A Fusion Method of 3D Object Detection Graph Neural Network Based on Local and Global Data Augmentation

Yi Zheng[1](✉), Xiaoyang Liu[1,2], Kaizhi Ruan[1,2], Wenhua Zhai[1,2], Yanbin Liu[1,2], and Ningjing Gong[2]

[1] Shanghai Electro-Mechanical Engineering Institute, Shanghai 201109, China
zhengyi_1121@163.com

[2] Shanghai Institute of Aerospace System Engineering, Shanghai 201109, China

Abstract. LiDAR-based 3D object detection is an important task for autonomous driving because it provides the location information of objects on the road. However, the existing methods perform poorly in the detection of distant objects since they suffer from sparse and incomplete point clouds. To overcome the problems caused by multi-scale and occupancy, we propose a novel graph-based framework, i.e. named a fusion method of 3D object detection graph neural network based on local and global data augmentation (LGDA-GNN), for accurate 3D object detection from the point clouds. Firstly, we summarize seven characteristics of point cloud data, and point out a new way to improve accuracy of this task. Secondly, to select effective vertices of graph neural network and to enhance local features and the proportion of point cloud at medium and long distances, we propose methods including self-adaptive multi-scale voxel downsampling and multi-step graph construction. Thirdly, to improve the accuracy of difficult category, a fusion method of self-attention module and visibility property is presented for global feature reinforcement. Finally, we integrate our refinement modules into a graph-based pipeline, and extensive experiments on the KITTI benchmark show that we achieve a great performance on the difficult car object for the bird eye view detection and 3D object detection task.

Keywords: Point cloud · Graph neural network · 3D object detection · Data augmentation

1 Introduction

3D object detection in point clouds plays an important role in autonomous driving. LiDAR is the most common instrument to collect such data. The 2D data does not provide depth information and the 3D data introduces a third dimension that contains more detailed object location and size information, and are less sensitive to illumination. In addition, LiDAR has many advantages over

Z. Ren et al. (Eds.): *Proceedings of 2021 5th Chinese Conference on Swarm Intelligence and Cooperative Control*, LNEE 934, pp. 605–616, 2023.
https://doi.org/10.1007/978-981-19-3998-3_58

cameras, for example LiDAR data is more stable at night and more robust. It could rarely be affected by changing illumination.

However, 3D object detection still faces many challenges due to the characteristics of the point cloud. For example, the farther away an object is from the LiDAR, the fewer 3D points it occupies on the point cloud. And the low resolution of far-away point cloud (sparse point clouds) results in the poor detection of the distant objects. It can be called multi-scale problem of point clouds. The point clouds of LiDAR are randomly distributed and the order of the point clouds should not have influence on the extracted features.

To solve the above problems, lots of methods have been proposed for 3D object detection in recent years. In general, these approaches can be classified into two categories, i.e. voxel-based methods and point-based methods. The voxel-based method is completely using the 2D methods on the 3D point cloud. The 2D methods uses a 2D convolution neural network to detect the objects after mapping the point clouds to the front view or bird-eye view. Therefore many 2D methods for the multi-scale problems can be used, such as the classical image pyramid, feature pyramid and convolution kernel pyramid, etc. The 3D methods processing point cloud straightly like [1] uses a skip connection to concatenate the low-level features to the high-level features, which can be summarized as the point-based methods. Since PointNet independently processed point clouds and did not get the local structural features, [2] proposed later used set abstraction operation to get the multi-scale features. However, the repeated grouping and sampling on a large point clouds in the PointNet++ are computationally expensive. Since this method allows direct operations on points, many similar methods have been proposed, including the method based on Graph Neural Network (GNN). Although GNN-based methods can acquire the local object features much more than other solutions, there are still many problems. For example, the background points account for a high proportion in point clouds, the asymmetrical density distribution problem and the low accuracy of the obscured objects.

