# Multi-view Based Clustering of 3D LiDAR Point Clouds for Intelligent Vehicles

Haoxiang Jie[1], Zuotao Ning[1(✉)], Qixi Zhao[1], Wei Liu[1,2], Jun Hu[1], and Jian Gao[1]

[1] Neusoft Reach Automotive Technology Company, Shenyang, China
ningzt@reachauto.com
[2] School of Computer Science and Engineering, Northeastern University, Shenyang, China

**Abstract.** 3D point clustering is important for the LiDAR perception system involved applications in tracking, 3D detection, etc. With the development of high-resolution LiDAR, each LiDAR frame perceives richer detail information of the surrounding environment but highly enlarges the point data volume, which brings a challenge for clustering algorithms to precisely segment the point cloud while running with a real-time processing speed. To meet this challenge, we innovate a multi-view (bird's eye view and front view) based clustering method, named MVC. The method contains two stages. In the first stage, we propose a density image based algorithm, PG-DBSCAN, to segment the point cloud in bird's eye view (BEV), which derives the preliminary division with fairly low computation resources. Then in the second stage, a front view (FV) clustering process is integrated to refine the under-segmented clusters. Our method takes both the speed and precision advantages of BEV and FV clustering, and this coarse-to-fine architecture reasonably allocates the computation resources and shows a real-time outstanding clustering performance. We evaluate the MVC algorithm both on the publicly available dataset with 64-line LiDAR and our own dataset with 128-line LiDAR. Compared with other clustering methods, MVC is able to derive more accurate clustering results. Specifically, toward the 128-line LiDAR with large data volume, our method shows an outperforming running speed, which perfectly fits on the LiDAR perception tasks.

**Keywords:** Point Cloud Segmentation · High Resolution LiDAR · PG-DBSCAN

## 1 Introduction

In the LiDAR perception system, Deep-Learning based 3D detection modules are widely used to provide important evidences for the free driving space prediction. However, sometimes such kind of modules may perform miss detection or incorrect detection when meeting untrained rare scenes, and further cause wrong


H. Aziz et al. (Eds.): AI 2022, LNAI 13728, pp. 57–70, 2022.
https://doi.org/10.1007/978-3-031-22695-3_5

drivable area prediction. That could be dangerous. For solving this problems, engineers and researchers have been developing 3D object clustering methods using as the back-up plan that is able to perceive the obstacle locations when detection modules make wrong judgments.

With the development of laser sensors and electronic chips, the resolution of LiDARs is designed higher and higher. For example, the LiDAR "Ruby" from Robosence company[1] is assembled with 128 lines and 0.2 °C horizontal resolution. Such LiDAR is able to reflect over 2.3 million 3D points in each frame and provides much richer 3D information of the surrounding environment compared with the lower resolution LiDARs. But the denser the point clouds are, the larger the data volume would be. So the high resolution LiDAR raises more requirements for the processing speed while maintaining the segmentation accuracy.

However, facing the dense LiDAR point cloud, the traditional clustering methods, such as DBSCAN [6], Mean Shift [4], pose difficulties of high computational complexity and fail in real-time processing. The point cloud clustering methods based on the range maps, such as [13] and [21], may be able to meet the real-time running requirement, but would bring serious over-segmentation issues when the LiDAR resolution goes up.

So, in this paper, in order to decrease the computational complexity and achieve a satisfactory clustering performance, we propose a multi-view based 3D point cloud clustering algorithm (MVC). This method is inspired by the real-world spatial distribution of objects on the streets, that in the bird's-eye view (BEV), the majority of objects in the driving scenes are naturally separated. Thus we project the point cloud in BEV and design a preliminary clustering stage. In order to improve the processing speed, we down-sample the points in BEV with polar grid maps. Meanwhile, we modify the traditional DBSCAN [6] method and reduce the computational complexity. However, the BEV clustering module cannot segment the objects located at the same place in BEV but different places on the vertical direction, such as a billboard and the car below. Thus, for improving the clustering accuracy, we introduce front-view (FV) refining clustering stage to solve the vertical under-segmentation problem.

