2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

Fast Multi-Pass 3D Point Segmentation Based on a Structured Mesh Graph for Ground Vehicles

Patrick Burger and Hans-Joachim Wuensche

***Abstract*— Point-cloud segmentation of 3D LiDAR scans is an important preprocessing task for autonomous vehicles in on-road and especially in off-road scenarios. Clustering point measurements with the same properties into multiple homogeneous regions is a challenging task due to an uneven sampling density and lack of explicit structural information. This paper presents a novel technique to achieve a robust and fast point-cloud segmentation using the characteristic intrinsic sensor pattern. This pattern is characterized by the mounting position of each laser diode. A structured mesh graph is created by taking the beam calibration and the chronology of incoming data packets into account. The proposed graph-based, multi-pass point segmentation algorithm compares this pattern with a flat-world model to detect discontinuities and to set label attributes such as *obstacle* or *free space* for each vertex. Furthermore, we directly detect missing measurements and therefore generate artificial vertices considering the laser beam intrinsics. Finally, a region-growing algorithm is applied in order to obtain cohesive objects. Experimental results show that we achieve a reliable overall performance and a good trade-off between segmentation quality and runtime of 15 ms in rough terrain as well as suburban areas.**

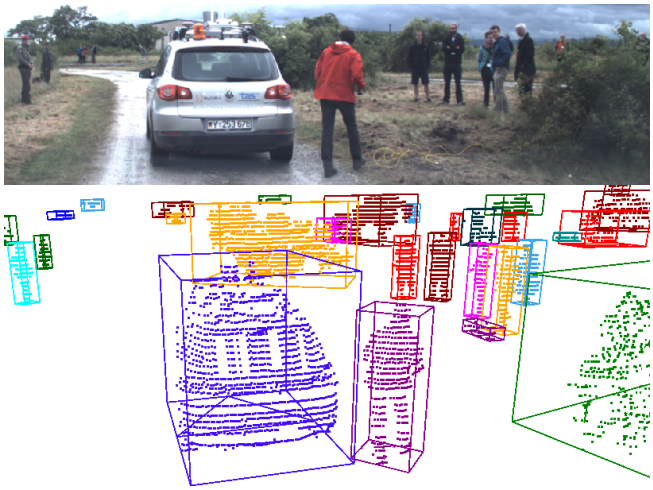


Fig. 1: Typical segmentation result of one scan in an off-road scenario [9]. Object instances (car, peoples, bushes...) are presented by a bounding box.

I. INTRODUCTION

The task of autonomous driving has received substantial attention in the last decade [1]–[8]. Fully understanding the environment is still a challenging task for autonomous vehicles [3]–[5]. Environment perception is the key component for intelligent vehicles to drive safely in urban and non-urban areas. In order to navigate successfully, demanding maneuvers are often required, which presuppose a detailed environment model [6], [7]. Furthermore, usually high-definition maps cannot be relied on, either because they are not available (off-roads) or impracticable [8].

In general, environment perception and tracking tasks have three different preprocessing steps in common. First, the incoming data is prepared and stored for further processing. Second, obstacle and ground plane points are separated. Third, points are clustered. The overall performance strongly depends on each of these preprocessing steps with continuous error propagation. In conclusion, point segmentation plays an import role but represents just a small part of the whole pipeline. Therefore, one has to find the balance between computation time and accuracy.

This paper presents a fast multi-pass algorithm for real-time segmentation of Velodyne point clouds on the basis of a mesh graph that takes all 3D measurements into account. The intrinsic sensor pattern of a Velodyne sensor is used as a prior to speed up the creation of neighborhood relations and to distinguish between different label classes.

In the first pass, key vertices are identified which are very likely to be obstacle points. In the second pass, a histogram-based multimodal distribution clustering detects additional obstacle vertices that correspond to the key vertices by taking the point density in polar coordinates into account. In the third pass, pattern matching is applied to identify vertices which deviate considerably from our flat-world pattern model. In the fourth pass, we enrich the graph by adding artificial vertices in order to overcome the problem of missing measurements. In the final pass, obstacle points are clustered using a region-growing algorithm.

This paper is organized into five sections. After discussing related work in II, a detailed description of our proposed method is given in III. In section IV, we present experimental results. Finally, we offer some conclusions and an outlook on future work in section V.

II. RELATED WORK

In the last decade, several methods addressing the environment perception task for autonomous cars have been published. On one hand, there are approaches that focus on vision exclusively [3], [10], [11]. These methods usually use color, texture features or neural networks to segment ground and object pixels. On the other hand, 3D range sensors are used to separate objects from the ground plane. Laser range scanners provide complementary data to video-based sensors

All authors are with the Institute for Autonomous Systems Technology (TAS) of the University of the Bundeswehr Munich, Neubiberg, Germany. patrick.burger@unibw.de

with accurate distance information and little interference from illumination.

In this paper, we will focus on 3D point clouds from modern LiDAR sensors. Therefore, a detailed comparison of state-of-the-art approaches will be shown in the following paragraphs. These algorithms can be separated into three subgroups.

1) Grid methods: Grid methods are the most common approaches due to their ability to reduce dimensionality and combine range measurements from different sensors in one representation. During the 2007 DARPA Urban Challenge many teams used this method to distinguish objects from the ground [2], [12]. Here, 3D measurements are projected into a 2.5D occupancy grid. Cells with a height difference below a predefined threshold are marked as ground. In general, these approaches are prone to over-segmentation [13].

Himmelsbach et al. [14] introduce a fast method by dividing a polar grid map into many segments to perform a 2D line-fit extraction on each segment. Over-segmentation is reduced by comparing each point distance to the estimated line. Furthermore, grid cells with similar heights are grouped together. Chen et al. [13] extend this approach by using a one-dimensional Gaussian-Process regression method with a non-stationary covariance function to determine ground and obstacle points for each segment. These methods are sensitive to slope changes and information is lost due to the grid discretization.