To solve these problems, in this work, we firstly summarize seven characteristics of point cloud data:**structurelessness**,**randomness**,**irregularity**,**data-heaviness**,**local-scale relevancy**,**global-scale relevancy**,**3D-illusion**. Structurelessness means that the point clouds do not have the stable structure like 2D image. We have to give them structure to represent them. Randomness means the input sequence should not affect the output. Irregularity means the local point cloud may be arranged in an irregular sequence and the density may be different in diverse regions. Data-heaviness means point clouds have tons of data to compute. Local-scale relevancy means point clouds in a local region is relevant. Global-scale relevancy means the point clouds of the same kind of objects are relevant all around the space. 3D-illusion means the point clouds show different objects in different angles of view. In order to solve **structurelessness** and **randomness**, we use a graph-based network shown in Fig. 1 as a compact representation of a point cloud. After that, we provide several methods to improve the effectiveness of vertex selection and enhance the global features of vertices respectively. Firstly, for the downsampling loss in the sparse area of the point clouds caused by the

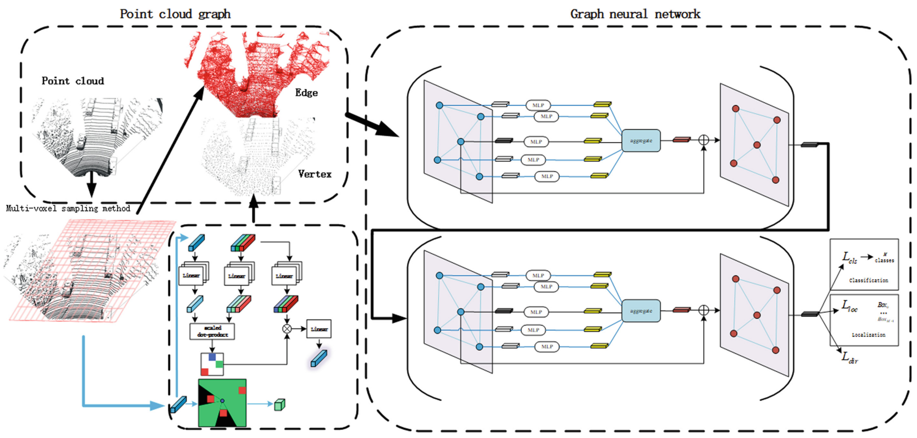

Fig. 1. The full architecture of our proposed method including multi-voxel sampling, the graph neural network and fusion feature of self-attention and visibility.

fixed-voxel sampled method, a self-adaptive multi-scale voxel downsampling method for graph construction is proposed and have dealt with **data-heaviness** and **irregularity** by the way. Through the new downsampling method, the distribution of the vertices is redetermined. To make sure the selected vertices are very precise, we propose multi-step graph construction to have some redundancy in the process of picking vertices. This is how we strengthen the local features to utilize **local-scale relevancy**. Secondly, considering the unevenness of the point cloud, a global feature enhancement method based on self-attention mechanism is proposed to make full use of **global-scale relevancy**. For the **3D-illusion** of point clouds, a global feature enhancement method based on visibility classification is also proposed, which divides vertex voxels into different categories based on occlusion and unocclusion.

In conclusion, the main contributions of our work are: (1)We innovatively summarize seven characteristics of point cloud data and point out a new way to improve accuracy of this task. (2)To accurately select the vertices and to raise the proportion of point cloud at medium and long distances, a multi-step graph construction is proposed. (3) To improve the accuracy of object detection on the difficult task, we propose a vertex global-feature enhancement method incorporating visibility and self-attention modules. (4) We integrate our refinement modules into a graph-based pipeline, and achieve great performance on the hard category of KITTI dataset for the 3D object detection task.

2 Related Works

Point-Based Methods for 3D Object Detection: Although the best algorithms in both categories are currently on the same level of accuracy and speed, intuitively the Point-based method has more room for improvement. Point-based methods represent the points by itself rather than the voxel. With Multi-Layer Perception (MLP), we can enlarge the size of features from three dimensions to higher dimensions. [1] computes the global feature vector of point cloud by stacking many MLP, and feeds it back to point-wise features by concatenating the global feature with each of the point features. [2] proposes the set abstraction operation that contained sampling and grouping, which enables flexible receptive fields for point cloud feature learning. It is able to progressively capture features at increasingly larger scales along a multi-resolution hierarchy. Then [3–5], and many point-based methods are mostly based on the PointNet++ series. And they all have the same problem as the Pointnet++ which takes too much time on repeated grouping and sampling. After many Point-based approaches are proposed, the way of point representation is generally accepted, but there is still much disagreement about the data flow transfer process and the feature extraction process. The physical feature of a point is a column of vectors, and how to deal with the combined vectors is a problem that many new methods are exploring. Taking PV-RCNN [6] as an example, the article combines the advantages of the methods of the Point and Voxel based methods to obtain 3D feature bodies from 3D voxels, converting them into 2D eye view feature maps of bird, and generate proposal boxes using the anchor-based method.[7] designs an auxiliary network improves positioning accuracy through deep mining of geometric information of 3D objects.

