2010 IEEE Intelligent Vehicles Symposium
University of California, San Diego, CA, USA
June 21-24, 2010



# Fast Segmentation of 3D Point Clouds for Ground Vehicles

M. Himmelsbach and Felix v. Hundelshausen and H.-J. Wuensche

*Abstract*— **This paper describes a fast method for segmentation of large-size long-range 3D point clouds that especially lends itself for later classification of objects. Our approach is targeted at high-speed autonomous ground robot mobility, so real-time performance of the segmentation method plays a critical role. This is especially true as segmentation is considered only a necessary preliminary for the more important task of object classification that is itself computationally very demanding. Efficiency is achieved in our approach by splitting the segmentation problem into two simpler subproblems of lower complexity: local ground plane estimation followed by fast 2D connected components labeling.**

**The method's performance is evaluated on real data acquired in different outdoor scenes, and the results are compared to those of existing methods. We show that our method requires less runtime while at the same time yielding segmentation results that are better suited for later classification of the identified objects.**

## I. INTRODUCTION

In this paper we address the problem of segmenting large-size long-range 3D scan data in a way that allows later classification of the objects detected based on their 3D points. Given the set of 3D points acquired by a range scanner, the goal of segmentation is to separate the point cloud into disjunct subsets representing individual objects. With fully three-dimensional scanners becoming available in the last years, the main challenge posed to segmentation algorithms is the vast amount of potentially unordered 3D data to be processed.

In [1], we presented a LIDAR based perception system capable of handling such data at real-time. An occupancy grid was built in a way similar to [2] and segmentation was entirely done in the 2.5D domain of this grid by finding connected components of grid cells believed to represent obstacles. With the grid acting like a hash-table for fast retrieval of 3D points, subsets of object point clouds were then extracted for each connected component found and fed into a 3D point cloud feature based classification framework to assign each object a class label.

While this system proved capable of real-time object detection and classification for the simplified task of distinguishing vehicles from other objects in the scene [1], some major drawbacks could easily be identified, the most prominent being under-segmentation of data due to the dimensionality reduction introduced by mapping 3D points to the 2.5D grid structure.

All authors are with department of Aerospace Engineering, Autonomous Systems Technology (TAS), University of the Bundeswehr Munich, Neubiberg, Germany.

Contact author email: michael.himmelsbach@unibw.de

Improving on our earlier work, the main contribution of this paper is a method for fast segmentation of long-range, unordered 3D point cloud data that overcomes the aforementioned problem of our previous work while still providing the information necessary for classification of objects. Especially, our new method keeps the full 3D information delivered by the sensor during the critical steps, making it less prone to under-segmentation. Still, the method is shown to perform in real-time given the large clouds of 100.000 points delivered by our Velodyne high-definition LIDAR sensor at a rate of 10 Hz. The method is not restricted to a particular robot or sensor, however we describe and demonstrate it using our vehicle MuCAR-3 (Munich Cognitive Autonomous Robot Car, $3^{rd}$ generation), a VW Touareg equipped with a Velodyne HDL-64 LIDAR.

Before giving a detailed description of our approach in section III, we begin by giving an outline of related work. In section IV we provide experimental results and conclude the paper in section V, giving an outlook for future research on the topic.

## II. RELATED WORK

Given the large amount of data delivered by modern 3D range scanners, the most efficient segmentation algorithms to date rely on first reducing dimensionality, achieved by projecting the 3D points to a 2.5D grid fixed to the ground. This approach was taken by many teams at the 2007 DARPA Urban Challenge robot competition in order to segment point clouds and to detect the other vehicles on the track, with great success ([3], [4], [5]). One drawback of the approach is that under-segmentation, in which points belonging to different objects are merged in the same segment, frequently occurs. This is because the grid only provides a 2.5D representation of the environment but does not model free vertical space between any pair of points. For example, treetops and the ground beneath them will thus be detected as one single object. This poses an additional difficulty to object classification, e.g. as in [1], as ground plane points are then common to otherwise different kinds of objects.

