Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 8151805, 13 pages
https://doi.org/10.1155/2022/8151805

Research Article

Research on 3D Point Cloud Object Detection Algorithm for Autonomous Driving

Haiyang Jiang,[1] **Yuanyao Lu**,[2] **and Shengnan Chen**[2]

[1]*School of Electrical and Control Engineering, North China University of Technology, Beijing, China*
[2]*School of Information Science and Technology, North China University of Technology, Beijing, China*

Correspondence should be addressed to Yuanyao Lu; luyy@ncut.edu.cn

Received 4 January 2022; Revised 22 January 2022; Accepted 25 January 2022; Published 17 February 2022

Academic Editor: Haruna Chiroma

In autonomous driving, lidar has become the main vehicle sensor due to its advantages such as long-range measurement and high accuracy. However, the collected point cloud data is sparse and unevenly distributed, and it lacks characterization capabilities when facing objects with missing or similar shapes, so that the detection accuracy is low while detecting long-distance small targets with similar shapes and a small number of point clouds. In order to improve the detection accuracy of small targets represented by point clouds, this paper adopts a method that fuses point cloud and RGB image to construct a 3D object detection network architecture based on two-stage complementary fusion. In the first stage of the fusion, we use the FPS method to select some points from the raw point cloud data as key points. Then, we voxelize the raw point cloud and use the 3D sparse convolutional neural network to extract multi-scale points cloud features, which would fuse features of different scales with key points. In the second stage of fusion, a 2D object detector is used to obtain the 2D bounding box and category information of the target in the image and take the camera as the origin to extend along the direction of the 2D bounding box to form a frustum; then the point cloud and target category information within the frustum are fused into the key points. This paper uses key points as a bridge to effectively combine the semantic information of the image such as texture and color with the point cloud features. The experimental results show that the network proposed in this paper has excellent performance on small objects.

1. Introduction

Deep learning, as a major artificial intelligence technology, has been widely used in computer vision, speech recognition, and natural language processing; more notably, the application of deep learning to 2D object detection [1–6] and semantic segmentation [7–10] techniques in computer vision has led to the rapid development of both in the past 10 years. However, with the rapid development of autonomous driving and mobile robots, only 2D object detection can no longer satisfy all the information required for environment perception. In the urban scenario of autonomous driving, a single sensor cannot accurately sense the complex and changing road traffic environment, so fusing data collected by multiple sensors is the preferred solution for current autonomous driving perception systems. LiDAR and camera are the two most commonly used sensors in current autonomous vehicles. Since the 3D point cloud generated by LiDAR has accurate depth and reflection intensity information, but due to its disadvantages such as sparse and uneven density, when facing objects with missing shape or similar shape, its representation ability is insufficient. Therefore, it often leads to missed or false detection when detecting small long-range targets with similar shape and small number of point clouds. While the 2D images generated by camera sensors are rich in semantic information such as texture and color, which can delineate targets and scenes in detail, but cause the loss of depth information by perspective projection, there is no way to guarantee the accuracy of depth estimation by either monocular [11, 12] or stereo binocular [13, 14] based algorithms, which makes it difficult to achieve high-precision 3D localization. Therefore, how to effectively combine the respective advantages of 2D images and 3D point clouds to achieve accurate and robust

3D object detection in urban road scenarios of autonomous driving is the main focus of this paper. The contributions of this paper include the following three main points.

(1) We designed a 3D object detection network architecture based on two-stage complementary fusion. The architecture uses key points as a bridge to successfully combine Point-based, Voxel-based, and Image-based methods. In this way, the complementary fusion of the geometric information collected by the lidar and the semantic information collected by the camera is realized.

(2) The innovation of this paper is proposing a Feature Fusion Model, which fuses features from voxels, original point cloud features, BEV features, and RGB image features, and conducts experimental analysis on the contribution of each part of the features to the accuracy of the network model, and the results show that, after adding RGB image features, the accuracy of point cloud object detection in 3D space is significantly improved on small targets, and objects with similar structures are also substantially improved.

(3) In order to solve the problem of overreliance on the detection performance of 2D detectors in point cloud and image fusion, this paper proposes a method for assigning the foreground point features and background point features within the cone according to the confidence score and fusing them with the point cloud features by applying parallel processing to the first and second stages to use the features provided by the images as auxiliary features.

2. Related Work

At present, most existing 3D object detection methods based on point cloud representation are mainly divided into two categories, namely, LiDAR + Image fusion method and Only LiDAR method. In the LiDAR + Image fusion method [15], it, respectively, fuses the top view and front view of the point cloud and the RGB image features with the candidate area generated in the top view, thereby achieving the data-level fusion between the point cloud and the image. However, a large amount of data processed in this method requires a lot of computing resources and communication bandwidth, and it also generates a large amount of redundant information. The 2D driving 3D method [16] is used to extract the point cloud of corresponding area to estimate the 3D bounding box, but the disadvantage of this type of method is that it relies too much on the accuracy of the 2D detection. The research work [17] uses the camera at the first stage, and it only processes the lidar point cloud at the second stage. The disadvantage of this method is that although two sensors are used, the data of the two sensors is not fused; thus it is very difficult to use this method to detect long distance and obscured objects. In Only LiDAR method, it is divided into Voxel-based method and Point-based method. Voxel-based methods usually convert the point cloud into a Bird's Eye View (BEV) [18–24] or voxelize it [25–29] at first, then use the 3D sparse convolutional neural network to extract the features. The advantage of this method is that it not only has high convolution efficiency but also can produce high-quality proposal. The disadvantage is that the loss of information in the voxelization process reduces the accuracy of location when locating the target frame. The Point-based method [30–35] directly extracts features from the raw point cloud to detect 3D objects. Although this method can get a larger receptive field, the calculation cost has also increased. Thereby, this paper combines the advantages of Voxel-based, Point-based, and Image-based methods to propose a 3D object detection network architecture based on two-stage complementary fusion. This architecture is based on 3D sparse convolutional neural network combined with mature 2D object detector, and the key points are utilized as a bridge, so that a series of precise and guided fusion methods can be designed to accurately predict the categories, 3D positions, and other information of the objects in the surrounding environment. Figure 1 shows some classic 3D object detection algorithms based on point cloud representation.

3. System Design

In order to fully combine the target object's geometric information collected by the lidar and the semantic information collected by the camera, this paper proposed a 3D object detection network structure based on two-stage complementary fusion. This structure uses key points as a bridge to combine Point-based, Voxel-based, and Image-based methods, which fully integrates and utilizes the geometric information of the point cloud and the semantic information of the image.