The main contributions of this paper are the following items.

- We innovate a new point cloud clustering method combining the BEV and FV, which utilizes the point cloud geographical features to accurately segment the 3D obstacles.
- We raise a PG-DBSCAN [6] algorithm which highly reduces the computational complexity for this 3D point clustering task.
- We compare our method with 4 effective traditional point cloud clustering methods on both semanticKITTI dataset [1] and a self-collected 128-line LiDAR dataset.

[1] https://www.robosense.ai/en/rslidar/RS-Ruby.

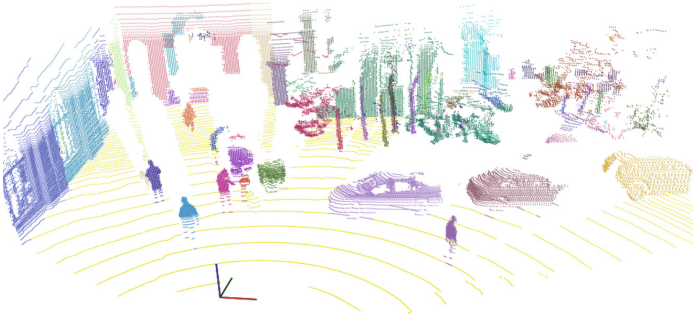


**Fig. 1.** A demonstration of MVC clustering result.

## 2 Related Work

In this section, through analysing existing technologies, we divide clustering algorithms into two categories: free-trained methods and Deep-learning based methods.

### 2.1 Free-trained Point Cloud Clustering Methods

**Clustering Method Based on Voxel/Grid Map.** Most of Voxel/Grid map based algorithms need minimal computational overhead and perform fast processing speed through down-sampling the point cloud. So this kind of clustering algorithms is widely utilized in the field of robots and unmanned vehicles due to their real-time requirement. For example, in the 2007 DARPA Challenge [19,20], many teams chose this kind of methods to separate objects from the ground. [5] created hybrid elevation maps to extract the non-ground objects, then down-sampled the non-ground point cloud by voxel cells and cluster the points according to the voxel connectivity. Similarly, [9] utilized the BEV grid map and 3D voxel to down-sample the non-ground points, then combined the connectivity of the grids and the height difference between the voxels to further cluster the point cloud [15] proposed the curved voxel clustering method considering the difference of horizontal and vertical angular resolutions for LiDAR. However, the segmentation accuracy of these methods highly depends on the size of grids or voxels, and some of the spatial information of point clouds is lost due to the point down-sampling.

**Clustering Method Based on Range Image.** The point cloud clustering methods based on the range image also attracted the interest of many researchers [2] projected 3D point clouds into range images. They performed a N4-searching by BFS algorithm and clustered the point cloud following a given angle threshold. On the basis of [2,12] further increased the constraints of distance and reflection intensity difference between adjacent points and reduced the over-segmentation rate. [22] proposed the Scan Line Run (SLR) clustering method based on the

range image. [8] combined density and connectivity information of the range image to achieve real-time clustering performance.

**Clustering Method Based on Graph Model.** Applying graph theory to achieve point cloud clustering is also a research direction. For example, [11] proposed a clustering method based on Radially Bounded Nearest neighbours (RBNN) graphs. They represented the 3D laser point clouds as directed graphs, then cluster the LiDAR point clouds based on a given threshold [14] generated point cloud undirected graphs based on the scanning characteristics of the mechanical rotating LiDAR. The point cloud is then separated according to the local convexity criterion which is calculated based on the normal vector of the nodes. Similarly, [3] considered the hardware parameters of Velodyne HDL-64 LiDAR when creating the undirected graphs. In order to balance the accuracy and running speed of the algorithm, they used a 4-connected region growing method to cluster 3D point clouds.

