Hindawi
Mobile Information Systems
Volume 2022, Article ID 1611097, 9 pages
https://doi.org/10.1155/2022/1611097

Research Article

3D Object Detection Algorithm Based on the Reconstruction of Sparse Point Clouds in the Viewing Frustum

Xing Xu,[1] **Xiang Wu**,[2] **Yun Zhao**,[1] **Xiaoshu Lü**,[3,4] **and Aki Aapaoja**[5]

[1]*School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China*
[2]*School of Mechanical and Energy Engineering, Zhejiang University of Science and Technology, HangZhou 310023, China*
[3]*Department of Electrical Engineering and Energy Technology, University of Vaasa, Vaasa 65101, Finland*
[4]*Department of Civil Engineering, Aalto University, Espoo 02130, Finland*
[5]*Solita Ltd., Oulu 90100, Finland*

Correspondence should be addressed to Yun Zhao; zy_super0201@163.com

Received 7 December 2021; Revised 22 August 2022; Accepted 30 September 2022; Published 15 October 2022

Academic Editor: Adrian Kliks

In response to the problem that the detection precision of the current 3D object detection algorithm is low when the object is severely occluded, this study proposes an object detection algorithm based on the reconstruction of sparse point clouds in the viewing frustum. The algorithm obtains more local feature information of the sparse point clouds in the viewing frustum through dimensional expansion, performs the fusion of local and global feature information of the point cloud data to obtain point cloud data with more complete semantic information, and then applies the obtained data to the 3D object detection task. The experimental results show that the precision of object detection in both 3D view and BEV (Bird's Eye View) can be improved effectively through the algorithm, especially object detection of moderate and hard levels when the object is severely occluded. In the 3D view, the average precision of the 3D detection of cars, pedestrians, and cyclists at a moderate level can be increased by 7.1p.p., 16.39p.p., and 5.42p.p., respectively; in BEV, the average precision of the 3D detection of car, pedestrians, and cyclists at hard level can be increased by 6.51p.p., 16.57p.p., and 7.18p.p., respectively, thus indicating the effectiveness of the algorithm.

1. Introduction

In recent years, with the continuous development of electric vehicles, automated driving technology has been tending to maturity, and 3D object detection has also become particularly important. In actual traffic scenes, various problems will be encountered, such as variations in illumination and bad weather conditions, which will cause the failure to fully perceive objects. If automated vehicles wanted to be reliable and safe, their observation systems must have at least the following features: high accuracy, high certainty, and high reliability. Nowadays, single sensor systems (e.g., radar, camera, and LiDAR) cannot provide this, and thus, sensor fusion [1] (e.g., combining camera and LiDAR) is needed. For example, LiDARs are excellent at providing rich, depth, and accurate information without interference from the light environment. On the contrary, camera systems are more cost-effective and capable of providing effective visual recognition when color or texture attributes have challenges in long-detection range, field-of-view, and dim conditions [2]. Hence, compared with traditional 2D object detection [3], the LiDAR point cloud data used for 3D object detection have richer semantic information. However, since the point clouds are discrete and random and possibly sparse, it is challenging to detect object instances from point clouds.

At present, for the processing of point cloud data, Chen et al. [4] proposed Multi-View 3D networks (MV3D) which can convert the 3D point clouds into 2D images through the projection of different views and then fuse them with RGB images to obtain the 3D bounding boxes. However, the networks use the views converted from point cloud data, not the original point cloud data, which will cause the loss of some information. In order to solve the problem of information loss, Yin and Tuzel [5] proposed VoxelNet which

uses VFE (voxel feature encoding) network to encode the points in each voxel and then performs 3DRPN(3D Region Proposal Network) detection without any artificial processing such as projection. However, due to the large point cloud data in space and the 3D convolution involved in the convolutional layer of the neural network, the computational burden is heavy, and the requirements for real-time detection in automatic drive scenes could not be met. Ku et al. [6] proposed the AVOD algorithm which takes RGB images and BEV (Bird's Eye View) as input, uses the FPN (Feature Pyramid Networks) to obtain the feature images of the two, and then performs feature fusion of the two to generate the 3D box estimation of the object. Since the algorithm uses low-level and high-level semantic information, it can improve the object detection effect, especially for small objects. However, due to poor point cloud processing, part of the point cloud information is lost, resulting in only a moderate detection effect for pedestrians and cyclists. Qi et al. [7] proposed a 3D object detection algorithm PointNet which can directly consume point cloud data, ensure the rotational invariance of the point cloud by multiplying each point with a transformation matrix, use multilayer perceptron to generate the global feature, and finally generate the 3D bounding box of the object. Although PointNet can make full use of the semantic information of point clouds, it could not process the local information of the point clouds well. On this basis, Qi et al. [8] proposed PointNet++ which can iteratively extract features from local areas of the point cloud with a density-adaptive feature extraction method and can learn deep point set features efficiently and robustly.