3.1. Fusion of Voxel-Based Feature. In the first stage of feature fusion, the Voxel-based method is used to conduct the irregular raw point cloud data voxelization. Then, we use the 3D sparse convolutional neural network [36, 37] as the backbone network for feature extraction. The output feature from the feature extraction network uses a two-layer Multilayer Perceptron (MPL) network to ascend the dimension of voxel features; then we convert its height information to channel feature to generate BEV. Secondly, 2048 points are sampled as key points from the raw point cloud data with the Farthest Point Sampling (FPS) method, and then the voxel features are fused to the key points. In the process of fusing voxel features to key points, we propose a Feature Fusion Module, which uses the set abstraction operation method [32] to aggregate voxel-wise feature volumes, at which time the voxels around the key points are regular voxels with multiscale semantic features encoded by a multilayer 3D voxel CNN. The set of all voxel feature vectors in the kth layer of the 3D voxel CNN is denoted by $\mathscr{F}^{(l_k)} = \left\{f_1^{(l_k)}, \ldots, f_{N_k}^{(l_k)}\right\}$, and $\mathscr{V}^{(l_k)} = \left\{v_1^{(l_k)}, \ldots, v_{N_k}^{(l_k)}\right\}$ denotes the set of 3D coordinates of all voxels computed from the voxel index of the kth layer and the actual voxel size, where N_k is the number of nonempty voxels in the kth layer. For each key point p_i, we first determine the number of

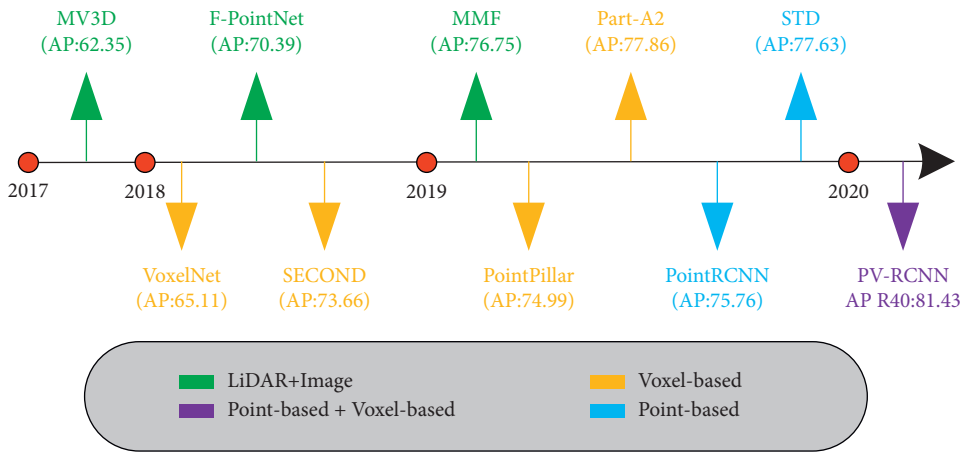

Figure 1: The development process of 3D object detection algorithm based on point cloud representation.

nonempty voxels within the radius r_k in the kth layer of the 3D voxel CNN to retrieve the set of feature vectors of valid voxels in the neighborhood of the key point r_k:

$$S_i^{(l_k)} = \left\{ \left[f_j^{(l_k)}; v_j^{(l_k)} - p_i \right]^T \middle| \begin{array}{l} \| v_j^{(l_k)} - p_i \|^2 < r_k \\ \forall v_j^{(l_k)} \in \mathscr{V}^{(l_k)} \\ \forall f_j^{(l_k)} \in \mathscr{F}^{(l_k)} \end{array} \right\}, \quad (1)$$

where $v_j^{(l_k)} - p_i$ is the local relative coordinate of the voxel and the key point, which represents the relative position of the voxel feature $f_j^{(l_k)}$. Then, the voxel features within $S_i^{(l_k)}$ are transformed using PointNetblock [31] to generate the features of key point p_i as

$$f_i^{(pv_k)} = \max\left\{ G\left(\mathscr{M}\left(S_i^{(l_k)} \right) \right) \right\}, \quad (2)$$

where $\mathscr{M}(\cdot)$ denotes the random sampling of up to $T_k = 16$ voxels from the set $S_i^{(l_k)}$ to save computational cost, and $G(\cdot)$ denotes the use of a multilayer perceptual network to encode the features and relative positions of each voxel. We perform the above strategy on different convolutional layers of the 3D voxel CNN to stitch the features aggregated in different convolutional layers to generate multiscale semantic features of key point p_i. Also, to compensate for the information loss caused by the voxelization process, we use a Point-based approach to extend the key point features by fusing the raw point cloud features $f_i^{(\text{raw})}$ and the BEV features $f_i^{(BEV)}$ to the key points, and the process of fusing the raw point cloud features to the key points is given in Figure 2.

3.2. Fusion of Image-Based and Point-Based Feature. In the second stage of network fusion, RGB images are used as input data, and yolov5 [38] is employed as a 2D object detector to classify and locate the target objects in the image data, so as to obtain the class of target objects and the location information of 2D bounding box. Through the known camera projection matrix, we project each 2D bounding box into the 3D point cloud data to form a frustum candidate area. Then, for the points in the frustum, we use the Point-based method to fuse the features into the key points, so that the effective fusion of Voxel-based, Point-based, and Image-based methods can be realized. In this way, the search range of the point cloud can be greatly narrowed down, and the operating efficiency of the network can be improved. 3D point cloud object detection by 2D-driven 3D is presented in [16]; the final detection result of the whole network tends to be overly dependent on the detection accuracy of the 2D detector. To solve this problem, this paper proposes a method to assign the foreground point features and background point features within the frustum according to the confidence score value, by using parallel processing for the first and second stages to fuse the features provided by the image as auxiliary features with the point cloud features. The 2D detector parameters are kept fixed during the entire training period of the network model since the 2D detector network model parameters are pretrained.