### 2.2 Deep-learning Based Point Cloud Clustering Methods

Pointnet [16] firstly generated Deeplearning network for solving the point cloud classification and clustering problem. They used multilayer perceptrons followed by max-pooling to extract the point feature and resulted a decent performance. A novel work, [7] proposed a proposal-free point cloud clustering method by a simplified framework with a Deeplearing-based solution. The method does not rely on any post-process, and is able to reach a good performance [18] presented a top-down Deep-learing based LiDAR segmentation architecture with a MASK R-CNN instance head. The method also formulated a pseudo labeling framework to enhance the clustering performance by training the network on unlabelled dateset. [10] used cylinder convolution extract grid-level features for each LiDAR frame and proposed Dynamic Shifting for complex point distributions then raised Consensus-driven Fusion to finally derive instance preditions.

## 3 The Proposed Method

Our work mainly focuses on the non-ground targets clustering. So a relevant ground segmentation method [14] is utilized for data preprocessing. The pipeline of MVC is briefly demonstrated in Fig. 2, which follows a coarse-to-fine architecture. It works with two stages: BEV coarse-segmentation and FV fine-segmentation.

The details of MVC are described in the following subsections.

### 3.1 Preliminary Clustering Based on BEV Projection

The demonstration of the BEV clustering procedure is shown in Fig. 3.

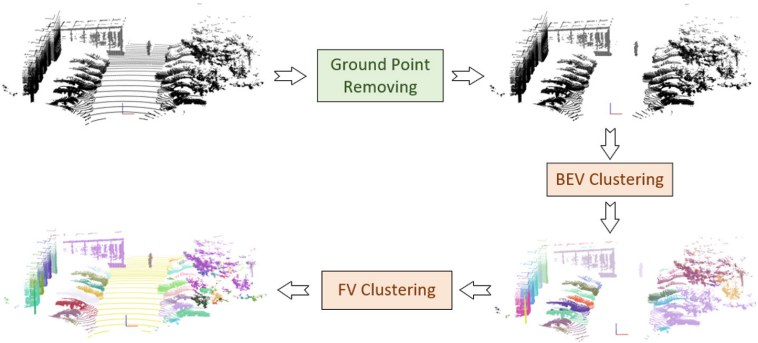



**Fig. 2.** The pipeline of MVC.

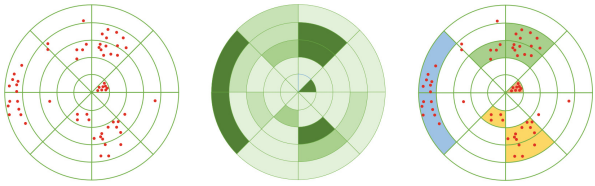



**Fig. 3.** (a) shows the BEV grids and the projected 2D points. We take the grids as pixels and count the point number in each grid as pixel values, then generate the density image as (b). The brightness of the color represents the density value. Using the modified DBSCAN method, pixels are clustered as shown in (c) and the points located in the grid with same color share the same cluster label.

Firstly, we project the object point cloud onto $x - o - y$ plane in Polor Coordinate $\{\rho, \theta\}$ by

$$\rho = \sqrt{x^2 + y^2} \tag{1}$$

$$\theta = i \times res \tag{2}$$

where $i$ is the horizontal index of the point anti-clockwise counted from the positive direction of the $x$ axis, and $res$ is the horizontal angle between two adjacent laser beams in the same scanning line. Clearly, the $\rho$ is equal to the range value of each target point.

Secondly, the grids of the BEV map are generated with a manually selected angle unit size $\theta_{thres}$ and range unit size $r_{thres}$. By using these grids we generate density image and count the point number in each gird as the density value $dv$. The reason why we choose Polar Coordinate in the BEV grid map is inspired by a related work CVC [15] that in such Coordinate the grid area expands with the range value increasing, which perfectly fits the near-dense-far-sparse geometrical characteristic of the LiDAR points.

Based on the traditional DBSCAN [6], we propose a density-based clustering method Polar-Grid-DBSCAN(PG-DBSCAN).

