2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

GNN-based Point Cloud Maps Feature Extraction and Residual Feature Fusion for 3D Object Detection

Wei-Hsiang Liao, Chieh-Chih Wang, and Wen-Chieh Lin

***Abstract*— LiDAR detection of long-range vehicles is challenging because very few and sparse points are measured in long distances and vehicles with similar shapes of targets could lead to false positives easily. To tackle these challenges, taking the environment information (HD maps) into account could be beneficial to predetermine where targets are more or less likely to appear. Compared with semantic maps, HD maps formed by point clouds provide much richer information from surrounding static objects and scenes. In this work, we construct a GNN-based feature extraction of point cloud maps to increase the receptive fields of learning map features. Our work is based on PVRCNN, the state-of-the-art LiDAR object detection method. With point-wise and voxel-wise features obtained from PVRCNN, residual feature fusion is proposed to fuse the features from PVRCNN and the map features from GNN. Our approach is evaluated on NuScenes dataset. It achieves a 24.78% average precision improvement for long-range objects at 40-50 meters, the farthest areas with ground truth annotation. Our approach also has a 4.22% reduction of false positives in the entire sensing areas.**

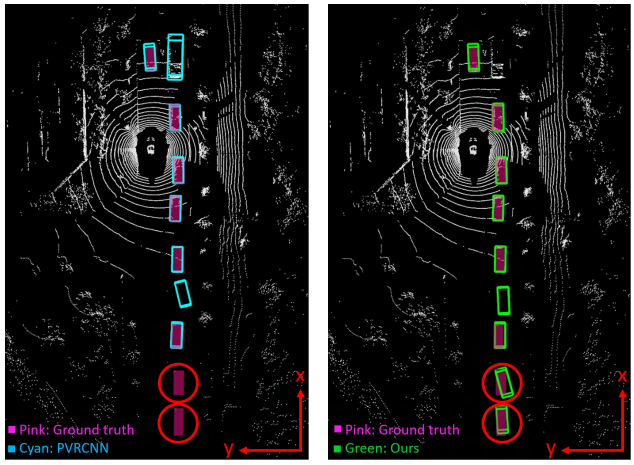
***Index Terms*— LiDAR, Object Detection, Self-Driving Cars, Point Cloud Maps.**

I. Introduction

LiDAR object detection is a crucial task for self-driving cars. Recently, many LiDAR object detection methods have emerged [1]–[3]. However, there are still some problems. First, the point clouds become sparse as the distance increases due to the physical properties of LiDAR sensing.Long-range vehicles usually only result in few points. The lack of point clouds leads to the feature of the vehicles being not enough and causes the vehicles to be not easily detected (Fig. 1).

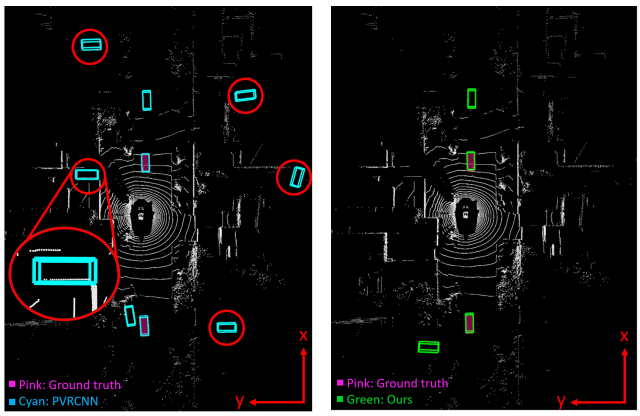
Second, LiDAR vehicle detection usually generates false positives when the shapes of point clouds are similar to the target object, such as building's corner and shrubberies. Fig. 2(a) shows that there is an edge composed of point clouds shown in the LiDAR measurements, caused by the building located on the side of the road. The shape is similar to the side of a car, usually leading to false positives.

As for the two problems we mentioned, consider that the environment information is beneficial to distinguish whether certain positions may appear vehicles. We propose a method that combines maps with LiDAR object detection to prompt the model to know where vehicles are located. Finally, Increasing the recall on long-range vehicles detection and reducing the false positives.



(a) PVRCNN (b) Ours

Fig. 1: **Long-range object detection improvement** (a) There is very few and sparse points are meatured in long distance. The vehicles are hard to be detected. (b) We combine point cloud maps into detection model and increase the recall on long-range.



