3D Ground Point Classification for Automotive Scenarios

Julia Nitsch[1,2], Julio Aguilar[1], Juan Nieto[2], Roland Siegwart[2], Max Schmidt[1] and Cesar Cadena[2]

***Abstract*— Autonomous driving applications must be provided with information about other road users and road side infrastructure by object detection modules. These modules often process point clouds sensed by light detection and ranging (LiDAR) sensors. Within the captured point cloud a large amount of points correspond to physical locations on the ground. These points do not hold information about road users, obstacles or road side infrastructure. Thus an important preprocessing step is identifying ground points to allow the object detection focusing on relevant measurements only.**
Within this paper we propose a ground point classification which relies on simple but effective geometric features. We evaluate the accuracy of the proposed algorithm on simulated data of different traffic scenarios. In addition, we evaluate the effectiveness of this preprocessing step based on the achieved speed up of an object detection algorithm on real world data.

I. Introduction

Object detection modules are crucial for fully autonomous driving applications, because they provide information about other road users like their poses and driving directions as well as information about road side infrastructure. For object detection modules LiDAR sensors are a suitable sensor choice [1]–[5]. Their accurate distance and long range measurements enable object detections up to 200m [6]. LiDAR sensors typically measure the distance to objects using a time of flight measurement principle and generate a point cloud representation of the environment. In automotive applications a high amount of these points correspond to physical locations on the ground. These ground points are usually of no interest for object detection algorithms since they do not hold information about other road users, obstacles or road side infrastructure. Actually if ground points are present in the point cloud we face two particular problems within the object detection:

- Higher computation time due to a larger input point cloud
- Potential error source for false positive objects (ghost objects)

In order to tackle these problems we present a ground point classification method, which is based on geometric features of the point cloud. Ground point classification is an extensively discussed topic in literature, hence the contributions of this work are explicitly stated:

- We propose a weighted normal vector estimation based on the inverse of each point neighbor distance. This metric takes the downwards tilted mounting position of LiDAR sensors in automotive applications into account.
- We propose a simple but effective per point *ground candidate quality measurement*. This is a good indicator for a point belonging to a ground surface.
- We show experimental results not only in usual traffic scenarios, but also on steep terrain as well as on different height levels of ground planes.

This paper is organized as follows. First we discuss related work on ground classification methods in Sec. II. Next the geometric feature computation is described in Sec. III-A and the ground point classification based on these features in Sec. III-B. In Sec. IV we show experimental results in different traffic scenarios and show that the ground point removal speeds up the object detection module. Finally our results are discussed in Sec. V.

II. Related Work

There are a number of approaches where ground points are filtered from the input point cloud through fitting a plane model [7]–[9]. However ground points cannot always be modeled as plane (e.g: Fig. 4). Even in scenarios where the ground is relatively flat like on highways, a wrong plane orientation could lead to errors in far distance measurements but also to misclassifications in near range as it is shown in our evaluations in Sec. IV.

Douillard *et al.* [10] presented a ground classification algorithm which computes geometric features on a voxel representation. Furthermore this representation is segmented based on the computed features. The authors classify the segment containing the most voxels as ground, following the assumption that largest contiguous set of the observed points within the point cloud belong to the ground. Other works also compute geometric features to detect the ground points within the input point cloud [5], [11], [12]. However, within these approaches the authors explicitly exclude elevated targets from the ground point classification. Following this approach could lead to a failure to classify flat ground surfaces after a lowered surface as visualized in Fig. 1.

The authors in [13] segment a point cloud through computing convex areas based on surface normals of local point neighborhoods. A segment is classified as ground segment after applying a histogram based classification. This approach is prone to fail due to noisy input data, where it is not able to find large enough convex segments to exploit the advantage of the ground plane being one large convex segment. We build upon their idea for computing surface normals as geometric features of the local neighborhood.

[1]Ibeo Automotive Systems GmbH, {julia.nitsch, julio.aguilar, max.schmidt}@ibeo-as.com
[2]Autonomous Systems Lab, ETH Zurich, {jnieto, rsiegwart, cesarc}@ethz.ch

In comparison to this approach we are using the weighted average based on neighboring point distances for computing the mean normal.

