2026 IEEE/SICE International Symposium on System Integration (SII)
January 11-14, 2026. Cancún, México

3D Object Segmentation Considering Density Variation and Scanning Order of Point Clouds

Takuma Tanaka*

* Graduate School of Science and Technology, Meiji University, Kanagawa, Japan

Yoshitaka Hara†

† Future Robotics Technology Center (fuRo), Chiba Institute of Technology, Chiba, Japan

Yoji Kuroda‡

‡ School of Science and Technology, Meiji University, Kanagawa, Japan

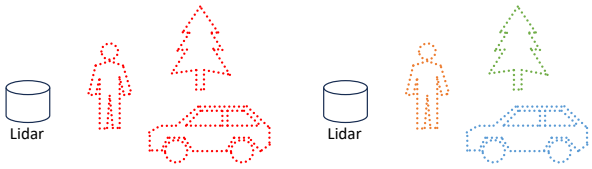
***Abstract*— In this paper, we propose a novel DBSCAN method for 3D object segmentation that considers point cloud density based on lidar range and scanning order of laser beams. Point clouds obtained by a lidar are dense at short ranges and sparse at long ranges. Furthermore, point clouds obtained by the lidar are organized in the scanning order of each beam. However, conventional DBSCAN has issues that its parameters are fixed and cannot adapt to density variation of point clouds, and that the computational costs for neighborhood search are high, resulting in significant processing time. Therefore, in the proposed method, parameters are adjusted according to lidar ranges to accommodate density differences. Furthermore, executing neighborhood search within only a specific beam scanning area reduces computational costs and shortens processing time. We conducted experiments in multiple environments to verify the effectiveness of the proposed method. By automatically determining parameters based on the point cloud density, the proposed method demonstrated that both nearby and distant objects can be correctly segmented. Additionally, our method demonstrated that processing time can be reduced by executing neighborhood search within only a specific scanning area of the beams. As described above, the proposed method achieves adaptive and high-speed object segmentation by considering both the differences in point cloud density due to lidar ranges and the beam scanning order.**

I. Introduction

Point cloud data obtained by light detection and ranging (lidar) is used in a wide range of fields, including robot environment recognition. One of the methods for processing point cloud data is 3D object segmentation.

Fig. 1 shows the concept of object segmentation for 3D point clouds. Object segmentation allows 3D point clouds to be separated by objects, such as pedestrians or trees. 3D point clouds separated by objects are useful for robot path planning and trajectory prediction of moving obstacles.

DBSCAN [1] is a typical object segmentation method for 3D point clouds and is a density-based algorithm. It treats high-density point clouds as segments and low-density point clouds as noise. Two parameters are used for DBSCAN object segmentation: eps [m] and min_points [num.]. Eps is a neighborhood distance of a query point, and min_points is a minimum number of neighborhood points required for a query point to form a segment. When performing object segmentation, these parameters are fixed for the entire point cloud.



(a) 3D Lidar Point Cloud (b) Object Segmentation

Fig. 1. Concept of object segmentation for 3D point clouds. Measured lidar point clouds are not segmented. Segmentation divides point clouds into each object.

One problem with using DBSCAN on point clouds obtained by a lidar is that the optimal parameters are different for nearby and distant objects. Point clouds obtained by the lidar have the characteristics that the point clouds are dense at short ranges and sparse at long ranges. Therefore, parameters adapted to nearby objects make the object segmentation of distant objects difficult, and parameters adapted to distant objects make the object segmentation of nearby objects difficult.

DBSCAN also has another problem of high computational costs. DBSCAN executes neighborhood search for query points by searching all points. Therefore, the search is executed even for points distant from the query point. As a result, unnecessary search is executed, which increases the computational costs.

In this paper, we propose Range DBSCAN, a novel method that achieves correct object segmentation regardless of the range of point clouds obtained by the lidar. By automatically determining the neighborhood point distance threshold eps based on the lidar measurement range, it can handle density variation of point clouds. Furthermore, to reduce computational costs, neighborhood search is executed only within a specific beam scanning area of the lidar.