2) Range-image methods: Range image based methods play a dominant role in mobile robotics [15], [16]. Bogoslavskyi et al. [17] create a 2D range image directly from the incoming LiDAR measurements. First, ground points are removed by a method similar to [14]. Second, they apply a breadth-first search and group points together if a angle criterion is satisfied.

3) Segmentation in 3D: Klasing et al. [18] present a method for segmentation in full 3D. The sensor characteristics are used to determine the neighborhood relations between points. Then, points are clustered according to their Euclidean and angular distance. The algorithm works well for indoor scenarios and is fast (10 ms).

The most closely related work to our method is the approach from [19]. A 2D graph is constructed by using the continuous sensor rotation for subsequent horizontal point neighborhoods within one beam. The vertical relations are determined by the yaw angles and the local surface in the N4 neighborhood of each point is approximated. Next, a generic local convexity criterion is applied to identify corresponding neighborhood surfaces. Finally, a histogram-based approach over all the normal vectors' z-values are used to identify obstacle and ground points. The proposed method works well for non-urban scenarios but the average processing time of 623 ms exceeds the typical Velodyne cycle time of 100 ms by far.

This paper proceeds by introducing a novel and fast point-wise labeling method that uses a model of how the Velodyne laser beams perceives a flat plane. Furthermore, the Velodyne intrinsic sensor pattern and the continuous sensor rotation are taken into account to directly create a mesh-graph with an N4-neighborhood. Edge connections are examined for each vertex by local cohesion criteria derived from the continuously spinning sensor pattern.

III. Proposed method

Our point-segmentation method is explained in four processing steps. First, the intrinsic Velodyne sensor pattern will be presented. This pattern plays a fundamental role in the segmentation of each point. Second, the data acquisition is introduced. Third, the mesh graph construction will be outlined. Finally, the multi-pass algorithm will be described in detail.

A. Intrinsic Velodyne Pattern

Velodyne sensors are 360° LiDAR sensors. The models HDL-64, HDL-32 and VLP-16 differ in their number of laser beams (64, 32, 16) and their vertical field of view. In the following sections, our method will be explained for the HDL-64, though it is possible to use any of the other models. In contrast to a LiDAR sensor with a single laser beam and a reflection mirror, the HDL-64 contains 4 groups of 16 laser diodes with different azimuth and elevation offset angles respectively as shown in Fig. 2. This configuration creates a unique intrinsic sensor pattern $\mathbb{S} = \{\mathbf{b}_i \mid i \in \mathbb{I}\}$, where $\mathbb{I} = \{1,\ldots,64\}$ is the set of beam IDs. Each beam $\mathbf{b}_i = \begin{pmatrix} \Psi_i^{\text{off}} & \Theta_i^{\text{off}} & d_i^{\text{off}} & h_i^{\text{off}} & v_i^{\text{off}} \end{pmatrix}^T$ is described by the calibration parameters: the azimuth offset angle Ψ_i^{off}, elevation offset angle Θ_i^{off}, distance offset correction d_i^{off}, horizontal h_i^{off} and vertical v_i^{off} parallax correction. For further information, see [20]. A geometric illustration of how the pattern appears on a orthogonal plane is shown in Fig. 3.

This pattern rotates around a pivot point. Each laser beam $\mathbf{b}_i$ fires $\mu_{\max} = \lfloor t_{\text{acq}}/\Delta t_{\text{f}} \rfloor \mid \mu_{\max} \in \mathbb{N}_0$ times per revolution, where Δt_{f} denotes the time between two successive firings and t_{acq} the acquisition time for one full revolution. After 360° a accumulated sparse point cloud $\{\mathbf{S}_\mu \mid \mu \in \mathbb{U}\}$ is obtained, where $\mathbb{U} = \{0,\ldots,\mu_{\max}\}$.

B. Data acquisition

For data acquisition, the Velodyne HDL-64E S2 is mounted on top of our research vehicle MuCAR–3. A detailed description can be found in [9]. The data arrives in 347 UDP packets within $t_{\text{acq}} = 100$ ms. Each of the UDP packets contains six sequential sensor patterns $\mathbf{S}_\mu$. In each rotation, $\mu_{\max} = 2083$ pattern segments are processed with approximately 130.000 points. Each point $\mathbf{p}_{\mu,i}$ is described by its rotation index μ and beam ID i. This data is the basis of our point cloud grid-mesh representation that will be introduced below.

C. Mesh graph construction

Our undirected mesh graph is constructed by using the beam calibration and the chronology of the incoming UDP packets. The graph structure is similar to a range image with $\mu_{\max} = 2083$ columns and $i_{\max} = 64$ rows. Generally,

each pixel cell is filled through the elevation and yaw angles of the measurements with a predefined angular resolution. This leads to quantization errors which especially occur for rotation LiDAR sensors. As a result, different measurements are merged into the same pixel cell and thus, not all measurements are taken into account. To solve this problem, a method is proposed to create a mesh graph without comparing or sorting angles during runtime by taking the static beam angle offset within pattern $\mathbf{S}$ into account.

The mesh graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ consists of a set of vertices $\mathbf{V} = \{\mathbf{v}_{\mu,i} = (\mathbf{p}_{\mu,i}, r_{\mu,i}, z_{\mu,i}, \omega_{\mu,i}, l_{\mu,i}) \mid \mu \in \mathbb{U}, i \in \mathbb{I}\}$ described by a point $\mathbf{p}$ in Cartesian coordinates, range measurement r, point height z in the sensor coordinate frame, index ω and label l. The index $\omega \in \mathbb{O}$ describes the vertical order with ascending elevation angle, where $\mathbb{O} = \{1 \dots 64\}$. The label l can take the following values: *free space*, *obstacle*, *noise*, *edge*, *invalid*. Moreover, we distinguish between horizontal and vertical edges $\mathbf{E} = \{\mathbf{E}^h, \mathbf{E}^v\}$.

