3D Point Cloud Target Detection Method Based on R-PointGNN

Nianfeng Li
Changchun University: Chang Chun University
Yan Li (Liyan27716@163.com)
Changchun University: Chang Chun University https://orcid.org/0000-0001-8333-3051
Yuying Wang
Changchun University: Chang Chun University
Zhiguo Xiao
Changchun University: Chang Chun University
Nan Zhao
Changchun University: Chang Chun University
Yupeng Li
Changchun University: Chang Chun University

Research

Keywords: point cloud, graph neural network, target detection, residual connectivity

Posted Date: March 8th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1355795/v1

3D Point Cloud Target Detection Method Based on R-PointGNN

Nianfeng Li [1], Yan Li [2], Yuying Wang [3], Zhiguo Xiao [4], Nan Zhao [5] and Yupeng Li [6]

1,4College of Computer Science and Technology, Changchun University, No.6543, Satellite Road, Changchun City, China
2,3,5,6Graduate School of Changchun University, Changchun University, No.6543, Satellite Road, Changchun City, China
Corresponding author: Yan Li (Liyan27716@163.com).

Abstract: Point cloud target detection completes the interaction of 3D features such as vector position and reflection intensity in the coordinate system and 3D visualization with visual enhancement effects, which are widely used in virtual reality, augmented reality and autonomous driving. However, the disorder, sparsity and overlap of LiDAR point clouds increase the difficulty of point cloud recognition. To address this problem, a 3D point cloud target detection model named R-PointGNN (Residual-Graph Neural Network) based on graph neural network is proposed. Deep residual connections are constructed on the graph neural network structure. Firstly, the semantic graphs of the point clouds are constructed by the nearest neighbor algorithm; then, the feature transfer and state update of the graphs are completed by the improved edge convolution; finally, the residual connections are introduced to connect multiple layers of states and extract deep features. Experiments on the KITTI dataset show that the method performs well in the point cloud target detection task with a high degree of discrimination.

Keywords: point cloud; graph neural network; target detection; residual connectivity

1 INTRODUCTION

Point cloud data is a collection of data describing the characteristics of real objects or environments, mainly from sensors such as 3D scanners, LiDARs, and RGB-D cameras, which can be used to analyze real-world object appearance data and environmental information [1]. Laser point clouds are formed with attributes including 3D coordinates and laser reflection intensity based on laser measurements. For the rendering of point cloud data, it helps the machine to perceive the 3D environment and complete human-machine interaction.

The target detection task based on point cloud data enhances the visualization of screen pixels and approximates the real world senses. Point clouds recognition technology is widely used in virtual reality, augmented reality and automatic recognition to enhance the immersive experience of users.

However, point clouds have characteristics of sparsity, disorder, and displacement invariance, which make structured convolutional models unsuitable for unstructured point cloud feature extraction, which greatly increases the difficulty of point clouds identification. The processing method of converting point cloud data into structured data increases the computational loss, and it is very easy to lose effective information. To address this current situation, the current research hotspot is to use point clouds as direct input without any mapping method. For example, Point-GNN [2] uses graph representation method to complete point cloud encoding without any normalization processing. Its sampling and clustering of point clouds are done by voxel sampling and the nearest neighbor algorithm (KNN). However, it has the problems of weak uniformity and susceptibility to noisy data; and the graph convolution (EdgeConv) algorithm used cannot avoid the problem of over-smoothing. Thus, in this paper, an improved 3D point cloud target detection model R-PointGNN is proposed based on graph neural network (GNN), and the performance is evaluated on the KIITI dataset.

The main work of this paper is as follows:

1) The nearest neighbor algorithm is used to complete the point cloud classification and construct the point cloud semantic graphs.

2) To make up for the insufficient information of point cloud features, in addition to the node coordinate information, the state features of source and target nodes are introduced in the edge convolution calculation method to further enhance the node features.

3) To increase the convolutional perceptual domain and capture the node features in multiple temporal states, deep residual connections are introduced in the graph convolution module, while alleviating the over-

smoothing phenomenon of the graph neural network.

2 RELATED WORK

The current research methods for 3D target detection can be summarized as LIDAR [3][4] based detection methods, and depth image [5][6] based detection methods. The depth image detection methods are classified into monocular [7][8], binocular [9], and multiocular 3D target detection by designing 2D image feature extractors to capture pixels. In addition, some studies combined multiple heterogeneous feature map with a fusion mode view study approach [10][11], which proved the superior performance of the fusion module by enhancing point-by-point features with semantic image features [12]. 3D target detection, as an important perceptual task for scene understanding, has a wide range of applications in virtual reality, autonomous driving, and other fields. Point clouds provide real-world 3D geometric perception and can accurately represent the 3D physical space coordinates of objects [13]. In recent years, many inspection tasks use LiDAR point clouds as input [14][15] and investigate effective methods for encoding the original point clouds [16]. Current methods for feature extraction of 3D point clouds can be broadly classified into three categories: projection-based methods, voxel-based sampling methods, and direct point cloud-oriented methods.

The projection-based approach [17] projects the point cloud data onto multiple view planes and uses traditional convolutional networks for feature extraction; the voxel sampling-based approach [18][19] transforms the point cloud into a voxelized grid and then used 3D convolution for processing. Both methods use the transformation of point clouds into structured data, but both suffer from uneven data distribution, computational complexity, and loss of feature information. And the latter creates a grid that takes up a lot of storage space because of the sparsity of the point cloud. Directly oriented point cloud processing methods are divided into point-based [20][21] methods and graph-based methods. Among the point-based processing methods, PointNet [22] pioneered the use of point clouds as direct input and solved the rotation problem and