The specific implementation of the method of assigning the foreground point features and background point features in the frustum according to the confidence is that the point cloud in the frustum is extracted using the Point-based method [31], where the point cloud in the frustum has the point cloud data of the target object framed by the 2D bounding box, which we call foreground points, and other points that are not the target object, which we call background points. Therefore, the foreground point features in the frustum should contribute more to the network model, and the background point features should contribute less to the network model. The weight values of the internal and external point cloud features of the bounding box are assigned by the confidence provided by the 2D object detector.

$$f_i^{(RGB)} = C \cdot f_i^{(fg)} + (1 - C) f_i^{(bg)}, \quad (3)$$

where C is the confidence of the target object provided by the 2D detector, f^{fg} is the front point feature within the frustum, f^{bg} is the background point feature within the frustum, and the final key point feature is

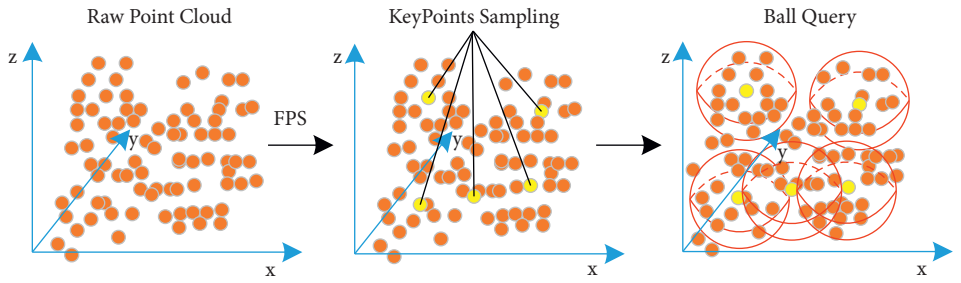

Figure 2: Process diagram of fusion of raw point cloud data to key points.

$$f_i^{(p)} = f_i^{(pv)} + f_i^{(\text{raw})} + f_i^{(BEV)} + f_i^{(RGB)}. \tag{4}$$

The overall structure of the network model is shown in Figure 3.

Generally, we compress the height of voxels from top to bottom directly in the process of generating BV, and the method of forced compression tends to miss a lot of features. In order to solve this problem, height features information of voxels will be converted to channel feature. The process of generating BEV based on the voxel method is shown in Figure 4. For instance, we assume that the voxel feature size output by the feature extraction network is $3 \times 3 \times 3\times$N, where $3 \times 3 \times 3$ is the length, width, and height of the voxel and N is number of feature channels. Then, the size of converted feature map is $3 \times 3 \times 1 \times 3$N, which is utilized to generate BEV.

4. Projection Transformation of the Point Cloud

The 2D object detector detects the 2D bounding box of the target object from the image and then takes the camera as the origin and extends along the direction of the bounding box to form a frustum. In order to make the generated frustum area more accurate, the first thing we have to do is to use the projection transformation method to project the point cloud in 3D space onto the RGB image, so as to obtain the intersection area between the point cloud and the image plane.

4.1. Converting Image Plane Coordinate System to Pixel Coordinate System. When a point on the target object in 3D place is projected onto the image plane, it does not directly correspond to the point observed in the digital image. Therefore, we need to project each point from the image plane to the digital image.

As shown in Figure 5, o is the center of camera and the coordinate system is i, j, k, where k points to the image plane. Then, point c intersecting with the k axis is recorded as the main point, which represents the center of the image plane. Right-hand plane coordinates which take the principal point as the origin $c - xy$ are used as the image plane coordinate system. Therefore, the first step to take after projecting the point $\vec{P}$ in space to the image plane is to subtract the principal point coordinates so that the discrete image has its own coordinate system.

For converting metric coordinates (m) to pixel coordinates, this paper uses parameters k and l provided by calibration procedure, which can be utilized to convert meter to pixel and easily inherit it into the equation as is shown in

$$\vec{P} \longrightarrow \vec{P}'(x, y, z)^T \longrightarrow \left(f \cdot k \cdot \frac{x}{z} + c_x, f \cdot l \cdot \frac{y}{z} + c_y\right), \tag{5}$$

where $f \cdot k$ is called α and $f \cdot l$ is called β in the conversion matrix. Therefore, the conversion equation for converting points in 3D space to 2D image pixels is

$$\vec{P} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \longrightarrow \vec{P}' = \begin{bmatrix} \alpha x/z + c_x \\ \beta y/z + c_y \end{bmatrix}. \tag{6}$$

For the points in the point cloud, we need to move each of them from the position of the lidar to the position of the camera through rotation and translation operations. As shown in formula above, the projection equation is related to the division of z, which makes the conversion nonlinear so that equation is difficult to be transformed. Therefore, we convert the original Euclidean coordinate system to a homogeneous coordinate system so that the projection transformation is converted from nonlinear to linear and the transformation equation can be transformed into matrix-vector multiplication. Equation (7) is the conversion formula from Euclidean coordinates to homogeneous coordinates:

$$(x, y) \longrightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \cdot (x, y, z) \longrightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{7}$$

With the help of homogeneous coordinate system, the projection equation expressed in matrix vector is