The horizontal edges $\mathbf{E}^h$ are created by taking consecutive measurements $\{\mu_0, \dots, \mu_{\max}\}$ of the same laser beam b_i into account with:

$$\mathbf{E}^h = \left\{ (\mathbf{v}_{\mu_1,i}, \mathbf{v}_{\mu_2,i}) \middle| |\mu_1 - \mu_2| = 1 \wedge \mu_1, \mu_2 \in \mathbb{U} \right\}.$$

The vertical neighborhood is determined by the static angle difference $\Delta\Psi_i = |\Psi_{38} - \Psi_i|$ between all beams $i \in \mathbb{I}$ and the corresponding reference beam Ψ_{38}, where $\omega_{\mu,38} = 1$ (smallest elevation angle). The horizontal pattern shift $\Delta\mu_i = \lceil \Delta\Psi_i / \alpha_h \rceil$ is used to find the almost parallel reference vertex, with delta angle α_h between two consecutive firings. This shift describes in a discrete manner how many patterns one has to receive or has already received to fill up the vertical aligned vertex neighborhood. Vertical edges $e_{i_1,i_2} = (\mathbf{v}_{\mu_1,i_1}, \mathbf{v}_{\mu_2,i_2}) \in \mathbf{E}^v$, with $\mu_1, \mu_2 \in \mathbb{U}$ and $i_1, i_2 \in \mathbb{I}$, are defined as follows:

$$\mathbf{E}^v = \left\{ e_{i_1,i_2} \middle| \mu_2 = g(\mu_{\max}, \mu_{\text{off}}) \wedge |\omega_{\mu_1,i_1} - \omega_{\mu_2,i_2}| = 1 \right\},$$

with offset $\mu_{\text{off}} = \mu_1 + \Delta\mu_2$ and pattern shift function $g(\mu_{\max}, \mu_{\text{off}})$:

$$g(\mu_{\max}, \mu_{\text{off}}) = \begin{cases} \mu_{\text{off}} & \mu_{\text{off}} \in M \setminus \{0, \mu_{\max}\} \\ 2\mu_{\max} - \mu_{\text{off}} & \mu_{\text{off}} > \mu_{\max} \\ \mu_{\max} + \mu_{\text{off}} & \mu_{\text{off}} < 0 \\ \mu_1 & else \end{cases}.$$

The result is stored in a lookup table to speed up runtime. This method allows a very fast graph construction within 1 ms. As shown in Fig. 4, an N4 neighborhood is applied.

Due to the separate processing of each incoming Velodyne pattern $\mathbf{S}$, we directly detect missing measurements. They are represented by vertices $\{\mathbf{v}_{\mu,i} \in \mathbf{V} \mid l_{\mu,i} = invalid\}$ which are labeled as *invalid*. Furthermore, we fill the gaps between valid vertices by adding vertical and horizontal edges. In section III-D.4, a new approach will be introduced to approximate *invalid* vertices by taking the laser diode intrinsics into account.

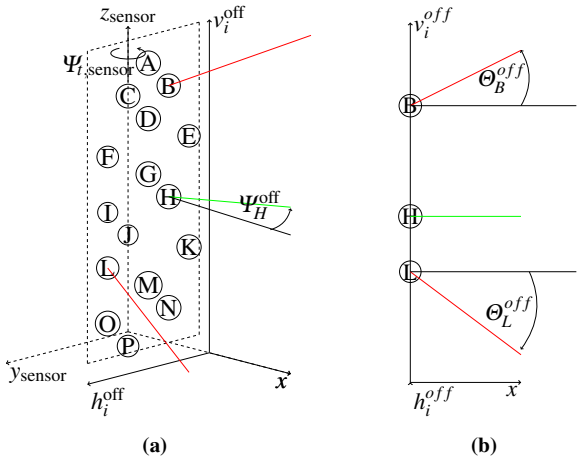


Fig. 2: Velodyne laser beams: One group of 16 laser diodes (A-P) with different azimuth Ψ^{off} (a) and elevation Θ^{off} (b) offset angle and the horizontal h_i^{off} and vertical v_i^{off} offsets for each laser beam. Four groups of this pattern rotate around a pivot point with sensor rotation angle $\Psi_{t,\text{sensor}}$.

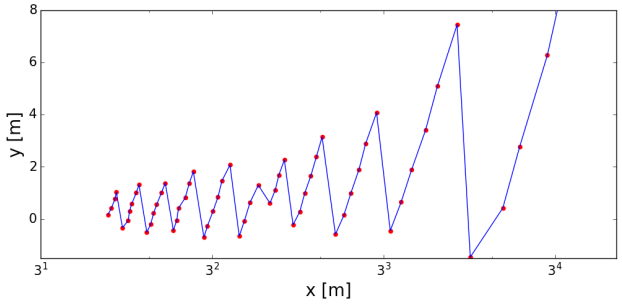


Fig. 3: Bird's eye view: Snippet of the Velodyne pattern $\mathbf{S}_\mu$ with 58 points (red) and logarithmic scale in x-axis. The blue line connects the beams with ascending elevation angle from left to right. On a flat surface, only beams with a negative elevation angle Θ^{off} reach the ground plane.

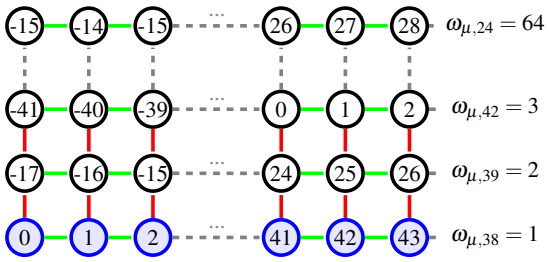


Fig. 4: Mesh graph representation: For each vertex $\mathbf{v}_{\mu,i}$, an N4 neighborhood is applied. The horizontal (green) and vertical edges (red) represent the connection. Blue vertices outline the reference vertices with ascending rotation index μ from left to right and ω from bottom to top. The remaining vertices are labeled by the horizontal pattern shift $\Delta\mu$ to the vertical reference beam, respectively.