disorder problem of point clouds using a deep network structure. However, its drawback is more obvious that only global features are considered, ignoring the point cloud local connection. So the performance is poor in the semantic segmentation task. pointNet++ [23] improved the feature extraction method to address this problem, but suffered from the problems of sensitivity to input noise and high time complexity. Methods [24][25] went one step further by constructing k-nearest neighbor graphs in the point cloud space to effectively obtain point cloud local information and reduced the time cost. On the other hand, pointRCNN [5] proposed a 3D target detection framework based on the original point cloud using PointNet++ as the backbone network, which was experimentally proven to have high robustness and accurate 3D detection performance. Most of the above deep learning models directly oriented to point cloud data have complex structures and cannot be applied to the real-time analysis of point clouds. Jing Bai [26] et al. proposed a lightweight real-time point cloud network, LightPointNet, with less than 0.07M parameters and good generalizability, to address the problem of more parameters in existing models. In addition, the literature [27] introduced the Progressive Population Augmentation (PPBA) algorithm to significantly optimize the StarNet detector for point clouded sparsity. The literature [28] conducted a study on the robustness of point cloud 3D target detection. A sparse LSTM multi-frame 3D target detection algorithm focusing on the time-domain information of point clouds was proposed in the literature [29]. All these studies provide new research perspectives for point cloud target detection.

Although the above point-based processing methods effectively capture the local point cloud features, they fail to capture the connection relationship between point clouds and cannot complete the deep feature interaction. The graph-based approach constructs the applicable graph representation structures by preprocessing the point cloud with sampling and clustering, which preserves the spatial features of the point cloud and makes up for the failure of the above networks to capture the topology of the point cloud. DGCNN

[30] computed the edge features between the neighboring points of each point in point clouds, thus extracting the dependency relationships between neighbors. But it fails to solve the problem of point sparsity and overlap. Wang Jiangan, He Jiao [31] combined PointNet and DGCNN and proposed a more optimized network model Linked-DGCNN to provide more reliable information for 3D scene interpretation. Point-GNN [2] efficiently encoded point cloud features in a fixed-radius nearest neighbor graph by graph neural network and proposed an automatic alignment mechanism to reduce the translation variance. The results show that the method achieves leading accuracy and even surpasses fusion-based algorithms. However, the above methods can hardly avoid the problems of insufficient point cloud feature and over-smoothing of graph neural network. Based on this, this paper proposes an improved 3D point cloud target detection model R-PointGNN based on graph neural network.

3 METHOD FOR TARGET DETECTION BASED ON R-PointGNN

The 3D point cloud target detection model based on R-PointGNN is shown in Figure $_1$, which is divided into four main steps: point cloud semantic graph construction, graph convolution construction, residual connection and targets detection.

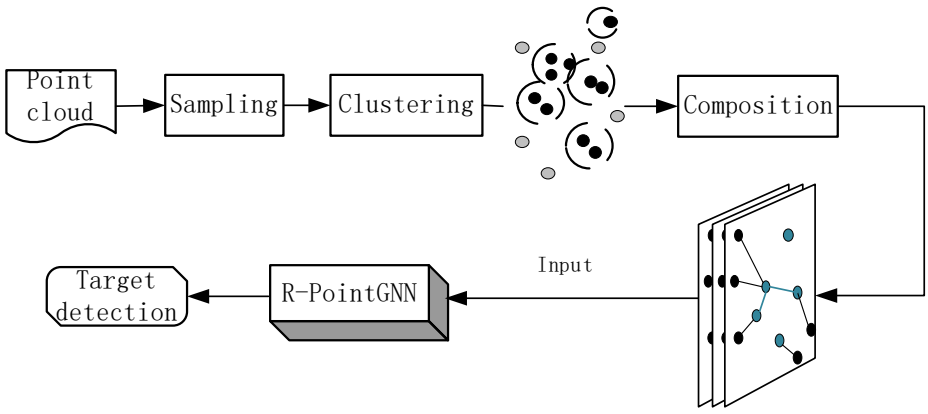

FIGURE 1. The structure of target detection model. It mainly includes modules of sampling, clustering, composition and target detection.

3.1 POINT CLOUD SEMANTIC GRAPH CONSTRUCTION

Semantic graphs are constructed by capturing the topological relationships between local points through

sampling and clustering algorithms to establish connected edges between the target and neighboring nodes [2]. In this paper, the construction method is selected from the nearest neighbor algorithm clustering algorithm based on the Kd-Tree index. Each node of the Kd-Tree is a binary tree of K-dimensional numerical points, and each node on it represents a hyperplane, and the retrieval space of the Kd-Tree is shown in Figure $_2$. The Kd-Tree is constructed by firstly determining the division dimension and then taking out the median of the data to divide the hyperplane until all points are divided. Compared with Ball-Tree index, Kd-Tree index is based on Euclidean distance, with time complexity close to $\log_2 n$. The search efficiency is higher. When querying the nearest neighbor points, the hyperplane division of Kd-Tree is more robust than hyperspherical division for sparse and uneven point clouds.

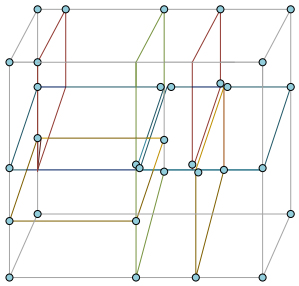

FIGURE 2. The retrieval space of Kd-Tree. Each node of the Kd-Tree is a binary tree of K-dimensional numerical points, and each node on it represents a hyperplane.

Define the set of point cloud data as $P = \{ P_1, P_2, P_3 \ldots P_n | (n \leq N)\}$, where P_i is represented by the attributes of $P_i = (X_i, S_i)$,and X_i is the three-dimensional spatial coordinate of the point $(x_i, y_i, z_i) \in \mathbb{R}^3$, and S_i denotes the state attribute of the point, and the laser reflection intensity is used for representation under the laser point cloud data set. For the given point cloud set, the sampled points and the topological relationships obtained by the KNN clustering algorithm are as follows.

$$F = \{ F_i, F_k, \ldots | D_{i,k} \text{ is } \mathrm{Min}(D_{i,1} \ldots D_{i,n}), k \in K, n \notin K, n \leq N - i + 1\} \quad (1)$$

$$e(i,j) = \{ (F_i, F_j) | j \in N_k(i)\} \quad (2)$$

where F denotes the set of sampled points, and the voxel downsampling algorithm is used to reduce the point cloud density [2] to obtain the target point cloud set F. $e(i,j)$ denotes the connection relationship, D denotes

the spatial distance metric, sampling Euclidean distance calculation. Max denotes taking the farthest distance, andMin denotes taking the closest distance. F_i is the target centroid, and F_k is the point to be sampled. $N_k(i)$ represents the neighborhood space of the point F_i. F_j represents the neighboring nodes in $N_k(i)$. To improve the query efficiency, the Kd-Tree index is used to determine the K-nearest neighbors of F_i. From this, the topological information between points is obtained and connected edges are constructed.