The Frustum-Pointnets [9] model is used in this study; that is, a 2D bounding box is generated through relatively mature 2D object detection at first; then, the viewing frustum is formed according to the positions of the camera and the 2D bounding box, and then, 3D object detection is performed for the original point cloud data within the viewing frustum. Through this method, the efficiency of 3D object detection can be greatly improved and the computational burden can be reduced.

The 3D point cloud reconstruction approaches can be mainly divided into the completion approach based on geometrical relationships and the complete approach based on deep learning.

The completion approach based on geometrical relationship is often to repair and complete incomplete 3D geometry information by adding various constraints. Kazhdan and Hoppe [10] proposed the approach of Poisson surface reconstruction which adopts the strategy of incorporating the points as interpolation constraints and obtains the surface reconstruction of the underlying 3D model by solving the Poisson equation. The completion approach based on geometrical relationship often requires that the missing part of the 3D model to be completed could not be too large; otherwise, the shape information of the part could not be inferred from the geometrical information of the model surface or adjacent parts. Pauly et al. [11] proposed the use of a 3D model database for providing geometrical information for the missing data region: similar models can be found in the database, and then, a complete 3D model can be obtained by mixing and distorting the models. Chaudhuri et al. [12] proposed 3D model assembly by learning the relationship between semantic encoding and geometrical model components. However, these approaches are more dependent on the number of models in the database. Alhashim et al. [13] proposed an algorithm for generating novel 3D models via topology-varying shape blending. Given a source and a target shape, the method can blend them topologically and geometrically, producing continuous series of in-betweens as new shape creations.

The completion approach based on deep learning is often used to restore 3D models using various types of deep neural networks. Dosovitskiy et al. [14] trained generative 'up-convolutional' neural networks on rendered 3D models of chairs, tables, and cars, allowing them to interpolate between given views to generate the missing ones. Firman et al. [15] proposed an algorithm that can complete the unobserved geometry of tabletop-sized objects, based on a supervised model trained on already available volumetric elements. The model maps from a local observation in a single-depth image to an estimate of the surface shape in the surrounding neighborhood. Soltani et al. [16] proposed the following approach: learning a generative model over multiview depth maps or their corresponding silhouettes and using a deterministic rendering function to produce 3D shapes from these images. They also demonstrated that their model has out-of-sample generalization power for real-world tasks with occluded objects. However, these networks are mainly based on voxel data input, rather than direct processing of point cloud data, which will easily lead to local information loss of the point clouds. Sarmad et al. [17] presented RL-GAN-Net, where a reinforcement learning (RL) agent provides fast and robust control of a generative adversarial network (GAN). This is the first attempt to train an RL agent to control the GAN, which effectively learns the highly nonlinear mapping from the input noise of the GAN to the latent space of the point cloud. In this case, the point cloud completion task can be completed without any prior knowledge about visibility or noise characteristics. Huang et al. [18] proposed a Point Fractal Network (PF-Net) which estimates the missing point cloud hierarchically by utilizing a feature-points-based multiscale generating network and adds up multistage completion loss and adversarial loss to generate more realistic missing region(s). Yu et al. [19] proposed a geometrically sensitive point cloud completion Transformer network (PoinTr) that adopts a Transformer Encoder-Decoder architecture for point cloud completion. By representing the point cloud as a set of unordered groups of points with position embeddings, they converted the point cloud to a sequence of point proxies and employ the transformers for the point cloud generation. At present, point cloud completion methods are mainly used in the 3D reconstruction direction of the target and are rarely used in 3D object detection.

In the scene of automatic drive, a point cloud missing may occur when the object is occluded or too far away [20]. Therefore, in order to obtain more accurate 3D object detection results, the deep learning method is adopted in this study to learn the multilevel features of each point in the

viewing frustum, different convolution operations are made use of for expansion in the feature space, and then the expanded features are decomposed and reconstructed into new point cloud data.

2. Modeling

The Frustum-Pointnets model is used in this study to directly process the original point clouds and achieve 3D object detection. The Frustum-Pointnets algorithm is a network framework for achieving end-to-end 3D object detection, as shown in Figure 1. At present, the 3D object detection algorithm which can achieve the fusion of images and point cloud data has become a hotspot in the academic research of the automatic drive field. However, the point cloud information collected by LiDAR is not complete [21]; reasons such as data missing, the occlusion of the object, insufficient threads of the LiDAR, and too far distance will cause the information loss of object point clouds [22], which will greatly affect the precision of the object detection algorithm [23]. Therefore, a 3D object detection algorithm based on sparse point cloud reconstruction in the viewing frustum is proposed in this study to improve the performance of object detection when the point clouds in the viewing frustum are severely occluded. Figure 2 shows the flow of the algorithm proposed in this study.