Meanwhile, [8–10] revealed that it is suitable to use the graph neural network to process the point clouds that are generated from non-Euclidean domains. Then, [11] proposed a method that used point representation on key points and sampling points, and used the data structure of the graph between key points and sampling points. After that, GNN iteratively updates its vertex features by aggregating features along the edges. The edge between two points allows the information sharing along the edge. The point clouds would obtain a new structure through this method. [12] introduced a point cloud completion module to recover high-quality proposals of dense points and entire views with original structures preserved based on a designed graph neural network module which comprehensively captures relations among points through a local-global attention mechanism.

3 Proposed Method

In this paper, we propose the LGDA-GNN based on graph neural network, aiming at solving the multi-scale 3D object detection problems through augmented data in point clouds. The overall architecture of our method contains three components: (a) self-adaptive multi-scale voxel downsampling mechanism; (b) multi-step graph construction and GNN module; (c) feature fusion of self-attention and visibility.

3.1 Self-adaptive Multi-scale Voxel Downsampling Method

As we have known, if the object is far away from the LiDAR, its point cloud density become lower. Therefore, the density of point clouds collected at different distances is not uniform. What is more, existing methods usually use downsampling method to extract the point cloud and reduce the number of point clouds, especially the voxel downsampling method. If the voxel downsampling method uses a fixed voxel to get the vertex, it will result in a bad situation that the number of sampling vertices in the front area is larger than the number in the back area. It aggravates the information loss especially the information loss of the distant objects. Existing methods that use the voxel downsampling have this shortcoming, but they are difficult to make some improvements. If the voxels of different sizes in different region were used, it will affect the regularity of voxel grids. Furthermore, the regular convolution layer cannot be used for feature extraction. In other words, a fixed-size voxel is a prerequisite for ensuring regularization of 3D data and the using of 3D convolution operation.

However, there is no limitation for graph-based methods, since the data regularization is not needed. Therefore, we design a self-adaptive multi-scale voxel downsampling method for graph construction to solve the downsampling loss of the sparse point-cloud area.

Multi-voxel Sampling Graph Generation Method. A graph network will make predictions from the vertices of the point clouds. In contrast, the features of a vertex are made up of a collection of points within a certain range around it. Therefore, the distribution of vertices and the location of the vertices are very important. Multi-voxel sampling graph generation method allows for some adjustment of the vertex distribution.

It can gradually reduce the size of voxels, and effectively increase the number of sampled point clouds in sparse areas. By using a smaller voxel to sample the point cloud in sparse area, the information loss will be alleviated. We modified the traditional point-cloud voxel downsampling method and proposed a self-adaptive multi-scale voxel sampled method for graph construction. As the formula shows, x represents the distance coordinate of the point and v represents the sampling voxel size.

$$V = V_0 - k_0 * mod(X/10) \tag{1}$$

It can effectively alleviate the problem of sparse point clouds in the distance and does not miss sparse faraway objects. Since there is only one hyper-parameter k_0 in Eq. (1), we only have to set k_0 to meet the required total amounts of point clouds, and it is superior to the methods which have to set many hyper-parameters that will decrease the stability of algorithm.

Invalid Altitude Suppression Method. Using Multi-voxel sampling method alone also leads to the problem that as voxels become smaller, more points are retained as vertices and a point-cloud map has more vertices to compute with, creating a significant computational burden. Furthermore, the vertices should be selected to achieve the balance of positive and negative samples. To solve this problem, we can restrict the sampling altitude area to a range.

Experimental comparisons show that positive samples are still easy to distinguish when the higher point cloud is removed. In the lower areas of the point cloud, most of the point cloud is about roads and walls. These background points account for a large proportion of the weight and we do not want the vertices to be created from the background, so the lower-altitude points are removed as well. After removing the noise of point cloud, we sort it by height from z_l to z_h. A certain range of altitude of point cloud is restrained. The percentage of the foreground points is greatly increased through this way.