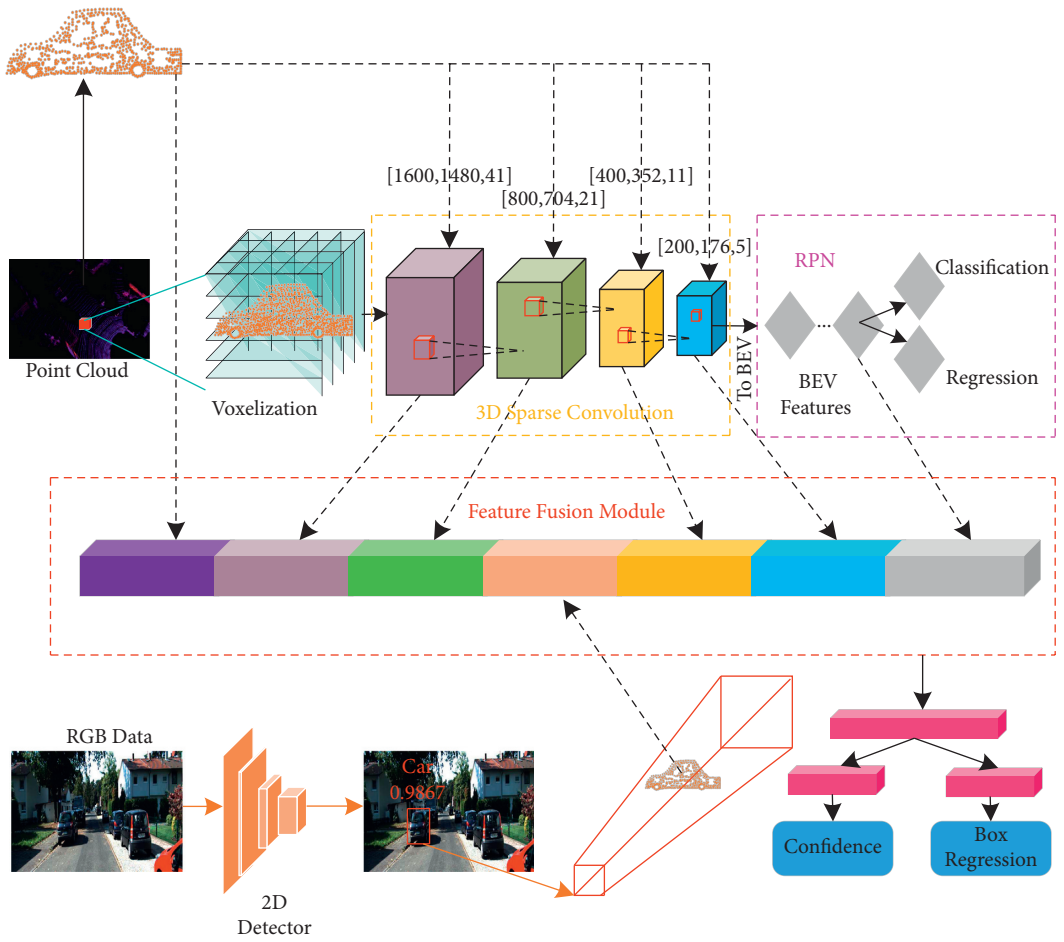

Figure 3: Network structure diagram of point cloud object detection.

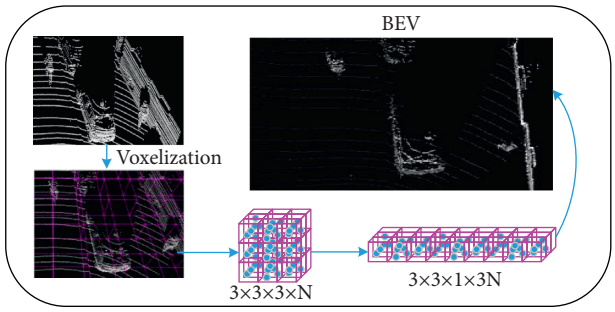

Figure 4: Process diagram of BEV generation.

$$\vec{P}_h' = \begin{bmatrix} \alpha \cdot x + c_x \cdot z \\ \beta \cdot y + c_y \cdot z \\ z \end{bmatrix} = \begin{bmatrix} \alpha & 0 & c_x & 0 \\ 0 & \beta & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{8}$$

We realized the mapping from the point $\vec{P}$ in 3D space in camera coordinate system to the point $\vec{P}'$ in 2D pixel plane through (8).

4.2. Point Cloud Projection on the Image Plane. To project the points measured in lidar coordinate system to camera coordinate system, extra conversion needs to be added to the operation of projection so that we can associate points in the vehicle coordinate system to the camera coordinate system. Generally, projection operation can be divided into three parts: translation, scaling, and rotation.

4.2.1. Translation. As is shown in Figure 6, the $\vec{P}$ point is linearly translated to $\vec{P}'$ by adding translation vector t.

We can get the translation formula as follows through Figure 6:

$$\vec{P}' = \vec{P} + t = \begin{bmatrix} x + t_x \\ y + t_y \end{bmatrix}. \tag{9}$$

In the homogeneous coordinate system, the translation (10) is obtained:

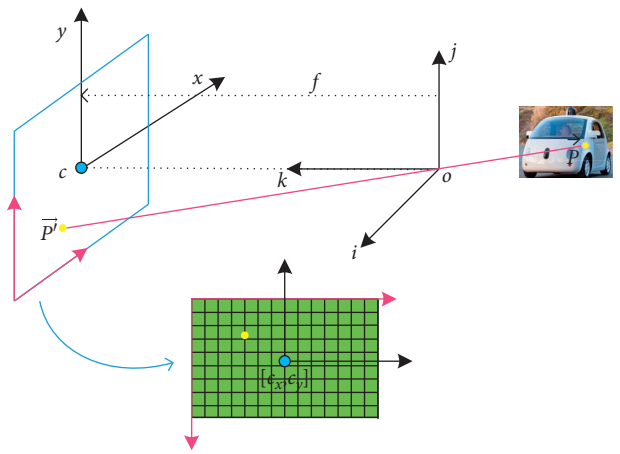

Figure 5: Process diagram of point in space projected onto image.

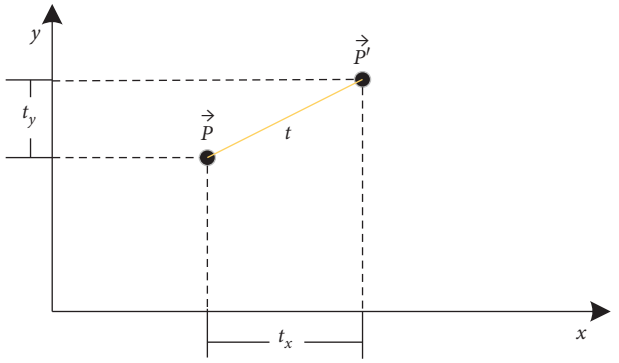

Figure 6: Point translation coordinate map.

$$\vec{P}_h' = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} I & t \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{10}$$

where I is a unit vector which is size of 2×2 and t is the coordinate increment t_x, t_y.

4.2.2. Scaling. We multiply the original vector by a scale vector to achieve scaling transformation. And the scaling formula is as follows:

$$\vec{P}_h' \longrightarrow \begin{bmatrix} s_x x \\ s_y y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = s \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{11}$$

4.2.3. Rotation. Figure 7 shows the process diagram of the vector $\vec{P}(x, y)$ after rotating θ counterclockwise to reach the vector $\vec{P}'(x', y')$.

We can get the following rotation matrix from Figure 7:

$$\begin{aligned} x' &= \cos\theta x - \sin\theta y, \\ y' &= \cos\theta y + \sin\theta x, \\ \vec{P}' &= \begin{bmatrix} x' \\ y' \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \\ &= R \cdot \vec{P}, \end{aligned} \tag{12}$$

where R is called the rotation matrix. In 3D space, the rotation of point P is realized around the three axes of x, y, and z. Therefore, the rotation matrix in 3D space is

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix},$$

$$R_y = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix}, \tag{13}$$

$$R_z = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

We can obtain projection matrix that projects 3D point cloud onto 2D image plane by cascading translation matrix and rotation matrix, which is shown in

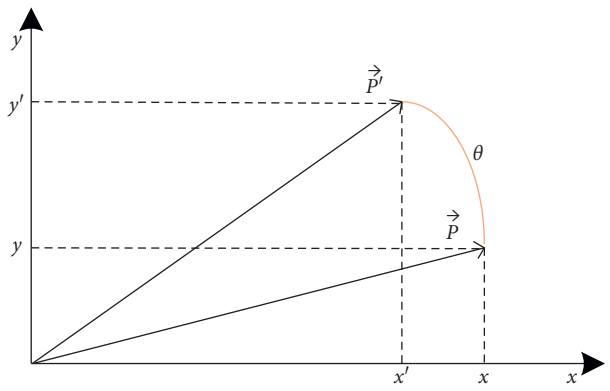

Figure 7: Point rotation coordinate map.

$$\vec{P}' = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} t_x & 0 & 0 \\ 0 & t_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & \frac{s}{t_x} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times [I|t] \times \begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix} \cdot \vec{P}. \tag{14}$$

Using (14) we can then project the point cloud data in 3D space onto a 2D image, and the simulation diagram of the projection result is shown in Figure 8.