The contributions of this paper are as follows:

- By automatically determining the DBSCAN threshold eps based on the lidar range, adaptive object segmentation is achieved that deals with density variation of 3D point clouds.
- By executing neighborhood search only within the point cloud of the specific beam scanning area of the lidar, computational costs are reduced to realize high-speed

978-1-6654-5784-2/26/$31.00 ©2026 IEEE

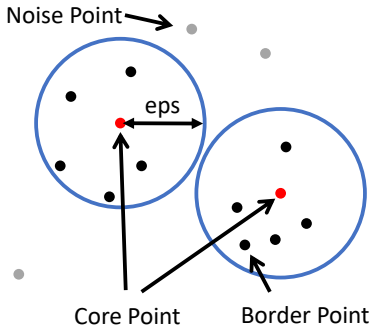


Fig. 2. DBSCAN performs segmentation by classifying each point into core points, border points, and noise points.

processing.

- Experiments show that the proposed method accurately performs object segmentation at various ranges and that processing time is reduced.

II. Related Work

A. Object Segmentation and Detection

Examples of major methods for object segmentation in 3D point clouds include k-means [2], k-means++ [3], and X-means [4]. k-means performs object segmentation by updating centroids, which are randomly placed segment centers. k-means++ is a method for efficiently placing initial positions of centroids. k-means and k-means++ require the number of segments to be predetermined. Whereas X-means is an object segmentation method that automatically determines the number of segments. However, these methods have difficulty with non-spherical object segmentation.

In the Euclidean object segmentation implemented in PCL [5], the point clouds are compared using the Euclidean distance. However, since the density of the point cloud is not considered, the segment is extended whenever even one neighborhood point exists near a query point. This means it is less able to deal with noise compared to DBSCAN.

Methods for object detection based on machine learning have also been proposed, including PointNet [6], PointNet++ [7], PointPillars [8], CenterPoint [9], CenterFormer [10], Point Transformer V3 [11]. These methods require pre-training on target objects to be recognized. In contrast, the proposed method can be used to segment any object and does not require any pre-training.

B. DBSCAN

Fig. 2 shows the points used for DBSCAN [1]. DBSCAN is a density-based object segmentation method. Three types of points are used for object segmentation: core points, border points, and noise points. Core points are points that have at least min_points neighborhood points in eps. Border points are points included in the neighborhood of core points, but whose own neighborhood contains less than min_points points. Noise points are points other than core and border points.

Fig. 3 shows the pipeline of DBSCAN processing. The following steps are used for object segmentation. First, a point is selected and set as the query point. Then, neighborhood search is executed on an entire point cloud. Points whose distance from the query point is less than eps are considered as neighborhood points. If the number of neighborhood points is greater than min_points, it is assumed to be a core point, and neighborhood points are included in the same segment. These neighborhood points are again selected as query points, and neighborhood search is executed to expand a segment. Even if the number of neighborhood points is fewer than min_points, points included in the neighborhood of core points are treated as border points and added to the same segment. Points that do not fall into either category are treated as noise points.

DBSCAN parameters must be set manually. The appropriate parameters depend on the density of point clouds.

C. Improvements of DBSCAN

Wang's method [12] estimates appropriate parameters for DBSCAN based on the overall shape of the point cloud. Calculate D_{k}, which is the maximum distance of k nearest neighbors for all points, and obtain the curve plotted by D_{k}. The boundary between dense segment areas and sparse noise areas is estimated from changes in the slope of the curve, and the optimal eps is determined. By estimating parameters in this way, it is possible to automatically determine DBSCAN parameters after statistically examining the overall structure of the point cloud. However, since eps is calculated using the same value for all points, it is difficult to perform correct object segmentation on point clouds with uneven density, such as a point cloud obtained by the lidar.

In Yabroudi's method [13], point clouds obtained by the lidar are divided into cells by area, and parameters are estimated for each cell. Specifically, it removes background elements such as walls and floors from the obtained point cloud, then divides it horizontally and vertically to assign it to cells. Evaluate the point cloud density for each cell, and then adaptively estimate eps and min_points. When processing DBSCAN, object segmentation is performed by setting parameters for each cell. This enables adaptation to density differences caused by the range of the lidar. However, there is the problem that parameters must be calculated for each cell, making the process complicated.

III. DBSCAN Considering Density Variation and Scanning Order of Point Clouds

A. Overview

Fig. 4 shows the processing pipeline of the proposed method, Range DBSCAN. By adjusting eps for each query point, it is possible to execute neighborhood search while accounting for local density differences. In addition, neighborhood search is executed only on the point cloud within a specific beam scanning area, not on the entire point cloud. By executing neighborhood search only within the scanning area of the specified beams, it is possible to omit search for points distant from the query point. As a result, it is possible to reduce the amount of computation required for neighborhood search.