(a) PVRCNN (b) Ours

Fig. 2: **False positives reduction**. (a) False positives are often caused by background point clouds. The features are only extracted by PV-RCNN (b) We reduce false positives by combining point cloud maps into the LiDAR object detection model.

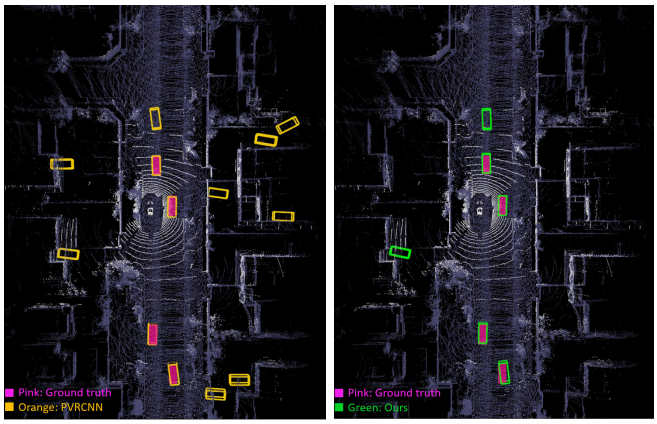
Point cloud maps are a result of multi-scans, which provide a more accurate scene. For example, the point cloud of the building's corner is a shape of right angle (As shown in Fig.4). This could help distinguish false positives from true ones. On the other hand, point cloud maps provide static scenes and objects in the long range. For example, the ground

Wei-Hsiang Liao is with the Graduate Degree Program of Artificial Intelligence, National Yang Ming Chiao Tung University, Hsinchu, Taiwan. E-mail: zxc741852741.ee08@nycu.edu.tw

Chieh-Chih Wang is with the Department of Electrical and Computer Engineering, National Yang Ming Chiao Tung University, and with the Mechanical and Mechatronics Systems Research Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan. E-mail: bobwang@ieee.org

Wen-Chieh Lin is with the Institute of Multimedia Engineering, National Yang Ming Chiao Tung University, Hsinchu, Taiwan. E-mail: wclin@cs.nctu.edu.tw

is a flat plane of point clouds, which can help to detect long-range vehicle. There are some challenges in combining point cloud maps with LiDAR object detection. Point cloud maps provide many point clouds about the environment. These additional point clouds may lead the model to generate more false positives if the receptive field is not enough (see Fig. 3(a)). Therefore, we construct a GNN-based feature extraction to increase the receptive field of point cloud maps. Based on the PVRCNN which provides point-wise features and voxel-wise feature of LiDAR scan. Another challenge is to fuse the map feature with other features. Because a general MLP layer is not easy to fuse the huge disparity of different features that contain a huge amount of information, we propose a residual feature fusion to fuse point-wise features, voxel-wise feature and map feature. First stage, the point-wise features, voxel-wise features are fused. Here, we call the fused feature PV-features. Second stage, learning a residual feature to fuse the map feature and PV-features by adding a skip connection. Therefore, the residual feature is capable of fusion map feature to produce more representative features.



(a) Without GNN (b) With GNN

Fig. 3: **Compare the model without or with GNN when add the point cloud maps**.

Contributions. We propose to combine point cloud maps which provide detailed information in the LiDAR object detection model to improve the detection performance. To learn the environment feature on point cloud maps, we construct a GNN-based feature extraction to extract the map feature and increase the receptive field by feature passing. After the map features are extracted, we propose a residual feature fusion to fuse point-wise features, voxel-wise features and map features. Finally, we trained our approach on the training data from the nuScenes dataset [4] and tested it on the validation data. For long-range task, our proposal obtains a 24.78% average precision improvement for the long-range vehicles in 40-50 meters. For the reduction of false positives, we have a 4.22% improvement for the overall areas.

II. Related Work

A. *LiDAR Object Detection*

Zamanakos et al. [5] and Qian et al. [6] introduced a wide variety of methods for LiDAR object detection. PVRCNN [3] is a top LiDAR object detection model among numerous