More recently, some work addressing the segmentation problem in full 3D appeared ([6], [7], [8], [9]). Among these, [6], [7] share some similarities in that both use the characteristics of the corresponding scanning device to establish neighborhood relations between points. This neighborhood is then used to extract local point features from estimated point normals. Then, Euclidean clustering of points is carried out, taking into account smoothness constraints derived from these features. Performance of [6] is reported to be close to, but not up to real-time, whereas in [7] real-time performance

978-1-4244-7868-2/10/$26.00 ©2010 IEEE

of their method is shown, which however depends on "sensor heuristics" special to their setup that do not apply in our case.

Anguelov *et al.* [8] apply machine learning techniques to the problem of point cloud segmentation and report good results. In a supervised learning framework, a Markov Random Field (MRF) is trained to label points with different class labels based on simple point features. However, inferring point labels from this kind of MRF presently can not be done in real-time for the large amount of points we are concerned with.

The approach most similar to our work is [9]. Like in our approach, local line fits are used to model the ground plane and the points labelled accordingly. Points labeled as non-ground are then clustered making use of euclidean point distances in a way similar to [10]. However, a rather coarse definition of locality is used resulting in some important information being discarded by the estimation process, especially at large ranges. Moreover, the method is not generally applicable as it is specific to a certain type of scanning device.

With the work presented in this paper, we put forth the idea of doing segmentation of point clouds in full 3D, presenting a general method that achieves real-time performance without imposing restrictions on the specific 3D scanner setup.

## III. FAST 3D POINT CLOUD SEGMENTATION

Instead of establishing complex neighborhood relations and extracting sophisticated point features as commonly done in point cloud segmentation, in our method we partition the data in a way that allows ground plane points to be estimated by simple comparisons to local line fits. Together, the partitioning and line fits can be considered an extension of the line extraction algorithms popular for 2D range data processing [11] to the domain of 3D point cloud data. In particular, in our work the fitted lines serve the additional purpose of identifying ground points. Grouping points labeled as not belonging to the ground is then done by mapping all non-ground points to a high-resolution grid structure imposed on the $xy$-plane and applying fast connected component algorithms to this grid.

We begin the description of the proposed method with some notes on data acquisition and preprocessing and go into the details of ground plane estimation afterwards. We finally show how to group non-ground points into disjunct subsets to obtain segmentation results suitable for later object classification.

### A. Data Acquisition And Preprocessing

LIDAR data arrives in small packets of 384 range and bearing measurements each at a rate of $\approx$ 2.5 kHz via 100 MBit Ethernet. These packets are collected for the time of one sensor revolution in what we call a scan. Before running our algorithms we first inertially correct the LIDAR scan, taking the vehicle's motion into account (exploiting an INS and odometric information). Formally, a scan at time $t$ will be denoted by the unordered set $P_t = \{p_1, \ldots, p_{N_t}\}$ with 3D points $p_i = \begin{pmatrix} x_i & y_i & z_i \end{pmatrix}^T$ given by their euclidean coordinates wrt. to the ego-coordinate system with origin at the robot's center of gravity.

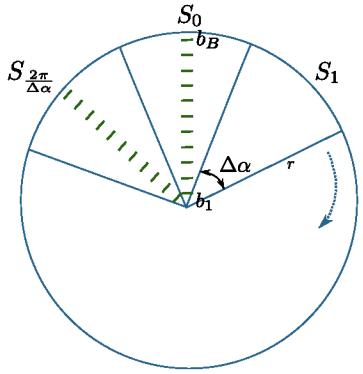



Fig. 1. Partitioning the 3D space into (small) segments of equal size.

### B. Local Ground Plane Estimation

A reasonable first step for segmentation of long-range 3D point clouds in outdoor environments is to establish a binary labeling of all points indicating whether a point belongs to the ground plane or not. As we're targeting outdoor scenarios, the ground plane model must be suited to describe both flat and sloped terrain as well as transitions between both. To capture such transitions, local surface properties need to be considered. This induces the need for some kind of neighboring structure being imposed on the unordered point cloud data, commonly done by utilizing special sensor characteristics [6], [9] or, more generally, by building static kd-trees [10]. While the former is not general in nature, the latter does not permit real-time use.