Compared with the traditional DBSCAN [6], instead of going through each point and calculating the surrounding data density for clustering, we go through each of the pixel in the density image to segment the point cloud. Firstly the pixels with density value $dv$ lower than 4 are marked as noise pixels. For the other pixels, we start from a random pixel as target and search its 8 neighbour pixels. If the neighbour pixels are not noise pixels, these neighbours are marked by the same label with the target. By recurrence, the whole density image is segmented to different areas, and the points located in each area share the same label.

Our PG-DBSCAN greatly accelerates the clustering speed compared with the traditional DBSCAN [6]. While dealing with large-scale point cloud data, as the area query operation of the traditional DBSCAN [6] is calculated based on the Euclidean distance, the average running time complexity is $O(\log(n))$, so the average computational complexity of traditional DBSCAN [6] is $O(n \times \log(n))$, where $n$ is the number of the points. However, PG-DBSCAN finishes the region query by inquiring the 8-neighbour of each pixel in the density image, so the computational complexity of one point becomes to $O(1)$, and the average computational complexity of one frame is reduced to $O(n)$, where $n$ is the pixel number of the density image.

After the operations mentioned above, we derive the preliminary segmentation result of the non-ground targets.

### 3.2 Refining Based on Range Image

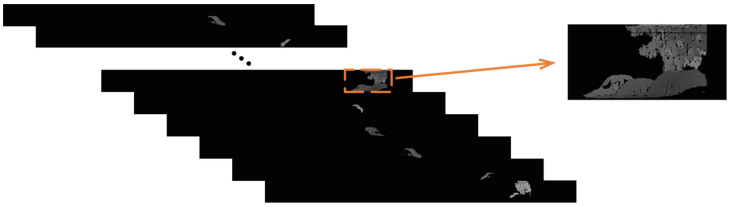


**Fig. 4.** After BEV preliminary segmentation, we project each clustered point cloud to a range image separately. The pixel brightness represents the range value of the points. Pure black pixels means no point or the point out of the region.

After BEV segmentation, for each cluster, we calculate the height difference $\Delta H$ between the highest point and lowest point. Only when $\Delta H$ is higher than 2 meters, we consider the cluster may be under-segmented that requires fine FV clustering.

For FV segmentation, the preliminary cluster is projected into range images [13], as Fig. 4.

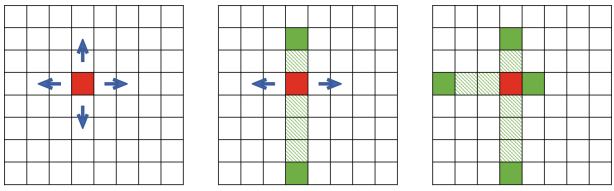



**Fig. 5.** The process of the modified N-4 neighbour searching. (a) indicates the searching directions. (b) and (c) are the searching process. We go through the pixels following the arrow directions and find the first non-zero-pixel specified by solid green blocks. The shadow green blocks represent the zeros pixels that are ignored.

Here we introduce a modified N-4 neighbor searching, and define neighbors of the target pixels as the adjacent pixels in the same row or column with the non-zero pixel values shown as Fig. 5

With this neighbour searching method, we cluster the pixels by judging if the height difference $\Delta h$ and range difference $\Delta r$ between the target and neighbour pixels are within a given threshold $\gamma$. We consider the points to belong to the same cluster on the condition that $\Delta h$ and $\Delta r$ meet Eq. (4) and Eq. (5). Here $r$ is directly derived from the pixel value and $h$ is calculated by:

$$h = r \times tan(\alpha) \tag{3}$$

$$\Delta r < \gamma \tag{4}$$

$$\Delta h < \gamma \tag{5}$$

where $\alpha$ is the vertical LiDAR beam angle which can be found from the LiDAR product specification. Through recursive searching method, all pixels in the range image are fine clustered, and the process is illustrated in Algorithm 1.

