2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

# Fast Range Image-Based Segmentation of Sparse 3D Laser Scans for Online Operation

Igor Bogoslavskyi Cyrill Stachniss

*Abstract*— **Object segmentation from 3D range data is an important topic in mobile robotics. A robot navigating in a dynamic environment needs to be aware of objects that might change or move. A segmentation of the laser scans into individual objects is typically the first processing step before a further analysis is performed. In this paper, we present a fast method that segments 3D range data into different objects, runs online, and has small computational demands. Our approach avoids the explicit computation of the 3D point cloud and performs all computations directly on a 2D range image, which enables a fast segmentation for each scan. A further relevant aspect of our method is that we can segment objects even if the 3D data is sparse. This is important for scanners such as the new Velodyne Puck. We implemented our approach in C++ and ROS and thoroughly tested it using different 3D scanners. Our method can operate at over 100 Hz for the 64-beam Velodyne scanner on a single core of a mobile CPU while producing high quality segmentation results. In addition to this, we make the source code for the approach available.**

## I. Introduction

Detecting objects in 3D laser range data is an important task in mobile robotics. A robot that is navigating in an unknown environment faces the complicated task of reasoning about its surroundings [2], [3], [9], [11], [12], [13], [15], [16], [22], [24], [25], [26]. There might be objects that constrain the possible actions of the robot or that may interfere with the robot's own plans. Thus, the interpretation of the robot's surroundings is key for robust operation. A first step in a standard perception pipeline is often a segmentation of the environment into individual objects. Therefore, we see the need for an efficiently computable online segmentation approach for 3D range scans. This will allow a robot to directly react to individual objects in its surroundings.

In addition to that, modern robots are able to build accurate maps using SLAM algorithms while moving through unknown environments. In dynamic environments such as busy streets with cars and pedestrians, the maps can be influenced by wrong data associations caused by the dynamic nature of the environment. A key step to enable a better reasoning about such objects and to potentially neglect dynamic ones during scan registration for mapping, is segmenting the 3D range data into different objects so that they can be tracked [6]. This segmentation should be available in real time as the robot needs to reason about what it sees right when the data becomes available in order to react appropriately.

Laser range sensors also called LIDARs keep gaining popularity as the price of the sensors keeps dropping. For example with recent introduction of an affordable 16-beam LIDAR by Velodyne, this type of sensors is becoming more popular and can also be installed on relatively low-cost platforms and not only on robotic cars. If we compare the data provided by the 16-beam LIDAR with the ones provided by the 64-beam variant, we observe a substantial drop in the vertical angular resolution. This poses additional challenges to a segmentation algorithm operating on such 3D data. Sparser point clouds lead to an increased Euclidean distance between neighboring points even if they stem from the same object. Thus, such sparse 3D points render it more difficult to reason about segments. The situation becomes even harder with increasing distance between the object and the sensor.

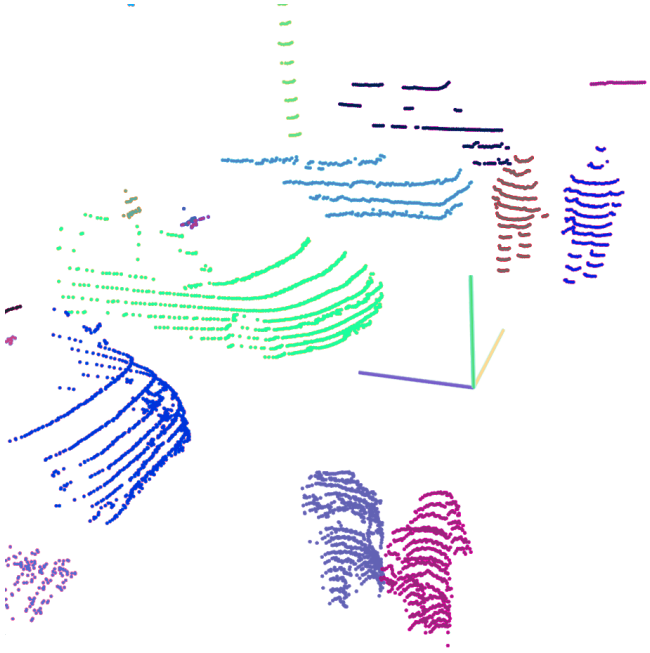

Fig. 1. Segmentation of typical objects. Here people, cars, and trees generated from sparse 3D range data recorded with Velodyne VLP-16. The segmentation runs at over 400 Hz on a mobile CPU. The sensor position is marked with a coordinate system in the center of the image. Note that even the cars, people and a tree trunk that are further away from the scanner are segmented in a meaningful way.