3.2 Multi-step Graph Construction

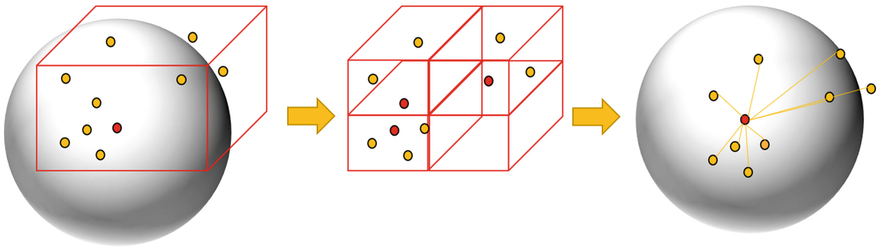

Fig. 2. Schematic diagram of multi-step graph construction.

In order to solve the irregular structure of point cloud, we use graph network to build a point-cloud representation. A point cloud with N points $P = \{p_1, p_2, \ldots, p_N\}$ is defined as a set, where each point $p_i = (x_i, s_i)$ includes its own coordinate position $x_i \in R^3$ and its own properties such as reflectivity $s_i \in R^k$, and so on. The point cloud graph contains two elements, vertex P and edge E. Therefore, given a point cloud set P, we describe the point cloud as a vertex and connect it with other point clouds within the fixed radius r. Thus, we create a graph $G = (P, E)$. The simple construction of the edge is shown in Eq. (2)

$$E = \big\{(p_i, p_j) \big| \|x_i - x_j\|_2 < r\big\} \tag{2}$$

where p_j denotes another point that is within the fixed radius r of the vertex p_i. This procedure is actually a nearest neighbor selective processing for a fixed radius with the vertex we have established. However,we can not make sure if the vertices we chose are really needed? And we have to make the process of choosing vertices much more precisely. As shown in Fig. 2, we used to pick centroid points in each voxel to be the vertex points, then we divide the procedure into two steps. Each voxel is separated into the small voxels,and we rename the centroid points in each small voxels to be the vertices of each respective voxel. The final vertex will be selected from the vertices of the minor voxels and replace the center point of construction circle. And the new vertex usually obtains more points as its edges and catches much more structural information which results in a higher detection accuracy.

After the graph was constructed, we process the graph with GNN. A typical graph neural network refines the vertex features by aggregating features along the edges. In the $(t+1)^{th}$ iteration, it updates each vertex feature as Eq. (3) and (4), where e^t and v^t are the edge and vertex features from the t^{th} iteration. The MLP $f^t(\cdot)$ computes the edge feature e^t_{ij} between the vertex i and vertex j. $x_i - x_j$ denotes the relative distance of two vertices. Then $p(\cdot)$ aggregates the edge features of one vertex. The MLP $g^t(\cdot)$ get updated feature v_i^{t+1} of the $(t+1)^{th}$ iterarion from the output of $p(\cdot)$ and the former feature v^t_j. These processes will be repeated in the next iteration.

$$e^t_{ij} = f^t([x_i - x_j, v^t_j]) \tag{3}$$

$$v_i^{t+1} = g^t(p(e^t_{ij}|(i,j) \in E), v^t_i) \tag{4}$$

The GNN is used to update the vertex feature to include the local structure information. After several iterations of the graph neural network, the vertex features are used to predict both the category and the bounding box of the object to which the vertex belongs.

3.3 Fusion of the Self-attention and Visibility Feature

The vertex features of the point-cloud graph contain local relationships between points and edges, and the self-attentive mechanism and visibility classification proposed in this paper are designed to enhance the vertex features so that they contain global-scale relevancy

Vertex Self-attention Method. Taking the vertex data queue of the point cloud as the input part of the self-attentive module, the corresponding output will be the interaction and fusion result of the point cloud map as a whole, with the features of other vertices fused in each vertex, just like [13] shows.

The input data of the self-attention module is a feature queue of vertices, which is used for obtaining the weight corresponding to each vertex among all vertices. This weight to some extent reflects the importance of the vertex in the queue, which is similar to the rest of the vertices, and allows new vertex features to be obtained. If a vertex represents a local region of the background in the point cloud, it can be clearly distinguished from the foreground features through the self-attention module. Else if a vertex represents a sparse foreground far away in the point cloud, it will be closer to the foreground and the fused new features will be closer to the clear foreground objects after the process of self-attention module.