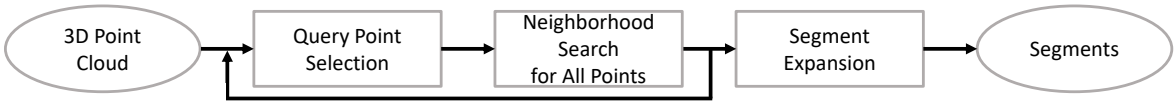


Fig. 3. Pipeline of DBSCAN (conventional method). Segmentation parameters are fixed for all points. Neighborhood search is executed for all points.

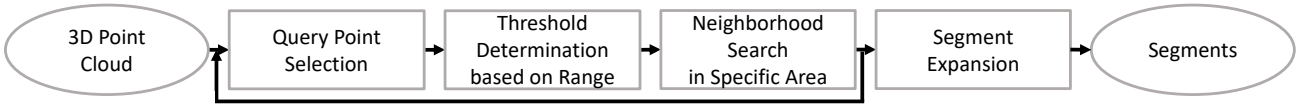


Fig. 4. Pipeline of Range DBSCAN (ours). Segmentation parameters are determined based on the range of each point. Neighborhood search is executed only within the specific area.

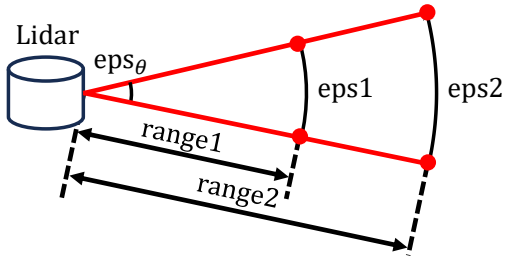


Fig. 5. Range DBSCAN automatically adjusts the threshold eps based on the measurement range of the lidar.

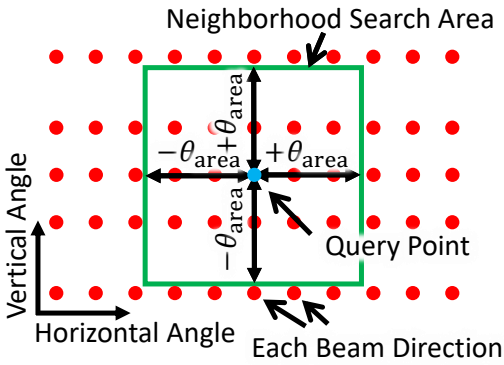


Fig. 6. Neighborhood search of Range DBSCAN is executed only within the specific area based on the query point.

B. Threshold Determination based on Range

In the proposed Range DBSCAN method, the eps used for neighborhood search is adjusted based on the measured range of each query point. Set an eps to a small value for areas with small ranges and high point cloud density, and set an eps to a large value for areas with large ranges and low point cloud density. As a result, changes in density caused by differences in range can be accounted for. This allows for the proper object segmentation of point clouds, both nearby and distant from the lidar.

Fig. 5 shows how to determine the threshold eps used in the proposed Range DBSCAN method. The calculation of an eps is based on the relationship between the central angle and the arc length of a sector.

A range from the lidar to a query point is range [m], a central angle is eps_θ [rad], and a constant is eps_{base} [m]. An eps is calculated using Eq. (1).

$$\text{eps} = \text{range} \times \text{eps}_\theta + \text{eps}_{\text{base}} \tag{1}$$

Here, a central angle eps_θ and a constant eps_{base} are parameters.

C. Neighborhood Search in Specific Area

Fig. 6 shows the concept of neighborhood search within only the scanning area of the specific beams. In the proposed Range DBSCAN method, neighborhood search is only executed on points obtained from beams within a specific scanning area of the beams that obtained the query point. A point cloud obtained by the lidar is organized in the order of beam scanning in angular space. It is possible to reduce the amount of computational costs by performing search only within a specific scanning area and omitting neighborhood search distant from the query point.

Using eps_θ used in the eps calculatin and the coefficient α, the search area θ_{area} [rad] is determined from Eq. (2).

$$\theta_{\text{area}} = \alpha \times \text{eps}_\theta \tag{2}$$

Here, the coefficient α is a parameter.

IV. Experiments

A. Experiment Conditions

Experiments were conducted at two locations: Meiji University and Tsukuba park. At Meiji University, data collection was performed using a stationary robot. Meanwhile, at Tsukuba park, the robot acquired data while moving. We obtained point cloud data from the 3D lidar in each environment. Then, we compared object segmentation results and processing time of Range DBSCAN with those of conventional DBSCAN.