The specific construction processes of the semantic graph are as follows:

1) Reduce of point cloud density through voxel downsampling algorithms.

2) Traverse the sampling space and determine nodes' K-nearest neighbors by means of the KNN algorithm.

3) Create adjacency edges in nodes' neighborhoods and complete semantic graph construction.

3.2 GRAPH CONVOLUTION CALCULATION

The graph neural network completes node feature interaction through connected edges between nodes, including two phases of feature transfer and state update. The computational process can be explained under the Message Passing Neural Net (MPNN) framework [32] as equation (3) and (4).

$$e_{i,j}^{t} = \mathcal{f}^{t}(s_i, s_j), \quad j \in N(i) \qquad (3)$$

$$s_i^{t+1} = \mathcal{U}^{t}(\rho(e_{i,j}^{t}), s_i^{t}), i, j \in e(i,j) \qquad (4)$$

where t denotes the number of iterations and s denotes the node feature. i represents the target node index and j represents the neighbor node index. $N_{(i)}$ denotes the neighborhood space of node i. $\mathcal{f}^{t}$ represents the message transfer function, which is a differentiable function. ρ is the aggregation function of neighbor node feature. $\mathcal{U}^{t}$ represents the state update method.

Generally, the graph convolution calculation is limited to the point cloud spatial features. And it is easy to ignore state attributes of point cloud, especially the target node state. In this paper, in addition to the point cloud coordinates, the target node states s_i and neighbor node states s_j are also introduced in the convolution

calculation method. The own attributes of target nodes are added to compensate for the lack of information in the point cloud and further enhance the node features. The optimized EdgeConv convolution is constructed. And the auto-alignment mechanism [2] is used to reduce the translation variance and achieve the point cloud translation invariance. Thus, equations (3) to (4) are rewritten that yields the calculation of feature aggregation (5) and state update (6).

$$s_i^{t\prime} = \rho(\not{f}^t(\ x_j - x_i\ + \mathfrak{h}^t(s_i^t)\ ,\quad s_i^t,\quad s_j^t\ )), i,j \in e(i,j) \qquad (5)$$

$$s_i^{t+1} = \quad \mathcal{U}^t(\ s_i^t,\ s_i^{t\prime}), i,j \in e(i,j) \qquad (6)$$

Among these, the $\not{f}^t$ uses multi-layer perceptrons (MLP) to complete feature mapping and obtains deep abstract features. The feature aggregation method ρ is chosen as Max. And the local displacement invariance is achieved by the Max method for point cloud disorder. $\mathfrak{h}^t(s_i^t)$ denotes the coordinate offsets that are calculated by the state of node i. While the output of $\mathfrak{h}^t$ is setting to zero, this auto-alignment mechanism is to disable. Equation (5) is used to complete the feature interaction and aggregation of local node pairs by introducing the target and neighboring node states.

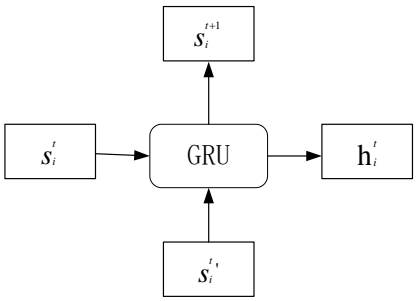

FIGURE 3. Input and output structure of GRU. s_i^t is the last time state, and $s_i^{t\prime}$ is the aggregate feature. s_{i+1}^t is the updated state.

The introduction of node states enhances the feature information, but also increases the influence of node noise in aggregation process, leading to the problem of over-smoothing of node features. Therefore, function $\mathcal{U}^t$ in this paper adopts Gate Recurrent Unit (GRU) as the state update model. The structure of input and output is shown in Figure 3. GRU is used to filter a large amount of node noise in the point cloud semantic

graph and mitigate the over-smoothing problem. The update expression of $\mathcal{U}^t$ is as follows:

$$s_i^{t+1} = (1-z) \circ s_i^t + z \circ s_i^{t\prime} \quad (7)$$

where z is the gating signal that controls the retention information and filtering information. The state properties of the target node at moment t+1 depend on the aggregated information in this state and the properties of the node itself at moment t. Function $(1-z) \circ s_i^t$ filters out some of the node noise, function $z \circ s_i^{t\prime}$ retain the focused information in the aggregated features. The mutual compensation of the two partial weights achieves the stability of the updated state. $\circ$ denotes the composite function calculation implementing by MLP.

3.3 R-PointGNN MODEL

In this paper, the structure of R-PoinGNN model is constructed as shown in Figure 4, where the details of GNN module are shown in Figure 5.

