Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 5585212, 13 pages
https://doi.org/10.1155/2021/5585212

Research Article

Combined Auxiliary Networks and Bird's Eye View Method for Real-Time Multicategory Object Recognition

Zhangpeng Gong,[1] **Luansu Wei**,[2] **Guoye Wang**,[1] **Dongxin Xu**,[1] **and Chang Ge**[1]

[1]*College of Engineering, China Agriculture University, Beijing 100083, China*
[2]*Jiangsu Key Laboratory of Construction Materials, School of Materials Science and Engineering, Southeast University, Nanjing 211189, China*

Correspondence should be addressed to Guoye Wang; wgy1615@126.com

Received 13 January 2021; Revised 16 February 2021; Accepted 28 February 2021; Published 9 March 2021

Academic Editor: Xiao Ling Wang

Copyright © 2021 Zhangpeng Gong et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Object recognition based on LIDAR data is crucial in automotive driving and is the subject of extensive research. However, the lack of accuracy and stability in complex environments obstructs the practical application of real-time recognition algorithms. In this study, we proposed a new real-time network for multicategory object recognition. The manually extracted bird's eye view (BEV) features were adopted to replace the resource-consuming 3D convolutional operation. Besides the subject network, we designed two auxiliary networks to help the network learn the pointwise features and boxwise features, aiming to improve the category and bounding boxes' accuracy. The KITTI dataset was adopted to train and validate the proposed network. Experimental results showed that, for hard mode, the total average precision (AP) of the category reached 97.4%. For an intersection over a union threshold of 0.5 and 0.7, the total AP of regression reached 93.2% and 85.5%; especially, the AP of car's regression reached 95.7% and 92.2%. The proposed network also showed consistent performance in the Apollo dataset with a processing duration of 37 ms. The proposed network exhibits stable and robust object recognition performance in complex environments (multiobject, unordered objects, and multicategory). And it shows sensitivity to occlusion of the LIDAR system and insensitivity to close large objects. The proposed multifunction method simultaneously achieves real-time operation, high accuracy, and stable performance, indicating its great potential value in practical application.

1. Introduction

Autonomous driving is a futuristic technology that will transform mobility industries and ease the burden of driving. Autonomous driving is currently supported by relatively mature planning, decision-making, and algorithm implementation but is mainly hindered by its poor perception. As an efficient and precise remote sensing technique, the LIDAR systems have been widely applied in real-time intelligent systems, such as self-driving vehicles [1, 2]. The data acquired from LIDAR are point clouds, which is a set of points containing coordinates and other feature-related information, such as reflectivity. Detecting objects accurately within a point cloud is crucial and has been a widespread research subject. However, the key challenge is that the raw point cloud data are irregular, unstructured, and unordered. Consequently, specific processing methods that require data with a regular form are not suitable for direct application.

The convolution operation is an efficient approach for extracting deep features [3–7], and it requires a regular grid as the input, which a point cloud does not satisfy. Therefore, the first step is to transform the unstructured point cloud into a regular style. The structured processed data can be graphical [8–12], ordered points [13–17], or voxels [18–22]. In graph-based methods, nodes represent points, and edges represent the relationships between points. The abstract expression is obscure. Although point-based methods can achieve better performance by taking the raw point clouds as their input and predicting bounding boxes based on each point, in general, their inference time cannot meet the demands of a real-time system. Therefore,

they are restricted primarily to offline analysis. Voxels are popular because they have a clear physical structure similar to images. VoxelNet [20] is an example of a classic voxelization method that performs impressively in 3D object recognition tasks. Its strong performance relies heavily on several 3D convolution operations, resulting in a time- and memory-consuming process. To avoid using 3D convolution operations, a structure that replaces 3D voxels with pillars, thereby erasing the vertical dimension, has been reported [21]. This method was also called the bird's eye view method, which led to an improved processing speed, although the performance was unstable due to the lack of vertical information. Due to the lack of color information, the unstable performance is more serious than the image recognition task [22]. Alternatively, it should be a compromise approach to keep the necessary information concisely by using the maximum height, the density of the point set, and the reflectivity of the highest point to express the pillar feature [23].

To achieve higher precision of the bounding boxes, RBG images are fused with LIDAR data [24], thus obtaining a richer expression of the environment. The introduction of camera data means that this method is based on the trigger consistency of two kinds of data [25, 26] and the calibration accuracy of the camera and the LIDAR coordinate system, which may cause robustness problems in practical applications. Inspired by the better performance of point-based methods, an alternative method involves aggregating the voxels into a small number of key points [27], thus combining the advantages of both voxel- and point-based methods. In addition, this study adopted the farthest point sampling (FPS) to sample key points. However, FPS is extremely time-consuming, specifically for a large-scale scene, and the sampling time is not discussed in [27]. Therefore, finding an optimal balance between performance and processing time is still a challenge.

Most researchers use only a single category of data when training networks and assign independent evaluation indexes for the recognition effect of single categories. This method excludes the interference of other types of categories in the result. Furthermore, it causes deviation from the requirement that results in the recognition of multiple categories through one forward propagation in the actual application, which cannot explain the actual effect of the application.

This present study focused on developing a LIDAR-based 3D object recognition method for road scenes. Considering the significant effect of image recognition, we expect to take the advanced image recognition methods to the point cloud recognition task. Hence, the proposed method is a voxel-based recognition method that can simultaneously predict multiple object categories. We evaluated the method based on the 3D localization and bounding box precision, object recognition accuracy, and processing time. Unlike most present methods that heavily rely on 3D convolutional operations, we considered that the bird's eye view (BEV) based method has not yet exhausted its performance potential. Thus, we improved the head network and designed an additional auxiliary network to improve the prediction accuracy. The network was trained and evaluated by the KITTI dataset and its benchmark. The results verify that the new part is beneficial to the network.

The rest of this paper is structured as follows: Section 2 presents the proposed network architecture; Section 3 outlines the implementation of the proposed network and presents the results; Section 4 discusses the specific recognition effects that are not obvious in the evaluation indicators; and Section 5 presents our conclusions.

2. Methods

The proposed network is divided into preprocessing, backbone network, neck network, head network, and auxiliary networks:

(1) The preprocessing stage transforms the unordered point cloud into ordered data.

(2) The backbone and neck networks are used to extract scene features.

(3) The head network transforms the scene features into predicted outputs.

(4) The auxiliary network is set up to help the subject network learn pointwise and boxwise features. It does not participate in the prediction process, so it will not cause an additional computing burden to the network.

2.1. Preprocessing. The method outlined in [23] is referred to. First, the irregular points are transformed into a pillar map according to their location. Besides the three channels mentioned in [23], we add a channel containing the pillar's minimum height, which expresses the difference between the edge and the inside of an object. Therefore, four channels represent the vertical distribution of the points in each pillar: the first channel records the number of points in the pillar; the second and the third record the maximum and minimum vertical coordinates of the points in the pillar; and the fourth records the reflectivity of the highest point in the pillar. Finally, a four-channel bird's eye view (4C-BEV) is obtained as the network input. This method is essentially equivalent to taking the upper cover shell of the spatial point cloud from a top-down perspective. Because of Earth's gravity, very few objects are suspended in the air, and obstacles can usually be clearly distinguished by direct observation of such shells. The channels' values need to be normalized, specifically, the first channel, because the point cloud density increases from far to near. Here, the distance factor K_d is added to make pillars with a similar degree of characteristic expression at different locations:

$$C_{[x,y]}{}^{1} = \frac{\log\left(N_p * K_d + 1\right)}{\log(64)}, \tag{1}$$

where N_p represents the number of points in each pillar and K_d is expressed as

$$K_d = \frac{\sqrt{x^2 + y^2}}{k_e}, \tag{2}$$

where k_e is the coefficient.