Velodyne VLP-32C was used for the 3D lidar. The number of laser channel was 32, acquiring 57,600 points per scan. The scan was 10 Hz. When performing object segmentation, ground point clouds were removed using a height threshold.

We used the conventional DBSCAN implemented in Open3D [14] [15]. DBSCAN neighborhood search in

Open3D utilizes KD-Trees of nanoflann [16] for radius search.

We set the value for min_points, one of DBSCAN parameters, to 4. This is because the original DBSCAN [1] showed that the optimal value for min_points is 4, and increasing this value beyond 4 has no significant effect on performance.

Parameters used in Range DBSCAN were eps_θ = 0.03 rad, eps_{base} = 0.5 m, α = 1.3. These values were determined experimentally.

B. Object Segmentation

Fig. 7 shows the object segmentation results at Meiji University. Fig. 7 (a) shows an aerial photograph of the experimental environment, Fig. 7 (b) shows the result of Range DBSCAN (ours), Fig. 7 (c) shows the result of DBSCAN (conventional method) with eps = 0.5 m, and Fig. 7 (d) shows the result of DBSCAN (conventional method) with eps = 1.5 m.

Fig. 7 (b) indicates that Range DBSCAN correctly segmented objects, including pedestrians and cars near the lidar as well as trees far from the lidar. Fig. 7 (c) indicates that, when using DBSCAN, pedestrians and cars near the lidar were correctly segmented as objects when the eps value was small. However, trees far from the lidar were not correctly segmented as objects. Fig. 7 (d) indicates that, in DBACAN, pedestrians and cars near the lidar were incorrectly segmented as a single object when the eps value was large. In contrast, trees far from the lidar were correctly segmented.

Fig. 8 shows the object segmentation results at Tsukuba park. Fig. 8 (a) shows an aerial photograph of the experimental environment, Fig. 8 (b) shows the result of Range DBSCAN (ours), Fig. 8 (c) shows the result of DBSCAN (conventional method) with eps = 0.5 m, and Fig. 8 (d) shows the result of DBSCAN (conventional method) with eps = 1.5 m.

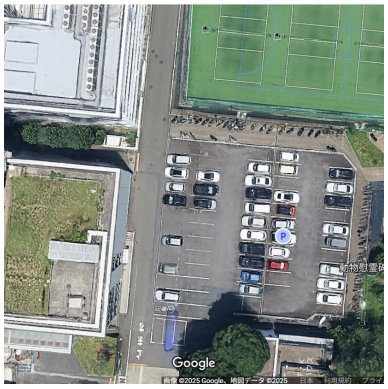
Fig. 8 indicates that, unlike DBSCAN, Range DBSCAN can correctly segment both nearby and distant objects, similar to Fig. 7. These experimental results showed that Range DBSCAN can perform more adaptive object segmentation than DBSCAN by considering the density variation of point clouds according to the measured range.

C. Processing Time

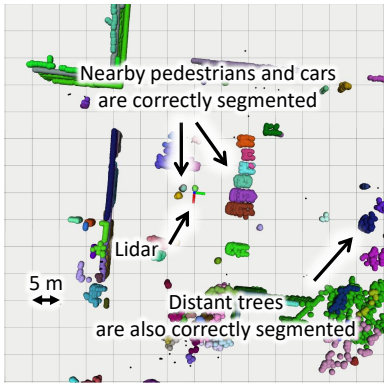
Fig. 9 shows the processing time for object segmentation using DBSCAN and Range DBSCAN at Meiji University, and Fig. 10 shows those at Tsukuba park. The average number of processed point clouds in the experiment at Meiji University was 46,580, while at Tsukuba park it was 41,867. In this experiment, the parameter eps = 1.0 m was used for DBSCAN.

In each environment, the processing time was measured 10 times, and the average was calculated. Data processing was performed using a PC equipped with CPU: Intel Core i9 10850K 3.60 GHz and RAM: 64 GB.