Instead, an efficient partitioning of data can be achieved by representing the $xy$-plane as a circle of infinite radius $r = \infty$ and dividing it into a discrete number of segments, as shown in figure 1. To account for specific sensor properties, we introduce the parameter $\Delta\alpha$ that describes the angle every segment covers. Thus, we come up with $M = \frac{2\pi}{\Delta\alpha}$ segments $S_i$. The index to a segment a point maps to is denoted by $segment(p_i)$ and is easily calculated to be

$$segment(p_i) = \frac{atan2(y_i, x_i)}{\Delta\alpha} \tag{1}$$

where $atan2(y, x)$ gives the the angle between the positive x-axis of the plane and the point $(x, y)$ and is ensured to be within $[0, 2\pi)$. We denote the set of all points mapped to the same segment $S_s$ by $P_s$,

$$P_s = \{p_i \in P | segment(p_i) = s\} \tag{2}$$

However, this only gives us an ordering of points wrt. their angular component. Especially, points are still 3D and so we cannot apply fast 2D line extraction, as is our intention.

To arrive at an ordering of points suitable for ground plane estimation, we introduce a mapping of all points $P_s$ of the same segment to one of many bins $b_j^s, j = 1..B$ discretizing the range component of the points. We denote the minimum and maximum range such a bin covers by $r_j^{min}$ and $r_j^{max}$,

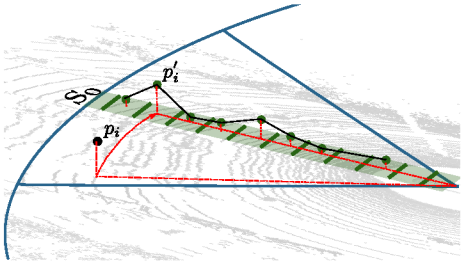



Fig. 2. Mapping of 3D points $p$ to a bin of the corresponding segment and resulting mapped point $p'$.

resp. Formally, a point $p_i \in P_s$ then maps to bin $b_j^s$ iff $r_j^{min} \leq \sqrt{x_i^2 + y_i^2} < r_j^{max}$, and we use $P_{b_j^s}$ to denote the set of all points mapping to $b_j^s$ this way. The same binning is used for all segments, thus $b_j^s$ and $b_i^{q \neq s}$ do not differ in the range they cover but only in the points mapped to them. This is illustrated in figure 2 for one arbitrarily chosen segment.

Once all points have been mapped to a segment and a corresponding segment bin, converting the points to 2D is then straightforward. Given a set $P_{b_j^s}$ of 3D points mapped to the same bin, we simply define a new set of 2D points $P'_{b_j^s}$ as

$$P'_{b_j^s} = \{p'_i = \begin{pmatrix} \sqrt{x_i^2 + y_i^2} & z_i \end{pmatrix}^T | p_i \in P_{b_j^s}\} \quad (3)$$

where the points are now ordered wrt. ascending range. Obviously, the point-to-bin mapping is not one-to-one, and $P'_{b_j^s}$ may contain more than one point, or no points at all. Thus, a so-called prototype point $p'_{b_j^s}$ is computed for every non-empty bin points $P'_{b_j^s}$. There are several possibilities of how to calculate the prototype points, for example by taking the mean of all bin points. We found that using the point with lowest $z$-coordinate yields good results as it is most likely to lie on the ground plane.

Summarizing, the prototype points $p'_{b_j^s}$ provide a reasonable reduction of the original 3D points suitable for 2D line extraction. Most importantly, the complexity of ground plane estimation by line extraction will not depend on the size of the point cloud $P$ but only on the parameters $\Delta\alpha$ and the number of bins $B$. Thus, one gains a greater flexibility in controlling the run-time of the algorithm.

Ground plane estimation now proceeds with extracting lines from the prototype sets of every segment. In principle, any of the algorithms outlined in [11] can be applied. However, as prototype points are already ordered, we can directly apply the *Incremental Algorithm* known for its simplicity and effectiveness. It only remains to formulate necessary conditions for a line $y = mx + b$ to be considered part of the ground plane:

- The line's slope $m$ must not exceed a certain threshold $T_m$, i.e. the ground plane should not show vertical structure.
- For small slopes $m < T_{m_{small}}$, the line's absolute $y$-intercept $b$ must not exceed a certain threshold $T_b$. This way, plateaus are excluded from the ground plane.