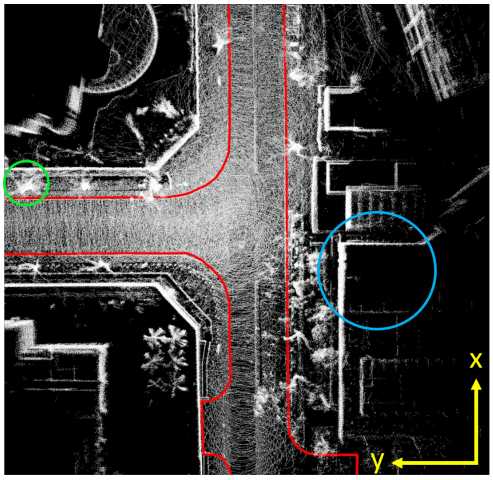


Fig. 4: **Compare two kinds of maps in self-driving car**. The red line is the semantic maps provided from the nuScenes dataset. White points are the point clouds maps. There is a corner of the building in the cyan circle. There is a tree in the green circle.

methods and achieved the first place in the KITTI dataset [7]. Therefore, we adopt PVRCNN [3] as a baseline method and verify the addition of the map to the model. Like mostly voxel-based methods, PVRCNN [3] first voxelizes point clouds and advances the convolutional neural network (CNN) to extract voxel-wise features. However, voxelization of these point clouds may lead to a loss of detailed features of point clouds. To address this problem, PVRCNN [3] proposes a fusion of voxel-wise features and point-wise features. After extracting voxel-wise features, the model will generate a certain number of rough bounding boxes. These bounding boxes will be refined by raw point clouds that are within the bounding boxes. The feature of point clouds that are within the bounding boxes will be extracted by Multilayer Perceptron (MLP) [8] and concatenated with voxel-wise features. Finally, these features will be used to refine the previous bounding boxes and generate more precise bounding boxes. PVRCNN++ [9] is an advanced version of PVRCNN [3], proposing an efficiency keypoints sampling module and VectorPool to improve the efficiency problem. Both PVRCNN and PVRCNN++ [9] exhibit comparable performance, but our method outperforms PVRCNN [3]. Based on our findings, it is reasonable to infer that our method will achieve better performance than PVRCNN++ [9].

Inspired by CenterNet [10], Yin et al. [11] propose a two-stage object detection method. The target of the first stage is to find the position of the target object in the LiDAR scan. The model considers the center of the object as a target and adopts a convolution as a feature extraction to find the position of the target. Second stage, generate the precise bounding box from first stage results. Inspired by DETR [12], Bai et al. [13] adopted a convolution network to extract features and generate object queries. Next, they use a transformer architecture to find the global relationship of the queries. Finally, the detection head produces the final detection results. SE-SSD [14] propose a pair of teacher and

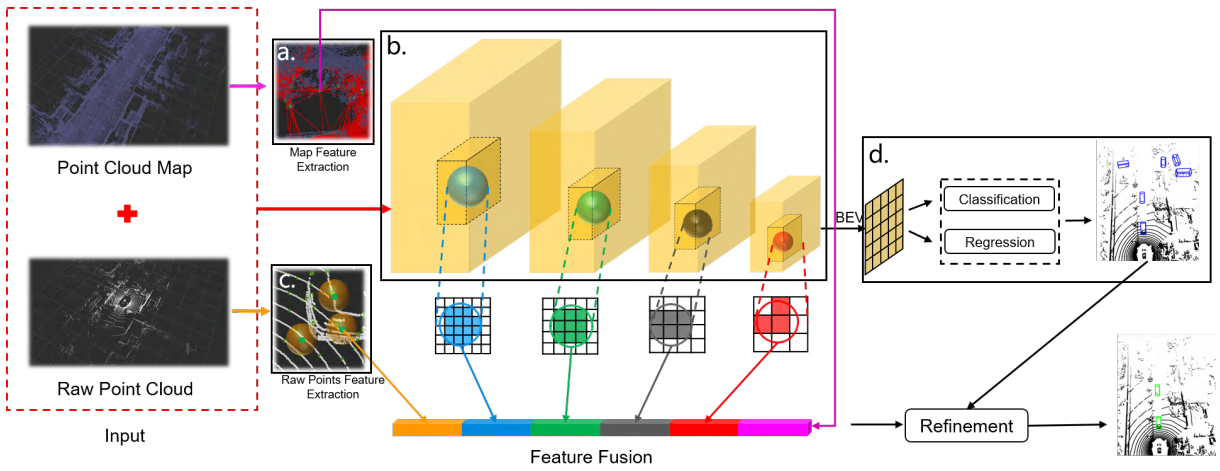


Fig. 5: **The architecture of the proposed approach**. There are three feature extraction modules, a. Graph Neural Network is used to extract features from point cloud maps, b. Convolution Neural Network is adopted to extract voxel-wise features, c. point-wise features are extracted by Multilayer Perceptron, d. The ROI is a certain number of rough bounding boxes. They are generated by voxel-wise features. Three types of features will be fused and used to refine ROI generated from voxel-wise features.

student detection model and design an effective IoU-based matching strategy to filter soft targets from the teacher and formulate a consistency loss to align student predictions with them.