The contribution of this paper is a fast and effective segmentation approach for 3D range data obtained from modern laser range finders such as Velodyne scanners. Our approach provides meaningful segmentations and runs multiple times faster than the acquisition of the scan. Even on a mobile CPU, we can process 64-beam Velodyne scans at over 100 Hz. We achieve this by performing all computations

Both authors are with Institute for Geodesy and Geoinformation, University of Bonn, Germany.

on a cylindrical range image. This has two advantages: First, the range image is often small, dense, and maintains the neighborhood information implicitly in its 2D structure. Second, operating on such range images is substantially faster than reasoning on the 3D point cloud. Our approach is also suited for scanners that provide comparably sparse point clouds. An example of such segmentation is depicted in Fig. 1 where people and cars are correctly segmented using data from a Velodyne VLP-16 scanner. At the same time, we explicitly target low computational demands for our segmentation approach. We implemented our approach in C++ using ROS and are sharing the source code.

## II. Related Work

Segmenting objects from 3D point clouds is a relatively well-researched topic. There is substantial amount of work that targets acquiring a global point cloud and segmenting it off-line, see for example [1], [9], [11], [12], [25]. These segmentation methods have been used on a variety of different data such as 3D range sensors or 2D lasers in push-broom mode. In this work, we focus on the segmentation of range data that comes from a 3D laser scanner such as a Velodyne that provides a 360 degree field of view in a single scan and is used for online operation on a mobile robot.

Segmentation techniques for single scans without requiring additional information can be divided into three groups. The first group performs the segmentation in the 3D domain by defining sophisticated features that explain the data in 3D [7], [8] or by removing the ground plane and segmenting the clouds with a variant of a nearest neighbor approach [5], [14]. Feature-based approaches, while allowing for accurate segmentation, are often comparably time-consuming and may limit the application for online applications to a robot with substantial computational resources.

The second group focuses on projecting 3D points onto a 2D grid positioned on the ground plane. The segmentation is then carried out on occupied grid cells [2], [13], [15], [22]. These algorithms are fast and suitable to run online. Quite often, however, they have a slight tendency to under-segment the clouds, i.e. multiple objects may be grouped as being one object if they are close to each other. This effect often depends on the choice of the grid discretization and may need to be tuned for individual environments. Additionally, some of these approach can suffer from under-segmenting objects in the z-direction.

The third group of approaches performs the segmentation on a range image and our approach belongs to this group of techniques. For example, Moosmann et al. present two approaches [17], [18] of that type. They use a range image to compute local convexities of the points in the cloud. In contrast to that, our approach is easier to implement and relies on a single parameter only, runs very fast and produces comparable results. We therefore believe that our approach is a valuable contribution to a vast and vibrant field of 3D point cloud segmentation and thus we intend to make our source code available.

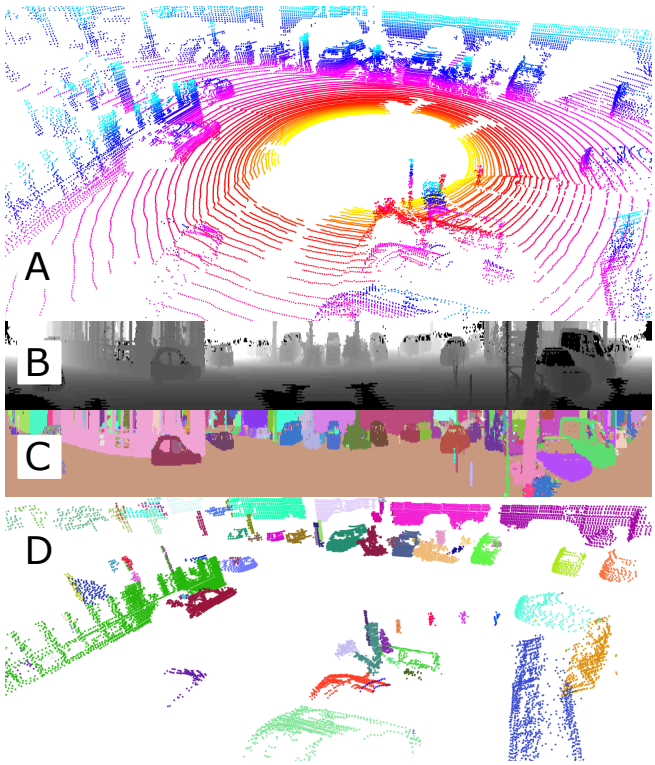


Fig. 2. Illustration of our method, best viewed in color. (A) Point cloud from Velodyne, which is shown for illustration reasons only. (B) We build up a range image not considering points lying on the ground plane and (C) perform the segmentation in the range image directly. (D) This allow us to provide individual small point clouds for the different segments. The different objects are shown with random colors. Range and label images are scaled for better visibility.

There are also several works that perform segmentation on RGBD data acquired from a LIDAR registered with a camera [20], [23]. Registering one or multiple cameras with the laser scanner requires more a sophisticated setup and the segmentation becomes more demanding. Using both cues may improve the results but it is seldom possible at 100 Hz. Therefore, we focus on segmenting unknown objects from pure 3D range data not requiring any additional visual or intensity information.