**Algorithm 1** Extraction of lines for one segment $S_s$

1: $L_s = \emptyset, c = 0, P_l = \emptyset$
2: **for** $i = 1$ to $B$ **do**
3:  **if** $P'_{b_i^s} \neq \emptyset$ **then**
4:  **if** $|P_l| \geq 2$ **then**
5:  $(m_c, b_c) = fitline(P_l \cup p'_{b_i^s})$
6:  **if** $|m_c| \leq T_m \wedge (m_c > T_{m_{small}} \vee |b_c| \leq T_b) \wedge fiterror(m_c, b_c, P_l \cup p'_{b_i^s}) \leq T_{RMSE}$ **then**
7:  $P_l = P_l \cup p'_{b_i^s}$
8:  **else**
9:  $(m_c, b_c) = fitline(P_l)$
10:  $L_s = L_s \cup \{(m_c, b_c)\}$
11:  $c = c + 1$
12:  $P_l = \emptyset$
13:  $i = i - 1$
14:  **else**
15:  **if** $distpointline(p'_{b_i^s}, m_{c-1}, b_{c-1}) \leq T_{d_{prev}} \vee c = 0 \vee P_l \neq \emptyset$ **then**
16:  $P_l = P_l \cup p'_{b_i^s}$

- The root mean square error of the fit must not exceed a certain threshold $T_{RMSE}$
- The distance of the first point of a line to the line previously fitted must not exceed $T_{d_{prev}}$, enforcing smooth transitions between pairs of successive lines.

The method for extracting ground plane lines $L_s = \{(m_i, b_i)\}$ for one segment is summarized in algorithm 1. For all line fits (lines 5,9), we use the common fitting method of *total-least-squares* (TLS) as described in [11].

With the lines $L_s$ describing the ground plane within one segment $S_s$ at hand, we may now label points as belonging to the ground plane or not. To this end, we find for each point in $P_s$ the line in $L_s$ closest to it in terms of the minimum distance to one of the line's endpoints. At large ranges, it happens that this closest line is far away from the point and can't be taken to provide a reasonable estimate of the ground beneath the point. In this case, we take the conservative view and label the point as non-ground. Otherwise, we evaluate the distance between the point and the line and label the point as ground if this distance is below a threshold $T_{d_{ground}}$ and non-ground if it is not.

### C. Segmentation of Non-Ground Points

Having labelled all points of a scan, the final step is to group close non-ground points into coherent objects. In experiments, achieving this by simple 3D clustering, e.g. as in [10], showed to violate the real-time requirements. We thus resort to the aforementioned grid technique in which fast connected components known from computer vision are applied to a binary 2D occupancy grid structure fixed to the ground.

As in [1], all non-ground points are mapped to grid cells $C$, rendering some cells of the grid to be occupied while leaving others untouched. Point cloud representations of

individual objects are then found by looking up the 3D points stored at connected components of occupied grid cells.

Note that even though this relies on an intermediate reduction of dimensionality, the problem of under-segmentation common to occupancy grids is not present in our approach. This is because non-ground points have already been separated from ground points in the previous step and cell occupancies can be determined based on the non-ground points only. Further, as a single non-ground point already renders a cell occupied, objects can be detected at larger ranges.

### D. 3D Voxel Grid Segmentation

Still, if a non-ground object is placed beneath another non-ground object, e.g. a car beneath a tree, the method as described so far would result in under-segmentation, assigning both the car and the tree to the same segment. We thus need to detect such rare situations, and refine segmentation results by performing final segmentation in full 3D, e.g. in a voxel grid. However, simply applying final 3D segmentation to all segments is not appropriate as this would violate real-time requirements. So we need to detect segments that need special 3D treatment in some way to not have to process all segments in this final step. Fortunately, we still know what points of a segment map to a single grid cell. Thus, in order to decide what segments need to be treated in full 3D, we propose to have a look at the $z$-coordinate differences encountered at each cell $c \in C_i$ of a segmented object $O_i$. Let $P_c^{O_i}$ denote all non-ground points of segment $O_i$ mapped to cell $c \in C_i$. We then propose to perform 3D voxel grid segmentation of all points $P_{c \in C_i}^{O_i}$ if the distance $|p_{i_z} - p_{j_z}|$ between any pair of points $p_i, p_j \in P_c^{O_i}$ such that