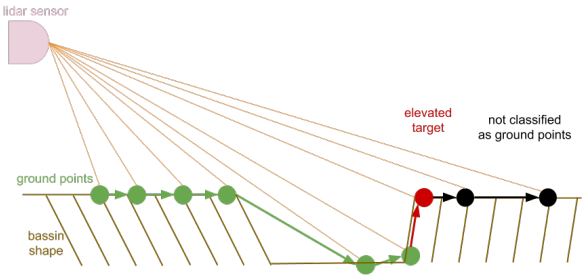


Fig. 1: A schematic basin is shown in brown. The measured point cloud by the LiDAR sensor is indicated as dots. The slopes between the single measurements are indicated as arrows following the ordering described in [12]. The measurement on the second edge of the basin is elevated compared to the last measurement within the basin. But the subsequent measurements are at the same height as the elevated target and thus not considered to be ground points. For further implementation details please see [12].

III. Methodology

Within this section we describe the proposed ground point [1] classification. We start by discussing the *Local Feature Extraction* (see Sec. III-A) which computes local geometric features and feature qualities for each point individually. These features are used to identify possible ground point candidates within the point cloud. In Sec. III-B *Clustering & Classification* the grouping of ground point candidates to identify ground surfaces is described.

Please note that we expect an ordered point cloud as input for the *Local Feature Extraction* represented in ego vehicle coordinates (see Fig. 2). Ordered point cloud means that neighbors are directly accessible for each point within the point cloud. If the LiDAR sensor does not provide this information natively, it can be easily computed from the physical sensor layout and the timestamps from each ray.

A. Local Feature Extraction

Within the *Local Feature Extraction* three local features are computed for each point:

1) Surface normal estimate n
2) Surface normal quality measurement n
3) Ground candidate quality measurement g

The surface normal estimate n of the current measurement point P is calculated based on surface normals between adjacent neighboring points (see Fig. 3). The direction vector d_i is given as $d_i = P_i \quad P$. The normals n_i for each pair are calculated as follows for the given neighborhood enumeration from Figure 3:

$$n_i = \begin{cases} d_i \quad d_{i-1}; & \text{if } 2 \quad i \quad 8 \\ d_1 \quad d_8; & \text{otherwise} \end{cases} \tag{1}$$

Following this order scheme resulting normals of flat horizontal surfaces are pointing in the direction $[0;0;1]^T$ which is also visualized in Figure 2. Once all n_i are computed the surface normal n is computed as weighted average using the inverse distance of the neighboring points:

$$n = \frac{\sum_{i=1}^{8} n_i \quad \frac{1}{|d_i|+|d_{i-1}|}}{\left|\sum_{i=1}^{8} n_i \quad \frac{1}{|d_i|+|d_{i-1}|}\right|} \tag{2}$$

Using this weighting scheme lets n rely more on physically closer points than points which are further away. Within an automotive setup lidar sensors (independent from its actual mounting position) are usually tilted towards the ground which is demonstrated in Fig. 2. This leads to different distances for neighboring points unlike the schematic representation in Fig. 3. Furthermore this weighting aims to reduce the influence of noise (e.g: one neighboring point is meters off from the other points due to reflections). In contrast to Moosmann *et al.* [13], we do not want to use a fixed distance threshold to discard far away neighboring points from P. This threshold would strongly depend on the mounting position and on the pitch angle of the sensor as well as the measured distance to P.

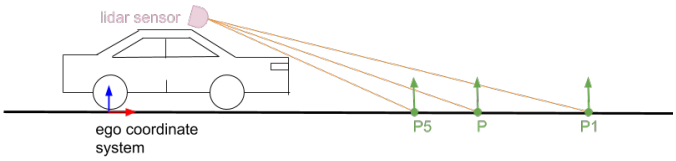


Fig. 2: The ego vehicle coordinate system is visualized on the bottom of the rear axis. As example setup the lidar sensor is mounted on the roof top and measures currently ground points P, P_5 and P_1. The point numbering corresponds to the same neighbor numbering scheme as in Fig. 3. For each point the computed surface normal is pointing in upwards direction of the ego vehicle coordinate system. These normals are visualized as green arrows.