By fusing the raw data from both sensors, the point cloud can correspond to the image, so that the object detected in 2D can find the corresponding point cloud in the frustum area formed in 3D.

5. Experiment

5.1. Introduction to Training Dataset. The dataset used in this article is the KITTI [41] dataset, which was jointly developed by the Karlsruhe Institute of Technology in Germany and the Toyota American Institute of Technology. It is a computer vision algorithm evaluation dataset in autonomous driving scenes which is widely used all over the world. KITTI contains real image data collected from scenes ranged from urban areas, villages to highways. Each image can contain up to 15 cars and 30 pedestrians with various degrees of masking and truncation. The entire dataset consists of 389 pairs of stereo images, optical flow diagrams, 39.2 km visual ranging sequence, and more than 200k 3D annotated object images, which are sampled and synchronized at a frequency of 10 Hz. The original dataset is classified as 'Road,' 'City,' 'Residential,' 'Campus,' and 'Person.' For 3D object detection, label is subdivided into car, van, truck, pedestrian, pedestrian (sitting), cyclist, tram, and misc. In this section, we visualized part of the image data and point cloud data of the 2D and 3D labeled objects in the KITTI dataset, and the visualization results are shown in Figure 9.

5.2. Training Parameter Settings. When training the network model with KITTI dataset, the position of the lidar on the point cloud collection vehicle is used as the origin of the coordinates. The valid data range of point cloud length is [0,70.4] *m*, the valid data range of point cloud width is [-40,40] *m*, and the valid data range of point cloud height is [-3,1] *m*. After point cloud voxelization, the length, width, and height of each voxel are, respectively, 0.05 m, 0.05 m, and 0.1 m. The batch-size used during training is 2, and the learning rate is 0.01.

In this section, we trained and verified our network model with the KITTI dataset and compared it with the current state-of-the-art 3D point cloud object detection algorithms such as F-PointNet, AVOD-FPN, ContFuse, and MV3D. The software and hardware configuration and version models used in the experiment are shown in Table 1.

5.3. Algorithm Performance Evaluation. In the field of object detection, the quality of an algorithm is mainly evaluated in two ways: qualitative and quantitative evaluation. Qualitative evaluation is mainly based on observation, which is a subjective evaluation mechanism. Quantitative evaluation uses mathematical statistics to quantify algorithm performance based on specific evaluation indexes. Compared with qualitative evaluation methods, quantitative evaluation can compare the differences between different algorithms more scientifically, fairly, and accurately.

As a very important evaluation index in the object detection algorithm, recall rate mainly reflects the algorithm's ability to cover positive examples. Its mathematical expression is

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{15}$$

In the formula, TP (True Positive) means "the number of positive cases that are correctly detected as positive cases," and FN (False Negative) means "the number of positive cases that are falsely detected as negative cases." Precision reflects the accuracy of the algorithm in predicting positive examples. The mathematical expression is

$$\text{precision} = \frac{TP}{TP + FP}. \tag{16}$$

In the formula, FP (False Positive) means "the number of negative cases that are falsely detected as positive cases."

The two evaluation indicators, recall rate and accuracy, show the performance of the algorithm from two different perspectives, but they are a pair of contradictory measures. Both affect each other and show a negative correlation trend. In order to balance them, we introduce the Precision-Recall (P-R) curve.

As shown in Figure 10, the P-R curve represents the corresponding recall rate (horizontal axis) and accuracy (vertical axis) when different IOU and confidence thresholds are picked. Generally, the more the P-R curve goes to the upper right corner, the better the result is. Take Figure 10 as an example; curve A is better than curve B.

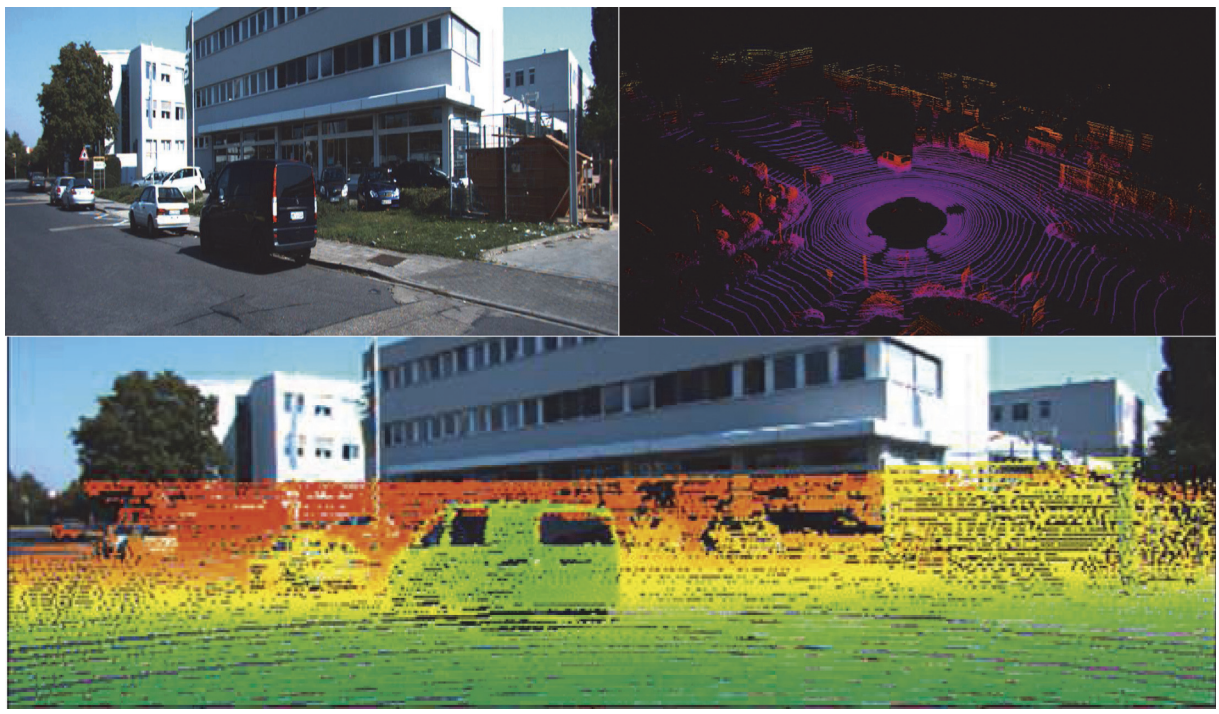

Figure 8: A simulation diagram of the point cloud projected onto the image.