$$p_{i_z} \leq p_{j_z} \wedge \nexists p_k \in P_c^{O_i} : p_{i_z} < p_{k_z} \leq p_{j_z} \tag{4}$$

is above a threshold $T_{3D}$, $|p_{i_z} - p_{j_z}| > T_{3D}$, for at least $N_{3D}$ cells $c \in C_i$. Literally, we perform 3D segmentation of a segment's points if at a few cells $c$ of a segmented object $O$ there is a large gap between any two points $z$-coordinates that no third point falls into. 3D voxel grid segmentation is then straight forward and proceeds similar to the 2D occupancy grid segmentation as described above. Results of performing final 3D voxel grid segmentation with $N_{3D} = 2$ and $T_{3D} = 0.4$ are shown in figure 3.

## IV. RESULTS

The main benefit of performing segmentation of 3D point clouds is to enable detection and tracking of smaller, dynamic objects in the scene, as this allows to accurately model the dynamic aspects of the scene and thus finally facilitates safe driving. Larger objects like vegetation and large buildings are of minor interest to the navigation task and are more efficiently modeled by simple 2D occupancy grids. As all our algorithms are applied to autonomous vehicle systems online, we are very concerned about real-time performance of segmentation, especially as there's a classification of detected segments to follow (e.g. as in [1]). While performing final 3D segmentation as described in section III-D still works

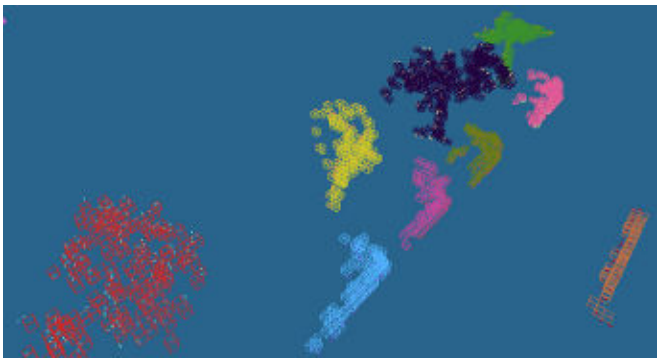


Fig. 3. Results of refining the segmentation obtained by our method with a final 3D grid segmentation. The cars are clearly separated from the trees above them.

in real-time if larger objects like vegetation and buildings are discarded at the previous steps of processing, it slightly misses real-time performance in a general setup where large objects (typically vegetation or buildings) are included. For generality, the final step of 3D voxel grid segmentation as described in section III-D has thus been omitted in the results presented in the following.

We have evaluated the proposed algorithm for numerous scans acquired while manually driving our robot in inner city traffic and countryside scenes. As no ground truth information is available, a qualitative performance evaluation is conducted. The parameters were fixed to $\Delta\alpha = 0.5\,\text{deg}$ and $B = 300$ throughout all experiments. The minimum range of the bin closest to the vehicle (i.e. that representing shortest range) was set to $r_1^{min} = 3\,\text{m}$ and the maximum range of the farthest bin to $r_{300}^{max} = 120\,\text{m}$, with range covered by the bins increasing from $0.05\,\text{m}$ to $2\,\text{m}$ to account for the many points in the vicinity of the vehicle.

An example result of ground plane labeling can be seen in figure 4. As can be seen, transitions between flat and non-flat terrain are well handled by our ground plane estimation method.