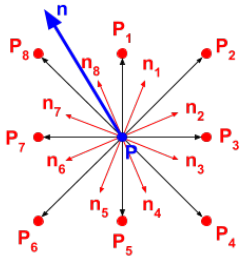


Fig. 3: The enumeration of points within the 8-neighborhood of the current point P is shown and the resulting normals n_1 to n_8 between adjacent neighbors. The *surface normal estimation* n for P is computed based on the weighted average of n_1 to n_8.

[1] We define ground points as points belonging to flat surfaces e.g. on the street or beneath the street which do not represent other objects of interest like road users or road side infrastructure.

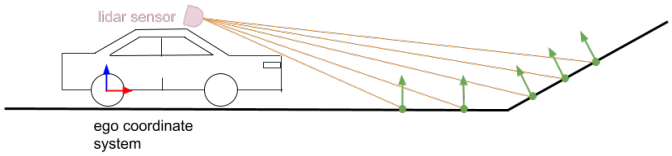


Fig. 4: This scenario shows the ego vehicle approaching a ramp and measuring ground points on the flat surface as well as on the ramp. The surface normals are visualized as green arrows and the difference of the direction of the surface normals on the flat surface compared to the surface normals on the ramp is clearly visible.

The second local feature n , namely the quality of the surface normal, is calculated based on the cosine similarity between n and its generating normals n_i (see Eq. 3). The more similar the direction of the mean normal is to each generating normal the higher is the quality measure. The quality decreases when the generating normals point towards other directions (e.g: on edges).

$$n = \frac{1}{8} \sum_{i=1}^{8} \left(\frac{< n; n_i >}{jnj \; jn_ij} \right) \tag{3}$$

Finally the third local feature g is computed. Since the point cloud is given in vehicle coordinates and we assume the ego vehicle to be on the ground the quality is calculated as follows:

$$g = n \; [0; 0; 1]^T \; n \tag{4}$$

One could argue that our approach would fail on steep terrain, because surface normals are compared with an upwards pointing vector. Since the point cloud is represented in ego vehicle coordinates and the ego vehicle is also on this steep surface, the computed surface normal has again an upwards direction. However this assumption does not hold for the scenario shown in Fig. 4 where the ego vehicle is approaching a ramp like it is the usual case in a car park. In this special case the assumption for ground surface normals pointing in an upwards direction is violated which leads to a lower g compared to the points on the flat surface. Nevertheless compared to vertical surfaces the normal quality n is higher on ramps. Considering this, the ramp scenario should be clearly separable from real obstacles. Nevertheless we consider this ramp scenario as special challenging case and therefore we show our performance explicitly in the experiments in Sec. IV on a parking garage ramp.

B. Clustering and Classification

As described in III-A the *Local Feature Extraction* assigns high g to points with upwards pointing normal vectors. Obviously high g values are not only assigned to ground points but to any points on flat horizontal surfaces. In order to discard non ground points the selected candidates get grouped into larger clusters based on their distance and g . Therefore the conditional Euclidean cluster method implemented in the point cloud library (PCL) [14] is used. Similar to [10] we assume that ground is represented by the largest cluster in the selected area. Thus, if smaller clusters fall into the same area as the identified ground cluster they are classified as non ground points. Differently to [10], other large clusters, which are outside of the ground area, are also treated as ground points. Very small clusters are identified as noise and therefore classified as non ground points.

IV. Experimental Evaluation

Within this section we evaluate the proposed ground point classification. First we show quantitative accuracy results on simulated data in Sec IV-A of the presented algorithm compared to a baseline plane fitting approach. Then we show the beneficial effect on the run time of an object tracking algorithm in Sec IV-B after filtering out ground points. Furthermore, in Sec. IV-C we show qualitative results on two challenging scenarios.

A. Simulated Data

We compare the accuracy of our presented algorithm on the Virtual KITTI Dataset [15] to a RANSAC [16] plane fitting algorithm implemented in [14]. Within this dataset the label $Terrain$ and $Road$ are considered for representing ground points. In particular we evaluate the algorithms on an urban scenario from dataset 0001 and on the rural scenario 0020. Furthermore we compare the accuracy of our presented algorithm to a ground representation where we accumulated the described features over time in a multi layer grid map representation implemented in [17]. Within this representation a point is classified as ground point if it is within 15cm of the corresponding ground grid patch. Further we compare the presented algorithm to only using the ground quality g as classification criteria, the clustering step is left out as an ablation study.

