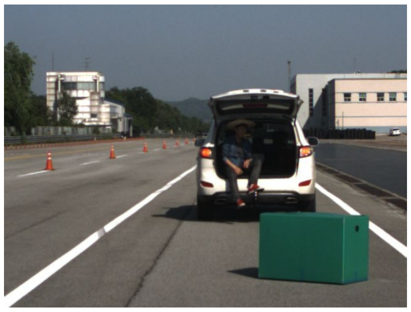
Figure 6: This figure shows a comparison between the clustering algorithm Fast-RNN and our method. In both images, the laser is represented by a triangle which is located in the $(0, 0)$ position. In the image (a) we can see how this algorithm has the problem that divides the long surfaces, like in the right wall, into different clusters while in (b) this the effect is not as pronounced.

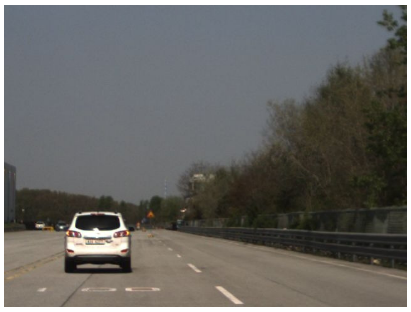
can run with an $O(n)$ complexity.

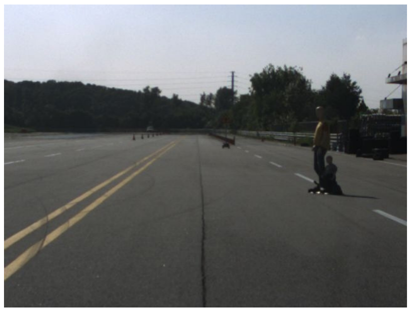
The pseudo-code of our Connected Components algorithm is give in 1. Due to the fact that we have a list of points, we can bypass the grid scanning and directly jump into labeling. Hence, just taking the first point of our list of points or just by choosing one randomly we can start to label. By taking a bin that is activated and occupied, we assign the current label to all of the points that fall into that bin. Then, we deactivate it and begin to expand the labeling process to the neighbor bins in a 4-connected or 8-connected version. When there are no activated bins left to expand, we will increase the label, by choosing a point that has not being labeled yet, then the previous process in repeated until all the points are labeled.

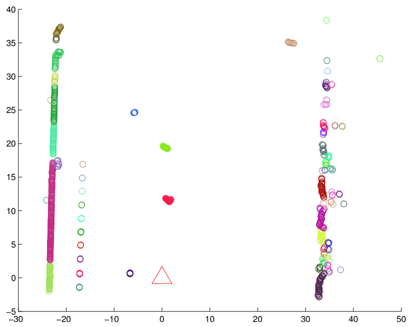
Implementation Details

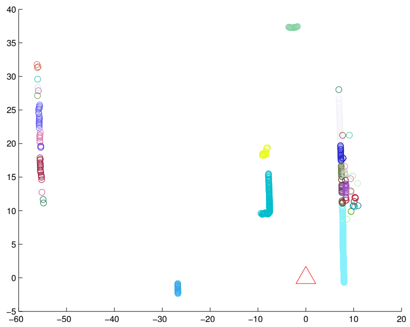
The above-descried algorithm runs in a linear time due to the fact that we know the locations of all the points in the grid from the beginning, so we achieved on average a rate of 238 segmentations per second. The figure 4 shows the data structures that we is used in the implementation. In order to make it fast and efficient in memory we are handling 4 data


(a) Obstacle avoidance.


(b) Car detection.


(c) Pedestrian detection.


(d) Obstacle avoidance.


(e) Car detection.

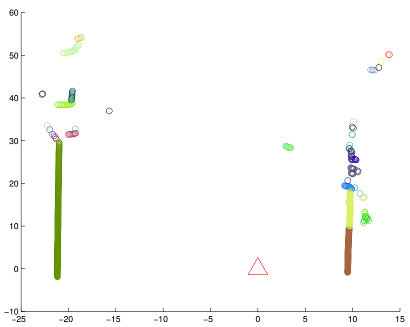

(f) Pedestrian detection.

Figure 7: This image shows the results of the proposed algorithm in a real application for an autonomous vehicle. The first row shows the contextual information which is provided by a conventional camera, while the second row shows the segmentation results. In each graph of the second row, each group is represented by a color. The position of the lidar device is always placed at $(0, 0)$ position and it is represented by a triangle.

structures at once. The first data structure is the raw input data from the LIDAR (represented by red the red color). This array contains all the p_l points from the LIDAR sorted from $-\theta$ to θ. This gives us a rigid data structure that will define the size of the output. The second data structure is a list that contains all the p_C points. This structure is just a linked list that points into the occupancy grid. Each bin in the grid is a data structure that points to the elements p_C, plus can hold two states, activated or deactivated. These two states allow us to avoid the labeling the points that has been already labeled. Therefore, the relationship between p_C and the map is bidirectional. In contrast, the relationship between p_l and p_C is in one direction as p_l is not modified or accessed again.

IV. Experiments

This section devoted to experimental analysis of the proposed algorithm. We analyze the quality of the proposed algorithm by comparing segmentation results with the above-described algorithms. The labeling obtained for the data extracted indoors and outdoors.

In both experimental setups we used a Sick lms511 pro device, that provides a maximum of 1141 points for a single scan and runs at 25Hz. The algorithm was running on the Intel Core i7-3820 CPU at 3.60GHz.