**Algorithm 1:** Refining based on Range Image

**Input:** An image in FV as *img* and its cluster as *cluster*
**Output:** *cluster*

```
foreach p_i in img do
    if p_i.flag! = is_visited then
        RecursiveClustering(p_i,cluster)
        Update(cluster)
    else
        continue
return cluster
```

**Function** RecursiveClustering($p_i$, $c_i$)

**Input:** A pixel of the image in FV as $p_i$ and its initialised cluster as $c_i$
**Output:** $c_i$

```
RecursiveClustering(p_i, c_i)
Neighbour_i ← FindNeighbour(p_i)
foreach n_i in Neighbour_i do
    if is_same_cluster(n_i,p_i) then
        Update(c_i)
        n.flag ← is_visited
        RecursiveClustering(n_i, c_i)
    else
        continue
```

## 4 Experiment

In this section, we test the proposed MVC algorithm and provide the experiment setup and evaluation metrics. We report the comparisons with different clustering methods on both SemanticKITTI dataset and our own dataset(NRS). Also, we carry ablation study for better understanding the advantages of the clustering processes of our method in the two views(BEV, FV).

We conduct experiments on a desktop with an Intel Xeon(R) CPU E3-1231 v3 @ 3.40 GHz × 8, 32 Gb RAM.

### 4.1 Experiment on SemanticKITTI

A Related work, [23] evaluated 4 different clustering methods on the SemanticKITTI dataset using Panoptic Quality(PQ) as evaluation metrics. For comparing the clustering performance of MVC with those 4 methods, we apply the same clustering process and the evaluation metrics. The result is shown in Table 1.

**Table 1.** Comparison between our method and the methods reported in [23] on SemanticKITTI dataset.

| Methods | Settings | PQ |
|---|---|---|
| Euclidean cluster | $d_{th} = 0.5$ m | 56.9 |
| Supervoxel cluster<br>Supervoxel cluster | $w_c, w_s, w_n = 0.0, 1.0, 0.0$<br>$w_c, w_s, w_n = 0.0, 1.0, 0.5$ | 52.8 52.7 |
| Depth cluster | $\theta = 10°$ | 55.2 |
| Scan-line run | $th_{run}, th_{merge} = 0.5, 1.0$ | 57.2 |
| Ours | $\theta_{thres}, r_{thres}, \gamma = 2, 0.5, 0.6$ | **58.8** |

**Performance Evaluation.** Following [23], all the clustering methods work as a post-process step after a semantic segmentation method, [24]. The experiment setting of the upper four methods remain same as in [23].

The experiment result shows that our method outperforms in the comparison group. It is worth mentioning that in this experiment, we abandon the ground point removing process, since the semantic segmentation process has already removed the ground point. Moreover, this pre-process also removes other background points, such as trees, which consequently deletes almost all the objects with large vertical size. However, our FV clustering processing happens only when the clusters from BEV segmentation are higher than 2 m. Thus, the FV clustering process seldom works in this experiment, but MVC still derives the best performance among all the methods.

### 4.2 Experiment on Self-Recorded Dataset

Importantly, for meeting engineering design requirements and feeding the needs from customers, it is necessary to test MVC method on our own dataset(NRS). NRS dataset is collected with RS-Ruby 128-line LiDAR sensor in the company NEUSOFT REACHAUTO[2] including the scenarios on the campus roads at Neusoft headquarters and the street of Shenyang city(China). However, because of the different labeling method between SemanticKITTI and NRS datasets, we have to change the evaluation metric from PQ to the method reported in [13,17].

We adopt the over-segmentation, under-segmentation and precision as criteria to evaluate the proposed algorithm. Further, we introduce four states of clustering results to quantify the clustering performance: Precision (P), True Positive (TP), Over-segmentation-rate (OSR) and Under-segmentation-rate (USR).

- TP is the number of clustered objects that are successfully segmented.
- OS is the total number of over-segmentation clusters.
- US is the total number of under-segmentation clusters.

Using the above-mentioned states, the following three metrics are formulated as:

$$OSR = 1 - \frac{TP}{TP + OS}, \tag{6}$$

$$USR = 1 - \frac{TP}{TP + US}, \tag{7}$$

$$P = \frac{TP}{TP + OS + US} \tag{8}$$

According to the related works, most of the previous algorithms are tested and validated via 64-beam LiDARs or even fewer. To the best of our knowledge, this paper is the first attempt to evaluate a clustering algorithm using a 128-line LiDAR.