Visual information is not the only information that aids segmentation. Temporal information and tracking are also shown to be useful to enhance the segmentation performance [10], [24]. While the benefit of using the information about the moving objects is clear, we show that it is possible to perform a fast and meaningful segmentation on single scans even without relying on temporal integration.

## III. Fast and Effective Segmentation Using Laser Range Images

This work focuses on fast 3D range scan segmentation for online processing on a mobile robot that is equipped with a rotating scanner such as one of the three popular Velodyne scanners with 16, 32, or 64 beams. The resolution of the sensors, especially the vertical one, has an impact on the difficulty of the segmentation problem. For every pair of

Fig. 3. Robot equipped with a 16-beam Velodyne used for our experiments.

neighboring points, one basically has to decide if the laser beams have been reflected from the same object or not.

In our approach, however, we avoid the explicit creation of the 3D point cloud and perform our computations using a laser range image, in our case a cylindrical one for the Velodyne scanners. This has two advantage: First, we can exploit the clearly defined neighborhood relations directly in the range 2D image and this makes the segmentation problem easier. Second, we avoid the generation of the 3D point cloud, which makes the overall approach faster to compute.

Most laser range scanners provide an individual range reading and an orientation per laser beam with a time stamp as raw data. This allows us to directly turn the data into a range image. The number of rows in the image is defined by the number of beams in the vertical direction, i.e., 16, 32 or 64 for the Velodyne scanners. The number of columns is given by the range readings per 360° revolution of the scanner. Each pixel of such a virtual image stores the measured distance from the sensor to the object. To speed up computations, one may even consider to combine multiple readings in the horizontal direction into one pixel.

In our implementation, we use the above described range images and build them directly from the raw measurements of the laser scanner. In case, however, a different laser scanner or a different device driver is used that only provides a 3D point cloud per revolution and not the individual range measurements, one can project the 3D points cloud onto a cylindrical image, compute the Euclidean distance per pixel, and proceed with our approach. This will increase the computational demands by up to a factor of 2 for the whole approach but still allows for a rather fast segmentation.

Throughout this work, we assume that the vehicle moves on the ground (see Fig. 3 for our setup) and we know the orientation of the sensor with respect to the wheels. Thus, we can quickly obtain a estimate of the ground plane by analyzing the columns of the range image, which can be seen as an approximation of the ground plane estimation in [13], [19]. The ground is then removed from the range image.

The key building block of our approach is the ability to estimate which measured points originate from the same object for any two laser beams. We present an easy to implement and fast to compute but yet effective approach

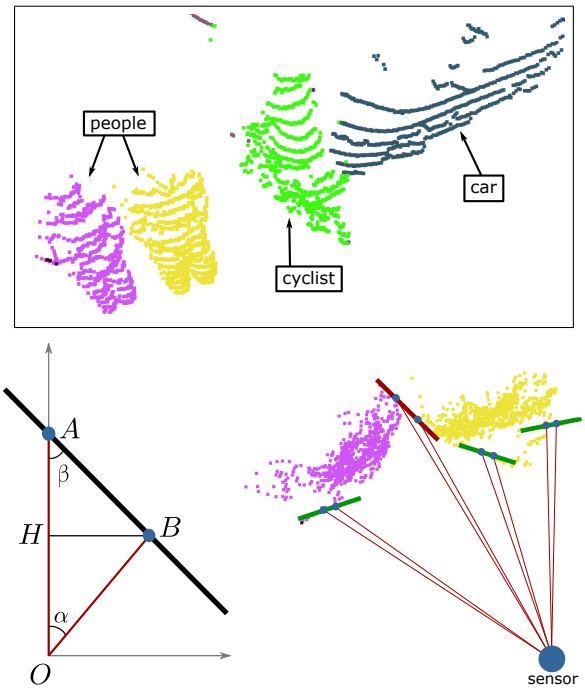


Fig. 4. *Top:* example scene with two pedestrians, a bicyclist and a car. *Bottom left:* Given that the sensor is in $O$ and the lines $OA$ and $OB$ represent two laser beams, the points $A$ and $B$ spawn a line that estimates the surface of an object should they both belong to the same object. We make the decision about this fact based on the angle $\beta$. If $\beta > \theta$, where $\theta$ is a predefined threshold, we consider the points to represent one object. *Bottom right:* a top view on the pedestrians from the example scene. The green lines represent points with $\beta > \theta$ while the red one shows an angle that falls under the threshold and thus labels objects as different.

to find the components that belong to one object. To answer the question if two laser measurements belong to the same object, we use a measure, which is illustrated in Fig. 4 and is described in the following paragraphs.