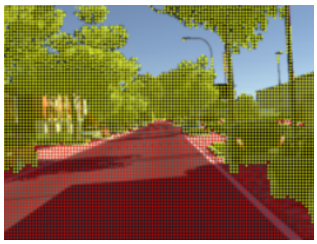
In Tab. I we state the overall accuracy (acc) Eq. 5 for each algorithm and the false positive rate (FPR) Eq. 6, which can be understood as the probability for a false detection.

$$acc = \frac{TruePositive + TrueNegative}{\#points} \tag{5}$$

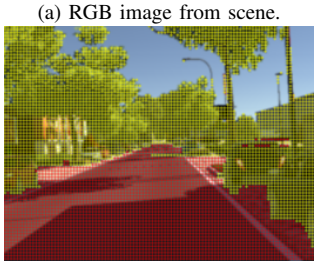
$$FPR = \frac{FalsePositive}{FalsePositive + TrueNegative} \tag{6}$$

As it can be seen in Fig. 5 only using g leads to misclassification on flat surfaces like on car roof tops. However the overall accuracy is not really affected by these points. This supports our hypothesis, that this simple ground quality measure is a good indicator for a point belonging to a ground surface. Nevertheless, the advantage of a cluster can be seen in FPR values where the proposed algorithm including the *Clustering and Classification* outperforms the classification on g . Furthermore the result shows that the proposed algorithm outperforms the plane fitting on simulated data within the classification accuracy. The plane fitting algorithm misaligns the plane in some frames as shown in Fig. 6. However within the city scenario (0001) the plane fitting algorithm has a lower FPR value than the proposed algorithm, because plane misalignment leads to only a few misclassified points in this special scenario. Comparing the

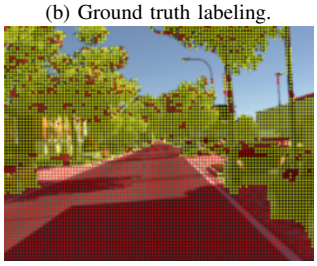
(a) RGB image from scene.



(b) Ground truth labeling.



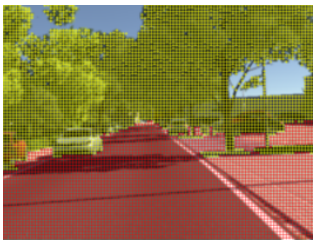
(c) Presented algorithm.



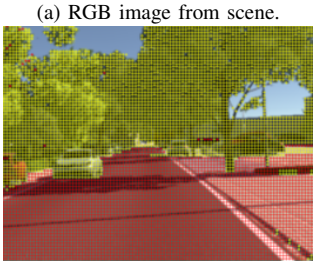
(d) No clustering of ground candidates.

Fig. 5: Classification of ground points using the presented algorithm with clusters compared to using g only as classification criteria. Yellow dots represent non ground points and red dots represent ground points.

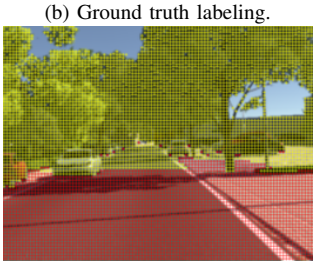
(a) RGB image from scene.



(b) Ground truth labeling.



(c) Presented algorithm.



(d) Plane fitting algorithm

Fig. 6: Simulated city scenario where the plane fitting algorithm misaligns the ground plane estimation. Note the misclassified points at the further away part of the road. Yellow dots represent non ground points and red dots represent ground points.