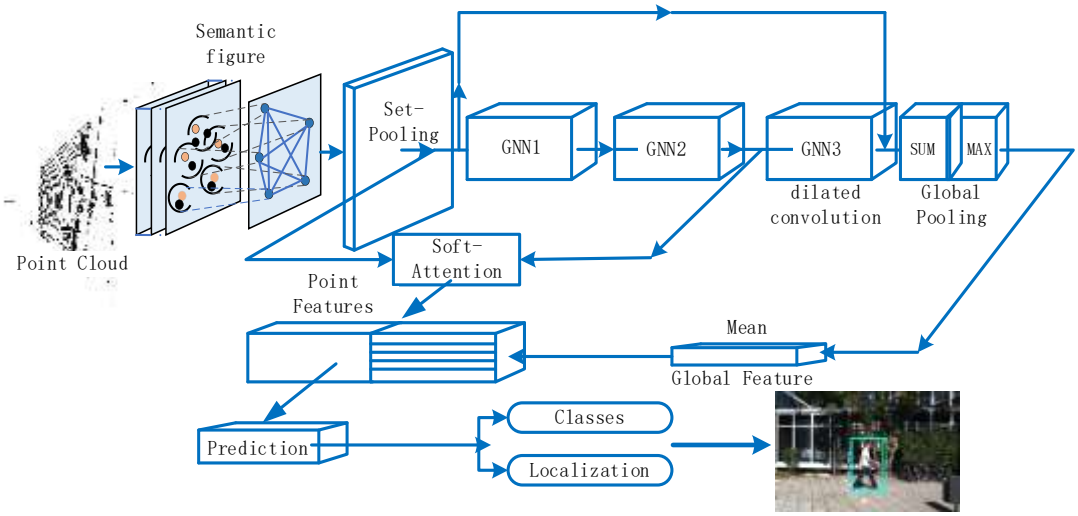

FIGURE 4. The tructure of R-PointGNN model. It has three layers of graph convolution. The layer of global pooling is made up by functions of sum and max. Then, the global feature combining with point features is got by using mean function. Finally, the graph feature is inputted to the layer of prediction.

R-PointGNN is built on the following six working modules.

1) First-order pooling: a pooling operation is on the semantic graph, extracting to local depth features, reducing the node size and shrinking the graph structure.

2) Second-order graph convolution: node feature aggregation is completed on the depth feature map, and the second-order neighborhood features of the target node are extracted through two iterations.

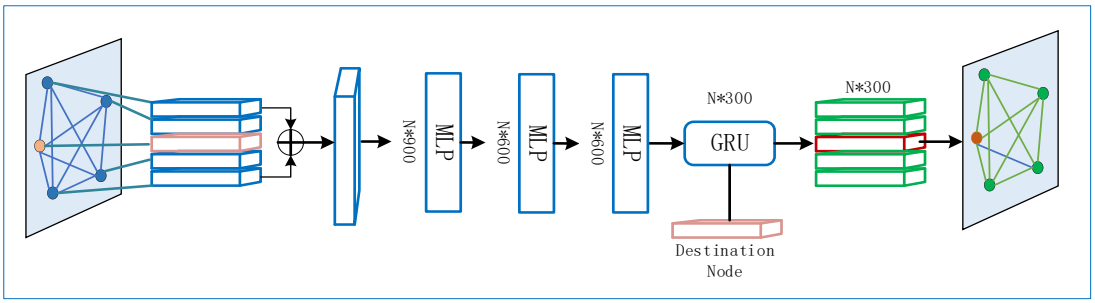

FIGURE 5. The module of GNN unit. The adjacent features aggregated of the target node transform the dimension through three layers of MLP, and the feature of the target node is updated through GRU module.

3) Dilated graph convolution: after the two-layers graph convolution operation, the reconstruction of the neighborhood is performed with the target node as the center. After the secondary composition of the point cloud data, the reconstructed semantic graph is input to the dilated graph convolution module to increase the node feature difference and reduce the influence of the neighborhood noise.

4) Residual concatenation: based on three iterations, the pooled features are regarded as the feature $State_0$ at moment t^0. Through a soft attention mechanism, the weight scores of the nodes are calculated; the scores are assigned to the node state $State_2$ at moment t^2 to calculate the weighted features; the $State_0$ and $State_3$ are feature concatenated for global pooling to obtain the semantic graph depth features.

5) Global pooling: the pooling functions are selected as MaxPooling and SumPooling methods. Max and Sum functions are symmetry functions that can achieve the displacement invariance of the point cloud and solve the disorder problem. By graph convolution feature extraction, the target nodes acquire local neighborhood features. To achieve the global feature transfer, the pooled feature vectors are on the MeanPooling operation again. Then the calculated features are used as the global feature vectors. The advantage of the Mean function over the Max function is that the Max function is prone to the loss of features, while the Mean function better preserves the node features.

6) Prediction: The point cloud features obtained from 4) are stitched with the global features obtained from 5) to perform node classification and localization prediction.

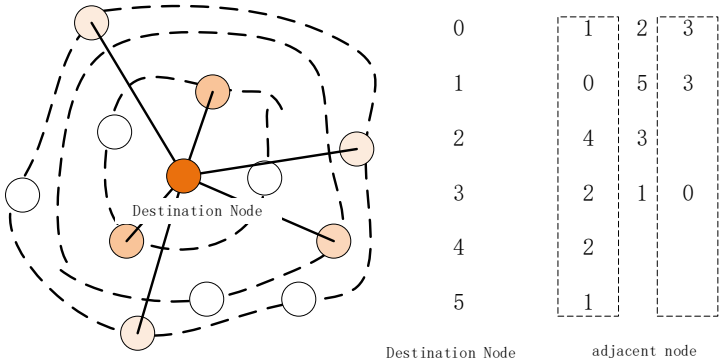

FIGURE 6. The structure of dilated graph convolution. The dilated rate is set to 2 and the Random rate is set to 0.2 during training.

R-PointGNN uses the two-time semantic graph construction method to reconstruct the neighborhood edges of point cloud after two layers of convolution. In the secondary composition, KNN is used to take 50 nearest neighbors of each target node, and then the point cloud neighborhood is calculated for each round of feature aggregation. The purpose of secondary composition is to reconstruct the connectivity relationship of point cloud to avoid the aggregated features under fixed graph structure from tending to be similar. It also increases the graph convolution field and reduces the neighborhood noise information by a layer of dilated graph convolution. The dilated graph convolution used in this paper is shown in Figure $_{6}$. Dilated convolution is generally used in image segmentation in combination with CNN. In this paper, it is extended to the non-Euclidean data space and combined with GNN to rapidly expand the neighborhood perception field. The Dilated rate is set to 2 and the Random rate is set to 0.2 during the computation to enhance the model robustness.