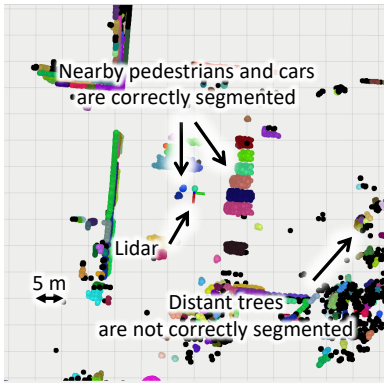
Fig. 9 indicates that in experiments conducted at Meiji University, the processing time for DBSCAN without parallelization was 486 ms, the processing time for DBSCAN



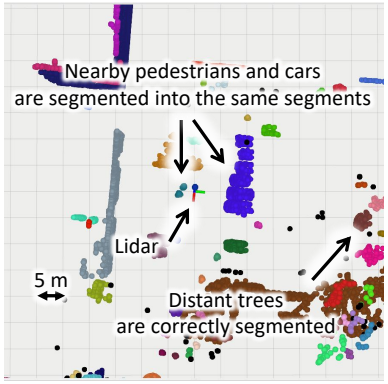
(a) Aerial Photograph



(b) Range DBSCAN (ours)
eps_θ = 0.03 rad

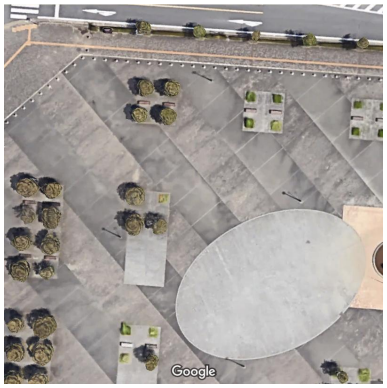


(c) DBSCAN (conventional method)
eps = 0.5 m

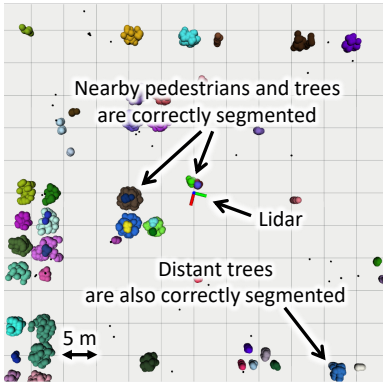


(d) DBSCAN (conventional method)
eps = 1.5 m

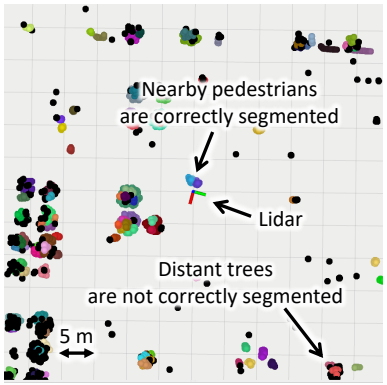
Fig. 7. Object segmentation results of experiments at Meiji University.



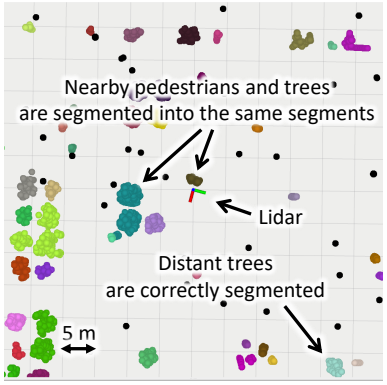
(a) Aerial Photograph



(b) Range DBSCAN (ours)
eps_θ = 0.03 rad



(c) DBSCAN (conventional method)
eps = 0.5 m



(d) DBSCAN (conventional method)
eps = 1.5 m

Fig. 8. Object segmentation results of experiments at Tsukuba park.

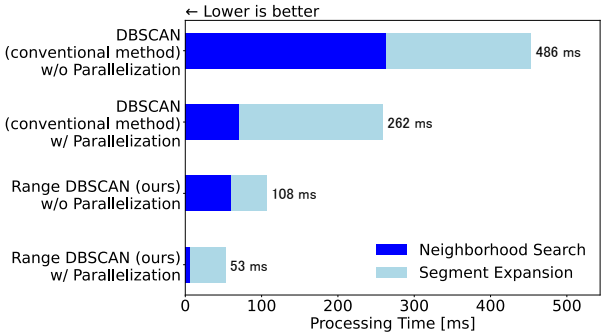


Fig. 9. Processing time of experiments at Meiji University.