TABLE I: Comparison of ground point classifications

virtual Kitti data set	city 0001		rural 0020	
	acc.	FPR	acc.	FPR.
Plane fitting algorithm	0.95	**0.026**	0.69	0.039
Presented algorithm (thresh. 0.6)	**0.97**	0.036	**0.93**	**0.037**
Presented algorithm (thresh. 0.7)	**0.97**	0.036	0.92	**0.037**
Grid accumulation of features (thresh. 0.7)	0.88	0.072	0.9	0.054
Presented algorithm w/o clustering (thresh. 0.6)	0.92	0.11	0.91	0.129
Presented algorithm w/o clustering (thresh. 0.7)	0.93	0.09	0.83	0.097
Grid accumulation of features w/o clustering (thresh. 0.7)	0.82	0.107	0.83	0.095

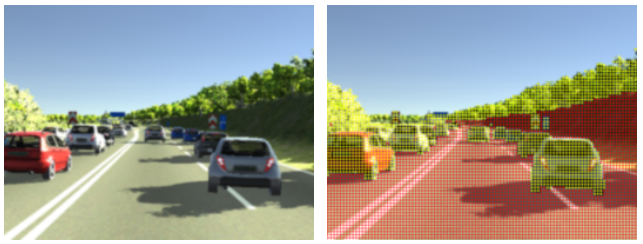
proposed algorithm to the accumulated features stored in a grid map representation there is no accuracy gain in the rural scenario (0020). One exemplary rural scene is visualized in Fig. 7 where the accumulated representation gives nearly the same classification results as the proposed algorithm. Within the city scenario (0001) we observe a drop in the overall accuracy of the grid representation by nearly 10%. One possible explanation is that geometric features on simulated data produce nearly perfect results so that accumulation which should aim to overcome noisy measurements has no positive effect but only introduces quantization errors.

B. Object Tracking Run Time after Ground Point Filtering

The presented ground point classification algorithm is evaluated on the effect on the run time of a subsequent object tracking module. This aims to support the claim that ground removal as preprocessing step is speeding up object tracking. The used tracking algorithm first clusters the point cloud, detects and, finally, tracks objects. The presented ground point classification is compared to a RANSAC plane fitting algorithm [14] and no ground classification in following scenarios:

1) Highway with three lanes in the same direction.
2) Rural road where fields and bushes are beneath the street and basins for collecting rain water.
3) Urban scene with many traffic participants.

The rural and urban scenarios provide a point cloud of four Ibeo Lux sensors, which are mounted on the rooftop facing in forward direction delivering a 16-layered point cloud. The highway scenario provides a point cloud of seven Ibeo Lux sensors delivering a 28-layered point cloud holding nearly twice as much points as within the other scenarios. Four sensors are mounted on the roof top and three are mounted on the back bumper of the car facing in backwards direction. The run time evaluation on the object tracking module of these three scenarios is visualized in Fig. 8. The mean percentage of points which are classified as ground points by each individual algorithm is visualized in Fig. 9. The presented algorithm classifies most points as ground points in all three scenarios, which further leads to the best run time of the object tracking module in all scenarios. Due to the lack of ground truth data we are unfortunately not able to report the accuracy of the tracking module. However visual inspection of the tracking results show no decline after applying the proposed ground filtering method. So we hypothesize that the run time improvement is due to the ground point filtering and not through detecting fewer objects. Within the highway scenario the plane fitting algorithm extracts nearly as many points as the presented algorithm. This leads to the suggestion that ground points on



(a) RGB image from scene. (b) Ground truth labeling.

(c) Presented algorithm. (d) Accumulated features within a grid representation

Fig. 7: Comparison of presented algorithm to accumulated features in rural scenario. Yellow dots represent non ground points and red dots represent ground points.

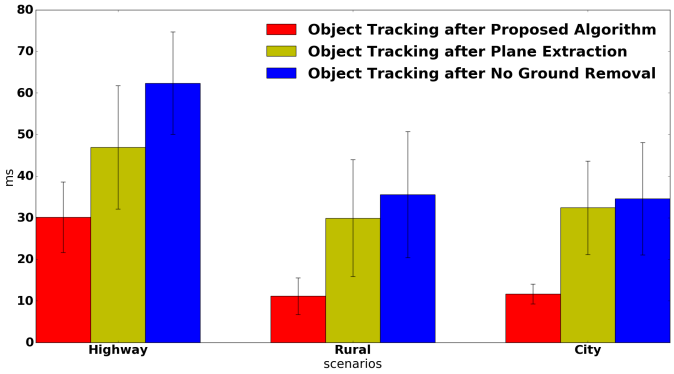


Fig. 8: Mean computation time of the object tracking module after preprocessing the point cloud with the presented algorithm, a simple plane fitting and no ground removal at all evaluated in a highway, rural and city scenario.