2.1. Frustum Point Cloud Region Generation. The overall structure of the 3D frustum point cloud region generation network is shown in Figure 3. First, we use mature 2D CNN for object detection in RGB images [24], obtain 2D bounding box region and object classification, and then use the known camera projection matrix to extend the 2D bounding box to the 3D point cloud data of the viewing frustum. The viewing frustum is normalized by rotating the frustum toward the central view so that the central axis of the viewing frustum is orthogonal to the image plane. Such a method can improve the rotational invariance of the algorithm. In addition, since the method can segment the point clouds outside the viewing frustum and process only the point clouds inside the viewing frustum, the computational efficiency can be improved significantly. The finally obtained 3D object frustum point cloud region is shown in Figure 4.

2.2. Reconstruction of Point Cloud Data in the Viewing Frustum. We adopt the deep learning approach and take the point clouds in the viewing frustum as input, assume n points, and make use of different convolution operations for expansion in the feature space through a 2-layer shared multilayer perceptron (MLP). The dimensions of the first layer are (128, 256), and the dimensions of the second layer are (516, 1024). In this case, the local feature information of the point clouds can be fully obtained, and then, the greatest feature of the dimensions of each point cloud data can be extracted through the maximum pooling layer. The shared MLP can play a very effective role in extracting the features of the point cloud data. Then, through the fusion of expanded local feature information and the global feature

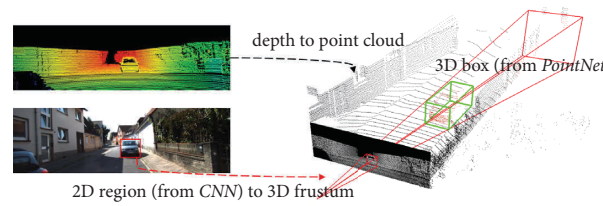

Figure 1: Overall framework of F-PointNet [9].

information obtained after the maximum pooling, the new point cloud features after reconstruction can be obtained via a layer of shared MLP with the dimensions of (1024, 1024), as shown in Figure 5.

2.3. 3D Instance Segmentation for Point Cloud Data in the Viewing Frustum. The PointNet is used for 3D instance segmentation, and the fusion of the local features and global features of the point clouds in the viewing frustum can be achieved so that the features of the point clouds can be better extracted and more accurate semantic information can be obtained. The structure is shown in Figure 6.

2.4. 3D Object Bounding Box Regression. After 3D instance segmentation, points classified as objects are extracted. Then, these segmented object points are obtained, and their coordinates are further normalized to enhance the translation invariance of the algorithm, following the same basic principle in obtaining the frustum point clouds. By subtracting the XYZ value from the centroid, the point cloud is converted into local coordinates, a special type of spatial transformation network [25]. T-Net is used to estimate the true center of the entire object, and then, the coordinates are transformed to make the predicted center become the origin. The 3D object detection is performed on the object point clouds in the viewing frustum, and the estimation of the 3D object bounding box of the object instance point clouds is achieved by PointNet and residual regression. The regression parameters include the center coordinates, length, width, and height of the 3D object bounding box [26], and the heading angle of the object [27]. The structure is shown in Figure 7.

2.5. Loss Function. The improved Frustum-PointNets model uses the same multitask loss function "Lmulti-task" as the original network, which includes the 3D object prediction loss function "Lmask" and the 3D object bounding box prediction loss function "Lbbox." The 3D object bounding box prediction loss function is the same as that of the original network, including the loss function Lc1-reg generated by T-Net, the loss function Lc2-reg predicted by the object bounding box center, the object heading loss functions Lh-cls and Lh-reg, and the size loss functions Ls-cls and Ls-reg. The corner loss function of 8 vertices in the bounding box of the object is Lcorner, and the 3D segmentation loss function is Lseg. The computational formula of Lmulti-task is as follows:

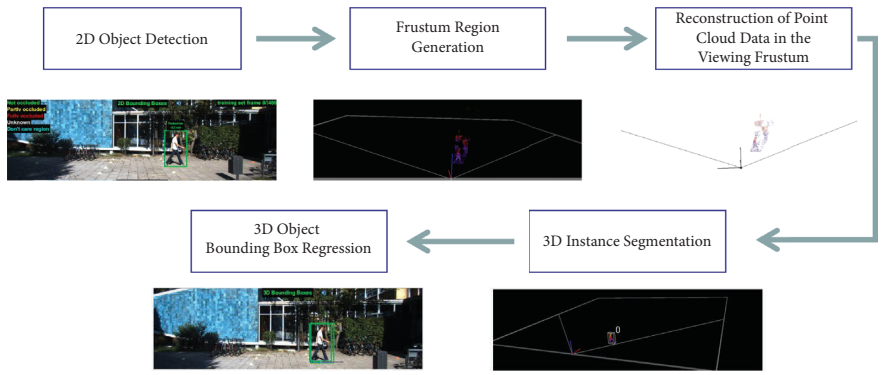

Figure 2: Algorithm flowchart.