D. Multi-pass labeling

In this subsection, a detailed description of our multi-pass labeling algorithm will be given. Further, only the vertical edges are considered. Therefore, we simplify the notation to a linear graph for each grid column separately. Vertices

can be listed in the order $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{64}$ such that the vertical edges are $\mathbf{E}^v = \{(\mathbf{v}_\omega, \mathbf{v}_{\omega+1}) \,|\, |\omega+1| \in \mathbb{O}\}$.

In order to find a label, each vertex will be observed and a labeling function L assigns a label to the vertex $\mathbf{v}_\omega$. Initially, all points are labeled as *free space*, because it is assumed that the vehicle operates in a flat world where the Velodyne pattern will not be deformed. Thus, our approach does not explicitly detect ground points or does a ground plane estimation like other approaches. Our approach detects discontinuities compared to the flat world pattern.

1) Key-vertices: First, we search for key-vertices which violate our Velodyne sensor-pattern model under a flat-world assumption. The strongest indication we found is the vertical index ω over the distance measurement. Therefore, each vertex range measurement r_ω is compared with the subsequent $r_{\omega+1}$. If $r_\omega > r_{\omega+1}$, the subsequent label $l_{\omega+1}$ is set to *obstacle* and the vertex $\mathbf{v}_{\omega+1}$ is added to the set of key-vertices $\mathbb{K}$.

2) Multimodal distribution clustering: Second, the key-vertices are used as priors for our multimodal distribution clustering. The density of the underlying point distribution is estimated using three different histograms taking the range (H_r), height (H_z) and vertical index (H_ω) into account.

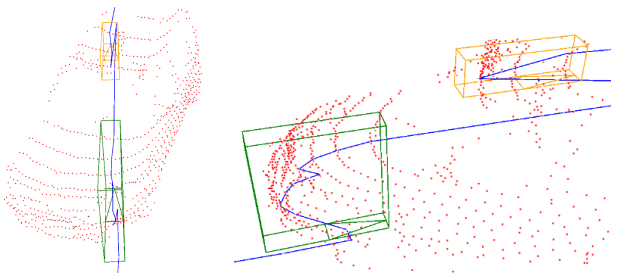
The range histogram H_r is initialized with the set of key-vertices. If key-vertices are in a radial neighborhood of $h_{\text{r,init}}$ and the point height of the next vertex (larger vertical index ω) is bigger than the previous point height, key-vertices are grouped into the same bin. Histogram bins are discarded if they contain less than two key-vertices or the mean bin height does not exceed 0.1 m. Next, we initialize the height and index histograms (H_z, H_ω) for all key-vertices in each range bin respectively. After initialization, a nearest neighborhood search is applied to find the closest histogram range bin for all unlabeled vertices $\mathbf{v}_\omega \notin \mathbb{K}$. If the unlabeled vertex is in radial distance of $h_{\text{r,label}}$ to the closest range bin, the vertex will be labeled $l_\omega = L(h_n(\omega))$ as follows:

$$h_n(\omega) = \begin{cases} \textit{obstacle} & \omega \in H_\omega \vee \\ & \big(\omega > \max(H_\omega) \wedge z_\omega > \min(H_z) \wedge \\ & \|\text{mean}(H_z) - z_\omega\| \geq h_{\text{height}}\big) \\ \textit{free space} & \text{otherwise} \end{cases}.$$

with height threshold h_{height}. Furthermore, we use Scott's normal reference rule [21] to update the range histogram's bin width $= 3.49 \cdot \sigma / \sqrt[3]{n_k}$ after we have found a new vertex that pertains to the bin, where σ is the standard deviation of the range measurement r_ω and n_k the number of key-vertices in the bin. This allows us to minimize the integrated mean squared error of the density estimate.

This clustering method provides good results for vertical objects, e.g., walls, poles and especially for obstacles with flat surfaces like cars. A result is shown in Fig. 5.

3) Pattern matching: In the third pass, all remaining and unlabeled vertices will be labeled. Therefore, we fit an eighth-degree polynomial function $f(\omega) = \sum_{j=0}^{8} a_j \omega^j$ that approximates the Velodyne pattern in a flat world with range over vertical index ω by minimizing the squared error:



(a) Front view **(b)** Side view

Fig. 5: Multimodal distribution clustering result of our multi-histogram-based approach to estimate the density of the underlying point distribution for each vertical column μ separately (blue line). Red points represent obstacle points and the green and orange bounding boxes show the density estimation of a vehicle. The bounding box dimensions: length and height are determined by the corresponding bin width H_r and height range of H_z. The width is just set for a better appearance.

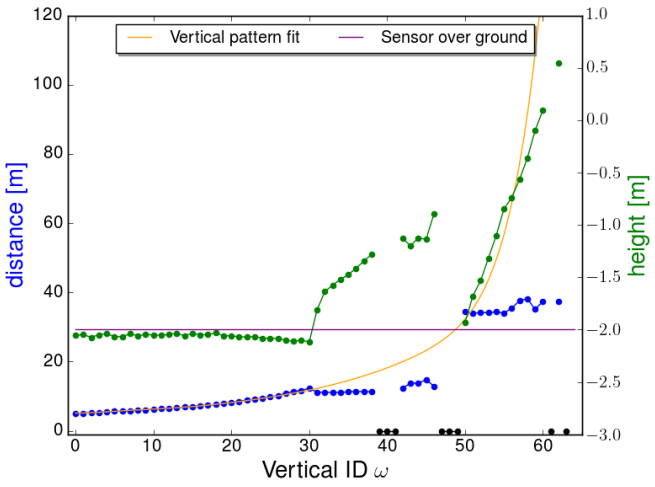


Fig. 6: Vertical pattern fit: This plot shows our fitted function f (orange) that approximates the range measurements of all points within one pattern assuming a flat world. Furthermore, the Velodyne sensor pattern $\mathbf{S}$ is shown with the vertex range r_ω (blue) and height z_ω (green) over the vertical index ω in a real-world scenario. The horizontal line (purple) plots the sensor height over ground which is in our case around –2 m. One can see the deviation from our model especially in the range $\{\omega\}_{31}^{38}$ and $\{\omega\}_{42}^{46}$ which correspond to the example in Fig. 5 with the green and orange bounding box. Black points represent invalid vertices.