Figure 6 shows the results of our segmentation method in various scenes compared to those obtained with our previous method described in [1]. In all scenes, usually all traffic participants are clearly segmented from the ground. A benefit of our new method over pure projection based methods can be seen in the detail view included in the comparison shown in figure 6(b). Here, point measurements on the roof of other vehicles are often not supported by a second point projected onto the same grid cell, and the cell will be ignored when building connected components of grid cells. Hence, when using pure projection based methods, such points will be missing in the point cloud representation extracted for a connected component (see detail views of figures 6(b) and 6(c)), hardening classification of such objects based on their point cloud. Large buildings (purple walls in figure 6(c)) and dense vegetation (colored green right to the car in figure 6(c)) still pose some problems, for both methods (and others, as reported in [6]). Vegetation is in general very difficult to deal

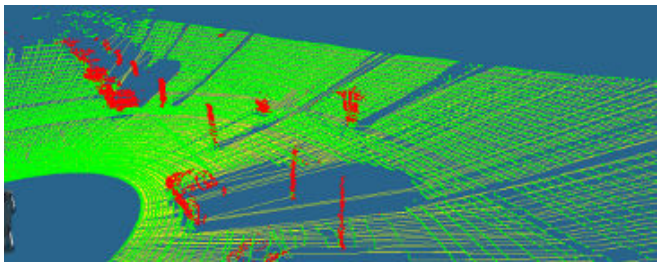


Fig. 4. Ground plane estimation by local line fits in sloped terrain. Note that objects placed on the hill are still separated from the ground.

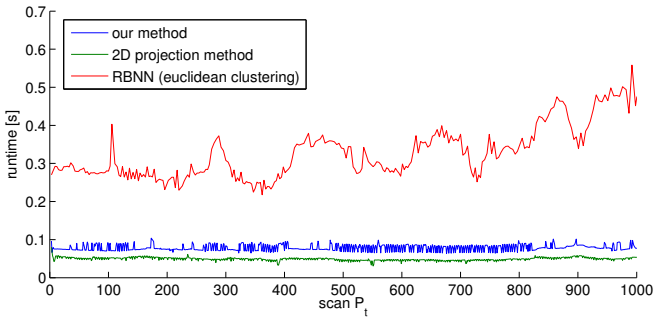



Fig. 5. Runtimes of our method (*blue*) compared to those of [10] (RBNN, *red*) and [1] (*green*) for 1000 scans on an Intel Core 2 Quad at 2.40GHz.

with, as range measurements are extremely noisy thereon. Generally, large buildings tend to be over-segmented, while vegetation often is under-segmented. Still, the problem is more severe for the projection based method, as sparse measurements at larger distances tend to render some cells unoccupied, thus breaking their connected structure. This case is shown in figure 6(a) for the wall in the scene's background.

Also shown are some challenging situations. In figures 6(a) and 6(d) one can see that, with our new method, under-segmentation of ground and non-ground points does not happen for overhanging structures, like tree branches. Instead, even the thin vertical structure shown in figure 6(d) is clearly separated from the ground by our method. On the contrary, the compared projection based method again fails in this case. Also, the results in figure 6(c) show that segmentation of objects close to each other, like the cars parking in the street going right, is possible with our approach. Finally, figure 5 shows that with an average of $0.075\,\mathrm{s}$ per run our method performs in real-time and much faster than 3D clustering approaches. Our method has runtimes comparable to 2.5D projection methods but does not share their problem of under-segmentation of data.

## V. CONCLUSIONS AND FUTURE WORK

We presented a fast novel algorithm for segmentation of large-size long-range 3D point clouds. At the core of the algorithm is a fast method for estimating local ground plane by applying 2D line extraction algorithms to the domain of unorganized 3D points. Comparing single points to the obtained ground plane lines then allows reliable separation of ground from non-ground points. We showed how this labeling of points can be used to arrive at a segmentation of 3D point cloud data that overcomes the most important problem of under-segmentation inherent to other segmentation methods based on data projection. We demonstrated that our method achieves good segmentation results on data acquired in a variety of scenarios, including flat and non-flat, sloped terrain. Finally, the method was shown to perform in real-time. Together with the good segmentation results, our method thus provides the necessary first step towards real-time object classification in 3D point clouds.

Future work will consist of integrating the new segmentation method into the object classification framework presented in [1]. It will be interesting to see how object classification will benefit given the new segmentation results.

## VI. ACKNOWLEDGMENTS

The authors gratefully acknowledge funding by German cluster of excellence CoTeSys, Cognition for Technical Systems (also see *http://www.cotesys.org*).

## REFERENCES

[1] M. Himmelsbach, T. Luettel, and H.-J. Wuensche, "Real-Time Object Classification in 3D Point Clouds Using Point Feature Histograms," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, St. Louis, USA, October 2009.