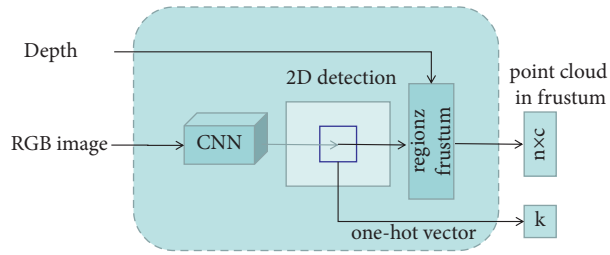

Figure 3: 3D frustum point cloud region generation network.

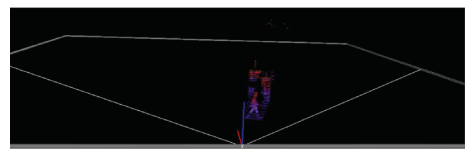

Figure 4: Initially obtained 3D object frustum point cloud region.

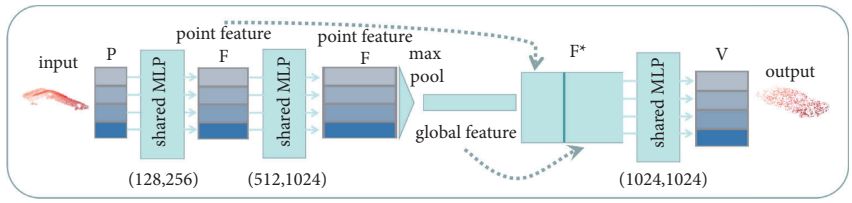

Figure 5: Point cloud reconstruction network within the viewing frustum.

$$\begin{aligned} L_{\text{multi-task}} &= L_{\text{mask}} + L_{\text{bbox}} \\ &= L_{\text{seg}} + \lambda\left(L_{c1\text{-reg}} + L_{c2\text{-reg}} + L_{h\text{-cls}} + L_{h\text{-reg}} + L_{s\text{-cls}} + L_{s\text{-reg}} + \gamma L_{\text{corner}}\right), \end{aligned} \tag{1}$$

where the corner loss is the sum of the distance between the eight corners of the predicted 3D bounding box and the ground truth bounding box. Since the corner is determined by the center, size, and orientation, the corner loss can constrain the training of each parameter of the 3D bounding box very well. Firstly, we define NS calibration bounding box and NH orientation angle and then convert them to the center of the estimated 3D bounding box. P_k^{ij} is the 3D vector of the kth corner of the calibration bounding box, where the index i stands for the serial number of the calibration object bounding box of 8 sizes and j stands for the 12 orientation angles. P_k^* is the 3D vector of the kth corner of the truth 3D bounding box. $\|P_k^{ij} - P_k^*\|$ is the distance between the kth corner of the 3D calibration bounding box and the kth

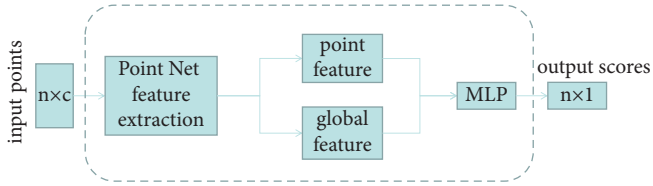

Figure 6: 3D instance segmentation network of point clouds in the viewing frustum.

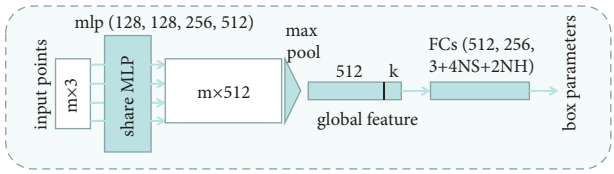

Figure 7: 3D object bounding box regression network.