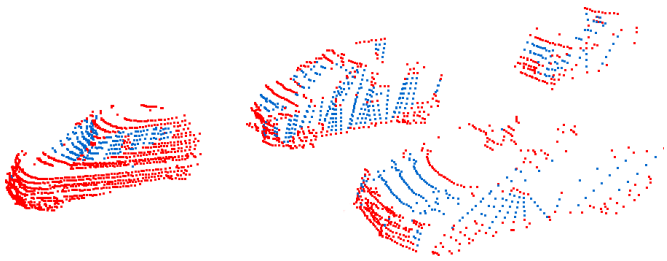


Fig. 7: Artificial vertices result: This figure shows the artificial points (blue) and obstacle points (red) with removed ground plane. One can see that this method approximates the contour of black cars (the three on the right side) and car windows very well.

$E = \sum_{\omega=1}^{64} |f(\omega) - r_\omega|^2$. The function is plotted in Fig. 6 (vertical pattern fit). The coefficients are listed in Tab. I. If the vertical vertices $\mathbf{v}_\omega$ deviate considerably from the model f, we have found another obstacle vertex. A decision tree is used to speed up the process of labeling by taking the vertical N2 neighborhood with previous vertex $\mathbf{v}_{\omega-1}$, current vertex $\mathbf{v}_\omega$ and next vertex $\mathbf{v}_{\omega+1}$ into account. In order to better illustrate the conditions, a decision table is shown in Tab. II. The table has six different entry states $\{A,\ldots,F\}$, which describe the incoming vertex labels $\{l_{\omega-1}, l_\omega, l_{\omega+1}\}$. In the matter of simplicity, labels are denoted with their first letter (underlined): *noise* (n), *free space* (f), *obstacle* (o), *edge* (e).

In each state, different conditions are checked. If they are fulfilled, a state transition is applied. This process is repeated until the END state is reached. If no entry state is reached or the conditions are not fulfilled, nothing is done. The current implementation runs only in the ascending vertical direction. The horizontal neighborhood is not yet taken into account.

In this pass, the two vertex labels *edge* and *noise* are used. They represent the transition, along a vertical graph path, between non-obstacle → obstacle (*edge*) and obstacle → non-obstacle (*noise*). The conditions are evaluated as follows with the binary decision function d, which takes the range difference between the vertices $\mathbf{v}_\omega$ and $\mathbf{v}_{\omega+1}$ and the pattern model f into account:

$$d(\mathbf{v}_\omega, \mathbf{v}_{\omega+1}) = \begin{cases} \text{false} & |r_\omega - f(\omega)| > d_{\text{t1}} f(\omega) \vee \\ & |r_{\omega+1} - r_\omega| > d_{\text{t2}} |f(\omega+1) - f(\omega)| \\ \text{true} & \text{otherwise} \end{cases},$$

with d_{t1}, d_{t2} as the deviation thresholds. The function returns false if a sufficient deviation from the pattern is found. Furthermore, we determine the slope in 3D and compare the result with the slope threshold s_t.

$$s(\mathbf{v}_\omega, \mathbf{v}_{\omega+1}) = \begin{cases} \text{true} & \text{slope}(\mathbf{p}_\omega, \mathbf{p}_{\omega+1}) < s_t \\ \text{false} & \text{otherwise} \end{cases}.$$

This threshold enables us to change the allowed slope accordingly to environment conditions and the vehicle we are using. This method is sensitive to errors in the extrinsic calibration of the sensor but stable to high-frequency angle changes in, e.g., pitch and roll as they especially occur in off-road scenarios.

4) Artificial vertices: Not all pulses find their way back to the LiDAR sensor due to diffuse reflections and specular surfaces. Missing pulses lead to missing range measurements, especially on black cars and vehicle windows. In this pass, a method is introduced which adds artificial vertices to the graph to enrich 3D objects by virtual measurements.

In the first step, a clustering is applied to group *invalid* vertices $g = \{\mathbf{v}_{\mu,i} \in \mathbf{G}_v \mid l_{\mu,i} = \textit{invalid}\}$, where the adjoining vertices must be labeled as *obstacles*. If the number of invalid vertices within one group is less than a threshold $t_{g_{\max}}$ and the Euclidean distance between the adjoining vertices is smaller than a threshold of $t_{d_{\max}}$, the ranges $r_{\mu,i}$ are approximated by linearly spaced vectors between the adjoining vertices. Consequently, the Cartesian coordinates $\mathbf{p}_{\mu,i}$ are determined by taking the beam calibration parameters $\mathbf{b}_i$ and the current sensor rotation angle $\Psi_{t,\text{sensor}}$ into account to simulate a linear surface in 3D. This method is a simple heuristic but works well for black cars and vehicle windows as shown in Fig. 7, where $t_{g_{\max}} = 30$ and $t_{d_{\max}} = 1.3\,\text{m}$. Moreover, the overall result of the following pass is improved.