R-PointGNN expands the graph convolution feature field and enhances the point cloud feature information by jump residual connectivity and dilated graph convolution to suppress the convergence of node states under the state difference at different moments to optimize the model feature extraction performance.

4 EXPERIMENTS AND RESULTS ANALYSIS

4.1 EXPERIMENTAL DATA SET

KITTI experimental dataset is a computer vision algorithm evaluation dataset for autonomous driving scenarios, which can be used for many tasks such as 3D target detection and stereo image evaluation. For 3D object detection, the labels of the original KITTI dataset include car, pedestrian, cyclist and so on. And there are different degrees of occlusion, which can be classified into three modes: easy mode, moderate mode and hard mode. KITTI data fields are described as follows.

TABLE 1. The descriptions of data section field. The fields of data section includes type, occluded, alpha, etc.

Values	Name	Description
1	type	The type of object
1	occluded	Integer indicating occulusion state
1	alpha	Observation angle
3	dimensions	3D object dimensions
3	location	3D object location in camera coordinate
1	score	for results

TABLE 2. Example of training data. The given data contains the fields in Table 1. The labels of the KITTI dataset include truck, car, pedestrian, cyclist, etc.

Type	Data
Truck	0.00 0 -1.57 599.41 156.40 629.75 189.25 2.85 2.63 12.34 0.47 1.49 69.44 -1.56
Car	0.00 0 1.85 387.63 181.54 423.81 203.12 1.67 1.87 3.69 -16.53 2.39 58.49 1.57
Cyclist	0.00 3 -1.65 676.60 163.95 688.98 193.93 1.86 0.60 2.02 4.59 1.32 45.84 -1.55
DontCare	-1 -1 -10 503.89 169.71 590.61 190.13 -1 -1 -1 -1 -1000 -1000 -1000 -10
DontCare	-1 -1 -10 511.35 174.96 527.81 187.45 -1 -1 -1 -1 -1000 -1000 -1000 -10
DontCare	-1 -1 -10 532.37 176.35 542.68 185.27 -1 -1 -1 -1 -1000 -1000 -1000 -10
DontCare	-1 -1 -10 559.62 175.83 575.40 183.15 -1 -1 -1 -1000 -1000 -1000 -10

The KITTI dataset is divided into a training set and a test set, with 7481 training samples and 7518 test samples respectively. Each sample contains point cloud data and camera images. For training, we further divide the training set into training data containing 3721 samples and validation data containing 3769 samples. In this paper, only the LiDAR point cloud data is used as input and only three objects included car, pedestrian and cyclist are used as detection targets, while other categories are ignored.

4.2 LOSS FUNCTION AND EVALUATION CRITERIAL

1) LOSS FUNCTION

The study utilized the same loss function as in the literature [2] and [34]. The classification loss is defined as the average cross-entropy loss function (Average-Loss).

$$\text{Average\_loss} = -\frac{1}{n}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{c_j}^{i}\log\left(\ p_{c_j}^{i}\ \right) \qquad (8)$$

y_c^i and p_c^i denote the label and predicted probability of node i.

The 3D detection bounding box and anchor parameters are (x, y, z, l, w, h, θ), where (x, y, z) represents the center of the bounding box, (l, w, h) represents the length, width, and height, and θ is the yaw angle. Encoding the bounding box uses the node coordinates yields.

$$\Delta x = \frac{x-x_v}{l_m},\ \Delta y = \frac{y-y_v}{h_m},\ \Delta z = \frac{z-z_v}{v_m}$$

$$\Delta l = \log\left(\frac{l}{l_m}\right),\ \Delta h = \log\left(\frac{h}{h_m}\right),$$

$$\Delta w = \log\left(\frac{w}{w_m}\right),\ \Delta\theta = \log\left(\frac{\theta-\theta_0}{\theta_m}\right) \quad (9)$$

Among these, l_m, h_m, w_m, θ_m are constant factors.

And it calculates the localization loss by Huber loss, where the mean value of the loss at all points is taken as follows.

$$\text{Loc\_loss} = \frac{1}{N}\sum_{I=1}^{N}\mathbb{I}\left(v_i \in b_{interest}\right)\sum_{\Delta\in\Delta_{b_i}} l_{huber}\left(\Delta - \Delta^{gt}\right) \quad (10)$$

Also, the L_2-regularized loss function is added to prevent overfitting of the model.

$$\text{Reg\_loss} = \sum_i(\mu_i x_i - X_i)^2 + \lambda(\mu_i)^2 \quad (11)$$

In summary, the total loss calculation is obtained as equation (12), where a, b, c are constant factors.

$$\text{Total\_loss} = a\text{Average\_loss} + b\text{Loc\_loss} + c\text{Reg\_loss} \qquad (12)$$

2) EVALUATION INDICATORS

Precision, recall, and average precision (AP) are selected as evaluation metrics to verify the model performance calculated as in equation (13) to (15).

$$\text{Precision} = \frac{TP}{TP + FP} \quad (13)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (14)$$

$$AP = S_{P\_R} \quad (15)$$

Where TP (True Positive) denotes a positive class sample with correct prediction, and FP (False Positive) denotes a positive class sample with incorrect prediction, and FN (False Negative) denotes a negative class sample with incorrect prediction. AP is the area under the P-R curve of a certain class, which is calculated by Precision and Recall.

4.3 EXPERIMENTAL RESULTS

The model training parameters are as follows.

TABLE 3. Training parameter settings. The Initial learning rate is 0.1, and optimizer uses SGD.

Parameters	Value
NUM_GPU	2
BATCH_SIZE	4
LR	0.1
OPTIMZER	SGD

While testing the data, the time performance of the model is shown in Table 4.

TABLE 4. Run time of model. It counts the run time of image generation and graph convolution to get the total time of model.

Runtime(s/per)	Car	Pedestrain_Cyclist
Gen_graph	0.1814	0.2462
GNN_inference	0.2956	0.1102
Total_time	0.5762	0.4382

In terms of run time, the R-PointGNN time performance is around 0.5s, which is faster. And KNN composition algorithm based on Kd-Tree index consumes a composition time of about 0.2s, which is more efficient. However, the GNN module has a large time gap when it tests different categories, which may be due to the fact that the point clouds obtained from such data sampling as pedestrain and cyclist are less sparse.