B. LiDAR Object Detection with Maps

In self-driving car, Maps can be divided into semantic maps and point cloud maps. Both maps have different advantages. Semantic maps provide high-level information but lack detailed environment information. Point cloud maps provide detailed point clouds of the environment, but usually come with a higher computational cost. HDNET [15] proposes that semantic maps provide strong priors that can improve the performance of LiDAR object detection. First, it replaces the z-axis of point clouds within drivable areas to zero. Second, voxelize point clouds into discretized representations. Next, transfer semantic maps into bird's eye view as a binary channel at the same resolution of voxelization of point clouds and then concatenate semantic maps with voxelization of point clouds. Finally, CNN will be used to extract the feature of point clouds and semantic maps. We also implemented the HDNet encoding method in PVRCNN. The performance is shown below,

TABLE I: Recall

	Drivable area	Non-drivable
PVRCNN	0.845	**0.817**
PVRCNN w/ semantic maps	**0.846**	0.806

TABLE II: Precision

	Drivable area	Non-drivable
PVRCNN	**0.750**	0.592
PVRCNN w/ semantic maps	0.676	**0.593**

The recall in non-drivable areas decreases. This is because if the object is located in non-drivable areas. The model tends to distinguish the object as the background. But targets often appear in non-drivable areas, and this will cause the recall decrease. On the other hand, the precision in the drivable area obviously decreases. This is because if the object is located in drivable areas, the model usually distinguishes the object as the target, regardless of whether the object is a true target or not.

Instead of semantic maps that provide high-level information, point cloud maps provide detailed point clouds of the environment. We construct a Graph Neural Network(GNN) to extract the geometric features of point cloud maps and to expand the receptive field of the model. Because the feature extracted from point cloud maps through GNN has various details of point clouds, these extracted features have various geometrical information about the environment. It can better interpret the information of the environment. For example, Fig.4 shown the comparison of semantic maps and point cloud maps. There is a corner of the building in the cyan circle. Point cloud maps extract the feature of the building through GNN, but semantic maps only provide "non-drivable areas" information. There is a tree in a green circle. point cloud maps extract the feature of tree through GNN, but semantic maps only provide "non-drivable areas" information. Point cloud maps provide detailed point clouds of the environment. Therefore, we adopt point clouds maps to improve the LiDAR object detection.

III. Proposed Method

In this section, we will first describe the feature extraction of the baseline model and the region of interest (ROI) generation module in Section III-A. We then describe how to combine point cloud maps into the LiDAR object detection model in Section III-B. The principle of GNN is described in Section III-C. Finally, Section III-D presents the residual feature fusion to produce the final detection results.

A. Voxel-wise and point-wise features Extraction and Region of Interest

We implement our proposal based on PVRCNN. PVRCNN deeply integrates voxel-wise features and point-wise features to learn more discriminative point cloud features. For voxel-wise features, PVRCNN adopt convolution as the voxel-wise features extraction backbone. Specifically, the input points P=$\{p_1, , p_n\}$ are first divided into small voxels with a spatial resolution of L×W×H, where the features of the voxels are directly calculated as the mean of all attributes of the points within the voxel. The common point attribute

uses 3D coordinates and reflectance intensities of each point. Then the convolution network uses sparse convolution to transform the input voxel into a feature map with 1×, 2×, 4×, 8× downsampled sizes (As shown in Fig. 5 (b)).

To extract point-wise features, PVRCNN first adopt the Furthest-Point-Sampling (FPS) algorithm to sample a small number of n keypoints K=$\{k_1, , k_n\}$ from the point clouds P. The strategy is to obtain a uniform distribution of sampling keypoints. Next, we search for most C raw points within a radius r of each keypoints. Finally, Multilayer Perceptron (MLP) is adopted to extract these neighboring points features, i.e., point-wise features. Given a points set P, the set of the neighboring points of the keypoints $k \in K$ is defined by

$$f_p = max(f(\{p_i \in P \mid \|k_j - p_i\| \leq r\})) \quad (1)$$

where $f(\cdot)$ denotes the MLP layers, f_p denotes the point-wise features, $max(\cdot)$ is the maximum pooling function adopted to aggregate the features (As shown in Fig. 5 (c))

The region of interest (ROI) represents a certain number of roughly bounding boxes generated from the last layer of the convolution feature map (As shown in Fig. 5 (d)). Each grid of the last feature map will generate two bounding box, which contain confidence scores and bounding boxes size. Next, ROI will be refined by point-wise features and voxel-wise features. To combine environment information, we add point cloud maps into model and extract a map feature to refine bounding boxes based on PV-RNN.