As shown in Figure 1, through observation with the naked eye, the objects are visible in the 4C-BEV, indicating that this method can preserve the point cloud's information in a vertical direction while compressing the data efficiently.

2.2. Backbone Network. The 4C-BEV is entirely consistent with the image in terms of data structure. Therefore, many popular backbone networks for image recognition can be used directly, such as ResNets [28], CSPDarknet53 [3], or VGG16 [29]. Besides, there are some differences between the object recognition tasks in image and point clouds. First, multiple scales are not necessary. In the image recognition task, the perspective phenomenon is one of the main factors requiring consideration in the network design. Therefore, the network contains output nodes representing different scales, or several preexisting boxes are predefined to represent different scales. When constructing the feature map using the LIDAR coordinates, as the objects have the same size as the real world, this perspective phenomenon is not encountered. Second, the scales of objects are different. In the image recognition task, in general, the recognition performance varies between large- and small-scale objects when using the same network. Typically, the area of interest appears near the observer, which means the identification accuracy of large targets is more important than others. The image passes through a multilayer network, which significantly reduces its scale and improves the recognition ability regarding large-scale objects. Taking CSPDarknet53 [3] as an example, after an input image was transmitted forward, the scales of the three outputs were reduced by 8, 16, and 32 times, respectively. Using the mentioned encoding approach, the feature vectors at each position can fuse with the features of the broader receptive field, identifying large-scale objects. However, for LIDAR-based tasks, the scene's object is relatively small compared to the scene size, with the large-scale output feature map affecting the recognition accuracy. Third, the orientation of the bounding boxes needs to be predicted. The maximum pooling layers play an essential role in a backbone network because they can prevent overfitting and improve the network's generalization ability; however, they can also enhance the rotation invariance.

The backbone network architecture is more similar to a tiny version of CSPDarknet53. Figure 2 illustrates the modified architecture. We use Conv (k, s, p, c_{out}) to represent a 2D convolutional operator, where c_{out} is the number of output channels; k, s, and p are the kernel size, stride, and padding size, respectively. The "Conv" operation contains a 2D convolutional operator, a group normalization (GN) layer, and an activation function layer sequentially when it acts as a convolutional middle layer. We used several small residual blocks to fuse features of the current layer and the previous layer. Then, we used big residual blocks to fuse shallow features and deep features. The nodes of the backbone network measure $h \times w \times c$, where h and w are the spatial dimensions, and c is the channel dimension. The input is a feature map with a fixed size of $h \times w \times 3$. The backbone network has two outputs: one with a fixed size of $h/2 \times w/2 \times 128$, while the other has a fixed size of $h/4 \times w/4 \times 512$.

2.3. Neck Network. The role of the neck network is to perform further feature extraction and connect the backbone to the head. Figure 3 shows the architecture of our neck network. The residual blocks are retained to aid further feature extraction. Upsampling operations are used to unify the scale of the feature map. Although there is no perspective phenomenon, different categories of objects have different sizes, and multiscale features play a positive role in the network.

2.4. Head Network. The head network is custom-designed for our specific 3D object recognition task and divided into three parts. The first part is used for confidence prediction, with the sigmoid function used to limit the result range to [0, 1]:

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}. \tag{3}$$

Two channels are assigned to each category, representing the regression confidence of this category based on horizontal and vertical anchors.

The second part is used for predicting bounding boxes. The spatial position and physical dimension are predicted in this part. As there is no perspective effect, it is reasonable that bounding box regression based on the standard reference value should arise. Therefore, we predefined an anchor map as the standard reference value in which each position has $2 \times N_c$ anchors, where N_c is the number of predicted categories. In general, the orientation and border predictions are conducted simultaneously [20, 21, 23]. This method cannot express the close relationship between the two ends of the interval. Inevitably, they produce the greatest divergence, which is incorrect. To keep the prediction of orientation continuous, we adopt an anchor-free [30] and anchor-based [20, 21, 23] combined method. Six channels are assigned to represent the regression parameters (except for orientation) of the two anchors at each position. Furthermore, the sine and cosine values are used to represent the orientation indirectly.

In most studies, there is little discussion on multicategory object prediction. By default, when predicting multicategory objects, the regression parameters for all categories are given, which leads to low information utilization (only $1/N_c$ information is useful). Thus, the convergence efficiency is greatly affected. In this study, it is designed to give only a set of border predictions at each position. The box center's category is determined according to the ground truth. The other positions' categories are determined by the overlap between the standard anchor and the ground truth bounding box.

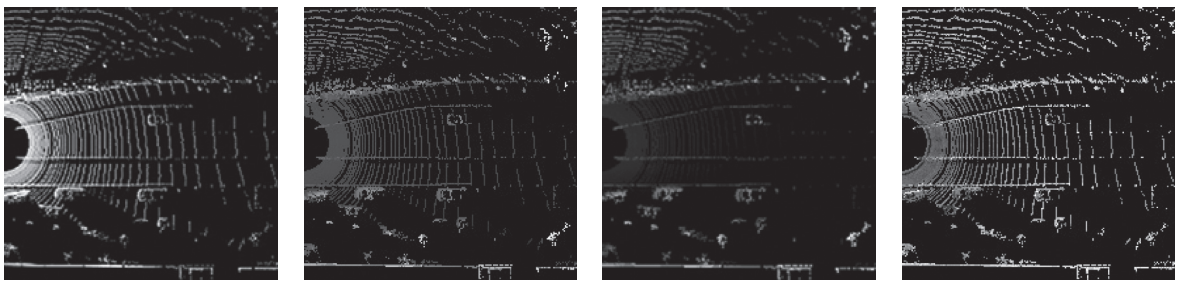


Figure 1: Four-channel bird's eye view: (a) the first channel, (b) the second channel, (c) the third channel, and (d) the fourth channel.