Based on the above model parameters, the P-R curves obtained on the validation data are shown in Figure 7. (a) and (b) represent the detection results of Car and Cyclist categories on 2D detection, 3D detection and bird's eye view respectively.

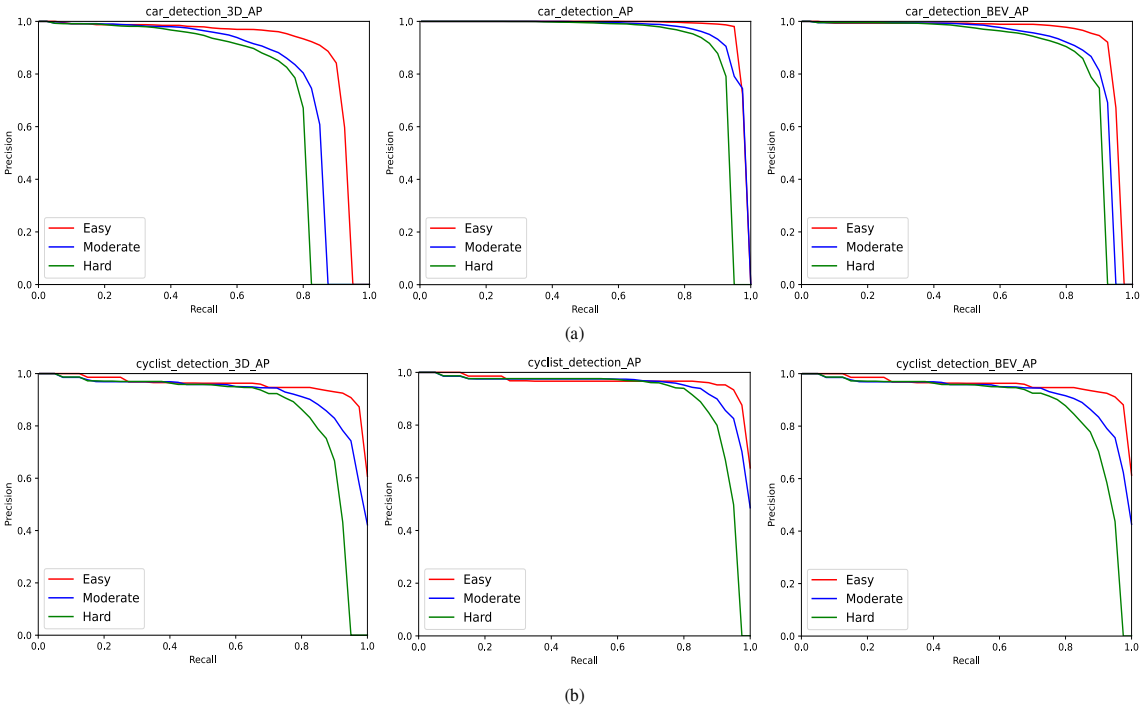

FIGURE 7. Validation P-R curves. (a) represents detection results for car under three detection methods of 2D detection, 3D detection and bird's eye view detection respectively. (b) represents detection results for cyclist under three detection methods.

Figure 8 shows the qualitative results measured by R-PointGNN on the validation set. (a) shows the detection results for the Car category; (b) detects only the Pedestrian and Cyclist categories; (c) shows the combined detection results. As shown in Figure 8, the green boxes represent the Car true values and the blue boxes represent the Pedestrian and Cyclist true values. The red boxes represent the predicted values. By verifying the results, the bounding boxes predicted by the model match well with the true values, and the model still has good detection performance even in complex scenes.

(a)

(b)

(c)

FIGURE 8. Detection effect on the validation set (a) detects only car; (b) detects pedestrian and cyclist; (c) is combined detection; in figure 8, the green, blue and yellow boxes represent the true values of car, pedestrian and cyclist; the red boxes correspond to the predicted values.

4.4 ABLATION EXPERIMENTS

In this paper, ablation experiments are conducted to compare the performance of the composition method with the update model, and the results are shown in Table 5. From the AP values, it is clear that the Kd-Tree based

retrieval method is more robust compared to Ball-Tree retrieval method for inhomogeneous point cloud data. And GRU as the update model effectively filters the node noise and optimizes the model performance.

TABLE 5. Comparison of AP for ablation experiments. It compares the performance of functions of graph generation and update.

Kd Tree	Ball Tree	GRU	BEV AP (Car)			3D AP (Car)		
			Easy	Moderate	Hard	Easy	Moderate	Hard
	√	√	89.71	87.47	86.39	87.34	77.59	75.21
	√		89.30	86.94	79.38	86.36	76.34	74.63
√		√	89.70	87.59	86.46	87.62	77.69	75.68

4.5 COMPARISON OF REALTED WORK

In this paper, we submit the R-PointGNN test results onto the official KITTI website (the test data are not provided with true values) and obtain the average precision (AP) under 2D detection, 3D detection and bird's eye view detection. We compare the test results with existing studies, and the comparison results are shown in the Table 6 to Table 8. Among the three detection methods, the proposed model has the best performance on 2D detection. Among the Car, Cyclist and Pedestrian categories, the model has the worst detection performance in the Pedestrian category. The possible reason is that the Pedestrian category has less point cloud data and is more sparse, which affects the detection effect.

TABLE 6. The Average Precision (AP) comparison of 2D object detection results. The best results are highlighted for each column.