B. *Point Cloud Maps*

Point cloud maps are established by stacking each LiDAR scan according to the ego-pose of each timestamp. Point cloud maps contain rich environment features, such as buildings, ground, and trees (As shown in Fig. 4). The static environment information is beneficial for distinguishing whether an object may appear in a certain position or not. We first combine point cloud maps with LiDAR scan as input into the model. Although the recall increases, there are also a lot of false positives increase too. Because the receptive field of the model is not enough to learn the surrounding environment features, the main distinguishing features still depend on local point clouds. To expand the receptive field from point cloud maps and extract the detailed geometry features, we propose to employ the GNN to extract features from point cloud maps and merge the map features with point-wise and voxel-wise features.

C. *Graph Neural Network*

Graph Neural Network [16] [17] is a classical feature extraction and aggregate model. The core concept is that passing features from neighboring vertices and updating each vertex. The receptive field enhances as the iterations of the feature passing increase. Wang et al. [18] and Zhang et al. [19] construct a GNN to extract the features of point clouds and focus on the classification and segmentation. Inspired by them, we construct a GNN to extract the feature of static environment scenes and objects from point cloud maps.

To construct a GNN (As shown in Fig. 6), the number of N_v vertices will be selected by using Furthest-Point-Sampling [20] to choose the uniform distribution points in all raw points. The vertices are connected to the vertices of its neighbors within a fixed radius r_g. The connection of two vertices indicates an edge of the graph, i.e.

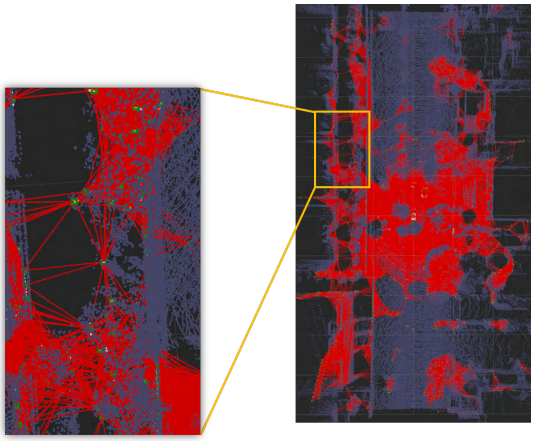


Fig. 6: **Graph neural network visualization** The red line represents the edges and the green point represents the vertex.

$$E = \{(k_i, k_j) \mid \|k_i - k_j\| \leq r_g\} \quad (2)$$

where E denotes the set of edges, k_i is the source point of the edge and k_j is the destination point of the edge. Each edge is composed of two components. The first component is the vector generated by the source point and each destination point. The second component is the feature of vertices. Each edge can be defined as,

$$e_{ij}^t = \{(k_i - k_j, v_j^t) \mid (i,j) \in E\} \quad (3)$$

where v_j^t denotes the feature of the vertices, the feature of the vertices will be updated by iterations of feature propagation, i.e.

$$v_i^{t+1} = \{g^t(max(f^t(e_{ij}^t \mid (i,j) \in E)))\} \quad (4)$$

where $f(\cdot)$ denotes edge feature extraction. $max(\cdot)$ is adopted to aggregate features. $g(\cdot)$ takes the aggregated edge features to update the vertices features. The initial feature of vertices is produced by aggregating neighboring point clouds within radius r from point cloud maps to represent the local environment features,

$$v_i^0 = \{q(max(p(\{p_i \in P^{'} \mid \|k_j - p_i\| \leq r\})))\} \quad (5)$$

where $P^{'}$ denotes the points in the point cloud maps. $p(\cdot)$ denotes points feature extraction. $max(\cdot)$ is adopted to aggregate features. $p(\cdot)$ denotes a layer for the embedding feature of points. The features of vertices will be passed to neighboring vertices. When the number of iterations of feature propagation increases, the receptive field of each vertex becomes larger. After the GNN is constructed, map features are extracted by GNN. So far, we have three types of features: voxel- wise features, point-wise features, and map features. Three features are fused to refine bounding boxes and generate the final bounding boxes. Because of the different properties between three types of features, we propose a residual feature fusion to fuse three types of features.