[2] https://www.reachauto.com/.

**Table 2.** Experimental settings of different methods

| | |
|---|---|
| Euclidean cluster | $d_{th} = 0.5m$ |
| Supervoxel cluster | $w_c, w_s, w_n = 0.0, 1.0, 0.0$ |
| Depth cluster | $\theta = 5^\circ$ |
| Scan-line run | $th_{run}, th_{merge} = 0.3, 0.5$ |
| Ours | $\theta_{thres}, r_{thres}, \gamma = 1, 0.2, 0.5$ |

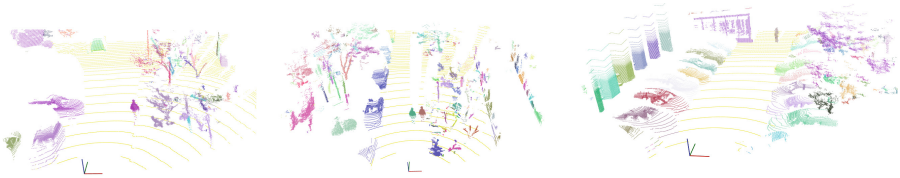
**Performance Evaluation.** Considering the point on NRS is denser than that on SemanticKITTI, we delicately adjust the coefficients of the methods for better performance, as shown in Table 2.

We separate NRS dataset into 3 scenario types: Easy, Medium and Hard. Easy type only consists of some sparse road participates without the vertical structure either, Fig. 6(a); Medium type has a dense road participates distribution, but there are not many vertical structures in this type of point clouds, Fig. 6(b); Hard type point cloud has crowded road participates and also objects with vertical structures such as trees and cars below, Fig. 6(c).

Since the NRS dataset does not have semantic segmentation labels, we cannot train a good segmentation network for pre-processing. So we choose a free-trained ground removing method to pre-process the point cloud.

Table 3 reports the comparison results as well as the processing speed of the group. In the Easy scenarios, all the methods have similar precision rate because of the dense point clouds but sparse objects. In the medium and hard scenarios, the precision gap between ours and the other methods becomes larger. Euclidean Cluster, Supervoxel Cluster and Scan-line Run suffer from under-segmentation caused by the background points (trees, traffic lights and etc.). Depth Cluster shows higher over-segmentation rate because smaller objects near the LiDAR would block the laser beams and truncate the big objects, which cannot be handled by this angle based clustering method. While being beneficial from the view combination in MVC, the performance of our method experiences a slight going down but still remain a decent precision rate.

Besides, we also report the clustering speed by frame-per-second (FPS) of all the methods in these three scenarios without taking the pre-processing stage of ground points removal into account. As the amount of points becomes larger, our method stably runs in a high speed at about 10 ms per frame, which satisfies the real-time requirement.



(a) Easy (b) Medium (c) Hard

**Fig. 6.** A demonstration of the three scenario types in NRS dataset.

**Table 3.** Segmentation results on NRS dataset

| Scenarios | Algs | OS | US | TP | OSR | USR | P | FPS |
|---|---|---|---|---|---|---|---|---|
| Easy | Ours | 34 | 44 | 1088 | 0.030 | 0.039 | **0.933** | **110** |
| | Euclidean cluster | 15 | 108 | 965 | **0.015** | 0.100 | 0.887 | 73 |
| | Supervoxel cluster | 18 | 143 | 927 | 0.019 | 0.134 | 0.852 | 24 |
| | Depth cluster | 109 | 31 | 948 | 0.103 | **0.032** | 0.871 | 70 |
| | Scan-line run | 75 | 35 | 978 | 0.071 | 0.035 | 0.899 | 54 |
| Medium | Ours | 104 | 217 | 2908 | 0.035 | 0.069 | **0.900** | **100** |
| | Euclidean cluster | 85 | 322 | 2822 | 0.029 | 0.102 | 0.874 | 67 |
| | Supervoxel cluster | 73 | 419 | 2737 | **0.026** | 0.133 | 0.848 | 16 |
| | Depth cluster | 346 | 90 | 2793 | 0.110 | **0.031** | 0.865 | 63 |
| | Scan-line run | 114 | 263 | 2852 | 0.084 | 0.038 | 0.883 | 54 |
| Hard | Ours | 190 | 407 | 2922 | 0.061 | **0.122** | **0.830** | **98** |
| | Euclidean cluster | 242 | 811 | 2276 | 0.096 | 0.263 | 0.684 | 63 |
| | Supervoxel cluster | 105 | 1126 | 2098 | **0.048** | 0.349 | 0.630 | 12 |
| | Depth cluster | 597 | 485 | 2248 | 0.210 | 0.177 | 0.675 | 60 |
| | Scan-line run | 383 | 391 | 2555 | 0.130 | 0.133 | 0.767 | 54 |