TABLE I: Coefficients of the Velodyne pattern function $f(\omega)$

$a_8 = 5.60519e-11$	$a_7 = -1.05775e-08$	$a_6 = 8.20012e-07$
$a_5 = -3.34620e-05$	$a_4 = 7.65166e-04$	$a_3 = -9.46870e-03$
$a_2 = 5.94245e-02$	$a_1 = -3.52446e-02$	$a_0 = 5.12710$

TABLE II: N-2 Neighborhood decision table

States \ Labels	$l_{\omega-1}$	l_ω	$l_{\omega+1}$	$\hat{l}_{\omega-1}$	$\hat{l}_\omega$	$\hat{l}_{\omega+1}$	Conditions
A → END	f	o	o	e	o	o	$\neg d(\mathbf{v}_{\omega-1}, \mathbf{v}_\omega) \wedge z_{\omega-1} < z_\omega$
A → END	f	o	o	o	o	o	$z_{\omega-1} > z_\omega$
B → END	o	f	o	o	o	o	$\neg d(\mathbf{v}_{\omega-1}, \mathbf{v}_\omega) \wedge \neg d(\mathbf{v}_\omega, \mathbf{v}_{\omega+1})$
B → END	o	f	o	o	n	o	otherwise
C → A	f ∨ o	o	f	f ∨ o	o	o	$\neg s(\mathbf{v}_\omega, \mathbf{v}_{\omega+1}) \wedge d(\mathbf{v}_\omega, \mathbf{v}_{\omega+1})$
C → A	f ∨ o	o	f	f ∨ o	o	o	$s(\mathbf{v}_\omega, \mathbf{v}_{\omega+1}) < 0 \wedge \neg d(\mathbf{v}_\omega, \mathbf{v}_{\omega+1})$
D → C	o	f	f	o	o	f	$\neg d(\mathbf{v}_{\omega-1}, \mathbf{v}_\omega) \wedge \neg s(\mathbf{v}_{\omega-1}, \mathbf{v}_\omega)$
D → END	o	f	f	o	n	f	otherwise
E → A	f	f	o	f	o	o	$\neg s(\mathbf{v}_{\omega-1}, \mathbf{v}_\omega) \wedge r_\omega < r_{\omega+1} \wedge \text{slope}(\mathbf{p}_\omega, \mathbf{p}_{\omega+1}) > 0$
F → C	f	f	f	f	o	f	$\neg s(\mathbf{v}_{\omega-1}, \mathbf{v}_\omega) \wedge \lvert z_{\omega-1} - z_{\omega+1} \rvert > 0.1$

5) Clustering: In the final pass, an N4-connected region growing algorithm is run on the graph $\mathbf{G}$ to group obstacle points. The algorithm can be described in three steps:

- Select initial $v_{\mu,i}$ randomly, with $l_{\mu,i} = \textit{obstacle}$
- Grow region until no more vertices are added
- Remove region from graph.

Vertices are added to a region, if the beta-angle criteria from [17] is met, whereas $\beta_h > \alpha_h$. Additionally we distinguish between the vertical and horizontal neighborhood with different angles: $\beta_v = 4.5°$ and $\beta_h = 10°$. The vertical alpha α_v is approximated by our fitted function f from Sec. III-D.3

IV. EXPERIMENTAL EVALUATION AND RESULTS

The goal of our experimental evaluation is to show the performance of our approach in suburban as well as off-road scenarios. The test data was collected in a suburb of Munich and at the European Land Robot Trial 2016 [9]. The runtime of our point-wise labeling is compared with a popular grid-based ground plane segmentation method [14]. In order to evaluate the segmentation result, we have labeled 20 different outdoor scans manually. Each point cloud has approximately 130.000 points, and point samples are marked as obstacle or ground. Furthermore, groups of obstacle points are described by a bounding box. Our final segmentation result is compared with the results of [17] and [22].

A. Runtime Evaluation

In this section, the runtime of our point labeling is compared with the ground plane segmentation approach from [14]. The runtime is evaluated on a desktop computer with an Intel i7-4790K 4GHz CPU for a total of 3000 scans. The dataset was recorded in a suburban area, and the results are shown in Fig. 8. Both algorithms utilize parallelization with up to four threads. Further runtime tests were performed in off-road scenarios. Our point labeling achieves a mean

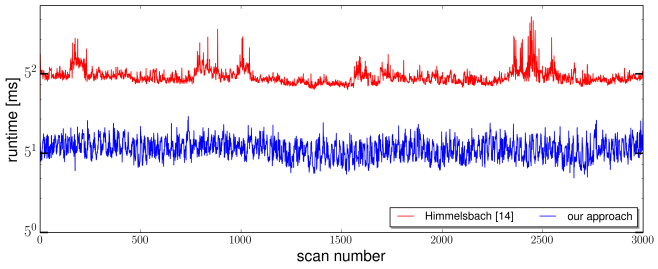


Fig. 8: Runtime results of [14] and our point-wise labeling plotted in log scale.

runtime of around 5.5 ms in all tested environments. Additionally, the algorithm has a nearly constant processing time without peaks due to the fact that we actively avoid a ground estimation with e.g. a least-squares fit, where the runtime strongly depends on the environment. The overall mean runtime of our approach including the clustering is approximately 15 ms.

B. Segmentation Results

In this section, the segmentation performance of the algorithms are compared in suburban areas and in rough off-road scenarios. Video footage recorded during practical experiments is available online[1]. The proposed parameters of [17], [22] are used to achieve the best results, respectively. Moreover, all parameters are fixed throughout the evaluation with: $h_{\text{r,init}} = 0.4\,\text{m}$, $h_{\text{r,label}} = 1.2\,\text{m}$, $h_{\text{height}} = 0.5\,\text{m}$, $d_{\text{t1}} = 0.25$, $d_{\text{t2}} = 1.1$, $s_t = 0.3$, $t_{g_{\max}} = 30$, $t_{d_{\max}} = 1.3\,\text{m}$. Finally, we remove regions that do not contain at least 10 vertices. Results for a sub-urban scenario are shown in Fig. 9.

Furthermore, we compute the precision $\mathscr{P}$ and recall $\mathscr{R}$ of the point labeling. The precision is calculated by counting the number of correctly classified obstacle points and then divided by the number of all detected obstacle points. The recall is calculated by counting the number of correctly classified obstacle points and then divided by the number of expected obstacle points.