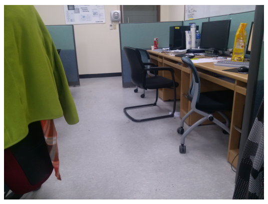
A. *Indoors experiments*

The LIDAR device was placed on a table of our lab (see image 8) facing various obejcts. The scale factor s that we used in order to build up the occupancy grid is 10cm.

The segmentation performance for indoors environment is shown in the figure 5b. The point-cloud that is used for labeling is exactly the same as the one shown in the figure 5a. It is visually easy to see that our algorithm based on CC has a better capability to properly segment the data into different object that segments that belongs to different objects. On the other hand the other method does not separate well some clearly defined object. Comparing the segmentation results obtained by the proposed algorithm (figure 6b), against the results produced by Fast-RNN algorithm (figure 6a) it is easy to see how our algorithm preserves the same label even for long and thin objects simultaneously well separating objects that are not connected.

B. *Outdoor Experiments*

The purpose of developing our algorithm was to create a method that provides robust and fast laser data segmentation for an autonomous vehicle perception system. Therefore, in


(a) View from lidar.

(b) Laboratory hardware setup.

Figure 8: This two images are shown for a better understanding of the scans of our lab.

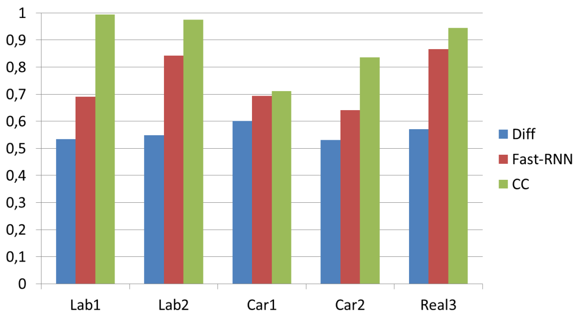


Figure 9: This graph shows a numerical comparison between the algorithm based on the ρ differences (Diff), Fast-RNN and the proposed method (CC)) with respect a handmade ground truth. The first column (*Lab1*) corresponds with the scan displayed on 5, the second one (*Lab2*) corresponds with 6 and finally the third, fourth and fifth (*Car1*, *Car2* and *Car3*), corresponds to 7 (d), (e) and (f) respectively.

this subsection we show some of the results obtained in the real outdoors environment.

We attached the LIDAR to the front bumper of our vehicle, placed in the middle of the bumper at 50cm above the ground. To provide contextual information for this experiment we synchronized LIDAR with a camera located inside of the vehicle. The scale factor s was fixed to 30cm. The figure 7 shows segmentation of the laser data into a number clusters. It can be see that most of the objects are correctly labeled. The segmentation time of one-scan takes on average about 0.0042 seconds; which is about 238 segmentations per second.

Finally, in order to show some numerical results (figure 9), on the performance of our algorithm, we measured the segmentation accuracy of each discussed algorithm by comparing them to hand-segmented maps. Our algorithm outperformed other algorithms in most of the tests except *Car1*. The reason why our algorithm underperformed is a large amount of noise on the right side of the scan that is labeled as many independent objects.

V. Conclusion

In this paper, we have presented a 2D point-cloud segmentation algorithm. We showed that algorithm outperforms the common 2D point-cloud segmentation approaches. We have compared the quality of segmentation and the computational time of our algorithm and the other segmentation approaches, based on the range data obtained indoors and outdoors. Discretizing the space, mapping the point-cloud onto a grid and then applying an efficient data structure that is used in Connected Components analysis achieves such good performance.

We are planning to extend the proposed algorithm by creating a non-equal cell sized grid. The cell size should depend on the distance to the points. The points located closer to the sensor are more densely distributed. Moreover, there is more uncertainty for the points located farther away from the sensor. The larger cell size will take into account such uncertainty.

References

[1] Y. Ioannou, B. Taati, R. Harrap, and M. Greenspan, “Difference of normals as a multi-scale operator in unorganized point clouds,” *ArXiv e-prints*, Sep. 2012.

[2] W. Yao, S. Hinz, and U. Stilla, “Object extraction based on 3d-segmentation of lidar data by combining mean shift with normalized cuts: Two examples from urban areas,” in *Urban Remote Sensing Event, 2009 Joint*, 2009, pp. 1–6.

[3] J. Aue, D. Langer, B. Muller-Bessler, and B. Huhnke, “Efficient segmentation of 3d lidar point clouds handling partial occlusion,” in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, 2011, pp. 423–428.

[4] M. Thuy and F. Leon, “Non-linear, shape independent object tracking based on 2d lidar data,” in *Intelligent Vehicles Symposium, 2009 IEEE*, 2009, pp. 532–537.

[5] R. J. Lopez-Sastre, D. Onoro-Rubio, P. Gil-Jimenez, and S. Maldonado-Bascon, “Fast reciprocal nearest neighbors clustering,” *Signal Processing*, pp. 270–275, 2012.

[6] R. Haralick, “Some neighborhood operations,” *Real Time/Parallel Computing Image Analysis*, p. 11Ž01335, 1981.

[7] L. He, Y. Chao, K. Suzuki, and H. Itoh, “A run-based one-scan labeling algorithm,” in *Proceedings of the 6th International Conference on Image Analysis and Recognition*, ser. ICIAR ’09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 93–102.

[8] T. G. Y. O. M. Y. Y. Shirai, “High speed algorithm for component labeling,” *Trans. IEICE J72-D-II*, p. 2472̌013255, 1989.

[9] Y. Han and R. A. Wagner, “An efficient and fast parallel-connected component algorithm,” *J. ACM*, vol. 37, no. 3, pp. 626–642, Jul. 1990. [Online]. Available: http://doi.acm.org/10.1145/79147.214077