	Car			Pedestrian			Cyclist		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
R-GCN [33]	96.19	92.67	87.66	N/A	N/A	N/A	N/A	N/A	N/A
PFF3D [34]	95.37	92.15	87.54	62.12	52.53	50.27	79.44	66.25	60.11
SCNet [35]	95.59	90.30	85.09	60.95	49.61	46.91	78.48	62.50	56.34
MAFF-Net [36]	94.38	91.46	83.89	N/A	N/A	N/A	N/A	N/A	N/A
MMLAB LIGA-Stereo [37]	**96.43**	**93.82**	86.19	**65.59**	52.18	49.29	74.40	54.57	48.11
CG-Stereo [38]	96.31	90.38	82.80	54.64	42.54	38.45	69.98	48.46	42.41
R-PointGNN	95.73	93.13	**87.94**	64.73	**54.19**	**50.49**	**79.46**	**69.73**	**63.46**

TABLE 7. The Average Precision (AP) comparison of 3D object detection results. The best results are highlighted for each column.

	Car			Pedestrian			Cyclist		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard

R-GCN [33]	83.42	75.26	68.73	N/A	N/A	N/A	N/A	N/A	N/A
PFF3D [34]	81.11	72.93	67.24	43.93	36.07	32.86	63.27	46.78	41.37
SCNet [35]	83.34	73.17	67.93	**47.83**	**38.66**	**35.70**	67.98	50.79	45.15
MAFF-Net [36]	**85.52**	75.04	67.61	N/A	N/A	N/A	N/A	N/A	N/A
MMLAB LIGA-Stereo [37]	81.39	64.66	57.22	40.46	30.00	27.02	54.44	36.86	32.06
CG-Stereo [38]	74.39	53.58	46.50	33.22	24.31	20.95	47.40	30.89	27.23
R-PointGNN	84.22	**76.08**	**69.01**	43.89	37.99	34.70	**69.59**	**57.07**	**51.33**

TABLE 8. The Average Precision (AP) comparison of Bird's eye view results. The best results are highlighted for each column.

	Car			Pedestrian			Cyclist		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
R-GCN [33]	**91.91**	86.05	81.05	N/A	N/A	N/A	N/A	N/A	N/A
PFF3D [34]	89.61	85.05	80.42	48.74	40.94	38.54	72.67	55.71	49.58
SCNet [35]	90.07	86.48	**81.30**	**56.87**	**46.73**	**42.74**	73.73	56.39	49.99
MAFF-Net [36]	90.79	87.34	77.66	N/A	N/A	N/A	N/A	N/A	N/A
MMLAB LIGA-Stereo [37]	88.15	76.78	67.40	44.71	34.13	30.42	58.95	40.60	35.27
CG-Stereo [38]	85.29	66.44	58.95	39.24	29.56	25.87	55.33	36.25	32.17
R-PointGNN	89.71	**88.06**	81.05	50.44	43.84	40.49	**75.14**	**62.41**	**55.52**

From the above table, it is clear that the proposed model has better detection performance on the Car and Cyclist categories. Among these results, the Cyclist category obtains the optimal values for the easy, moderate and hard modes of the benchmark method under the three detection methods of 2D detection, 3D detection and bird's eye view detection respectively.

5 CONCLUSION

In order to address the problems of sparsity and disorder of point clouds in 3D target detection tasks, this paper adopts a direct point cloud-oriented research method to construct a point cloud semantic graph, capture the topological features of the scene graph, and complete the feature interaction between point clouds through graph neural network for target detection under the global field of view. Based on the graph neural network, this paper proposes an improved 3D point cloud target detection model R-PointGNN. Through the point cloud composition algorithm of KNN, the problem of weak data uniformity and susceptibility to noise is optimized. And a more uniformly distributed point cloud semantic graph is constructed; the edge convolution calculation

method is improved and GRU unit is introduced to enhance the point cloud features and filters the neighborhood noise; the dilated graph convolution and residual connection are introduced to rapidly expand the convolutional perception field and optimize the model over-smoothing problem. The experimental results show that R-PointGNN has high robustness and better performance. However, the current stage of research also suffers from the problem of high time complexity, which is caused by the complexity of the graph convolution. Also, the detection performance is not so good for less sample scenarios where the point cloud is more sparse, such as the Pedestrian category.

In the next stage, we will further optimize the model structure and parameters, and study lightweight graph convolution detection algorithms to improve model timeliness and detection performance in sparse point cloud scenarios.

Abbreviations

KNN: The nearest neighbor algorithm; EdgeConv: The graph convolution; GNN: Graph neural network; PPBA: Progressive Population Augmentation.

Declarations

Authors' contributions

NFL: methodology, formal analysis. YL: conceptualization, writing—original draft preparation, validation. YYW: formal analysis. ZGX: Contrast test, formal analysis. NZ: formal analysis. YPL: formal analysis. All authors read and approved the fnal manuscript.

Acknowledgements

The work was supported by the project plan of science and technology development center of the Ministry of Education (No. 2020hyb03002) and in part by the Jilin Science and Technology Development Plan Project (Project No. 20200404221; 20210201083GX) and in part by Jilin Provincial Department of Education Plan Project (Project No. JJKH20210632KJ).

Availability of data and materials

The datasets generated and/or analysed during the current study are available in the KITTI repository, [www.cvlibs.net/datasets/kitti/]

Funding

There was no funding for the research.

Competing interests

There are no fnancial competing interests.

Author details

[1,4] College of Computer Science and Technology, Changchun University, No.6543, Satellite Road, Changchun City, China.

[2,3,5,6] Graduate School of Changchun University, Changchun University, No.6543, Satellite Road, Changchun City, China.