The top image of Fig. 4 shows an example scene with two people walking close to each other in front of a bicyclist, who passes between them and a parked car. This scene has been recorded using our Velodyne VLP-16 scanner. The bottom left image shows an illustration of two arbitrary points $A$ and $B$ measured from the scanner located at $O$ with the illustrated laser beams $OA$ and $OB$. Without loss of generality, we assume the coordinates of $A$ and $B$ to be in a coordinate system, which is centered in $O$ and the $y$-axis is oriented along the longer of two laser beams. We define the angle $\beta$ as the angle between the laser beam and the line connecting $A$ and $B$ in the point that is further away from the scanner (in our example that is $A$). Intuitively, the angle $\beta$ relates the distance in depth that two point on the same object may have with the distance to the scanned object and this allows to elegantly capture this information in a single parameter. In practice, the angle $\beta$ turns out to provide valuable information to determine if the points $A$ and $B$ lie on the same object or not.

Given the nature of the laser range measurements, we know the distance $\|OA\|$ as it corresponds to the first laser measurement as well as $\|OB\|$ (second laser measurement).

We will call these range measurements $d_1$ and $d_2$ respectively and can use this information to calculate $\beta$ by applying trigonometric equations

$$\beta = \arctan \frac{\|BH\|}{\|HA\|} = \arctan \frac{d_2 \sin \alpha}{d_1 - d_2 \cos \alpha},$$

where $\alpha$ is the known angle between the beams and is usually provided in the documentation of the scanner. The bottom right image in Fig. 4 illustrates the computation in the $x-y$ plane from a top-down view of the scene. The same approach can be taken in a coordinate system spawned by a projection of the laser beam to the $x-y$ plane and the $z$ axis. The logic behind the computations stays intact and both are needed given the neighborhood relation defined through the pixels of an (range) image.

The intuition behind the angle $\beta$ is that it stays relatively large for most objects and only takes small values if the depth difference between neighboring points given the range image is substantially larger than their displacement in the image plane that is defined through the angular resolution of the scanner. This insight allows us to define a parameter $\theta$ that acts as a threshold on the angle $\beta$. This threshold enables us to make a decision about whether to separate any two points in the range image into separate cluster or merge them into one. If $\beta$ is smaller than the user-defined value $\theta$, we argue that the change in depth is too large and make the decision to separate the points into different segments. Otherwise, the points are considered as lying on the same object.

A threshold-based criterion on $\beta$ is clearly a heuristic but works well in practice as we will illustrate in the experimental evaluation. A failure case can be a situation in which the scanner is located close to a wall. For the endpoints located far away from the scanner but still on the wall, the angle $\beta$ will be small and it is therefore likely for the wall to be split up in multiple segments. This essentially means that if $\beta$ is smaller than $\theta$, it is difficult to reason if we look at points that originate on two different objects or just lie on a wall nearly parallel to the beam direction. However, despite this shortcoming, our experiments suggest that the method is still useful in practice and the aforementioned behavior occurs rarely and if so, it usually results only in an over-segmentation of particularly inclined planar objects.

With the separating threshold in mind, we approach the segmentation directly in the range image. We regard two endpoints as being neighbors stemming from the same objects if they are neighbors in a the depth image and the angle $\beta$ between them is larger than $\theta$. Given this definition of a neighborhood, we can view the segmentation problem as the problem of finding the connected 2D components exploiting the structure of the depth image and the constraint on $\beta$.

Alg. 1 depicts the algorithm that we use to find the connected components. We use a variant of a pass-through filter with complexity $\mathcal{O}(N)$, where $N$ is the number of pixels, i.e. the number of range readings per scan. The algorithm guarantees visiting each point in the range image at maximum twice.

We start in the top left corner of the range image and pass through every pixel from top to bottom, left to right (line 4–5). Whenever we encounter a non-labeled pixel (line 6), we start a breadth-first search from this pixel on (line 7). The goal of this search is to label every pixel of this component. For that, the breadth-first search (BFS) uses a queue (line 10–12) and an N4 neighborhood consisting of the left, right, lower and top pixels (line 14). The decision if a point in the N4 neighborhood should be added to the queue of the BFS is made based on the angle $\beta$ generated by the neighbor and the current point (line 15–18). This procedure guarantees that the whole connected component will receive the same label. Once the queue of BFS is empty, we continue to traverse the range image sequentially until we reach a new unlabeled point.

This approach yields a fast execution time with an $\mathcal{O}(N)$ worst-case complexity and visits any pixel in the depth image at maximum twice. The connected components algorithm in itself, however, is not the main contribution of this work but its effective application to segmentation through range images considering the value of $\beta$ for two neighboring measurements. For more information on the comparison between different implementations of connected components algorithms, we refer the reader to [4].

Overall, our approach yields an easy-to-implement and fast method with a single parameter that even has a physical meaning. Therefore, our approach requires only minimal parameter tweaking to achieve good segmentation performance.