corner of the truth 3D bounding box; P_k^{**} is the 3D vector of the kth corner after the 3D bounding box is flipped by π angles. This vector is introduced to avoid great loss caused by the flipped heading estimation. $\|P_k^{ij} - P_k^{**}\|$ is the distance between the kth corner of the 3D calibration bounding box and the kth corner of the flipped 3D bounding box. The computational formula of Lcorner is as follows:

$$L_{\text{corner}} = \sum_{i=1}^{NS}\sum_{j=1}^{NH} H_{ij} \min\left\{\sum_{k=1}^{8}\left\|P_k^{ij} - P_k^{*}\right\|, \sum_{i=1}^{8}\left\|P_k^{ij} - P_k^{**}\right\|\right\}, \quad (2)$$

where H_{ij} is the Huber loss regression function, which can enhance the robustness to outliers compared with the mean square error loss function MSE (mean square error). The parameter "a" stands for the residual; δ is the parameter of the Huber loss regression function: square error shall be adopted for parameter a (residual) less than δ and linear error for parameter a (residual) more than δ. The computational formula of $H_\delta(a)$ is as follows:

$$H_\delta(\mathrm{a}) = \begin{cases} \frac{1}{2}a^2, & \text{for} |a| \le \delta, \\ \delta \cdot \left(|a| - \frac{1}{2}\delta\right), & \text{otherwise.} \end{cases} \quad (3)$$

3. Experimental Results and Analysis

3.1. Experimental Environment Configuration and Dataset. In order to verify the improved 3D object detection algorithm proposed in this study, the data used are the 3D KITTI dataset in the automatic drive scene [28]. The dataset has RGB images and point cloud data in different scenes and classifies the object detection difficulty into three levels (easy, moderate, and hard) based on the occlusion of the objects. There are a total of 7481 scenes in this dataset, corresponding to 7481 RGB images and relevant point cloud data. The computer configuration used for the calculation is GTX 2080ti. The operating system is Ubuntu 16.04, the language used is Python 2.7, and the deep learning framework is TensorFlow 1.13. Training parameter setting: we divide the KITTI dataset into 3712 training sets and 3769 validation sets. The optimizer of the model selects the Adam algorithm, with an initial learning rate of 0.001, an initial attenuation rate of 0.5, and an attenuation rate of 800,000; that is, the learning rate is halved every 800,000 iterations; the batch_size is set to 32; that is, 32 point clouds are processed each time; num_point is 1024 (specify that, for each sample, only 1024 points are extracted from the point cloud frustum for training), and max_epoch is 200 (the number of training executions).

3.2. Evaluation Index. The evaluation index is the average precision (AP). The AP value is one of the most widely used evaluation indexes in the field of deep learning. It is the average value of the multiclassification precision P and can be calculated by the following formula:

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \quad (4)$$

where TP stands for true positive, that is, the number of samples whose predicted and true values are both true, and FP stands for false positive, that is, the number of samples whose predicted value is true but true value is false.

3.3. Verification Results and Comparison. We make a comparison between the obtained results and those of other algorithms with the same evaluation index and set the detection IoU (Intersection over Union) for vehicles to 0.7 and that for pedestrians and cyclists to 0.5. The larger the IoU is, the smaller the error required for detection will be and the stricter the evaluation will be. The results are listed in Table 1 and Table 2. It can be seen that this algorithm has improved performance in both 3D view and BEV (Bird's Eye View). For the detection of pedestrians, cyclists, and vehicles that are difficult to be detected, greater improvement can be achieved. In the 3D view, the AP of "easy" level vehicles in the 3D view has been improved less by 3.19 p.p., but greatly for detection of vehicles at the "moderate "and "hard "levels increased by 7.1 p.p. and 8.1 p.p., respectively, especially for the detection of pedestrians at the "moderate "level and cyclists at the "easy" level (the AP is increased by 16.39p.p. and 9.39p.p, respectively). In BEV (Bird's Eye View), for the detection of pedestrians and cyclists at the "easy" level, the AP is improved by 17.57 p.p. and 10.92 p.p.; for the detection of pedestrians and cyclists at the "hard "level, the AP is increased by 16.57 p.p. and 7.18 p.p. Compared with other fusion algorithms, our algorithm achieves a great improvement in the 3D object detection of pedestrians and cyclists. The detection results are shown in Figure 8, and distant cyclists can also be accurately identified.

We divide the algorithm in this study into four steps, namely, the view frustum point cloud area generation, the sparse point cloud reconstruction, the 3D instance segmentation, and the 3D box regression. The time required for the four steps is obtained through experiments, as shown in Table 3.

Table 1: 3D object detection AP values of various algorithms in the 3D view with the KITTI test set.

Method	Cars			Pedestrians			Cyclists		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
MV3D	71.29	62.68	56.56						
VoxelNet	81.97	65.46	62.85	39.48	33.69	31.50	61.22	48.36	44.37
Pointpillars	79.05	**74.99**	**68.30**	52.08	43.53	41.47	75.78	**59.07**	52.92
F-PointNet	80.62	64.70	56.07	50.88	41.55	38.04	69.36	53.50	52.88
Ours	**83.81**	71.80	64.17	**65.57**	**57.94**	**50.77**	**78.75**	58.92	**55.70**

Table 2: 3D object detection AP values of various algorithms in the BEV with the KITTI test set.