highway scenarios could be approximated with a plane model due to the wide street. However within the results we observe that the plane fitting algorithm sometimes tilts the ground plane as shown in Fig. 10. It can be seen that the presented algorithm is able to classify the ground points to a large extent, whereas the plane fitting has already misclassified some points in the near field. These misclassified points pose a potential risk of being detected as ghost objects. Within the city scenario the plane fitting extracts nearly as many ground points as the presented algorithm but the run time of the object tracking is nearly as long as if no ground points had been extracted. In comparison to the highway scenario not as many points represent the ground and the plane fitting algorithm misclassifies even more points.

C. Special scenarios

Additionally two special scenarios are shown namely one containing a ramp to a parking garage and one containing a

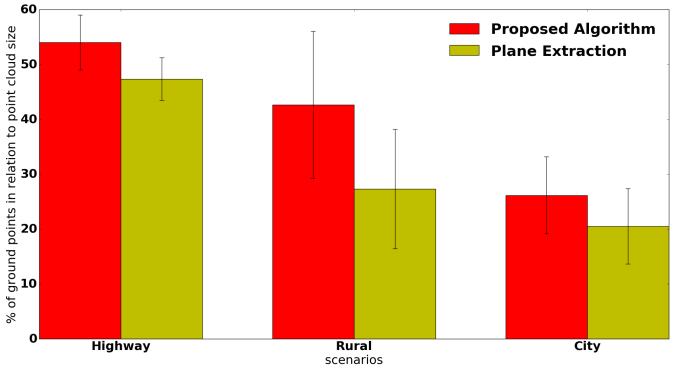
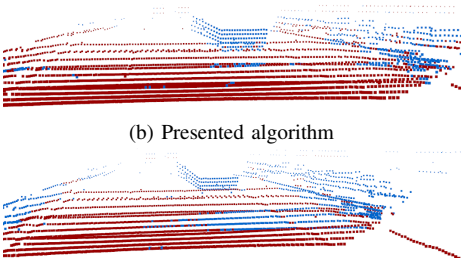


Fig. 9: Mean percentage of detected ground points within the point cloud in highway, rural and city scenario by the presented algorithm and the plane fitting algorithm. The no ground removal algorithm obviously detects zero ground points and is therefore not listed in this figure.

(a) RGB image of highway scenario



(b) Presented algorithm

(c) Classification using a fitted plane model

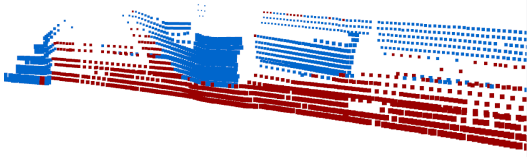
Fig. 10: The rgb image of a highway scenario is visualized in Fig. 10a. The camera faces in the same direction and captures the same scene as the lidar sensor. Dark red points indicate ground points and blue points indicate non ground points. The presented algorithm captures mainly the ground points whereas the plane fitting algorithm slightly tilts the plane which leads to misclassifications in the near range

basin next to the street. Within *the ramp* scenario it is shown that the presented algorithm is also able to detect ground points on a ramp while the ego vehicle is still driving on a flat surface (see Fig. 11). Within *the basin* scenario in Fig. 12 it can be seen, that the presented algorithm is able to detect a ground surface although a basin is next to the street (schematically shown in Fig. 1).

V. Conclusion and Future Work

Within this work we present a ground point classification approach based on geometric point features, which significantly speeds up a subsequent detection algorithm. We compare therefore the effect of no ground classification, ground plane fitting and the presented ground classification approach. We achieved a speed up of the object tracking over no ground removal within all scenarios by more than

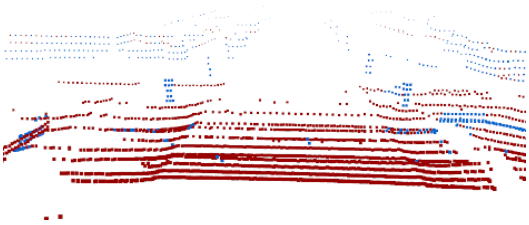
(a) RGB image



(b) Presented algorithm

Fig. 11: The rgb image of a parking garage ramp is visualized in Fig. 11a. The camera faces in the same direction and captures the same scene as the lidar sensor. Within this scenario there are two ramps. The one on the left side is leading to the upper park floors and the one on the right side is leading to the lower floors. In Fig. 11b the classified point cloud is shown. Dark red points indicate ground points and blue points indicate non ground points.