## IV. Experimental Evaluation

The main focus of this work is a fast and easy to implement segmentation approach for 3D range data that runs at 100 Hz or faster and provides a meaningful segmentation of the scene into objects so that the robot can exploit this information online to improve its understanding of the

**Algorithm 1** Range Image Labeling

```
1:  procedure LabelRangeImage
2:      Label ← 1, R ← range image
3:      L ← zeros(R_rows × R_cols)
4:      for r = 1 ... R_rows do
5:          for c = 1 ... R_cols do
6:              if L(r, c) = 0 then
7:                  LabelComponentBFS(r, c, Label);
8:                  Label ← Label + 1;
9:  procedure LabelComponentBFS(r, c, Label)
10:     queue.push({r, c})
11:     while queue is not empty do
12:         {r, c} ← queue.top()
13:         L(r, c) ← Label
14:         for {r_n, c_n} ∈ Neighborhood{r, c} do
15:             d_1 ← max(R(r, c), R(r_n, c_n))
16:             d_2 ← min(R(r, c), R(r_n, c_n))
17:             if arctan (d_2 sin α / (d_1 − d_2 cos α)) > θ then
18:                 queue.push({r_n, c_n})
19:         queue.pop()
```

TABLE I

AVERAGE RUNTIME AND STD. DEV. PER 360° LASER SCAN.

| scanner | mobile i5 U5200 2.2 GHz | desktop i7 4770K, 3.5 GHz |
|---|---|---|
| 16 beams | 2.4 ms ± 0.5 ms ≈ 416 Hz | 1.5 ms ± 0.2 ms ≈ 667 Hz |
| 32 beams | 4.4 ms ± 1.2 ms ≈ 227 Hz | 2.6 ms ± 0.5 ms ≈ 385 Hz |
| 64 beams | 8.6 ms ± 2.6 ms ≈ 116 Hz | 4.7 ms ± 1.2 ms ≈ 212 Hz |

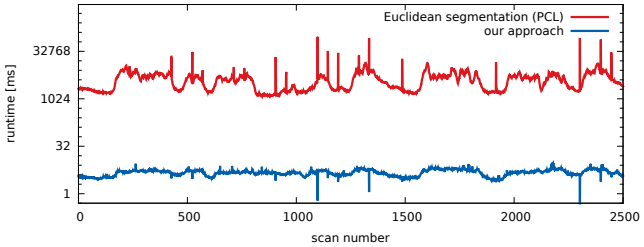


Fig. 5. Timings for segmenting approximately 2,500 scans from a 64-beam Velodyne dataset with our approach and Euclidean segmentation from PCL.

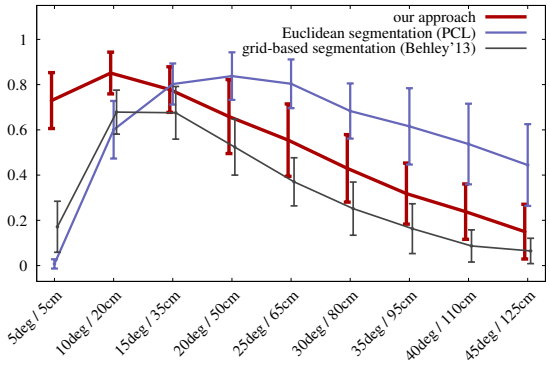


Fig. 6. Precision of our algorithm compared to the grid-based segmentation from Behley et al. [2] and segmentation through Euclidean clustering as provided by PCL for varying parameters on 30 different, manually labeled outdoor 3D scans. On the x-axis, the first value is the parameter $\theta$ for our method and the second one serves as both the cell size for the grid-based approach and as the distance threshold for the Euclidean clustering approach.

surroundings. Our experiments are designed to show the capabilities of our method and to support our key claims, which are: (i) all computation can be executed fast, even on a single core of a mobile CPU with at least 100 Hz, (ii) we can segment typical 3D range data obtained by mobile robots into meaningful segments, and (iii) the approach performs well on sparse data such as that obtained from a 16-beam Velodyne Puck scanner.

We furthermore provide comparisons to a popular grid-based method for segmentation proposed in [24] as used for example in [2] and to segmentation through Euclidean clustering as provided by PCL [21]. We perform the evaluations on own datasets as well as on publicly available ones. Throughout all these experiments, we set the only parameter of our approach to $\theta = 10°$ as this provides the best peformance as shown in Fig. 6.

### A. Runtime

The first experiment is designed to support the claim that our approach can be executed fast to support online processing on the robot in real time. We therefore tested our approach on point clouds computed with different Velodyne laser scanners and processed the data on different computers. On the robot, we used an Acer notebook with an i5 5200U 2.2 GHz CPU but we also processed the data on a desktop computer with an i7 4770K 3.5 GHz CPU, in both cases using only one core of the CPU.

Tab. I summarizes the runtime results for nearly 2,500 point clouds recorded in urban outdoor environments. The numbers support our first claim, namely that the computations can be executed fast and in an online fashion. The frame rate of our segmentation pipeline is more than one order of magnitude larger than the frame rate of the laser scanner. On a mobile i5 CPU, we achieve average frame rates of 116 Hz-416 Hz depending on the scanner and 212 Hz-667 Hz on an i7 desktop computer.