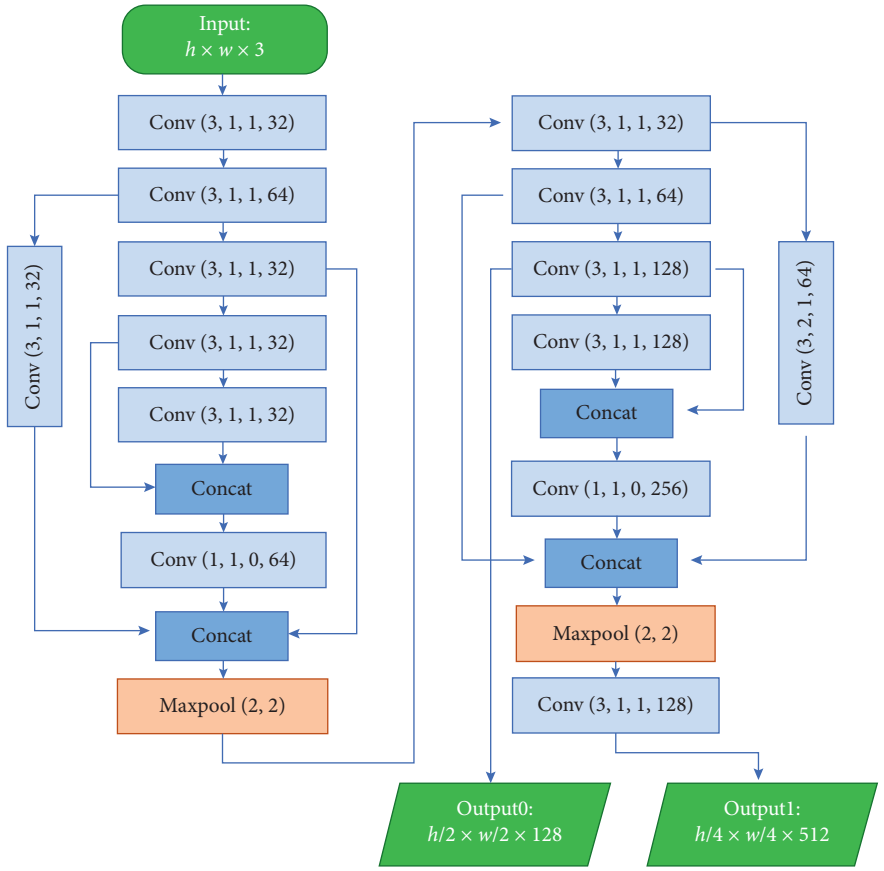


Figure 2: Architecture of the backbone network. The green blocks are input or output nodes (size labeled inside). The shallow blue blocks are the Conv operating set, and the deep blue blocks are the fusion process. The orange blocks are the maximum pooling layers. The values in parentheses indicate the filter size and stride.

For convenience, it is assumed that the category is determined, and there are two anchors for each position. The ground truth of the bounding box regression value R_{gt} of one anchor at each location can be expressed as follows:

$$R_{gt} = \left[\Delta x, \Delta y, \Delta z, \Delta h, \Delta w, \Delta l, \ \sin\theta_{gt}, \ \cos\theta_{gt}\right]^T, \tag{4}$$

$$\theta_{gt} = \arctan\frac{\sin\theta_{gt}}{\cos\theta_{gt}}, \tag{5}$$

$$\mathbf{A} = \left[x_a, y_a, z_a, h_a, w_a, l_a, \theta_a\right]^T, \tag{6}$$

$$\Delta x, \Delta y, \Delta z = \frac{x_{gt} - x_a}{d_a}, \frac{y_{gt} - y_a}{d_a}, \frac{z_{gt} - z_a}{h_a}, \tag{7}$$

$$d_a = \sqrt{w_a^2 + l_a^2}, \tag{8}$$

$$\Delta h, \Delta w, \Delta l = \log\left(\frac{h_{gt}}{h_a}\right), \ \log\left(\frac{w_{gt}}{w_a}\right), \ \log\left(\frac{l_{gt}}{l_a}\right), \tag{9}$$

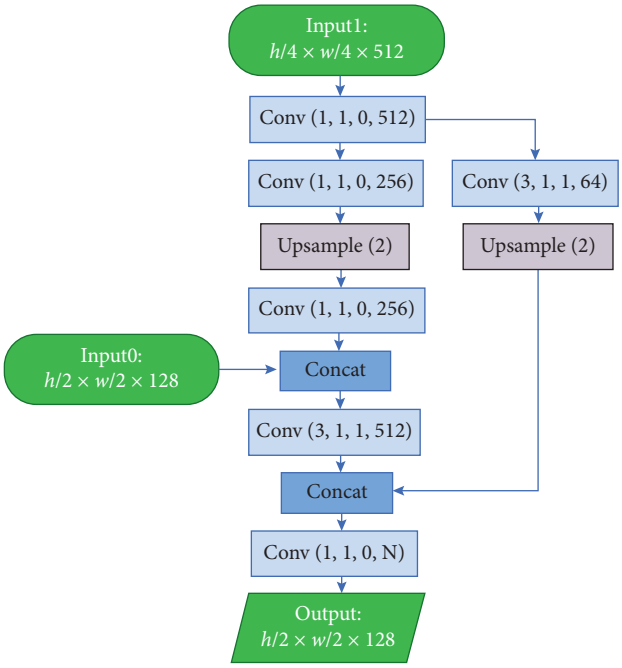


Figure 3: Architecture of the neck network. The gray blocks represent upsampling operations (the values in parentheses indicate the magnification rate).

where **A** denotes the parameter of one anchor in each position.

The third part is used for category prediction, for which N_c channels are assigned. The softmax function is used to transform the result to N_c probabilities, whose range is limited to [0, 1]:

$$\operatorname{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{N_c} e^{x_j}}. \tag{10}$$

2.5. Auxiliary Network. Because of the ability to obtain more detailed pointwise characteristics, point-based methods usually achieve higher accuracy than voxel-based methods. To enhance our method's accuracy, the pointwise feature was introduced to the network. Referenced by the SA-SSD [31], the pointwise feature learning network was set as an auxiliary network that only works during training, does not play a role in predicting, avoiding additional computational overhead caused by the additional feature extracting. The penultimate layer of the neck network was set as the former feature extraction layer of the auxiliary network, which is ultimately a voxelwise category prediction network. The auxiliary network is elaborated in Figure 4. The accuracy of border regression is highly dependent on the accuracy of category prediction. Therefore, the primary task is to improve the accuracy of object category prediction by the increased category information of the point cloud. Unlike the category prediction part in the head network, which only focuses on the category prediction of the bounding boxes' center voxels, the auxiliary network focuses on the category prediction of the voxels around the bounding box center. Since each voxel contains only one highest point, voxel features are equivalent to pointwise features. We randomly extract no more than 1000 internal points and no more than 250 external points of bounding boxes to save memory space. We recreate the voxel category label, depending on whether its highest point is within the bounding box. The whole operation is similar to an additional "droop-out" process, which improves the generalization performance of the network. The second task of the auxiliary network is to enhance the accuracy of bounding box regression. In this step, we randomly sample no more than 50 highest points within each bounding box and calculate the inverse distance to the bounding box center as its weight. The weighted average and maximum pointwise features among all points in the bounding box region are combined to express a boxwise feature:

$$f_b = \left[\frac{\sum_{j=1}^{50} w_j(p_j) f_j}{\sum_{j=1}^{50} w_j(p_j)}, \max(f_0, \cdots, f_j) \text{ where } f_j \text{ is sampled}\right], \tag{11}$$

$$w_j(p_j) = \begin{cases} \dfrac{1}{1 + \|p_j - p_c\|_2}, & \text{if } p_j \text{ is sampled,} \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

2.6. Loss. The loss contains the central part and the auxiliary part. The central part contains confidence loss, regression loss, and category loss. The auxiliary part contains point category loss and box regression loss. We adopted the smooth$_{L1}$ function [5] to calculate the bounding box regression loss:

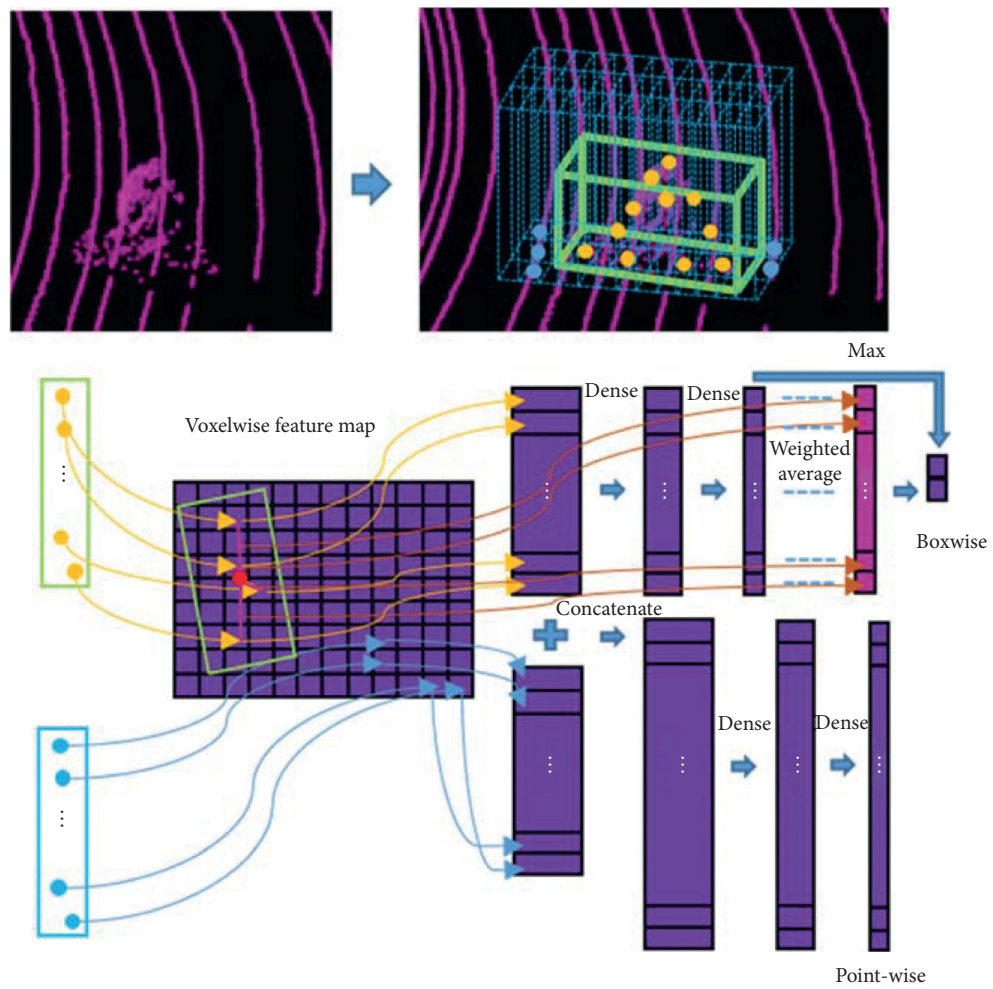


Figure 4: The architecture of the auxiliary network. The blue dash in the scene is the voxel boundary. The green box is the ground truth bounding box. The yellow points and blue points are the voxels' highest points, which are included and not included in the bounding boxes, respectively. The red point is the center of the ground truth bounding box. The boxwise feature is then followed by two fully connected layers (the operation is called "Dense" customarily), generating bounding box regression values similar to R_{gt} (see equation (4)).

$$\mathrm{SMOOTH}_{L1}\left(\mathbf{X}_p, \mathbf{X}_{gt}, a\right) = \frac{\sum_n \mathrm{smooth}_{L1}\left(\mathbf{X}_p - \mathbf{X}_{gt}, a\right)}{n}, \quad (13)$$

$$\mathrm{smooth}_{L1}(x, a) = \begin{cases} 0.5ax^2, & |x| < \frac{1}{a^2}, \\ |x| - \frac{0.5}{a}, & |x| \geq \frac{1}{a^2}. \end{cases} \quad (14)$$

Smooth_{L1} function has stable convergence characteristics in the case of large deviation and adequate convergence in small variation. The predictions of category and confidence are converted into the probability value prediction within the interval of [0, 1]. The cross-entropy function was applicable to calculate their losses:

$$\mathrm{CE}\left(\mathbf{X}_p, \mathbf{X}_{gt}\right) = -\frac{\sum_n \log\left(\mathbf{X}_p\right) * \mathbf{X}_{gt}}{n}, \quad (15)$$

where $\mathbf{X}_p$, $\mathbf{X}_{gt}$ are the predicted value and ground truth value, respectively.

The ground truth values of the category are labeled as a one-hot form. The focus loss [32] can solve the problem that when the proportion of positive and negative samples is unbalanced, the negative ones are submerged in the positive ones. Although the positive and negative labeled data distributions are incredibly uneven, the ratio of positive and negative samples is given. To avoid the focus loss affecting the rate of convergence, we do not adopt focus loss.

Not all losses in each position are calculated in a feature map. Some grids that are far from the center of the object are inaccurate and can be neglected. The positive confidence label is vital because it can be used as a mask to filter out untrusted data not to be included in the loss calculation. In Section 2.4, an anchor map was established. We excluded the angle parameters and determined the confidence by calculating the intersection over union (IoU) between the ground truth bounding box and the anchors in the map. Because the confidence feature map does not rely on the vertical direction position, the projection plane in the vertical direction of the ground truth bounding box and anchors are used when calculating the IoU:

The final loss L is defined as

$$L = L_{\text{conf}} + L_{\text{reg}} + L_{\text{cls}} + L_{\text{point}} + L_{\text{box}}. \tag{16}$$

Among the final loss, the confidence loss is expressed as

$$L_{\text{cof}} = \frac{\sum \mathbf{P}_{gt} * \text{CE}\left(\mathbf{P}_p, \mathbf{P}_{gt}\right)}{\sum \mathbf{P}_{gt}} + \frac{\sum \mathbf{N}_{gt} * \text{CE}\left(1 - \mathbf{P}_p, \mathbf{N}_{gt}\right)}{\sum \mathbf{N}_{gt}}, \tag{17}$$

where $\mathbf{P}_p$ represents the positive confidence prediction, $\mathbf{P}_{gt}$ denotes the positive corresponding ground truth, and $\mathbf{N}_{gt}$ denotes the negative corresponding ground truth.

The regression loss is expressed as

$$L_{\text{reg}} = \frac{\sum \mathbf{P}_{gt} * \text{SMOOTH}_{L1}\left(\mathbf{R}_p, \mathbf{R}_{gt}, a\right)}{\sum \mathbf{P}_{gt}}, \tag{18}$$

where $\mathbf{R}_p$ is the bounding box regression prediction and $\mathbf{R}_{gt}$ is the corresponding ground truth.

The category loss is expressed as

$$L_{\text{cls}} = \frac{\sum \mathbf{M}_{gt} * \text{CE}\left(\mathbf{C}_p - \mathbf{C}_{gt}\right)}{\sum \mathbf{M}_{gt}}, \tag{19}$$

where $\mathbf{M}_{gt}$ is the maximum last channel value of $\mathbf{P}_{gt}$.

The auxiliary parts of the loss are defined as

$$L_{\text{box}} + L_{\text{point}} = \frac{\sum_{n_b}\left(\sum \text{SMOOTH}_{L1}\left(\mathbf{B}_p, \mathbf{B}_{gt}, a\right) * \mathbf{W}_b / \sum \mathbf{W}_b\right)}{n_b} + \frac{\sum_{n_c} \text{CE}\left(\mathbf{C}^p{}_p, \mathbf{C}^p{}_{gt}\right)}{n_c}, \tag{20}$$

where $\mathbf{W}_b$ denotes the weights calculated by equation (12). $\mathbf{B}_p$, $\mathbf{C}^p{}_p$ denote the boxwise regression feature and category prediction of the sample point; and $\mathbf{B}_{gt}$, and $\mathbf{C}^p{}_{gt}$ denote the corresponding ground truth, respectively.

2.7. Dataset. Most 3D object recognition networks are trained using the KITTI dataset [33]. The KITTI dataset contains 7481 frames, among which we selected 2000 frames as the verification set and the remaining 5481 frames as the training set. We were interested in cars, trucks, vans, pedestrians, and cyclists among the object categories. Besides, trucks and vans were merged into one class. In this study, the Apollo dataset [34] was also adopted.