(a) RGB image



(b) Presented algorithm

Fig. 12: Classification of ground points with a basin on the right side of the street. Within Fig 12a the rgb image of the scene is visualized. The camera faces in the same direction and captures the same scene as the lidar sensor. In Fig. 12b the classified point cloud is shown. Dark red points indicate ground points and blue points indicate non ground points.

a factor of two. Using a ground point removal algorithm as preprocessing step confirmed the statement in [10] and [12] that subsequent modules' results get improved by a previously performed ground removal. Following our approach we are able to detect 97% of ground points in a simulated data urban scenario and 93% in a simulated rural scenario. Furthermore, we are able to handle steep scenarios without excluding points behind elevated targets.

In future work the clustering step could be further improved in taking environmental knowledge into account. For example, previously detected objects can be used as to distinguish clusters from ground points. Furthermore the performance of the ground point classification could be evaluated on different traffic scenarios like high density traffic and in challenging weather conditions (e.g snow covered roads).

References

[1] M. Himmelsbach, A. Mueller, T. Lüttel, and H.-J. Wünsche, "Lidar-based 3d object perception," in *Proceedings of 1st international workshop on cognition for technical systems*, vol. 1, 2008.

[2] H. Zhao, Y. Liu, X. Zhu, Y. Zhao, and H. Zha, "Scene understanding in a large dynamic environment through a laser-based sensing," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2010, pp. 127–133.

[3] T.-D. Vu, O. Aycard, and F. Tango, "Object perception for intelligent vehicle applications: A multi-sensor fusion approach," in *Intelligent Vehicles Symposium Proceedings*. IEEE, 2014, pp. 774–780.

[4] R. O. Chavez-Garcia and O. Aycard, "Multiple sensor fusion and classification for moving object detection and tracking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 525–534, 2016.

[5] J. Rieken, R. Matthaei, and M. Maurer, "Toward perception-driven urban environment modeling for automated road vehicles," in *International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2015, pp. 731–738.

[6] J. Fischer, A. Menon, A. Gorjestani, C. Shankwitz, and M. Donath, "Range sensor evaluation for use in cooperative intersection collision avoidance systems," in *Vehicular Networking Conference (VNC)*, Oct 2009, pp. 1–8.

[7] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun, "3d object proposals for accurate object class detection," in *Advances in Neural Information Processing Systems*, 2015, pp. 424–432.

[8] K. Lai and D. Fox, "3d laser scan classification using web data and domain adaptation." in *Robotics: Science and Systems*, vol. 2, 2009.

[9] X. Chen, K. Kundu, Y. Zhu, H. Ma, S. Fidler, and R. Urtasun, "3d object proposals using stereo imagery for accurate object class detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[10] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel, "On the segmentation of 3d lidar point clouds," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 2798–2805.

[11] M. Himmelsbach, F. V. Hundelshausen, and H.-J. Wuensche, "Fast segmentation of 3d point clouds for ground vehicles," in *Intelligent Vehicles Symposium (IV)*. IEEE, 2010, pp. 560–565.

[12] J. Rieken, R. Matthaei, and M. Maurer, "Benefits of using explicit ground-plane information for grid-based urban environment modeling," in *International Conference on Information Fusion (Fusion)*. IEEE, 2015, pp. 2049–2056.

[13] F. Moosmann, O. Pink, and C. Stiller, "Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion," in *Intelligent Vehicles Symposium (IV)*. IEEE, 2009, pp. 215–220.

[14] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *International Conference on Robotics and automation (ICRA)*. IEEE, 2011, pp. 1–4.

[15] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[16] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," in *Readings in computer vision*. Elsevier, 1987, pp. 726–740.

[17] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS) – The Complete Reference (Volume 1)*, A. Koubaa, Ed. Springer, 2016, ch. 5. [Online]. Available: http://www.springer.com/de/book/9783319260525