Figure 9: KITTI visualization data.

The average accuracy is derived from the P-R curve. For a continuous P-R curve, the calculation of AP is shown in (17). In the formula, p represents precision, r represents recall, and p is a function which takes r as a parameter. The calculation of the discrete P-R curve AP is shown in (18), where N represents the number of samples, $p(k)$ represents the accuracy when k samples are detected, and $\cdot r(k)$ represents the change in recall rate when the number of detected samples is changed from $k-1$ to k. Mean average precision (mAP) is to average AP of multiple categories and measure

Table 1: Experimental environment used for training and testing.

Software and hardware configuration	Version of the model
Processor	Intel Xeon(R) W-2135 CPU @3.70 GHz
Graphics card	GeForce RTX 2080 Ti
RAM	32G
Operating system	Ubuntu 16.04
Frame	Pytorch 1.3.1、TensorFlow
Programming language	*Python* 3.7、*C++*
CUDA/CUDNN	CUDA 10.1/CUDNN v7.6

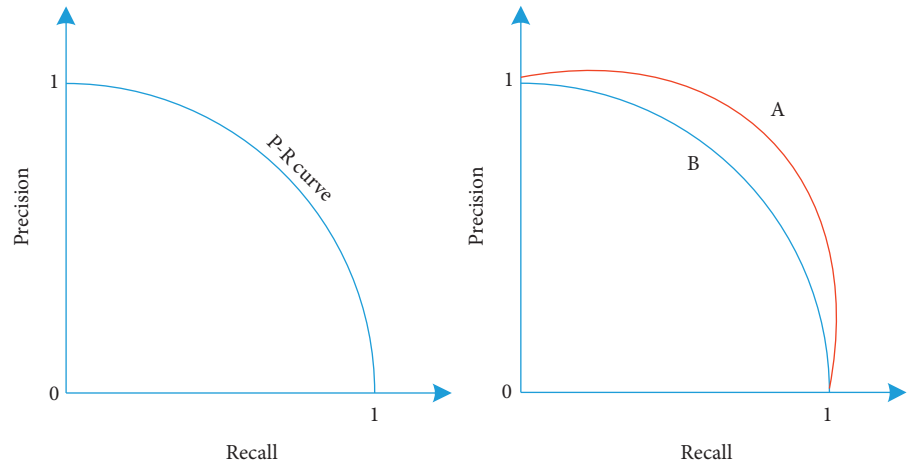

Figure 10: P-R curve diagram.

Table 2: Comparison of algorithm performance at a moderate level.

Class	Method	Modality	3DmAP
Car	F-PointNet [16]	RGB + LiDAR	70.92%
	AVOD-FPN [18]		74.44%
	ContFuse [20]		73.25%
	MV3D [15]		62.68%
	Ours		**75.02%**
Pedestrian	F-PointNet [16]	RGB + LiDAR	61.32%
	AVOD-FPN [18]		58.8%
	ContFuse [20]		—
	MV3D [15]		—
	Ours		**62.51%**
Cyclist	F-PointNet [16]	RGB + LiDAR	56.49%
	AVOD-FPN [18]		49.7%
	ContFuse [20]		—
	MV3D [15]		—
	Ours		**57.26%**

the accuracy of the algorithm in all categories. The calculation formula is shown in (19), where C is the number of categories.

$$AP = \int_0^1 p(r)d(r), \tag{17}$$

$$AP = \sum_{k=1}^{N} p(k) \cdot r(k), \tag{18}$$

$$mAP = \frac{\left(\sum_{i=1}^{C} AP_i\right)}{C}. \tag{19}$$

In this paper, AP and mAP are used as evaluation indexes to measure the performance of the model. In the following tables, the test results of all algorithms participating in the comparison are the results published by the authors of the algorithm in their articles, and those undisclosed or unavailable experimental results are indicated

Table 3: Comparison of 3D positioning performance between our algorithm and the state-of-the-art algorithm.

Method	Modality	Car			Pedestrian			Cyclist		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
PointPillars [22]	Only LiDAR	90.07	86.56	**82.81**	57.60	48.64	45.78	79.90	62.73	55.58
PointRCNN [33]		92.13	87.39	82.72	54.77	48.13	42.84	82.56	67.24	60.28
Part-A^2 [29]		94.07	85.35	75.88	59.04	49.81	45.92	83.43	68.73	61.85
PV-RCNN [39]		90.25	81.43	76.82	52.17	43.29	40.29	78.60	63.71	57.65
SE-SSD [40]		91.49	82.54	77.15	-	-	-	-	-	-
AVOD-FPN [18]	RGB + LiDAR	90.99	84.82	79.62	58.49	50.32	46.98	69.39	57.12	51.09
F-PointNet [16]		91.17	84.67	74.77	70.00	61.32	53.59	77.26	61.37	53.78
ContFuse [20]		94.07	85.35	75.88	—	—	—	—	—	—
MV3D [15]		86.62	78.93	69.80	—	—	—	—	—	—
Ours		**95.01**	**88.32**	80.57	**79.67**	**66.89**	**56.36**	**90.12**	**72.41**	**63.21**

Table 4: The detection result diagram of the network model.