In contrast to the KITTI dataset, the Apollo dataset contains continuous frame data. When a vehicle turns, the surrounding objects show disordered orientations and spatial positions, which are more complicated than those in the nonturning state. This representative disordered data frequently occurs in continuous frame data. The Apollo dataset contains 16 scenes. Each scene contains 2–5 sections of continuous frame data collected at a frequency of 2 Hz, lasting for 1 min. We take a section of each scenario as the verification set and the rest as the training set. The final training set consisted of 3,943 frames, while the validation set consisted of 1,650 frames. We were interested in four types of labeled data: small vehicles, big vehicles, pedestrians, and riders (i.e., motorcyclists and bicyclists), which were labeled using the abbreviations "VEH, TRU, PED, CYC," respectively, during the visualization step. The data augmentation technique [34] was adopted during the training process.

2.8. Details. In this study, points inside the range covered from 41.6 m in front, 20.8 m left and right, and 2 m above and below the LIDAR coordinate system were used to construct the BEV feature map. The resolution of the grid was set as 0.2 m. Therefore, the BEV feature map was divided into 208×208 grids.

A minibatch gradient descent was conducted with a batch size of 1. We placed all batch normalization layers in the network with the group normalization layers because of the small batch size. Each training set involved in the training was defined as an epoch. Epochs were set as 100 and the first 75 epochs had a learning rate of 0.001, while the remaining epochs had a learning rate of 0.0001. All algorithms were run on a workstation with a Core i7 CPU, 8G RAM, an NVIDIA 1080Ti GPU, and the open-source deep learning framework TensorFlow. Nonmaximum suppression was deployed to filter out excess bounding boxes, with the IoU threshold set as 0.1.

2.9. Evaluation Indicators. It is assumed that n_p objects are predicted with n_{gt} objects labeled in the ground truth. First, the prediction object needs to be paired with the ground truth object by calculating the IoU. Considering the predicted objects as the benchmark and matching all predicted objects with the labeled objects using the maximum IoU, the calculated result is denoted as the precision. Similarly, considering the labeled objects as the benchmark and matching all labeled objects with the predicted objects using the maximum IoU, the calculated result is denoted as the recall. As the recall rises, the precision drops. Using recall as the horizontal axis and precision as the vertical axis, the area surrounded by the plotted recall-precision curve and the coordinate axis is the average precision (AP), which is widely adopted to evaluate the performance of the network. We set the parameters consistent with the KITTI benchmark, where the IoU threshold of cars is 0.7, and pedestrians and cyclists are 0.5.

3. Results and Discussion

3.1. Loss Curves. Validation was performed using one batch of data in the validation set per 100 iterations. During the training process, the loss function fluctuated violently with the weight decline of 0.99, representing the changing trend of the loss. The evolution of the loss curves throughout the training process is shown in Figure 5. By the time the

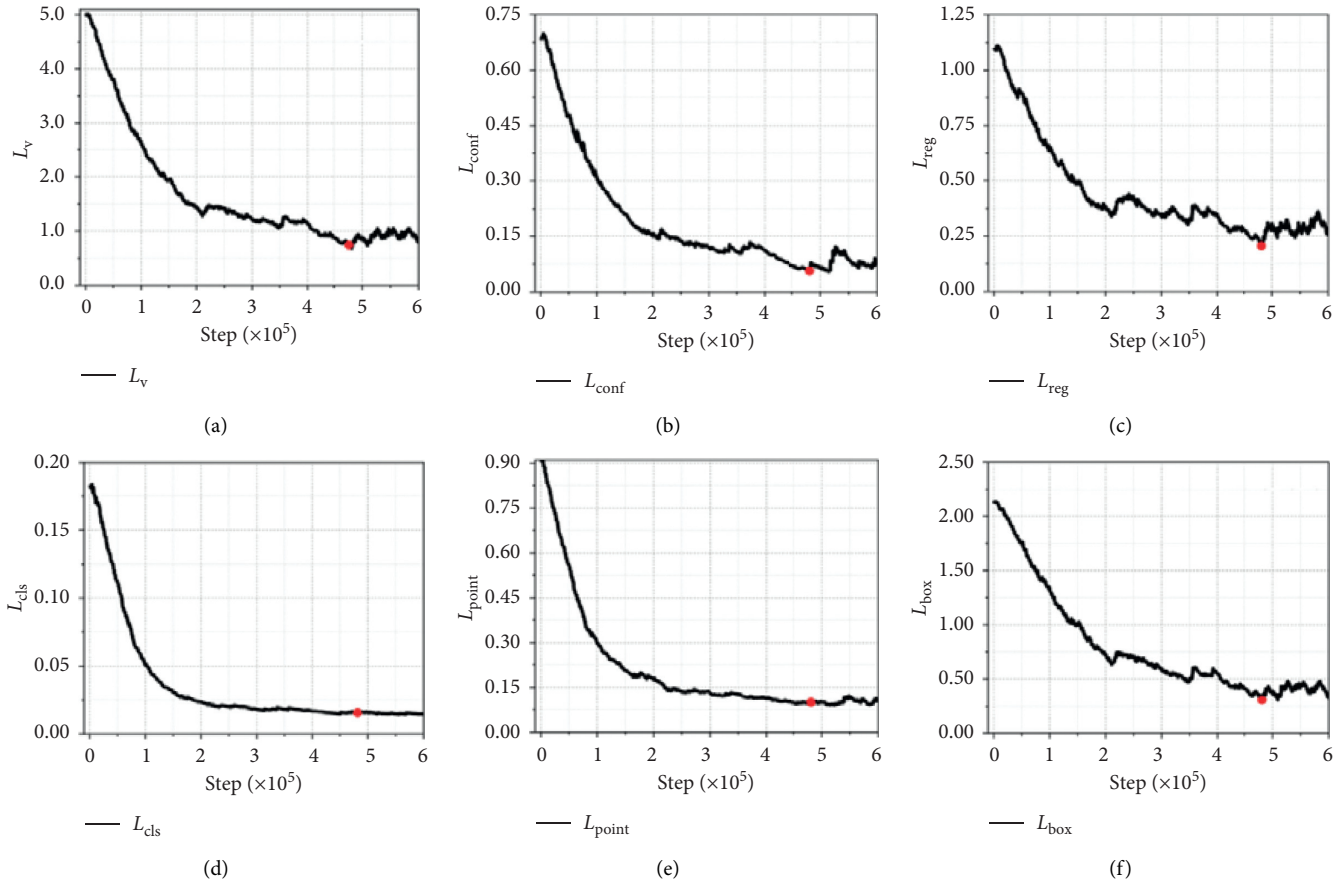


Figure 5: Loss curves for the training process: (a) validation loss L_v, (b) confidence loss L_{conf}, (c) regression loss L_{reg}, (d) category loss L_{cls}, (e) point loss L_{point}, and (f) box loss L_{box}.

iteration reaches 500,000, the network has converged. Our trained model was marked as red points in the figures.

3.2. Speed. The acquisition frequencies commonly used by LIDAR are 5, 10, 15, and 20 Hz. The entire recognition process is divided into preprocessing and inference. The mean preprocessing and inference process duration was approximately 5.7 and 31.2 ms, respectively. The mean recognition process duration was approximately 37 ms, which meets the real-time requirement.