We also compared the speed of our segmentation pipeline to Euclidean clustering for segmentation as provided by PCL. Note that we do not perform any voxelization of space, neither for our approach nor for Euclidean clustering, as we aim to maintain all information. Fig. 5 shows the comparison for the 64-beam Velodyne. As can be seen, our approach is on average around 1,000 times faster than Euclidean clustering in the 3D space.

### B. Segmentation Results

The next set of experiments in designed to show the obtained segmentation results of our approach. We consider the results on sparse (16 beams) and dense (64 beams) laser range data. For the 64-beam evaluation, we rely on the publicly available street scenes dataset provided by Moosmann [17] while we recorded the 16-beam datasets using our robot in Bonn, Germany.

We evaluate the precision of our method and compare it to a popular grid- based approach [2] and to segmentation through Euclidean clustering as provided by PCL. For that, we have manually segmented 30 point clouds from different scenes and ran all three methods varying their parameters. For our method, we have chosen different values for $\theta$, while for the grid- based approach we have varied the size of the grid cells. We have chosen values for $\theta$ from 5° to 45° and for the grid cell resolution (grid-based) and the distance threshold (Euclidean) values between 0.05 m to 1.25 m. We have evaluated the precision of the algorithms by counting how many of the manually labeled objects have been found by the algorithms. For every ground truth cluster, we search for a found segment with the biggest overlap. We consider the cluster as correctly found if the point-wise overlap is substantial. We then count the number of successful matches and divide them by the number of expected ground truth clusters. We compute this precision value for every scan and present the mean and standard deviation of these values with relation to the chosen parameter in Fig. 6. As can be seen with the default parameter of $\theta = 10°$, our method outperforms the grid-based approach in terms of segmentation quality in all parameters settings. In comparison to Euclidean

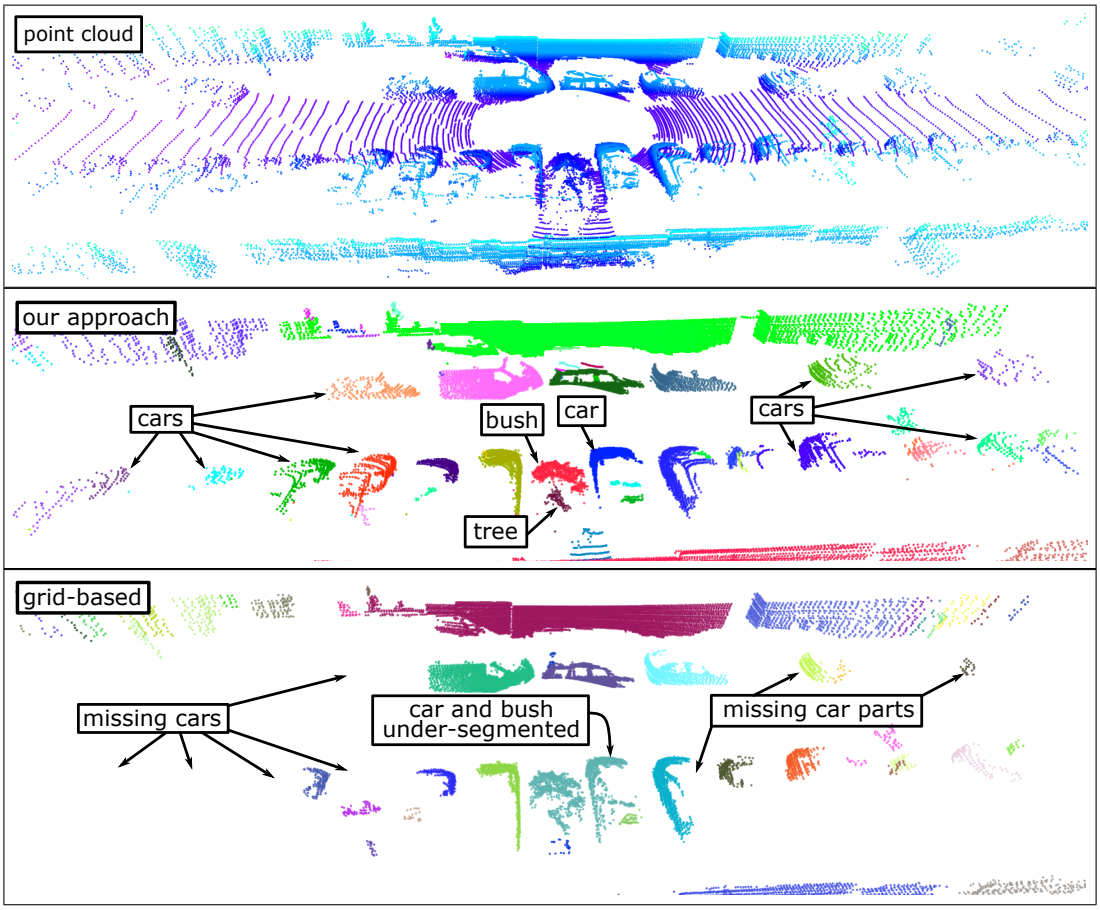


Fig. 7. *Top:* Point cloud of an outdoor scene taken with a 64-beam Velodyne (shown for illustration only). *Middle:* Our segmentation that provides correct segmentation even for distant objects while not under-segmenting the close ones. *Bottom:* Segmentation provided by a grid-based approach with cell size set to 0.2. There is a number of cars that are situated further from the sensor missing and one car is merged with a bush. Images are best viewed in color.