REFERENCES

[1] Guo, Yulan, et al, Deep learning for 3d point clouds: A survey, IEEE transactions on pattern analysis and machine intelligence 43(12), 4338-4364(2020)

[2] Shi, Weijing, and Raj Rajkumar, 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, (2020)

[3] Shi, Shaoshuai, et al, From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network, IEEE transactions on pattern analysis and machine intelligence 43(8): 2647-2664 (2020)

[4] Chen Q, Sun L, Wang Z, et al, Object as Hotspots: An Anchor-Free 3D Object Detection Approach via Firing of Hotspots, (2019)

[5] Shi, Shaoshuai, Xiaogang Wang, and Hongsheng Li, Pointrcnn: 3d object proposal generation and detection from point cloud, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, (2019)

[6] Liang Z, Zhang M, Zhang Z, et al, RangeRCNN: Towards Fast and Accurate 3D Object Detection with Range Image Representation, (2020)

[7] Beker, Deniz, et al, Monocular differentiable rendering for self-supervised 3d object detection, European Conference on Computer Vision, Springer, Cham, (2020)

[8] Shi X, Chen Z, Kim T K, Distance-Normalized Unified Representation for Monocular 3D Object Detection, European Conference on Computer Vision, Springer, Cham, (2020)

[9] Chen, Yilun, et al, Dsgn: Deep stereo geometry network for 3d object detection, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, (2020)

[10] Yoo J H, Kim Y, Kim J, et al, 3D-CVF: Generating Joint Camera and LiDAR Features Using Cross-View Spatial Feature Fusion for 3D Object Detection, (2020)

[11] Pang, Su, Daniel Morris, and Hayder Radha, CLOCs: Camera-LiDAR object candidates fusion for 3D object detection, 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, (2020)

[12] Huang, Tengteng, et al, Epnet: Enhancing point features with image semantics for 3d object detection, European Conference on Computer Vision, Springer, Cham, (2020)

[13] Meng, Qinghao, et al, Weakly supervised 3d object detection from lidar point cloud, European Conference on Computer Vision. Springer, Cham, (2020)

[14] Chen, Jintai, et al, A hierarchical graph network for 3d object detection on point clouds, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, (2020)

[15] Li, Jiale, et al, P2V-RCNN: Point to Voxel Feature Learning for 3D Object Detection from Point Clouds, IEEE Access 9, 98249-98260 (2021)

[16] Ge R, Ding Z, Hu Y, et al, AFDet: Anchor Free One Stage 3D Object Detection, (2020)

[17] Qi, Charles R, et al, Volumetric and multi-view cnns for object classification on 3d data, in Proceedings of the IEEE conference on computer vision and pattern recognition, (2016)

[18] Engelcke, Martin, et al, Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks, 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, (2017)

[19] Xu, Xu, and Sinisa Todorovic, Beam search for learning a deep convolutional neural network of 3d shapes, 2016 23rd International Conference on Pattern Recognition (ICPR), IEEE, (2016)

[20] Zhang Zhiyong, Wang Lei, Cheng Haixia. Classification of large field attraction clouds based on Feature-pointNet, Computer and Information Technology 29(01), 6-9+34 (2021)

[21] MA Jinghui, PAN Wei, WANG Ru. 3D point cloud classification based on K-means clustering, Computer Engineering and Applications 56(17), 181-186 (2020)

[22] Qi, Charles R., et al, Pointnet: Deep learning on point sets for 3d classification and segmentation, in Proceedings of the IEEE conference on computer vision and pattern recognition, (2017)

[23] Qi C R, Yi L, Su H, et al, PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space, (2017)

[24] Wang Xujiao, Ma Jie, Wang Nannan, Ma Pengfei, Yang Lichuang, A deep learning point cloud classification model based on graph convolutional networks, Advances in Lasers and Optoelectronics 56(21), 56-60 (2019)

[25] Ma Jie, Wang Xujiao, Ma Pengfei, Yang Lichuang, Wang Nannan, A deep learning point cloud classification network incorporating kd tree neighborhood query, Journal of Shenzhen University (Science and Technology Edition) 37(01), 79-83 (2019)

[26] Bai Jing, Si Qinglong, Qin Feiwei. LightPointNet, a lightweight real-time point cloud classification network, Journal of Computer-Aided Design and Graphics 31(04), 102-111 (2019)

[27] Cheng, Shuyang, et al, Improving 3d object detection through progressive population based augmentation, European Conference on Computer Vision. Springer, Cham, (2020)

[28] Liu Z, Zhao X, Huang T, et al, TANet: Robust 3D Object Detection from Point Clouds with Triple Attention, in Proceedings of the AAAI Conference on Artificial Intelligence 34(7), 11677-11684 (2020)

[29] Huang, Rui, et al, An lstm approach to temporal 3d object detection in lidar point clouds, European Conference on Computer Vision. Springer, Cham, (2020)

[30] Wang Y, Sun Y, Liu Z, et al, Dynamic graph cnn for learning on point clouds, Acm Transactions On 8 Computer Engineering and Applications Graphics (tog) 38(5), 1-12 (2019)

[31] WANG Jiangan, He Jiao, Pang Dawei, A point cloud classification and segmentation network based on dynamic graph convolutional network, Advances in Laser and Optoelectronics 58(12), 470-477 (2021)

[32] Gilmer, Justin, Schoenholz, Samuel S, Riley, Patrick F, Vinyals, Oriol，Dahl, George E, Neural Message Passing for Quantum Chemistry, (2017)

[33] Zarzar, Jesus, Silvio Giancola, and Bernard Ghanem, PointRGCN: Graph convolution networks for 3D vehicles detection refinement, arXiv preprint arXiv:1911.12236 (2019).

[34] Wen, Li-Hua, and Kang-Hyun Jo, Fast and accurate 3D object detection for lidar-camera-based autonomous vehicles using one shared voxel-based backbone, IEEE Access 9, 22080-22089 (2021)

[35] Wang, Zhiyu, et al, SCNet: Subdivision coding network for object detection based on 3D point cloud, IEEE Access 7, 120449-120462 (2019).

[36] Z. Zhang, Z. Liang, M. Zhang, X. Zhao, Y. Ming, T. Wenming and S. Pu, MAFF-Net: Filter False Positive for 3D Vehicle Detection with Multi-modal Adaptive Feature Fusion., arXiv preprint arXiv:2009.10945 (2020)

[37] Guo, Xiaoyang, et al, Liga-stereo: Learning lidar geometry aware representations for stereo-based 3d detector, in Proceedings of the IEEE/CVF International Conference on Computer Vision, (2021)

[38] Li, Chengyao, Jason Ku, and Steven L. Waslander, Confidence guided stereo 3d object detection with split depth estimation, 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, (2020)