[2] S. Thrun, M. Montemerlo, and A. Aron, "Probabilistic Terrain Analysis For High-Speed Desert Driving," in *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

[3] C. Urmson et. al., "Autonomous Driving In Urban Environments: Boss and the Urban Challenge," *Journal of Field Robotics Special Issue on the 2007 DARPA Urban Challenge, Part I*, vol. 25, no. 1, pp. 425–466, June 2009.

[4] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, "Junior: The Stanford Entry in the Urban Challenge," *Journal of Field Robotics*, 2008.

[5] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schröder, M. Thuy, M. Goebl, F. von Hundelshausen, O. Pink, C. Frese, and C. Stiller, "Team AnnieWAY's autonomous system for the DARPA Urban Challenge 2007," *International Journal of Field Robotics Research*, 2008.

[6] F. Moosmann, O. Pink, and C. Stiller, "Segmentation of 3D Lidar Data in non-flat Urban Environments using a Local Convexity Criterion," in *Proceedings of the IEEE Intelligent Vehicles Symposium, IV 09*, Xi'an, China, June 2009.

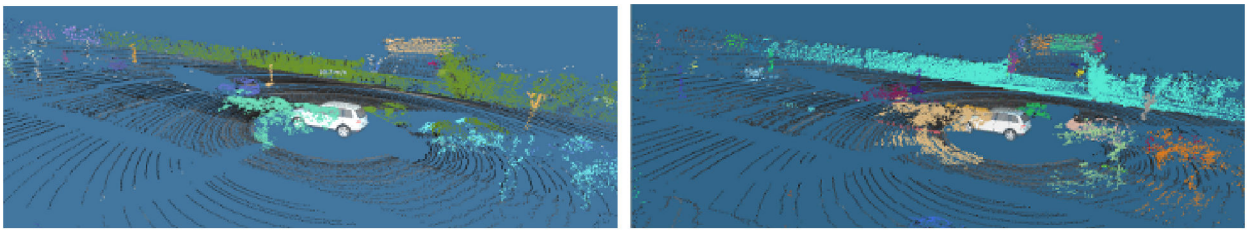
[7] K. Klasing, D. Wollherr, and M. Buss, "Realtime Segmentation of Range Data Using Continuous Nearest Neighbors," in *Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, 2009.

[8] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng, "Discriminative Learning of Markov Random Fields for Segmentation of 3D Scan Data," in *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 169–176.

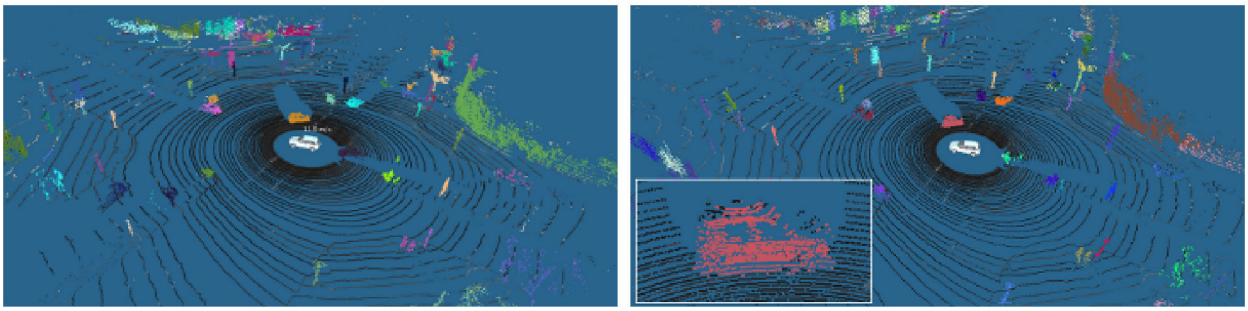
[9] D. Steinhauser, O. Ruepp, and D. Burschka, "Motion segmentation and scene classification from 3D LIDAR data," in *Intelligent Vehicles Symposium, 2008 IEEE*, June 2008, pp. 398–403.