3.3. Accuracy. The recall precision curves (trained by the KITTI dataset) are given in Figure 6. The APs of the regression and category are listed in Table 1 (trained by the KITTI dataset). The data we mainly focus on is marked in bold. The AP for cars was 92.5, which is relatively high, considering that the regression is based on the anchor determined by the category prediction. The trucks, which are not considered in the KITTI benchmark, are included in the identification. Due to the uneven distribution of training data, the AP of other categories is slightly lower. The total AP (0.5), the total AP (0.7), and the total AP (categories) can reach 93.2, 85.5, and 97.4, respectively. In Apollo dataset, the labeled bounding boxes are the objects' visible parts, which vary from the actual physical size, resulting in low indicators. As for apparent objects, using the Apollo dataset shows similar performance to the KITTI dataset, which is described in Section 3.5.

3.4. Comparison. The main contribution of this paper lies in the anchor-free and anchor combined prediction method and auxiliary networks specially designed for the bird's eye view network. The contrast effect is shown in Table 2. Due to the use of the data augmentation technique, the training results can be slightly different. Indicators within ±1% were regarded as the same performance. Our proposed method significantly improves the prediction results of pedestrians and cyclists, and the data we mainly focus on is marked in bold.

3.5. Scene Analysis. The method described in this paper was designed for a complex environment (multiobject, unordered objects, multicategory). In this section, some typical scenes are selected to analyze the network's performance concerning object recognition. The continuous frame data from the Apollo dataset provides sufficient verification of the stability of the recognition effect.

Figure 7 shows three frames of a congested traffic scene. This scene features many vehicles in a dense array, precisely what the proposed network has been designed to identify.

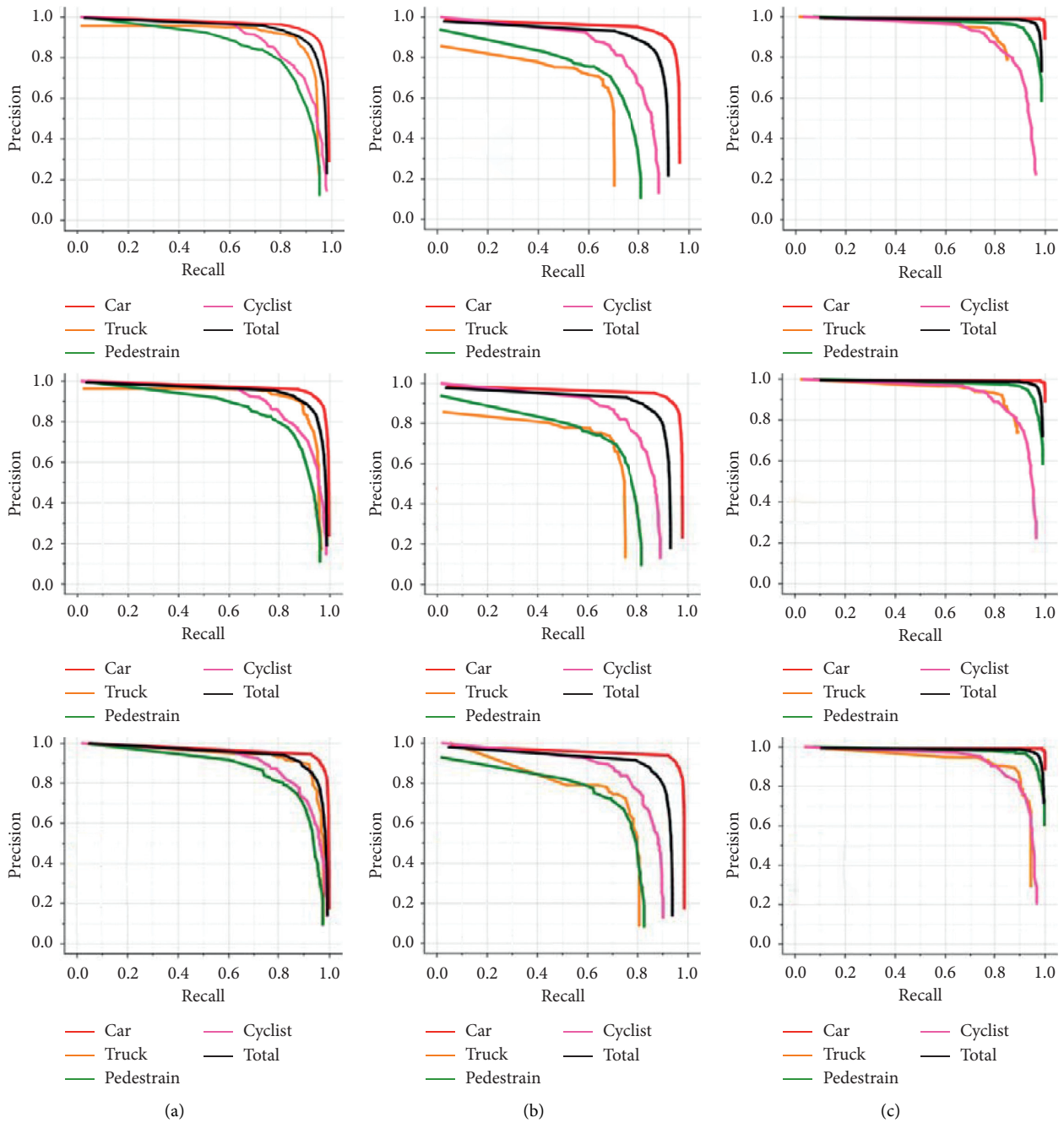


Figure 6: Recall-precision curves (trained by the KITTI dataset): (H, M, E) the hard, moderate, and easy modes separately; (a–c) the recall-precision curve for bounding box regression (IoU > 0.5), the recall-precision curve for bounding box regression (IoU > 0.7), and the recall-precision curve for category separately.

The recognition result corresponding to the turning scene is shown in Figure 8. Most objects are recognized accurately, and the performance remains stable. Despite the large size of the rotation scene, the direction of the objects was also predicted accurately.

Figure 9 depicts a scene containing several objects belonging to different categories, including a vehicle, truck, and cyclist. Each object was recognized consistently and accurately across the consecutive frames.

The recognition visualization of the KITTI dataset is shown in Figure 10, and a series of typical complex scenarios are selected, including multiobject, unordered objects, and multicategory.

3.6. Discussion. The orientation of the bounding box is expressed by sine and cosine values indirectly in this paper. Compared with the directly predicted method, when the

Table 1: Evaluation indicators trained/tested by the KITTI dataset.

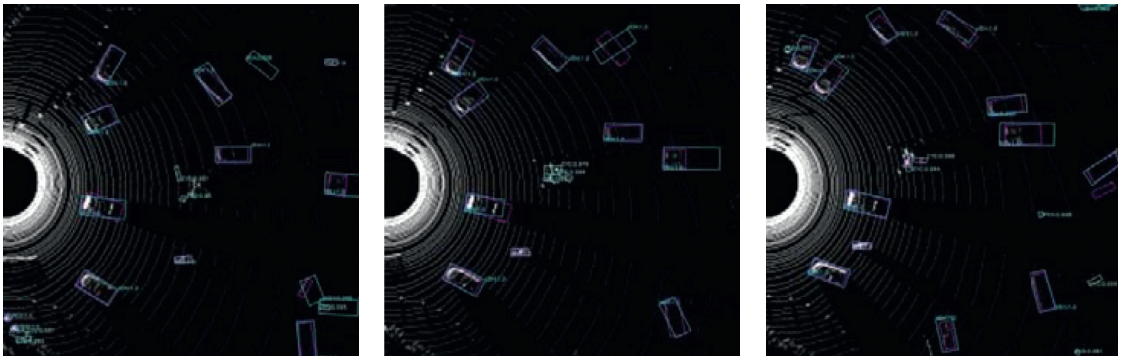
Category	AP (regression) IoU > 0.5			AP (regression) IoU > 0.7			AP (category)		
Mode	Hard	Moderate	Easy	Hard	Moderate	Easy	Hard	Moderate	Easy
Car	95.7	96.8	96.6	**92.2**	**94.1**	94.3	**99.1**	**99.2**	99.1
Truck	88.5	90.4	92.7	54.5	59.7	68.3	**82.5**	**85.9**	89.6
Pedestrian	**83.3**	**84.6**	86.3	64.3	65.4	66.4	**95.9**	**96.9**	97.8
Cyclist	**88.3**	**89.9**	90.2	78.3	80.2	81.2	**89.1**	**90.4**	91.1
Total	**93.2**	**94.1**	94.2	**85.5**	**87.0**	87.1	**97.4**	**97.8**	98.2

Table 2: Results of comparison trained/tested by the KITTI dataset.