### 4.3 Ablation Study

The ablation study mainly focuses on assessing the clustering process in different views. We evaluate the clustering performance and running speed in this experiment and report the statistics in Table 4.

**Table 4.** The ablation study of different modules in MVC

| Group | OS | US | TP | OSR | USR | P | FPS |
|---|---|---|---|---|---|---|---|
| 1. BEV(PG-DBSCAN)+FV | 104 | 217 | 2908 | 0.035 | 0.069 | **0.900** | 100 |
| 2. BEV(PG-DBSCAN) | 63 | 1052 | 2014 | **0.028** | 0.332 | 0.655 | **250** |
| 3. BEV(DBSCAN [6])+FV | 97 | 236 | 2886 | 0.032 | 0.075 | 0.897 | 2 |
| 4. FV | 328 | 81 | 2820 | 0.104 | **0.028** | 0.873 | 18 |

**FV Clustering Module.** As shown in Table 4, we compare the performance of FV clustering removed MVC (Group 2) with the original MVC (Group 1). Without the FV clustering, the number of over-segmentation vehicles slightly goes down and OSR remains steady, however, the amount of under-segmentation rate increases sharply. Because, in some scenes, background points may combine different target objects together, as show in Fig 7. Without FV clustering process, the points from these objects are clustered into the same cloud and cause under-segmentation.

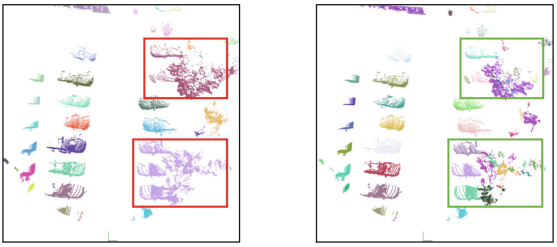


**Fig. 7.** Left image shows only the BEV clustering result, and the right image is the result of whole MVC algorithm

**BEV Clustering Module.** In this part, we analyse the importance of BEV clustering process. Comparing Group 1 with Group 4 in Tab. 4, we conclude that PG-DBSCAN efficiently accelerates the clustering process and slightly improve clustering precision.

Also, from Group 3 and 1, we can see that, with nearly same clustering precision, the running speed of MVC with traditional DBSCAN [6] is 50 times slower than that with PG-DBSCAN.

## 5 Conclusion

In this paper, a multi-view based clustering method is proposed for the 3D point cloud. The algorithm adopts the coarse-to-fine architecture. First, the non-ground point cloud is projected to the BEV density image and down-sampled. We propose PG-DBSCAN based on the traditional DBSCAN [6] for the preliminary segmentation. Then we further separate the under-segmented clusters on vertical direction based on range images. We compare our method with 4 traditional clustering algorithms on both SemanticKITTI and NRS dataset. The experiment results show the real-time performance, stability and accuracy of the MVC algorithm, and prove that this method is suitable for clustering the dense point clouds in various driving scenes.