[10] K. Klasing, D. Wollherr, and M. Buss, "A clustering method for efficient segmentation of 3D laser data," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, May 2008, pp. 4043–4048.

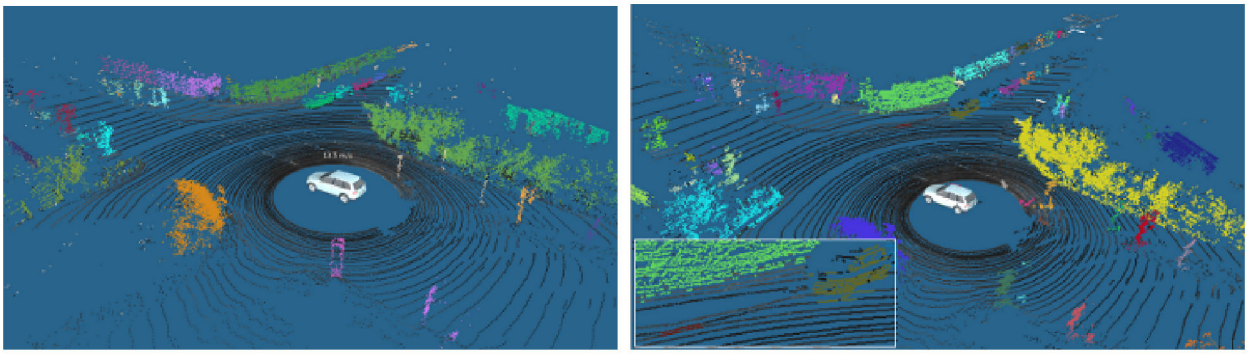
[11] V. Nguyen, S. Gächter, A. Martinelli, N. Tomatis, and R. Siegwart, "A comparison of line extraction algorithms using 2D range data for indoor mobile robotics," *Auton. Robots*, vol. 23, no. 2, pp. 97–111, 2007.



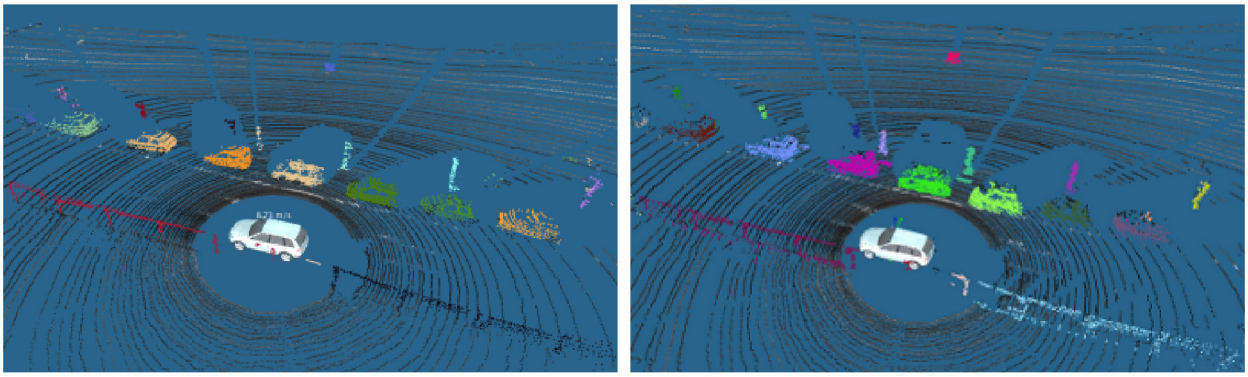
(a) Compared to projection based methods, overhanging tree branches are clearly separated from the ground by our method.



(b) A typical urban traffic scene showing other vehicles, traffic signs and signals, tree trunks and other vegetation detected by our method. The problems of projection based methods are illustrated in the detail view and are explained in the main text.



(c) Another traffic scene with some vehicles. Even though vehicles are parking close to each other, our method is still able to separate them.



(d) In contrast to projection based methods, our new method can also cope with thin vertical structures.

Fig. 6. Segmentation results of our method (left column) for a variety of urban traffic scenes, compared to the results of the projection based method of [1] (right column). Points extracted for connected components found in the grid are randomly colored according to their connected component identity, ground points are colored in gray scales according to their intensity. See main text for details.