Category	AP (regression) IoU > 0.5	AP (regression) IoU > 0.7	AP (category)	L_{point}	L_{box}	$\sin\theta/\cos\theta$
Vehicle	90.0	77.4	97.7			
Pedestrian	66.5	42.2	88.6			
Cyclist	68.2	32.5	73.2			
Vehicle	95.6	93.0	99.2			✓
Pedestrian	77.6	61.8	96.0			✓
Cyclist	85.1	68.3	83.1			✓
Vehicle	94.2	91.2	99.3	✓		✓
Pedestrian	81.4	63.2	95.3	✓		✓
Cyclist	86.7	79.2	87.8	✓		✓
Vehicle	95.7	92.2	99.1	✓	✓	✓
Pedestrian	**83.3**	64.3	95.9	✓	✓	✓
Cyclist	**88.3**	78.3	**89.1**	✓	✓	✓

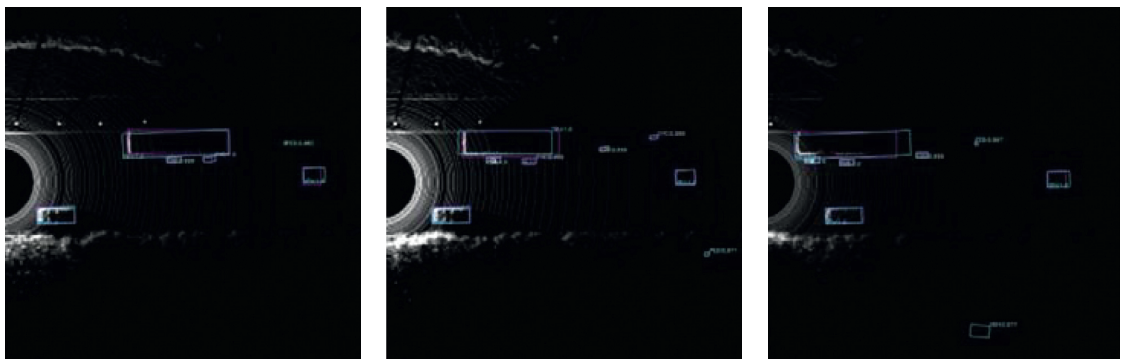
(a) (b) (c)

Figure 7: Scenes of congested traffic: (a–c) Three frames from a single scene.



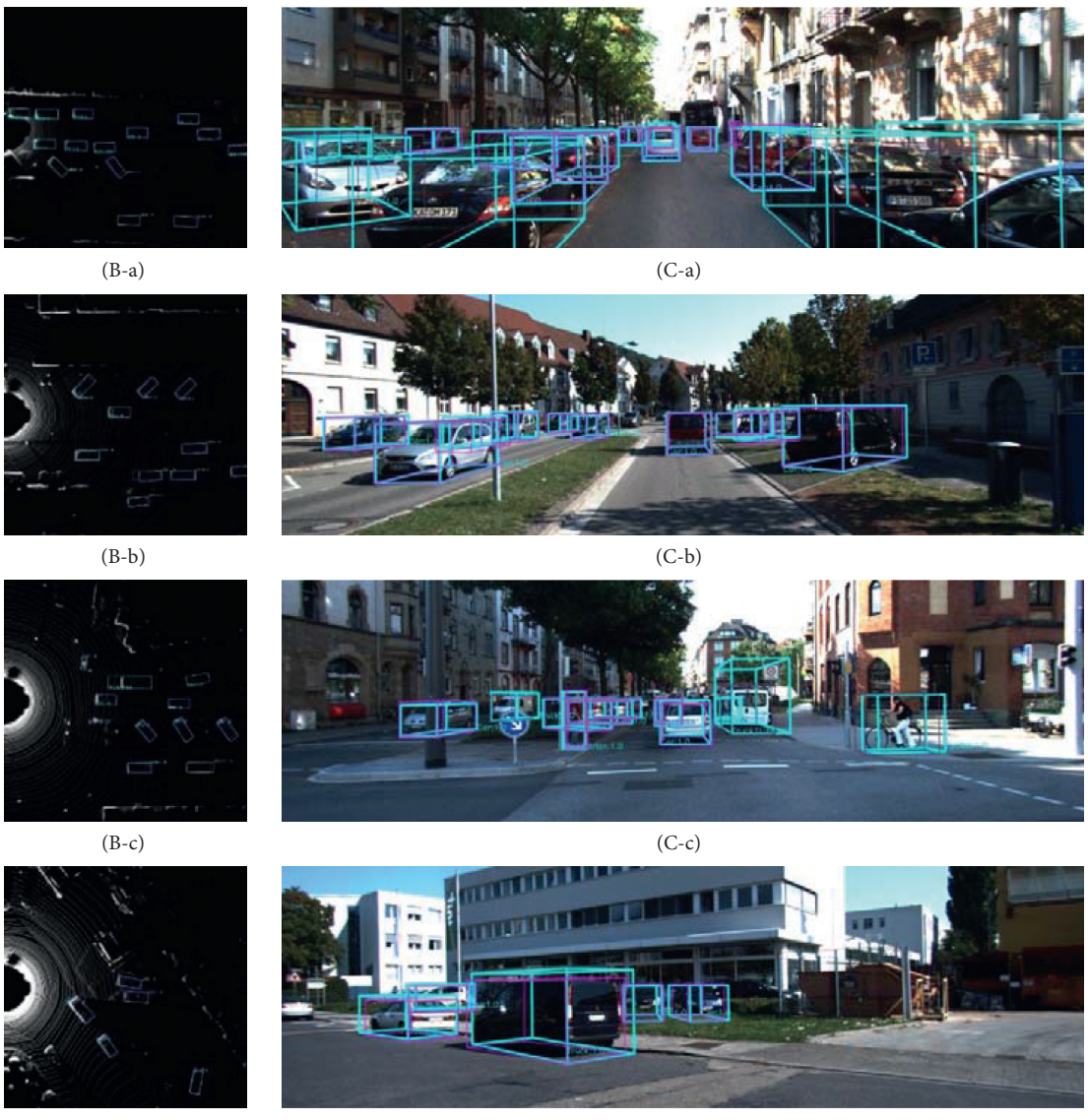
(a) (b) (c)

Figure 8: Three consecutive frames from a turning scene, (a), (b), and (c) are the recognition results for three consecutive frames, with an acquisition interval of 0.5 s.



(a) (b) (c)

Figure 9: Set of consecutive frames from a multicategory scene: (a–c) the recognition results for three consecutive frames, with an acquisition interval of 0.5 s.