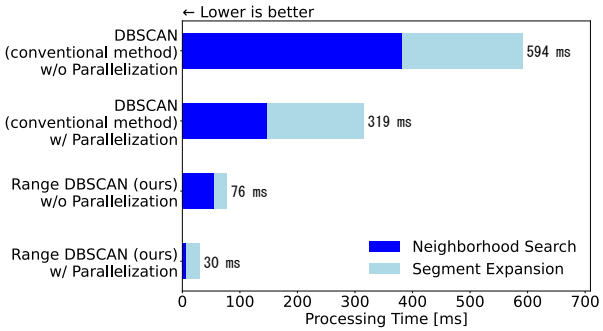


Fig. 10. Processing time of experiments at Tsukuba park.

with parallelization was 262 ms, the processing time for Range DBSCAN without parallelization was 108 ms, and the processing time for Range DBSCAN with parallelization was 53 ms. At Meiji University, the processing time for the parallelized Range DBSCAN was reduced by approximately 89% compared to the non-parallelized DBSCAN.

Fig. 10 indicates that in experiments conducted at Tsukuba park, the processing time for DBSCAN without parallelization was 594 ms, the processing time for DBSCAN with parallelization was 319 ms, the processing time for Range DBSCAN without parallelization was 76 ms, and the processing time for Range DBSCAN with parallelization was 30 ms. At Tsukuba park, the processing time for the parallelized Range DBSCAN was reduced by approximately 95% compared to the non-parallelized DBSCAN.

V. Conclusion

In this paper, we proposed Range DBSCAN, a novel method that achieves correct object segmentation regardless of the range of point clouds obtained by the lidar. By automatically determining the neighborhood point distance threshold eps based on the lidar measurement range, it could handle density variation of point clouds. Furthermore, to reduce computational costs, neighborhood search was executed only within a specific beam scanning area of the lidar.

Experiments have demonstrated that the proposed method can deal with density variation of 3D point clouds, enabling adaptive object segmentation at various ranges. In addition, we demonstrated that by executing neighborhood search

within only a specific beam scanning area, computational costs can be reduced and the processing time for object segmentation can be shortened.

In the future, we plan to develop a system capable of performing object segmentation that can handle more complex environmental shapes and to perform research applying this technology to tasks such as predicting the trajectories of moving obstacles.

ACKNOWLEDGMENT

This study was conducted under the Research Cluster for Autonomous Robotic Systems at Meiji University. The author wishes to express sincere gratitude here.

REFERENCES

[1] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu: "A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", *Proc. of Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, 1996.

[2] James B. MacQueen: "Some Methods for Classification and Analysis of Multivariate Observations", *Proc. of Berkeley Sympo. on Mathematical Statistics and Probability*, 1967.

[3] David Arthur and Sergei Vassilvitskii: "k-means++: The Advantages of Careful Seeding", *Proc. of Annual ACM-SIAM Sympo. on Discrete Algorithms (SODA)*, 2007.

[4] Dan Pelleg and Andrew W. Moore: "X-means: Extending k-means with Efficient Estimation of the Number of Clusters", *Proc. of Int. Conf. on Machine Learning (ICML)*, 2000.

[5] Radu B. Rusu and Steve Cousins: "3D is Here: Point Cloud Library (PCL)", *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[6] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas: "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation", *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[7] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas: "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space", *Proc. of Annual Conf. on Neural Information Processing Systems (NeurIPS)*, 2017.

[8] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom: "PointPillars: Fast Encoders for Object Detection from Point Clouds", *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[9] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl: "Center-based 3D Object Detection and Tracking", *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[10] Zixang Zhou, Xiangchen Zhao, Yu Wang, Panqu Wang, and Hassan Foroosh: "CenterFormer: Center-based Transformer for 3D Object Detection", *Proc. of European Conf. on Computer Vision (ECCV)*, 2022.

[11] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao: "Point Transformer V3: Simpler, Faster, Stronger", *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[12] Chunxiao Wang, Min Ji, Jian Wang, Wei Wen, Ting Li, and Yong Sun: "An Improved DBSCAN Method for LiDAR Data Segmentation with Automatic Eps Estimation", *Sensors*, vol. 19, no. 1, pp. 172–197, 2019.

[13] Mohammad El Yabroudi, Khalfalla Awedat, Rakan C. Chabaan, Osama Abudayyeh, and Ikhlas Abdel-Qader: "Adaptive DBSCAN LiDAR Point Cloud Clustering for Autonomous Driving Applications", *Proc. of IEEE Int. Conf. on Electro Information Technology (eIT)*, 2022.

[14] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun: "Open3D: A Modern Library for 3D Data Processing", *arXiv:1801.09847*, 2018.

[15] Open3D: A Modern Library for 3D Data Processing. `https://github.com/isl-org/Open3D`

[16] nanoflann: a C++11 header-only library for Nearest Neighbor (NN) search with KD-trees. `https://github.com/jlblancoc/nanoflann`