D. *Residual Feature Fusion and Bounding Box Refinement*

In this section, we will describe how to fuse features and use it to refine the bounding boxes generated from convolution. So far, we have three types of features: voxel-wise features, point-wise features, and map features. As

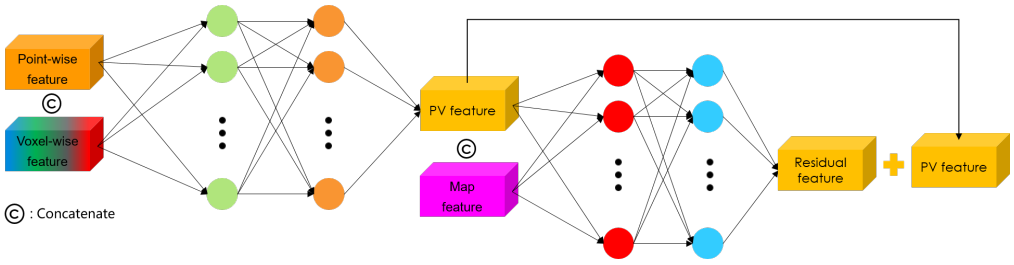


Fig. 7: **The residual feature fusion**

described in the PVRCNN [3], Multilayer Perceptron is used to fusion voxel-wise features and point-wise features to refine the roughly bounding boxes produced by convolution. We first concatenated the map features with the point-wise features and voxel-wise features,then adopted the MLP layer to fuse three types of feature.

$$f_{all} = MLP(f_p, f_v, f_m) \tag{6}$$

where f_{all} denotes the fusion feature, f_p denotes the point-wise features, f_v denotes the voxel-wise features, and f_m denotes the maps features. There is no obvious improvement. We are conscious that the fusion of different features only depending on a MLP layers is not enough, so we adopt residual feature fusion into feature fusion. We consider two possible reasons and modify the strategy of feature fusion. First, because we want to extract the feature of environment information, the receptive field of map features is larger than point-wise features and voxel-wise feature. Second, point-wise features and voxel-wise features are extracted from LiDAR scan, and map features are extracted from point cloud maps through GNN. The feature of LiDAR scan is closer to the target object, leading to a model that can converge to results only with point-wise features and voxel-wise features.

Different properties of the feature may lead to using a MLP to fuse three types of feature that are not easy to converge to a better loss. Therefore, we adopt the two-stage feature fusion (Fig. 7). In the first stage of fusion, we merge point-wise features and voxel-wise features. The input data of both features contain LiDAR scan. Here, we call the fusion feature PV-features.

$$f_{pv} = h(f_p, f_v) \tag{7}$$

where f_{pv} denotes the fusion features of the point-wise features, f_p, and the voxel-wise features, f_v. $h(\cdot)$ is a MLP layer to fusion f_v and f_p.

In the second stage, referring to the advantages of ResNet [21], we consider combining residual learning into the feature fusion module. The core concept of residual learning is that using a "shortcut" to prompt the model learning the residual feature and to let the model converge to a better loss.

$$y = F(x) + x \tag{8}$$

where $F(x)$ denotes the residual feature, F is a weight layer, and x denotes an input feature.

Consider the characteristic of three types of feature and residual learning, we bring in a residual learning into network and use map features learning a residual features and combine with PV-features to generate better results.

The target of the second stage is to use the map features to learn residual features and fuse environment information. We concatenate the PV features f_{pv} and the map features f_m, then transform them into a MLP layer, and the residual results will be added to the features f_{pv}.

$$f_{all} = l(f_{pv}, f_m) + f_{pv} \tag{9}$$

where $l(\cdot)$ denotes a layer that learns a residual feature. Finally, after fusion of the features, these features will be used to refine the bounding boxes generated from convolution. So far, the keypoints chosen from the raw points contain all the features that are extracted from the data. To refine the bounding boxes, for each rough bounding box, we choose keypoints that are inside the bounding boxes. These keypoints potentially contain the feature of the surrounding data of the bounding boxes, including the feature of the LiDAR scan and point cloud maps. We transform these features into another feature space by the MLP layer and use the detection head, which can be divided into a classification head and a regression head to finish the final detection step that generates the final bounding boxes.