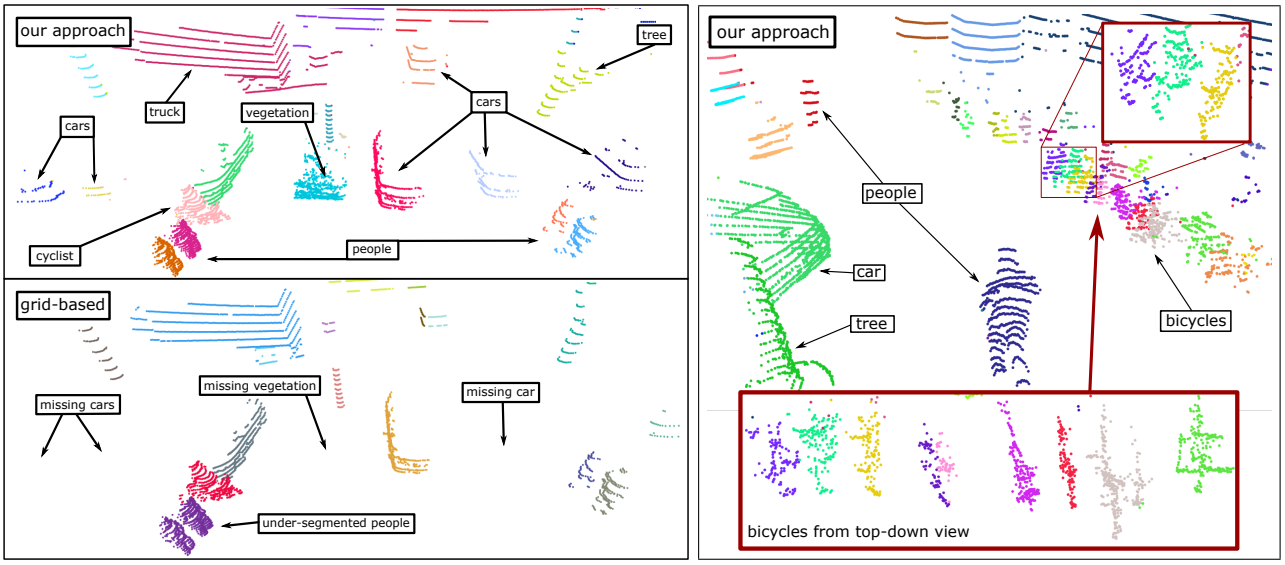


Fig. 8. *Left Top:* Our segmentation of an example outdoor scene taken with a 16-beam Velodyne. Our approach was able to find objects omitted by the grid-based method while correctly segmenting people that stand close to each other. *Left Bottom:* Grid-based segmentation result. Some objects are missing and people on the bottom left are under-segmented. *Right:* An outdoor scene recorded with a 16 beam Velodyne that shows that our approach is able to segment even complicated scenes with multiple small objects like bicycles placed very close to each other. The grid-based approach in this scene merged all the bicycles into two big clusters. The images are omitted for space reasons. Best viewed in color.

clustering, our approach shows quality-wise a comparable performance on the 64-beam datasets, while being around three orders of magnitude faster (4 ms vs. 4 s per scan). This nicely illustrates the benefits of our method for online processing. A typical example of a segmentation is shown in Fig. 7.

Finally, we aim at supporting our claim that our segmentation pipeline handles sparse data coming from a scanner with 16 beams in the vertical direction (Velodyne VLP-16) well. For this, we analyzed the results using data recorded from our scanner and compared them to manually labeled ground truth clouds. Examples are depicted in Fig. 8. Although this is only a qualitative evaluation, we can clearly see that our approach handles the sparse range data better than the approaches that work in the space of 3D points. We believe that the main reason for that is the fact that we operate directly on the range images and thus can better find the neighboring points that may result from scanning the same object.

In summary, our evaluation suggests that our method provides competitive segmentation results compared to existing methods on dense 3D range scans and outperforms them on sparse scans. At the same time, our method is fast enough for online processing and has small computational demands. Thus, we supported all our claims with this experimental evaluation.

## V. Conclusion

In this paper, we presented a fast and easy to implement method for 3D range data segmentation. Our approach operates directly on the range images and does not need to explicitly compute the point cloud in the 3D space. This simplifies the segmentation of the individual range scans as we can exploit the neighborhoods relation given by the range image. This allows us to successfully segment even sparse laser scans like those recorded from a 16-beam Velodyne scanner. Our method exploits an efficient computation of connected components and has only one parameter, which even has a physical motivation. We implemented and evaluated our approach on different datasets and provided comparisons to other existing techniques. On a mobile i5 CPU, we obtain segmentation results at average frame rates between 116 Hz and 416 Hz and up to 667 Hz on an i7 CPU.