Method	Cars			Pedestrians			Cyclists		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
MV3D	86.02	76.90	68.49						
VoxelNet	**89.35**	79.26	77.39	46.13	40.74	38.11	66.70	54.76	50.55
AVOD	88.53	**83.79**	**77.90**	58.75	51.05	47.54	68.09	57.48	50.77
F-PointNet	87.28	77.09	67.90	55.26	47.56	42.57	73.42	59.87	52.88
Ours	87.98	83.60	76.26	**72.83**	**66.56**	**59.14**	**84.34**	**64.57**	**60.06**

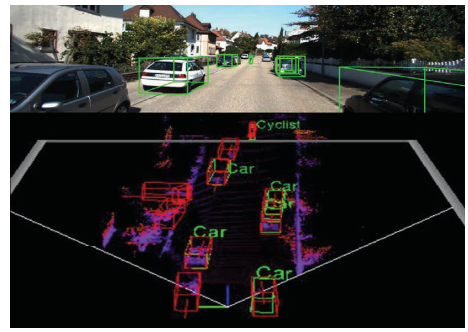

Figure 8: Visualization of 3D object bounding box.

Table 3: Algorithm running average time (ms).

The algorithm part of this study	Time (ms)
View frustum point cloud area generation	45
Sparse point cloud reconstruction	6
3D instance segmentation	22
3D box regression	15
Total	88

Through the experimental results, we can find that the reconstructed part of the sparse point cloud takes less time, but the experimental results are greatly improved. The proportion of each part is shown in Figure 9. The comparison with other algorithms is shown in Table 4.

Figure 10 shows a PR (Precision-Recall) curve chart for the detection precision and regression of vehicles, pedestrians, and cyclists. By selecting different confidence thresholds, different points can be obtained on the PR coordinate system, and the PR curve can be obtained by connecting these points. The recall rate is the correct proportion predicted in the real positive example data. For a certain recall rate, the higher the detection precision is, the better the detection performance of the algorithm will be. Through experimental comparison and analysis, compared with the original algorithm, our algorithm improves the regression rate and detection precision of 3D object detection for vehicles, pedestrians, and cyclists.

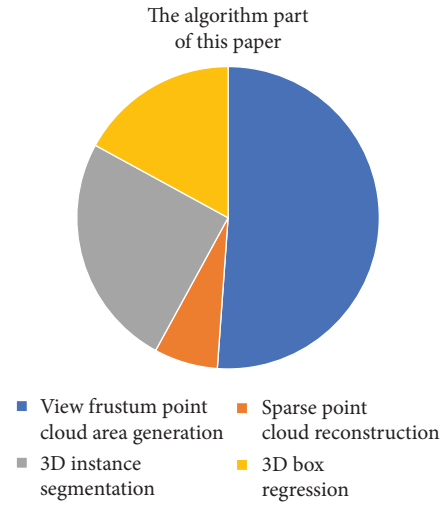

Figure 9: The proportion of the algorithm part of this study.

Table 4: 3D detector runtime for various algorithms.

Method	MV3D (ms)	F-ConvNet (ms)	AVOD (ms)	F-PointNet (ms)	Ours (ms)
Time	243	476	80	81	88