Feature	Pedestrian			Runtime (s)
	Easy	Moderate	Hard	
Voxel feature only	70.87%	58.50%	50.13%	0.04
+Raw point cloud feature	+1.91%	+1.37%	+0.22%	0.02
+BEV feature	+2.82%	**+3.73%**	+2.75%	0.00
+RGB feature	**+4.07%**	+3.29%	**+3.26%**	0.04
Full model	**79.67%**	**66.89%**	**56.36%**	**0.10**

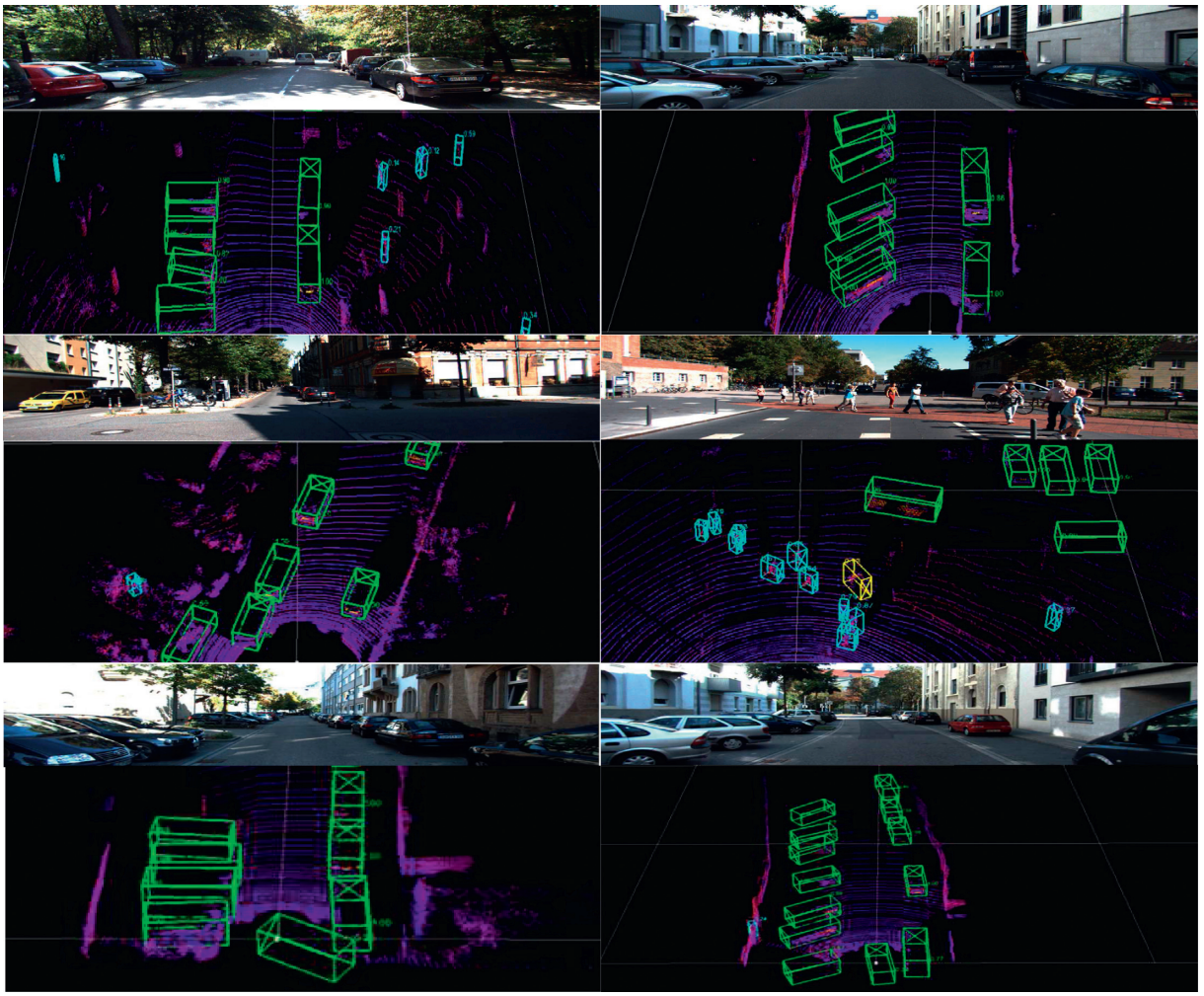

Figure 11: The detection result diagram of the network model.

by "-" or not included in the comparison. The optimal value of each test is shown in bold. The results of comparing our work comparing with current state-of-the-art fusion algorithms are shown in Tables 2 and 3.

Table 2 shows that, among all the fusion methods of LiDAR and RGB images, our method has a significant improvement in mAP in the detection of pedestrians and cyclists. The main reason is that there are less data for pedestrians and cyclists in the pure point cloud data; thus it is difficult to detect. By adding the semantic information of the image, we can determine the boundary box of the target object in the image and then project it into the point cloud, which ensures that the projected frustum definitely contains a certain type of object. In this way, the detection accuracy of pedestrians and cyclists with less point cloud data can be increased.

In Table 3, we show the comparison results of the 3D positioning accuracy AP(BEV)% comparing with some state-of-the-art algorithms.

It can be found from Table 3 that the location results of our algorithm are comparable to the state-of-the-art algorithms, and the performance is more obvious in the easy category of pedestrians and cyclists. The result proves that when our algorithm generates BEV, using redundant information for compression helps information loss.

To verify whether the semantic information of RGB images we added improves the network model, we, respectively, trained the network model only extracts voxel features, the network model with raw point cloud features added, the network model with BEV features added, and the network model with image information added, and the contribution of adding different feature information to the network model is shown in Table 4.

Through Table 4 we can see that the detection accuracy of the network model is improved due to the inclusion of image semantic features. Since the network uses parallel processing, the detection accuracy of the network model depends mainly on the voxel features of the point cloud, and the RGB image features are only used as auxiliary features to improve the detection accuracy of the algorithm, so the overall performance of the algorithm does not depend too much on the detection accuracy of the 2D detector. Finally, at the end of the paper, we give a set of final results of our algorithm as shown in Figure 11.

6. Conclusion

In order to integrate point cloud data and image data better, this article proposed a 3D object detection network architecture based on two-stage complementary fusion. The architecture uses key points as the connection bridge and successfully combines Point-based, Voxel-based, and Image-based methods. Detecting through 3D point cloud targets driven by 2D detectors further improved the detection result of pedestrians and cyclists, which have a small amount of point cloud data. The disadvantage is that there is redundant information between extracting features using Point-based methods for point cloud data within a frustum and using Point-based methods for the raw point cloud, which increases the running time of the network model. In the future we plan to introduce point cloud feature alignment methods, so that we can apply feature similarity alignment to the global features from the original point cloud and the local features from the point cloud inside the frustum. For the two features with high similarity, we discard the global features in the original point cloud and keep the point cloud features in the frustum and the target class and 2D bounding box information provided by the 2D detector, so that the retained features can participate in the subsequent calculation. In this way, we can reduce the redundancy of information and improve the efficiency of the network.

Data Availability

All data and program included in this study are available upon request by contact with the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (61971007 and 61571013).

References

[1] Z. Dai, B. Cai, Y. Lin, and J. Chen, "UP-DETR: unsupervised pre-training for object detection with transformers," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[2] K. J. Joseph, S. Khan, F. S. Khan, and V. N. Balasubramanian, "Towards open world object detection," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA, June 2021.