For the loss function, we follow PVRCNN [3] to adopt focal loss as a classification loss and adopt smooth-L1 loss as regression loss.

IV. Experiments

We first describe the implementation details in Section IV-A and then the experimental results in Section IV-B.

A. *Implementation Detail*

We implement our method on nuScenes [4] dataset and use the AP (Average Precision), precision, and recall as evaluation metric.

1) Point Cloud Maps: We construct point cloud maps using LiDAR scans, ego-poses, semantic segmentation annotations. There are two steps to construct a point cloud maps. First, point cloud maps are static pre-built data, so we need to remove the moving objects of the LiDAR scan. NuScenes dataset provides a precise semantic segmentation annotation that contains 32 types of class, and we keep the point clouds that belong to the static objects and remove the point clouds that belong to the moving objects. Second, nuScenes provides a precise ego-pose. we stack the LiDAR scan according to ego-pose to construct point cloud maps. Finally, point cloud maps are constructed. Second, nuScenes provides a precise ego-pose. we stack the LiDAR scan according to ego-pose to construct point cloud maps. Finally, point cloud maps are constructed.

2) Construct Graph Neural Network: To construct a graph, the number of vertices must be determined. We use $N_v = 4096$ as the number of vertices. Such a strategy encourages that the vertices are uniformly distributed around LiDAR scan. After the vertices are determined, we choose

the neighboring points within a radius $r_g = 5m$ of vertices. The center is a vertex that is regarded as a source point. Neighboring points are regarded as destination points. Each vertices have a initial feature. The initial feature of vertices is produced by aggregating neighboring raw points within radius (0.8, 1.6m) through the two MLP layers of units (32, 32) and maximum aggregation. The feature of vertices will be updated by aggregating neighboring edges through the MLP layers of units (64, 128, 256) and another MLP layers of units (256, 256) after maximum aggregation. When the feature of vertices is updated, the receptive field of features will be expanded. We choose the number of iteration $t = 3$ in our method to update the feature of vertices.

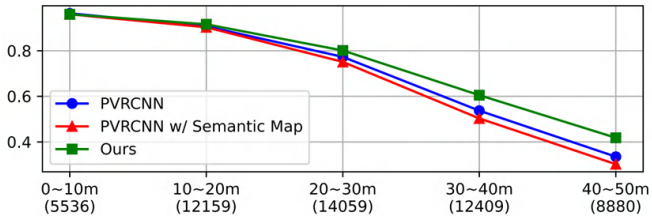
3) Residual Feature Fusion: We first merge point-wise features and voxel-wise features using MLP layers of units (128, 128) which result here we call PV-features, then we concatenate PV-features with map features and learn a residual feature through MLP layers of units (256, 256, 128). The residual features will add to the PV-features.

B. *Experimental Results*

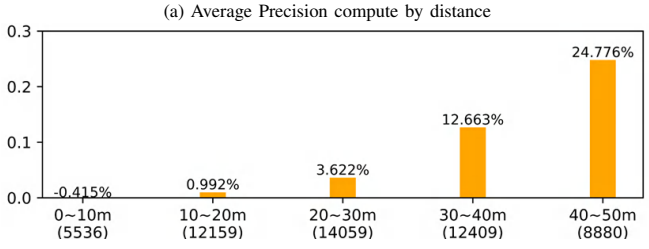
1) Performance: We adopt several object detection strategies to evaluate the performance in the above situation. First, to evaluate performance on long-range, we compute the AP (Average Precision) by distance (Table III). For results, there is a higher improvement rate as the distance increases. There is a 24.776% improvement rate for the range of 40m to 50m (Fig. 8).

TABLE III: Average Precision for different range of distance

	0-10m	10-20m	20-30m	30-40m	40-50m	Overall
PVRCNN	**0.964**	0.907	0.773	0.537	0.335	0.742
PVRCNN w/ semantic maps	0.960	0.903	0.751	0.503	0.302	0.728
Ours	0.960	**0.916**	**0.801**	**0.605**	**0.418**	**0.781**



(a) Average Precision compute by distance



(b) Improvement rate

Fig. 8: **Average Precision for different range of distance**. The numbers in parentheses represent the number of ground truth

Second, we evaluate the performance on false positives reduction. False positives often appear on the shape of point clouds similar to the target object which are located in non-drivable areas, so we specifically compute the precision of non-drivable areas. For overall precision performance, there is a 4.22% improvement. For non-drivable areas, there is a 8.11% improvement (Table IV).