Suppose we have n vertices, each one with m columns of features. The Value $V \in R^{m\times 1}$, Key $K \in R^{m\times 3}$ and Query $Q \in R^{m\times 3}$. With the self-attention method proposed by [14], we can get the output in Eq. (5).

$$output_j = concat(\sum_{i=1}^{m} \frac{e^{{Q_j}^T K_i}}{\sqrt{3}\sum_{i=1}^{m} {Q_j}^T K_i} V_i, ...)W_0, j = 1, 2, ..., m \tag{5}$$

Visibility Classification. With inspiration of [15], we conclude that 3D-illusion is an important characteristics that should not be overlooked in the 3D detection task, and it is not a totally detrimental property. Point clouds from LiDAR lack a lot of angular information about the target object, but also contain beneficial information such as object occlusion, freespace, etc. We propose the method to make full use of visibility information, which means whether a point-cloud region or an object is visible.

There is a conception of occupancy grid map in SLAM, which shows the states of voxels should be divided into three catogories: freespace, unknown and occupied.

Firstly, we voxelize the point clouds, ignore the effect of height and look at all the vertex voxels in projection and initialise all the vertex grids to be seen as unknown. Secondly, we start with a vertex voxel at random, connect its vertex to the origin point, mark all the connected vertex voxels that the line passes through as freespace, and mark the current vertex grids as occupied. Then start again with the next vertex voxel until all the vertex voxels have been traversed and there should be no unknown voxels. We finally add a visibility feature to the vertex feature queue, and unify the vertex features in height (Fig. 3).

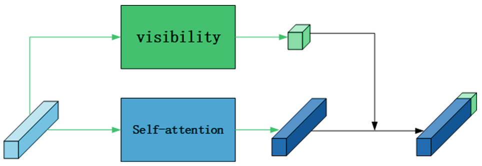

Fig. 3. Fusion process of the features of visibility and self-attention.

4 Experiments

We evaluate our method based the widely-used KITTI [16] object detection benchmark. The KITTI benchmark evaluates the average precision (AP) of three types of objects: Car, Pedestrian and Cyclist. Due to the scale difference, we follow the common practice and train only one network for the Car to testify our approaches. For training, we remove samples that do not contain objects of interest.

Car: We treat a side-view car with $\theta_2 = [-\pi/4, \pi/4]$ and a front-view car $\theta_2 = [\pi/4, 3\pi/4]$ as two different classes. Together with the Background class and

DoNotCare class, 4 classes are predicted. We construct the graph with $r_2 = 0.5r_1$. We set P as a downsampled point cloud by a maximum voxel size of 1 m in training and $k_0 = 0.075$ in multi-voxel sampling method.

4.1 Data Augmentation

To prevent overfitting, we perform data augmentation on the training data. Unlike some methods in [4] that use sophisticated algorithms to create new ground truth boxes, we choose a simple scheme with global rotation, global flipping, box translation and vertex jitter. During training, we randomly rotate the point cloud by yaw in the normal distribution $N(0, \pi/8)$ and then flip the x-axis by a probability of 0.5.

Observing the data, we found that the altitude of some point clouds ranges from –15 m to +5 m but only very few points locate from +2 m to +5 m and –2 m to –15 m. To prevent the influence of noise, we remove 0.5% of the total number of points from the top and from the bottom. After that, We use a 10% larger box to select the points to prevent cutting the object. During the translation, we check and avoid collisions among boxes. During graph construction, we use a random voxel downsampling to induce vertex jitter.

4.2 Implementation Details

Our framework is trained from scratch in an end-to-end manner with the ADAM optimizer. We train the entire network with the batch size on 1 RTX 2080 Ti GPU. Because the network is too deep, a small learning rate is required when starting training. The detection accuracy is almost the same when the number of iteration time T is 2 or 3, but the training time and testing time of the network are increased by almost 50% when T is 3, so we finally choose T to be 2. The common practice of point cloud sampling is to take 0.8 m as the voxel size, but this is very likely to ignoring the distant objects. By observing the characteristics of the object at about 70 m, we find that the point cloud of the car body is mainly concentrated in one face, and is extremely sparse. For acquiring the enough points of the distant objects, we choose the farthest voxel size of 0.4 m. And we decide the nearest voxel size to be 1.0m since this can almost retain the amount of key points.