[3] Q. Chen, Y. Wang, T. Yang, X. Zhang, J. Cheng, and J. Sun, "You only look one-level feature," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA, June 2021.

[4] C. Yang, Z. Wu, B. Zhou, and S. Lin, "Instance localization for self-supervised detection pretraining," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA, June 2021.

[5] T. Liang, Y. Wang, Z. Tang, G. Hu, and H. O. P. A. N. A. S. Ling, "One-shot path aggregation network architecture search for object detection," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA, June 2021.

[6] V. S. Vibashan, P. Oza, V. A. Sindagi, V. Gupta, and V. M. Patel, "MeGA-CAD: memory guided attention for category-aware unsupervised domain adaptive object detection," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA, June 2021.

[7] M. Fan, S. Lai, J. Huang, and X. Wei, "Rethinking BiSeNet for real-time semantic segmentation," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA, June 2021.

[8] B. Sun, B. Li, S. Cai, Y Yuan, and C Zhang, "FSCE: few-shot object detection via contrastive proposal encoding," in

Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, June 2021.

[9] S. Zheng, J. Lu, and H. Zhao, X Zhu, Z Luo, Y Wang et al., Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA, June 2020.

[10] Y. Li, H. Zhao, X. Qi et al., "Fully convolutional networks for panoptic segmentation," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[11] X. Chen, D. K. Ku, and Z. Zhang, "Monocular 3d object detection for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2147–2156, Las Vegas, NV, USA, June 2016.

[12] T. He and S. Soatto, "Mono3D++: monocular 3D vehicle detection with two-scale 3D hypotheses and task priors," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 8409–8416, Honolulu, HI, USA, 2019.

[13] X. Chen, D. K. Ku, and Y. Zhu, "3d object proposals using stereo imagery for accurate object class detection[J]," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 5, pp. 1259–1272, 2017.

[14] H. Hu, T. T. Zhao, and Q. Wang, "R-CNN based 3D object detection for autonomous driving," in *Proceedings of the International Conference of Transportation Professionals*, Xi'an, China, August 2020.

[15] X. Chen, H. M. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, July 2017.

[16] R. Qi, W. Liu, C. Wu, H. Su, and L. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, June 2018.

[17] X. X. Du, M. H. Ang, S. Karaman, and D. Rus, "A General Pipeline for 3d Detection of Vehicles," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, Australia, May 2018.

[18] J. Ku, M. Mozifian, J. Lee, H. Harakeh, and S. Waslander, "Joint 3d Proposal Generation and Object Detection from View Aggregation," in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, October 2018.

[19] B. Yang, W. J. Luo, and R. Urtasun, "PIXOR: real-time 3d object detection from point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pp. 7652–7660, Salt Lake City, UT, USA, June 2018.

[20] M. Liang, L. Yang, S. H. Wang, and R. Urtasun, "Deep Continuous Fusion for Multi-Sensor 3d Object Detection," in *Proceedings of the European Conference on Computer Vision*, Munich, Germany, September 2018.

[21] B. Yang, M. Liang, and R. H. Urtasun, "HDNET: exploiting hd maps for 3d object detection," in *Proceedings of the 2nd Conference on Robot Learning (CoRL)*, Zürich, Switzerland, October 2018.

[22] A. H. Lang, S. Vora, H. Caesar, L. B. Zhang, J. Yang, and O. P. Beijbom, "PointPillars: fast Encoders for Object Detection from point Clouds," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, June 2019.

[23] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task Multi-Sensor Fusion for 3d Object Detection," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, June 2019.

[24] H. W. Yi, S. S. Shi, M. Y. Ding et al., "Segvoxelnet: exploring semantic context and depthaware features for 3d vehicle detection from point cloud," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Paris, France, August 2020.

[25] S. R. Song and J. X. Xiao, "Deep sliding shapes for amodal 3d object detection in rgb-d images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 808–816, Las Vegas, NV, USA, June 2016.

[26] Y. Zhou and O. Tuzel, "Voxelnet: end-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pp. 4490–4499, Salt Lake City, UT, USA, June 2018.

[27] Y. Yan, Y. Mao, and B. Li, "Second: sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[28] Y. L. Chen, S. Liu, X. Y. Shen, and J. Y. Jia, "Fast point r-cnn," in *Proceedings of the IEEE international conference on computer vision (ICCV)*, Seoul, Korea (South), November 2019.

[29] S. S. Shi, Z. Wang, J. P. Shi, X. G. Wang, and H. S. Li, "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*, vol. 43, 2020.

[30] J. Q. Ngiam, B. Caine, W. Han et al., "Starnet: targeted computation for object detection in point clouds," https://arxiv.org/abs/1908.11069.

[31] R. Qi, H. Su, K. C. Mo, and L. P. Guibas, "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, Honolulu, HI, USA, June 2017.

[32] R. Qi, H. Su, K. C. Mo, and L. P. Guibas, "PointNet plus plus: deep hierarchical feature learning on point sets in a metric space," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 5099–5108, Long Beach, CA, USA, December 2017.

[33] S. S. Shi, X. G. Wang, and H. S. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pp. 770–779, Long Beach, CA, USA, June 2019.

[34] Z. X. Wang and K. Jia, "Frustum Convnet: Sliding Frustums to Aggregate Local point-wise Features for Amodal 3d Object Detection," in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, November 2019.

[35] Z. T. Yang, Y. N. Sun, S. Liu, X. Y. Shen, and J. Y. Jia, "STD: Sparse-To-Dense 3d Object Detector for point Cloud," in *Proceedings of the International Conference on Computer Vision (ICCV)*, Seoul, Korea (South), November 2019.

[36] B. Graham, M. Engelcke, and V. Laurens, "3D semantic segmentation with submanifold sparse convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, Salt Lake City, UT, USA, June 2018.

[37] B. Graham and L Maaten, "Submanifold Sparse Convolutional Networks," in *Proceedings of the CoRR*, June 2017.

[38] G. Jocher, yolov5, https://github.com/ultralytics/yolov5, 2021.

[39] S. S. Shi, C. X. Guo, L. Jiang et al., "PV-RCNN: point-voxel feature set abstraction for 3D object detection," in *Proceedings of the CVPR*, pp. 10529–10538, 2020.

[40] W. Zheng, W. Tang, and J. L. Se-Ssd, "SE-SSD: self-Ensembling Single-Stage Object Detector From Point Cloud et al," in *Proceedings of the CVPR*, 2021.

[41] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition*, Providence, RI, USA, June 2012.