Additionally, we determine the precision $\mathscr{O}$ of the object segmentation by comparing the amount of correctly detected objects with the number of expected objects. Therefore, we manually select all bounding boxes that fit best and are closest to the expected objects with [23]. The segmentation results are shown in Tab. III. As one can see, our algorithm outperforms the other approaches in both scenarios. The *obstacle* and *free space* (ground plane) labeling results are shown in Fig. 9b for [14] and our approach in Fig. 9c. It shows that our approach is less prone to under-segmentation. Additionally, we avoid incorrect labels by taking the uncertainty of the point density and the previous point label into account and set these points as *noise*. The object segmentation result of multiple vehicles (V1-V5), bushes (B1-B7) and a pole (P1) are shown in Fig. 9d, 9e, 9f. These figures clearly demonstrate how important it is to obtain a precise point discrimination between obstacle- and free-space points. The approach from [22] erroneously

[1] http://www.mucar3.de/iv2018-segmentation/

TABLE III: Segmentation Results

	off-road			sub-urban		
	$\mathscr{P}$	$\mathscr{R}$	$\mathscr{O}$	$\mathscr{P}$	$\mathscr{R}$	$\mathscr{O}$
[17]	0.61	0.71	0.84	0.70	0.71	0.85
[22]	0.81	0.85	0.75	0.83	0.85	0.81
our approach	0.91	0.92	0.93	0.93	0.96	0.97

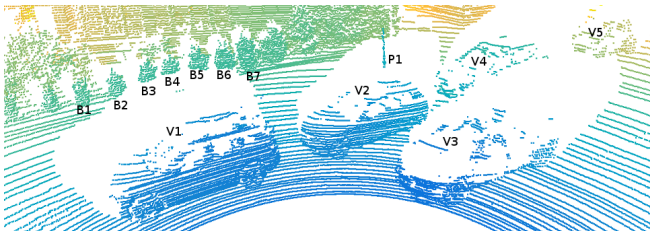
fuses free-space with obstacle points as one can see at the bottom of the three vehicles V1 (magenta), V2 (blue), V3 (dark-blue). Tracking algorithms are negatively affected by this under-segmentation. It highly increases the uncertainty of the object's position and dimension due to high-frequency dimension changes. Moreover, points which correspond to vehicles (V4, V5) are not or just partially clustered due to the low point density and grid discretization. To overcome this problem, we have introduced the artificial vertices in Sec. III-D.4 to find the vertical correspondences even if the points are far apart from each other. The approach from [17] tend to over-segmentation especially for the vehicles (V1,V3,V4,V5). Fig. 9f shows the overall performance of our approach. All vehicles (V1-V5), most bushes (B2-B6) and the pole P1 are detected completely. The presented examples show that our approach delivers coherent point clusters with respect to the underlying point density.

V. CONCLUSION AND FUTURE WORK

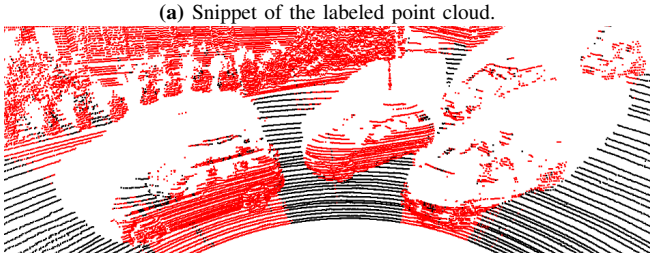
We have developed a fast multi-pass 3D point segmentation approach that incrementally improves the labeling result. First, key-vertices are identified which are highly likely to be an obstacle. Second, we use these key-vertices as prior for our multimodal distribution clustering. This step greatly improves the labeling results of flat surfaces on objects, e.g., hoods. Third, the unlabeled vertices and their neighborhood are compared with our pattern function that describes how one scan appears in a flat world without obstacles. The sensor pattern is used to evaluate the coherent neighborhood of each vertex. Fourth, we found a criterion to determine points that belong together by taking missing measurements into account. *Invalid* vertices are approximated, if these conditions holds. As a result, the appearance of objects is improved. In addition, we evaluated the proposed method on real-life applications and demonstrated that our approach delivers reliable labeling results in sub-urban but also in challenging off-road scenarios. Finally, our method is very fast with a nearly constant processing time.

In future research, we will use the mesh graph as basis for our road segmentation, vegetation classification and localization algorithms. Furthermore, the segmented point clouds are used as input for our multi-object tracking algorithms [9], [22] and the environment model [5]. In future work, we will improve the point labeling by our multimodal distribution clustering from Sec. III-D.2 as prior and by taking the horizontal neighborhood into account. Additionally, the system's performance has to be evaluated with different sensors and compared with publicly available datasets.

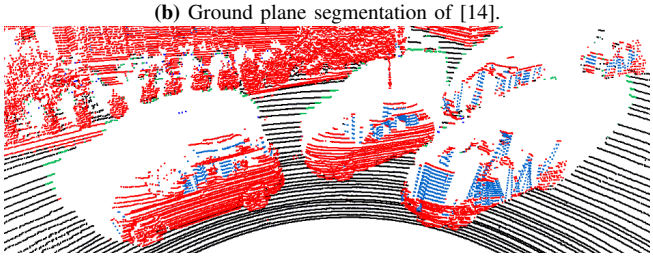
Due to the highly specific mesh graph construction we could not finish the integration of, e.g., the KITTI dataset because no raw UDP data packets are available.



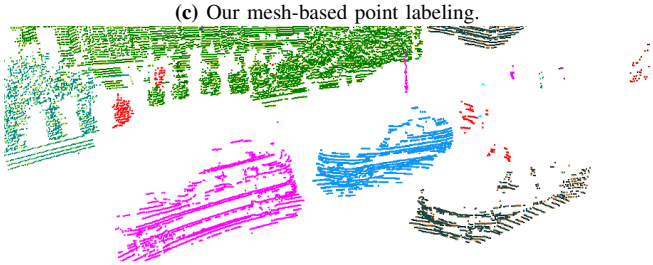
(a) Snippet of the labeled point cloud.