4.3 Ablation Study

In this section, we conduct extensive ablation experiments to validate individual components of our proposed method shown as the last column in Table 1. We classify the Self-adaptive multi-scale voxel down-sampling method and Multi-step graph construction into enhancement of local features and sort Fusion feature of self-attention and visibility into enhancement of global features. All models are trained on the train split and evaluated on the val split for the car class of KITTI dataset. We focus on the detection of Car because of its dominant presence in the dataset.

Table 1. The Average Precision (AP) comparison between the local features and global features of 3D object detection on the KITTI validation dataset

Method	Car_detection_3D_AP			Car_detection_AP		
	Easy	Moderate	Hard	Easy	Moderate	Hard
Local features	90.04	80.86	76.78	96.79	93.45	91.66
Global features	90.25	80.55	77.03	96.75	93.55	91.34
OADA-GNN	90.89	82.28	77.78	96.85	95.48	93.00

Table 2. The Average Precision (AP) comparison of 3D object detection on the KITTI validation dataset

Method	Car_detection_3D_AP			Car_detection_AP			Car_detection_BEV_AP		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
PointPillars	82.58	74.31	68.99	94	91.19	88.17	90.07	86.56	82.81
TANet	84.39	75.94	68.82	93.67	90.67	85.31	91.58	86.54	81.19
STD	87.95	79.71	75.09	96.14	93.22	90.53	94.74	89.19	86.42
SA-SSD	88.75	79.79	74.16	97.92	95.16	90.15	95.03	91.03	85.96
3DSSD	88.36	79.57	74.55	97.69	95.1	92.18	92.66	89.02	85.86
Voxel R-CNN	90.9	81.62	77.06	96.49	95.11	92.45	94.85	88.83	86.13
Point-GNN	88.33	79.47	72.29	96.58	93.5	88.35	93.11	89.17	83.9
PV-RCNN++	90.14	81.88	77.15	96.08	95.05	92.42	92.66	88.74	85.97
PC-RGNN	87.94	81.38	76.88	95.8	94.68	92.2	92.08	88.43	85.81
SE-SSD	91.49	82.54	77.15	96.69	95.6	90.53	95.68	91.84	86.72
Ours	90.89	82.28	**77.78**	96.85	**95.48**	**93.00**	93.39	89.55	**87.01**

4.4 Results

Experimentally, we found that about half of the vertex positions changed after the multi-step graph construction, indicating that new local key points were found.

We have submitted our results to the KITTI 3D object detection benchmark and the Bird's Eye View (BEV) object detection benchmark. In Table 2, we compare our results with the existing SOTA 3D object detection method. All results are evaluated by the mean average precision (mAP) with a rotated IoU threshold 0.7 for cars. Since the KITTI website does not accept unpublished methods, we split the training dataset into training and validation dataset and test our result on the latter one. And the rusult shows that a great progress has been made on the hard category.

In Fig. 4, we provide qualitative detection results on all categories. The results on both the camera image and the point cloud can be visualized. It must be noted that our approach uses only the point cloud data. The camera images are purely used for visual inspection since the test dataset does not provide ground truth labels

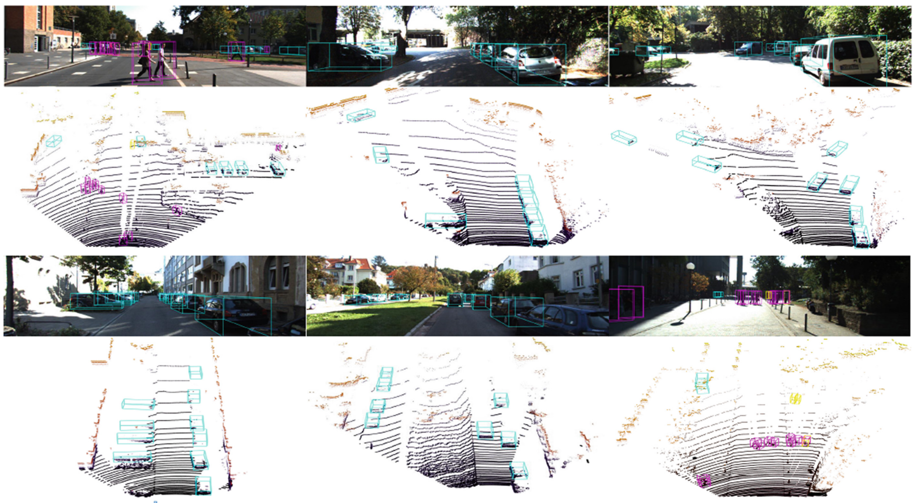

Fig. 4. Qualitative results on the KITTI test dataset using LGDA-GNN. We show the predicted 3D bounding box of Cars (green), Pedestrians (red) and Cyclists (blue) on both the image and the point cloud. Best viewed in color.