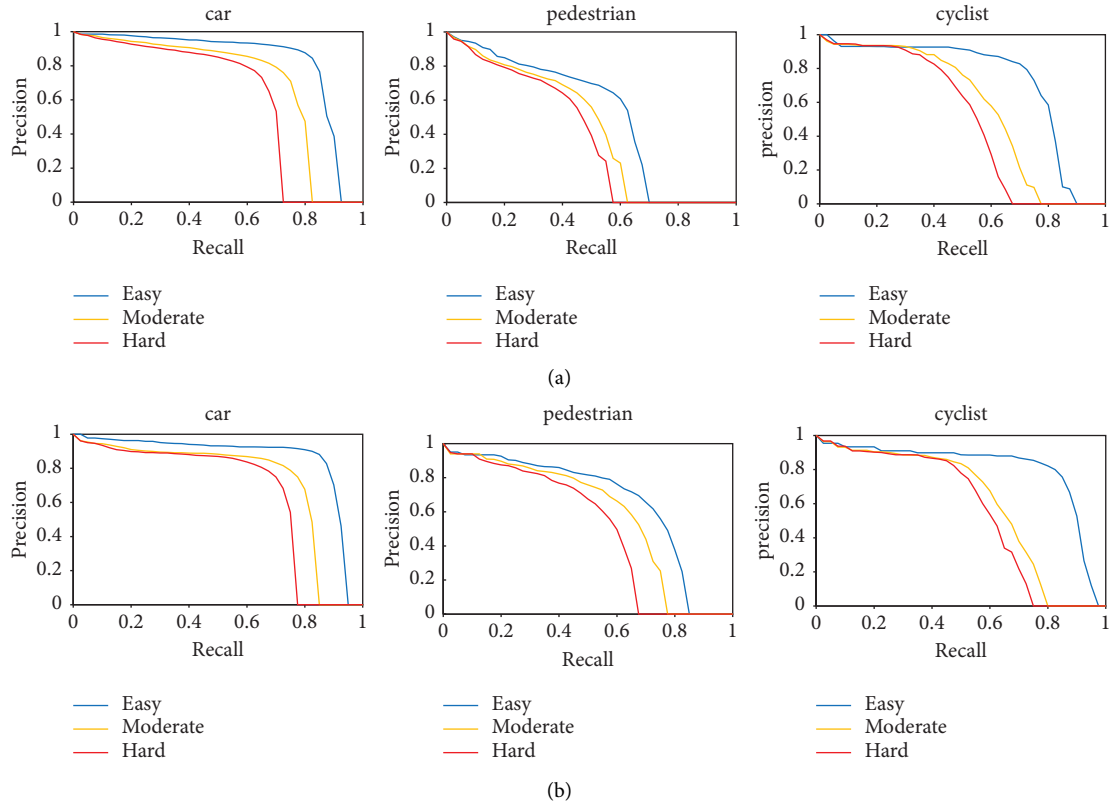

Figure 10: 3D object detection P-R curve comparison. (a) Original algorithm. (b) Our algorithm.

4. Conclusion

Since RGB images have rich color information, the object detection precision of vehicles, pedestrians, and cyclists with RGB images has always been higher than that with point cloud data. While, on the contrary, since point cloud data will not be easily affected by external factors such as illumination and weather changes, through using point cloud data, the safety of automatic drive can be improved and traffic accidents can be reduced. However, point cloud data also have shortcomings. Due to the threads and distance of the LiDAR, the collected point clouds may be relatively sparse so that it will be difficult to identify a specific object. Due to the shortcomings of different sensors, automated vehicles are already now equipped using varying configuration of machine vision sensors. Ultimately, it is matter of calculation time cost and performance of sensor, and hence, there is growing need for finding not only more reliable and accurate but also cost effective solutions. Therefore, a 3D object detection algorithm based on the reconstruction of sparse point clouds in the viewing frustum is proposed in this study, which can effectively improve the precision of small object detection. We increase the dimensions of the point cloud data for the sparse point clouds in the viewing frustum with shared MLP, perform the fusion of the obtained local point cloud data features and the global point cloud data features, and then obtain the point clouds after reconstruction with the shared MLP again. Finally, experiments prove that, in the 3D view, compared with the original F-PointNet model. The improvement of the 3D object detection rate of the car at three different difficulty levels is 3.19p.p., 7.1p.p., and 8.1p.p., respectively. The precision of 3D object detection for pedestrians is increased by 14.69p.p., 16.39p.p., and 12.73p.p., respectively. In BEV (Bird's Eye View), compared with the original F-PointNet model, the precision of 3D object detection for pedestrians is increased by 17.57p.p., 19p.p., and 16.57p.p., respectively. The improvement of 3D object detection rate for cyclists at three different difficulty levels is 10.92p.p., 4.7p.p., and 7.18p.p., respectively. In the process of automatic driving, the target is often blocked or dynamically changed, and the target object cannot be accurately identified. The algorithm in this study improves the detection accuracy of the target to reduce the occurrence of accidents. However, compared with the precision of 2D object detection, the precision of 3D object detection still has a large room for improvement, and in-depth studies on the features of point clouds will be continued in the next stage.

Data Availability

The data used to support the findings of this study can be obtained from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Xing Xu, Xiang Wu, Xiaoshu Lü, and Aki Aapaoja contributed equally to this work.

Acknowledgments

This research was supported by the National Key Research and Development Program of China (2019YFE0126100) and the Key Research and Development Program in Zhejiang Province of China (2019C54005).

References

[1] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving," *Common Practices and Emerging Technologies*, vol. 8, 2019.

[2] B. Chandrasekaran, S. Gangadhar, and J. M. Conrad, "A survey of multisensor fusion techniques, architectures and methodologies," in *Proceedings of the SoutheastCon 2017*, Concord, NC, USA, March 2017.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the Computer Vision and Pattern Recognition(CVPR), 2014 IEEE Conference on*, IEEE, pp. 580-587, Columbus, OH, USA, June 2014.