## References


1. Behley, J., et al.: Semantickitti: a dataset for semantic scene understanding of lidar sequences. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9297–9307 (2019)
2. Bogoslavskyi, I., Stachniss, C.: Fast range image-based segmentation of sparse 3D laser scans for online operation. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 163–169. IEEE (2016)
3. Burger, P., Wuensche, H.J.: Fast multi-pass 3D point segmentation based on a structured mesh graph for ground vehicles. In: 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 2150–2156. IEEE (2018)
4. Cheng, Y.: Mean shift, mode seeking, and clustering. IEEE Trans. Pattern Anal. Mach. Intell. **17**(8), 790–799 (1995)
5. Douillard, B., et al.: Hybrid elevation maps: 3D surface models for segmentation. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1532–1538. IEEE (2010)
6. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, vol. 96, pp. 226–231 (1996)
7. Gasperini, S., Mahani, M.A.N., Marcos-Ramiro, A., Navab, N., Tombari, F.: Panoster: end-to-end panoptic segmentation of lidar point clouds. IEEE Robot. Autom. Lett. **6**(2), 3216–3223 (2021)
8. Hasecke, F., Hahn, L., Kummert, A.: Fast lidar clustering by density and connectivity. arXiv e-prints pp. arXiv-2003 (2020)
9. Himmelsbach, M., Hundelshausen, F.V., Wuensche, H.J.: Fast segmentation of 3D point clouds for ground vehicles. In: 2010 IEEE Intelligent Vehicles Symposium, pp. 560–565. IEEE (2010)
10. Hong, F., Zhou, H., Zhu, X., Li, H., Liu, Z.: Lidar-based panoptic segmentation via dynamic shifting network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13090–13099 (2021)
11. Klasing, K., Wollherr, D., Buss, M.: A clustering method for efficient segmentation of 3D laser data. In: 2008 IEEE International Conference on Robotics and Automation, pp. 4043–4048. IEEE (2008)
12. Li, M., Yin, D.: A fast segmentation method of sparse point clouds. In: 2017 29th Chinese Control And Decision Conference (CCDC), pp. 3561–3565. IEEE (2017)
13. Li, Y., Le Bihan, C., Pourtau, T., Ristorcelli, T.: Insclustering: instantly clustering lidar range measures for autonomous vehicle. In: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), pp. 1–6. IEEE (2020)
14. Moosmann, F., Pink, O., Stiller, C.: Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion. In: 2009 IEEE Intelligent Vehicles Symposium, pp. 215–220. IEEE (2009)
15. Park, S., Wang, S., Lim, H., Kang, U.: Curved-voxel clustering for accurate segmentation of 3D lidar point clouds with real-time performance. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 6459–6464. IEEE (2019)
16. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 652–660 (2017)
17. Shin, M.O., Oh, G.M., Kim, S.W., Seo, S.W.: Real-time and accurate segmentation of 3-D point clouds based on gaussian process regression. IEEE Trans. Intell. Transp. Syst. **18**(12), 3363–3377 (2017)

18. Sirohi, K., Mohan, R., Büscher, D., Burgard, W., Valada, A.: Efficientlps: efficient lidar panoptic segmentation. IEEE Trans. Robot. (2021)
19. Thrun, S., Montemerlo, M., Aron, A.: Probabilistic terrain analysis for high-speed desert driving. In: Robotics: Science and Systems, pp. 16–19 (2006)
20. Urmson, C., et al.: High speed navigation of unrehearsed terrain: Red team technology for grand challenge 2004. Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-04-37 1 (2004)
21. Wen, M., Cho, S., Chae, J., Sung, Y., Cho, K.: Range image-based density-based spatial clustering of application with noise clustering method of three-dimensional point clouds. Int. J. Adv. Robot. Syst. **15**(2), 1729881418762302 (2018)
22. Zermas, D., Izzat, I., Papanikolopoulos, N.: Fast segmentation of 3D point clouds: a paradigm on lidar data for autonomous vehicle applications. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 5067–5073. IEEE (2017)
23. Zhao, Y., Zhang, X., Huang, X.: A technical survey and evaluation of traditional point cloud clustering methods for lidar panoptic segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 2464–2473 (2021)
24. Zhou, H., et al.: Cylinder3d: An effective 3D framework for driving-scene lidar semantic segmentation. arXiv preprint arXiv:2008.01550 (2020)