## Acknowledgments

We thank Jens Behley for fruitful discussions and for providing his implementation of grid-based segmentation. Further thanks to Frank Moosmann for sharing his data.

## References

[1] S.M. Abdullah, M. Awrangjeb, and G. Lu. Lidar segmentation using suitable seed points for 3d building extraction. *Intl. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(3):1, 2014.

[2] J. Behley, V. Steinhage, and A. Cremers. Laser-based segment classification using a mixture of bag-of-words. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[3] I. Bogoslavskyi, L. Spinello, W. Burgard, and C. Stachniss. Where to park? minimizing the expected time to find a parking space. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2015.

[4] L. Cabaret, L. Lacassagne, and L. Oudni. A review of world's fastest connected component labeling algorithms: Speed and energy estimation. In *In Proc. of the Intl. Conf. on Design and Architectures for Signal and Image Processing*, 2014.

[5] Y. Choe, S. Ahn, and M.J. Chung. Fast point cloud segmentation for an intelligent vehicle using sweeping 2d laser scanners. In *Proc. of the Intl. Conf. on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 38–43, 2012.

[6] A. Dewan, T. Caselitz, G.D. Tipaldi, and W. Burgard. Motion-based detection and tracking in 3d lidar scans. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2016.

[7] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel. On the segmentation of 3d lidar point clouds. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

[8] B. Douillard, J. Underwood, V. Vlaskine, A. Quadros, and S. Singh. A pipeline for the segmentation and classification of 3d point clouds. In *Proc. of the Int. Symposium on Experimental Robotics (ISER)*, 2014.

[9] F. Endres, C. Plagemann, C. Stachniss, and W. Burgard. Unsupervised discovery of object classes from range data using latent dirichlet allocation. In *Proc. of Robotics: Science and Systems (RSS)*, Seattle, WA, USA, 2009.

[10] G. Floros and B. Leibe. Joint 2d-3d temporally consistent semantic segmentation of street scenes. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2823–2830, 2012.

[11] A. Golovinskiy and T. Funkhouser. Min-cut based segmentation of point clouds. In *Proc. of the Computer Vision Workshops (ICCV Workshops)*, pages 39–46, 2009.

[12] M. Hebel and U. Stilla. Pre-classification of points and segmentation of urban objects by scan line analysis of airborne lidar data. *Intl. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 37(B3a):105–110, 2008.

[13] M. Himmelsbach, F. v Hundelshausen, and H. Wuensche. Fast segmentation of 3d point clouds for ground vehicles. In *IEEE Intelligent Vehicles Symposium*, pages 560–565, 2010.

[14] K. Klasing, D. Wollherr, and M. Buss. A clustering method for efficient segmentation of 3d laser data. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 4043–4048, 2008.

[15] D. Korchev, S. Cheng, Y. Owechko, and K. Kim. On real-time lidar data segmentation and classification. In *Proc. of the Intl. Conf. on Image Processing, Computer Vision, and Pattern Recog. (IPCV)*, 2013.

[16] C. Merfels and C. Stachniss. Pose fusion with chain pose graphs for automated driving. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[17] F. Moosmann. *Interlacing self-localization, moving object tracking and mapping for 3d range sensors*. KIT Scientific Publishing.

[18] F. Moosmann, O. Pink, and Ch. Stiller. Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion. In *Proc. of the Intelligent Vehicles Symposium*, pages 215–220, 2009.

[19] A. Petrovskaya and S. Thrun. Model based vehicle tracking for autonomous driving in urban environments. In *Proc. of Robotics: Science and Systems (RSS)*, volume 34, 2008.

[20] T. Pylvanainen, K. Roimela, R. Vedantham, J. Itaranta, and R. Grzeszczuk. Automatic alignment and multi-view segmentation of street view data using 3d shape priors. In *Proc. of the Symp. on 3D Data Processing, Visualization and Transmission (3DPVT)*, 2010.

[21] R.B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[22] D. Steinhauser, O. Ruepp, and D. Burschka. Motion segmentation and scene classification from 3d lidar data. In *Proc. of the Intelligent Vehicles Symposium*, pages 398–403, 2008.

[23] J. Strom, A. Richardson, and E. Olson. Graph-based segmentation for colored 3D laser point clouds. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[24] A. Teichman and S. Thrun. Tracking-based semi-supervised learning. In *Robotics: Science and Systems*, Los Angeles, CA, USA, 2011.

[25] J. Wang and J. Shan. Segmentation of lidar point clouds for building extraction. In *Proc. of the Anual Conf. of the American Society for Photogrammetry and Remote Senssing*, pages 9–13, 2009.

[26] K.M. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008.