[4] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-View 3D Object Detection Network for Autonomous Driving," in *Proceedings of the 2017 IEEE conference on computer vision and pattern recognition (CVPR)*, IEEE, Honolulu, Hawaii, July 2017.

[5] Z. Yin and O. Tuzel, "VoxelNet: end-to-end learning for point cloud based 3D object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition 2018*, Salt Lake City, UT, USA, June 2018.

[6] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3D proposal generation and object detection from view aggregation," in *Proceedings of the 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, October 2019.

[7] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: deep learning on point sets for 3D classification and segmentation," in *Proceedings of the 2017 IEEE conference on computer vision and pattern recognition (CVPR)*, July 2017.

[8] C. R. Qi, Y. Li, S. Hao, and L. J. Guibas, "PointNet++: deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[9] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D object detection from RGB-D data," in *Proceedings of the 2018 IEEE CVF conference on computer vision and pattern recognition (CVPR)*, IEEE, Salt Lake City, UT, USA, June 2018.

[10] M. Kazhdan and H. Hoppe, "Screened Poisson surface reconstruction," *ACM Transactions on Graphics*, vol. 32, no. 3, pp. 1–13, 2013.

[11] M. Pauly, N. J. Mitra, J. Giesen, and L. J. Guibas, "Example-based 3D scan completion," *Symposium on Geometry Processing*, Eurographics Association, Yarmouth Port, MA, USA, 2005.

[12] S. Chaudhuri, E. Kalogerakis, L. Guibas, and V. Koltun, "Probabilistic reasoning for assembly-based 3D modeling," *ACM Transactions on Graphics*, vol. 30, no. 4, pp. 1–10, 2011.

[13] I. Alhashim, H. Li, K. Xu, J. Cao, R. Ma, and H. Zhang, "Topology-varying 3D shape creation via structural blending," *ACM Transactions on Graphics*, vol. 33, no. 4, pp. 1–10, 2014.

[14] A. Dosovitskiy, J. T. Springenberg, M. Tatarchenko, and T. Brox, "Learning to generate chairs, tables and cars with convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 692–705, 2017.

[15] M. Firman, O. M. Aodha, S. Julier, and G. J. Brostow, "Structured prediction of unobserved voxels from a single depth image," in *Proceedings of the 2016 IEEE conference on computer vision and pattern recognition (CVPR)*, IEEE, Las Vegas, NV, USA, June 2016.

[16] A. A. Soltani, H. Huang, and J. Wu, "Synthesizing 3D shapes via modeling multi-view depth maps and silhouettes with deep generative networks," in *Proceedings of the IEEE conference on computer vision & pattern recognition*, November 2017.

[17] M. Sarmad, H. J. Lee, and Y. M. Kim, "R. L.-Gan-Net: A reinforcement learning agent controlled GAN network for real-time point cloud shape completion," in *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[18] Z. Huang, Y. Yu, J. Xu, F. Ni, and X. Le, "PF-net: point fractal network for 3D point cloud completion," in *Proceedings of the 2020 IEEE CVF conference on computer vision and pattern recognition (CVPR)*, June 2020.

[19] X. Yu, Y. Rao, Z. Wang, Z. Liu, J. Lu, and J. Zhou, "PoinTr: diverse point cloud completion with geometry-aware transformers," in *Proceedings of the IEEE/CVF international conference on computer vision 2021*, Montreal, Canada, October 2021.

[20] S. Thrun and B. Wegbreit, "Shape from symmetry," in *Proceedings of the ICCV 2005. Tenth IEEE international conference on*, IEEE, Beijing, China, October 2005.

[21] L. C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV) 2018*, Munich, Germany, September 2018.

[22] Y. Yang, F. Chen, F. Wu, D. Zeng, Y. Ji, and X. Y. Jing, "Multi-view semantic learning network for point cloud based 3D object detection," *Neurocomputing*, vol. 397, pp. 477–485, 2020.

[23] R. Cheng, C. Agia, Y. Ren, X. Li, and L. Bingbing, "S3CNet: a sparse semantic scene completion network for LiDAR point clouds," 2020, https://arxiv.org/abs/2012.09242.

[24] T. Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid networks for object detection," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, July 2017.

[25] M. Jaderberg, K. Simonyan, and A. Zisserman, "Spatial transformer networks," *Advances in neural information processing systems*, MIT Press, vol. 28, , 2015.

[26] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3D bounding box estimation using deep learning and geometry," in *Proceedings of the 2017 IEEE conference on computer vision and pattern recognition (CVPR)*, November 2017.

[27] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[28] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: the kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.