TABLE IV: Precision improvement

	Overall	Non-drivable (Building etc.)
PVRCNN	0.616	0.592
PVRCNN w/ semantic maps	0.565	0.563
Ours	**0.642** (4.22%↑)	**0.640** (8.11%↑)

It is noteworthy that point cloud maps bring in a richer information and prompt the improvement of the performance but also company by higher computational complexity. The inference time of our model is double that of the baseline model.

2) Ablation Study: In this work, the model can be divided into three modules. First, we only add point cloud maps to the baseline model and are aware that there are more false positives that appear because of the shape of local point clouds of map that are similar to the target object. Second, we consider building a GNN to expand the receptive field and extract the detailed geometry feature from point cloud maps. Third, we are conscious that the fusion of different features only depending on a MLP layers is not enough, so we adopt residual feature fusion into feature fusion. The features we extract from point cloud maps will prompt the model weighting convergence to a better solution.

We use PVRCNN [3] as a baseline to validate our proposal. The performance and each experiment setting are shown below:

TABLE V: The ablation study of our proposal. Four experiments are verified in the nuScenes dataset.

Baseline	Point Cloud Map	Graph Neural Network	Residual Feature Fusion	AP
✓				0.741
✓	✓			0.769
✓	✓	✓		0.772
✓	✓	✓	✓	0.781

3) Localization error: The point cloud maps provide static environment information in the real world. If localization errors appear, there is a negligible effect. We give a control variable for the localization error. The results are shown below, In fact, the localization precision rarely occurs

TABLE VI: Average Precision for localization error

Shift	0m	0.5m	1m	1.5m	2m	2.5m	3m	3.5m	4m
AP	0.781	0.778	0.768	0.759	0.749	0.746	0.739	0.737	0.735

with an error greater than 0.5m in the real world. In general, our method can work and is not affected by a localization error.

V. Conclusion

We propose to combine point cloud maps which provide rich environment information into the LiDAR object detection model to extract static environment features. Static environmental information is beneficial for distinguishing whether an vehicle may appear in a certain position or not, and finally achieving two goals. First, the model can detect vehicles through the LiDAR scan and environment information and improve the performance of long-range vehicle detection. Second, we construct a Graph Neural Network to extract the features of point cloud maps and expand the receptive field of the model to reduce false positives. To fuse different features, we propose a residual feature fusion to merge point- and voxel-wise features with map features and make lower confidence scores on non-drivable areas.

REFERENCES

[1] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.

[2] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 12 689–12 697.

[3] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "Pv-rcnn: Point-voxel feature set abstraction for 3d object detection," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 526–10 535.

[4] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020.

[5] G. Zamanakos, L. Tsochatzidis, A. Amanatiadis, and I. Pratikakis, "A comprehensive survey of LiDAR-based 3D object detection methods with deep learning for autonomous driving," *Computers & Graphics*, vol. 99, pp. 153–181, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0097849321001321

[6] R. Qian, X. Lai, and X. Li, "3d object detection for autonomous driving: A survey," *CoRR*, vol. abs/2106.10823, 2021. [Online]. Available: https://arxiv.org/abs/2106.10823

[7] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[8] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[9] S. Shi, L. Jiang, J. Deng, Z. Wang, C. Guo, J. Shi, X. Wang, and H. Li, "Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection," *International Journal of Computer Vision*, pp. 1–21, 2022.

[10] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6569–6578.

[11] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3d object detection and tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 11 784–11 793.

[12] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229.

[13] X. Bai, Z. Hu, X. Zhu, Q. Huang, Y. Chen, H. Fu, and C.-L. Tai, "Transfusion: Robust lidar-camera fusion for 3d object detection with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1090–1099.

[14] W. Zheng, W. Tang, L. Jiang, and C.-W. Fu, "Se-ssd: Self-ensembling single-stage object detector from point cloud," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 494–14 503.

[15] B. Yang, M. Liang, and R. Urtasun, "Hdnet: Exploiting hd maps for 3d object detection," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87. PMLR, 29–31 Oct 2018, pp. 146–155. [Online]. Available: https://proceedings.mlr.press/v87/yang18b.html

[16] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[17] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.

[18] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *Acm Transactions On Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.

[19] K. Zhang, M. Hao, J. Wang, C. W. de Silva, and C. Fu, "Linked dynamic graph cnn: Learning on point cloud via linking hierarchical features," *arXiv preprint arXiv:1904.10014*, 2019.

[20] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0304397585902245

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.