5 Conclusion

In this paper, we proposed a LGDA-GNN framework to investigate the graph neural networks (GNN) on 3D point-cloud object detection, improving the accuracy and precision of the vertex selection range, using the global-relevancy to learn the distant and obscured objects. We have outlined seven characteristics of point cloud data and designed different solutions according to the different characteristics of point clouds to solve the problem of 3D object detection and improve the effectiveness of detection algorithms. We have proposed local enhancement method, including multi-voxel sampling graph generation method which solves the problem of disproportion of point cloud, multi-step graph construction which increases the foreground vertices and gives a certain amount of tolerance to vertex selecting, and fusion feature of self-attention and visibility which enlarges the importance of foreground vertices and promotes the role of validate vertices. We operate an ablation study on all these methods and finally it shows a great progress compared with Point-GNN.

All these methods based on graph neural network are flexible and transferable. The way that GNN integrates sparse information of distant objects and makes them recognizable is also effective for object detection on missles. Considering the different kinds of data of missile seekers are not always robust [17], the graph neural network may play an important role on the future techniques of missile object detection, and the guidance precision of weapons can be improved.

References

1. 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)
2. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: deep hierarchical feature learning on point sets in a metric space (2017). arXiv preprint, arXiv:1706.02413
3. Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3d object detection from RGB-D data. In: Proceedings of the IEEE Conference On Computer Vision And Pattern Recognition, pp. 918–927 (2018)
4. Shi, S., Wang, X., Li, H.: Pointrcnn: 3d object proposal generation and detection from point cloud. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 770–779 (2019)
5. Yang, Z., Sun, Y., Liu, S., Shen, X., Jia, J.: Std: sparse-to-dense 3d object detector for point cloud. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1951–1960 (2019)
6. Shi, S., et al.: Pv-RCNN: point-voxel feature set abstraction for 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10529–10538 (2020)
7. He, C., Zeng, H., Huang, J., Hua, X.S., Zhang, L.: Structure aware single-stage 3d object detection from point cloud. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11873–11882 (2020)
8. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE Trans. Neural Netw. Learn. Syst. **32**, 4–24 (2020)
9. Qi, X., Liao, R., Jia, J., Fidler, S., Urtasun, R.: 3d graph neural networks for RGBD semantic segmentation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5199–5208 (2017)
10. Arnold, E., Al-Jarrah, O.Y., Dianati, M., Fallah, S., Oxtoby, D., Mouzakitis, A.: A survey on 3d object detection methods for autonomous driving applications. IEEE Trans. Intell. Transp. Syst. **20**(10), 3782–3795 (2019)
11. Shi, W., Rajkumar, R.: Point-GNN: graph neural network for 3d object detection in a point cloud. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 1711–1719 (2020)
12. Zhang, Y., Huang, D., Wang, Y.: PC-RGNN: point Cloud Completion and Graph Neural Network for 3D Object Detection, December 2020. arXiv e-prints, arXiv:2012.10412
13. Bhattacharyya, P., Huang, C., Czarnecki, K.: Self-attention based context-aware 3d object detection (2021). arXiv preprint, arXiv:2101.02672
14. Vaswani, A., et al.: Attention is all you need (2017). arXiv preprint, arXiv:1706.03762
15. Hu, P., Ziglar, J., Held, D., Ramanan, D.: What you see is what you get: exploiting visibility for 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11001–11009 (2020)
16. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: the kitti dataset. Int. J. Robot. Res. **32**(11), 1231–1237 (2013)
17. Qi, S., Zang, Y., Lyu, G., Du, M.: Research on air target image generation algorithm based on generative adversarial networks. Air Space Defense **14**(11), 67–73 (2021)