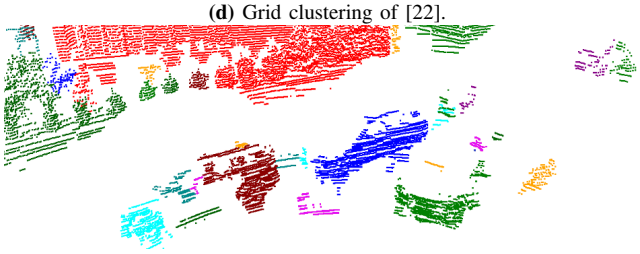
(b) Ground plane segmentation of [14].



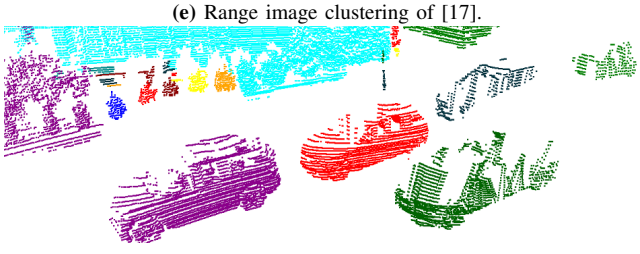
(c) Our mesh-based point labeling.



(d) Grid clustering of [22].



(e) Range image clustering of [17].



(f) Our multi-pass point segmentation approach.

Fig. 9: Results of the different algorithms: Fig. a displays the input point cloud with manually labeled objects. In Fig. b, c the point labeling is shown with color labels: free space (black), obstacle (red), artificial vertices (blue) and noise (green). Fig. d, e, f present the overall segmentation performance of the algorithms. Coherent objects have the same color.

ACKNOWLEDGMENT

The authors gratefully acknowledge funding by the Federal Office of Bundeswehr Equipment, Information Technology and In-Service Support (BAAINBw).

REFERENCES

[1] S. Thrun, "Winning the DARPA Grand Challenge: A Robot Race through the Mojave Desert," in *21st IEEE/ACM International Conference on Automated Software Engineering (ASE'06)*, 2006.

[2] C. Urmson, *et al.*, "Autonomous Driving in Urban Environments: Boss and the Urban Challenge," *J. Field Robotics*, vol. 25, no. 8, 2008.

[3] C. Guo, *et al.*, "A Vision System for Autonomous Vehicle Navigation in Challenging Traffic Scenes Using Integrated Cues," in *Proc. IEEE Intelligent Transportation Syst. Conf. (ITSC)*, 2010.

[4] A. Geiger, *et al.*, "3D Traffic Scene Understanding From Movable Platforms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 5, 2014.

[5] H. Jaspers, *et al.*, "Multi-modal Local Terrain Maps from Vision and LiDAR," in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, USA, 2017.

[6] J. Rieken, *et al.*, "Benefits of Using Explicit Ground-Plane Information for Grid-based Urban Environment Modeling," in *18th International Conference on Information Fusion (Fusion)*, July 2015.

[7] D. Fassbender, *et al.*, "Motion Planning for Autonomous Vehicles in Highly Constrained Urban Environments," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, 2016.

[8] J. Ziegler, *et al.*, "Video Based Localization for BERTHA," in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, June 2014.

[9] C. Fries, *et al.*, "How MuCAR Won the Convoy Scenario at ELROB 2016," in *Proc. IEEE Intelligent Transportation Syst. Conf. (ITSC)*, Yokohama, Japan, Oct. 2017.

[10] N. Hautiere, *et al.*, "Road Scene Analysis by Stereovision: a Robust and Quasi-Dense Approach," in *Int. Conf. on Control, Automation, Robotics and Vision*, 2006.

[11] S. Ramos, *et al.*, "Detecting Unexpected Obstacles for Self-Driving Cars: Fusing Deep Learning and Geometric Modeling," in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, June 2017.

[12] M. Montemerlo, *et al.*, "Junior: The Stanford Entry in the Urban Challenge," *J. Field Robotics*, vol. 25, no. 9, 2008.

[13] T. Chen, *et al.*, "Gaussian-Process-Based Real-Time Ground Segmentation for Autonomous Land Vehicles," *Journal of Intelligent & Robotic Systems*, vol. 76, no. 3, 2014.

[14] M. Himmelsbach, *et al.*, "Fast Segmentation of 3D Point Clouds for Ground Vehicles," in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, San Diego, CA, USA, 2010.

[15] D. Holz and S. Behnke, "Fast Range Image Segmentation and Smoothing Using Approximate Surface Reconstruction and Region Growing," *Proc. Int. Conf. Intelligent Autonomous Syst. (IAS)*, 2013.

[16] A. Lejeune, *et al.*, "Probabilistic Framework for the Characterization of Surfaces and Edges in Range Images, with Application to Edge Detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017.

[17] I. Bogoslavskyi and C. Stachniss, "Fast Range Image-Based Segmentation of Sparse 3D Laser Scans for Online Operation," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, Oct. 2016.

[18] K. Klasing, *et al.*, "Realtime Segmentation of Range Data Using Continuous Nearest Neighbors," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2009.

[19] F. Moosmann, *et al.*, "Segmentation of 3D Lidar Data in non-flat Urban Environments using a Local Convexity Criterion," in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, June 2009.

[20] G. Atanacio-Jimnez, *et al.*, "Lidar velodyne hdl-64e calibration using pattern planes," *International Journal of Advanced Robotic Systems*, vol. 8, no. 5, 2011.

[21] D. W. Scott, "On Optimal and Data-Based Histograms," *Biometrika*, vol. 66, no. 3, pp. 605–610, 1979.

[22] M. Himmelsbach and H.-J. Wuensche, "Tracking and Classification of Arbitrary Objects with Bottom-Up/Top-Down Detection," in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, Spain, 2012.

[23] B. Naujoks and H.-J. Wuensche, "An Orientation Corrected Bounding Box Fit Based on the Convex Hull under Real Time Constraints," in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, China, 2018.