(B-a) (C-a)

(B-b) (C-b)

(B-c) (C-c)

(B-d) (C-d)

Figure 10: Recognition visualization of the KITTI dataset. (B, C) The bird's eye view and corresponding camera view separately; (a, b) multiobject scenes; (c) multicategory scene; (d) unordered objects scene.

predicted value is not accurate, it will not deviate wildly. It is the advantage of continuous prediction, which makes the network robust. However, because of an additional prediction dependence in the calculation, the accuracy will worsen if only the prediction results with high quality are compared. Similarly, in the proposed multicategory object recognition network, the bounding box regression is also highly dependent on the category prediction, making the regression not achieve the highest precision, but makes the prediction more robust.

Our approach is not very sensitive to large objects that are very close to the observer. It is the reason that the AP of cars for easy mode was slightly worse than that for the moderate mode. Expanding the receptive field can alleviate such problems, but it will increase the depth of the network. Our experimental results showed that deeper networks increase the inferencing time but have no significant effect on accuracy, which is very different from image recognition tasks.

The indicators in this paper are much higher than those on the KITTI ranking list. The main reason is that the scope of perception selected in this paper is smaller than the standard. Our range covers 41.6 m (70.4 m in standard) in front, 20.8 m (40 m in standard) left and right, and 2 m above and below the LIDAR coordinate system, which has met our application's commands.

4. Conclusions

The main aim of this study was to design a LIDAR-based object recognition method for autonomous vehicle systems. Thus, we proposed a new multifunctional network that operates in real-time with high accuracy and stable performance. As several recognition methods achieve considerable performance differences in different datasets, the Apollo dataset was also adopted besides the KITTI dataset in this study, making the validation results more consistent with actual application scenes. Hence, the proposed recognition method has a high practical value. The key findings of this study are outlined below:

(1) The proposed network realizes the accurate recognition of multiple types of objects in real time.

(2) To tackle the inaccurate category prediction, an auxiliary network was designed to help network auxiliary learning pointwise features. It is not limited to the object's center point category, making the prediction result more robust.

(3) To tackle the inaccurate bounding box prediction, firstly, the validity of the indirect expression of orientation angle by sine and cosine values is verified. Besides, another auxiliary network was designed to help network auxiliary learning box-wise features.

(4) The proposed network delivers a stable and robust object recognition performance in complex environments (multiobject, unordered objects, and multicategory), reflecting its high practical value.

(5) The proposed network's performance is impacted negatively when the LIDAR system is obscured and is not sensitive to large objects that are very close to the observer. Further research is necessary to address this weakness of the network.

In this study, we have considered several possible problems in practical application scenarios. Although our proposed method needs to be further improved, it has demonstrated a very high practical application potential. Based on the phenomenon that most current methods rely heavily on the 3D sparse convolutional operation [35], our research's stable performance showed that artificial bird's eye view features can do the same thing as three-dimensional convolution.

Data Availability

The data used to support the findings of this study are available upon request to the corresponding author.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was funded by the National Natural Science Foundation of China, under Grant no. 51775548.

References

[1] W. Liu, J. Sun, W. Li, T. Hu, and P. Wang, "Deep learning on point clouds and its application: a survey," *Sensors*, vol. 19, no. 4188, pp. 1–22, 2019.

[2] M. Li and Y. Chen, "Registration of laser scanning point clouds: a review," *Sensors*, vol. 18, no. 1641, pp. 1–25, 2018.

[3] C. Wang, H. Liao, Y. Wu et al., "CSPNet: a new backbone that can enhance learning capability of CNN," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 390-391, Seattle, WA, USA, June 2020.

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

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[6] J. Redmon and A. Farhadi, "YOLOv3: an incremental improvement," 2018, https://arxiv.org/abs/1804.02767.

[7] A. Bochkovskiy, C. Wang, and H. Liao, "YOLOv4: optimal speed and accuracy of object detection," 2020, https://arxiv.org/abs/2004.10934.

[8] Y. Wang, Y. Sun, Z. Liu et al., "Dynamic graph CNN for learning on point clouds," *ACM Transactions on Graphics*, vol. 38, no. 5, pp. 1–12, 2019.

[9] P. Veličković, A. Casanova, P. Liò et al., "Graph attention networks," in *Proceedings of the International Conference on Learning Representations*, pp. 1–12, Vancouver, BC, Canada, May 2018.

[10] C. Wang, B. Samari, and K. Siddiqi, "Local spectral graph convolution for point set feature learning," in *Proceedings of the European Conference on Computer Vision*, pp. 52–66, Munich, Germany, September 2018.

[11] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3693–3702, Honolulu, HI, USA, July 2017.

[12] L. Yi, H. Su, X. Guo, and L. Guibas, "SyncSpecCNN: synchronized spectral CNN for 3D shape segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6584–6592, Honolulu, HI, USA, July 2017.

[13] K. Roman and V. Lempitsky, "Escape from cells: deep KD-networks for the recognition of 3D point cloud models," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 863–872, Venice, Italy, October 2017.

[14] Y. Li, R. Bu, M. Sun et al., "PointCNN: convolution on x-transformed points," 2018, https://arxiv.org/pdf/1801.07791.pdf.

[15] J. Pamplona, C. Madrigal, and A. Escalera, "PointNet evaluation for on-road object detection using a multi-resolution conditioning," in *Proceedings of the Iberoamerican Congress on Pattern Recognition*, pp. 513–520, Madrid, Spain, November 2018.

[16] C. Qi, H. Su, K. Mo, and L. 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. 77–85, Honolulu, HI, USA, July 2017.

[17] C. Qi, L. Yi, H. Su, and L. Guibas, "PointNet++: deep hierarchical feature learning on point sets in a metric space," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 5100–5109, Long Beach, CA, USA, December 2017.

[18] Z. Wu, S. Song, A. Khosla et al., "3D ShapeNets: a deep representation for volumetric shapes," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1912–1920, Boston, MA, USA, June 2015.

[19] J. Wu, C. Zhang, T. Xue et al., "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," 2016, https://arxiv.org/abs/1610.07584.

[20] 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*, pp. 4490–4499, Salt Lake City, UT, USA, June 2018.

[21] A. Lang, S. Vora, H. Caesar et al., "Pointpillars: fast encoders for object detection from point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 12697–12705, Long Beach, CA, USA, June 2019.

[22] C. Sun, W. Zhan, J. She et al., "Object detection from the video taken by drone via convolutional neural networks," *Mathematical Problems in Engineering*, vol. 2020, Article ID 4013647, 10 pages, 2020.

[23] W. Ali, S. Abdelkarim, M. Zidan, M. Zahran, and A. E. Sallab, "YOLO3D: end-to-end real-time 3D oriented object bounding box detection from LiDAR point cloud," *Lecture Notes in Computer Science*, vol. 11131, pp. 716–728, 2019.

[24] P. Cao, H. Chen, Y. Zhang, and G. Wang, "Multi-view frustum pointnet for object detection in autonomous driving," in *Proceedings of the IEEE International Conference on Image Processing*, pp. 3896–3899, Taipei, Taiwan, September 2019.

[25] X. Wang and H. Su, "Completely model-free RL-based consensus of continuous-time multi-agent systems," *Applied Mathematics and Computation*, vol. 382, Article ID 125312, 2020.

[26] X. Wang, H. Su, X. Wang, and G. Chen, "Fully distributed event-triggered semiglobal consensus of multi-agent systems with input saturation," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 6, pp. 5055–5064, 2017.

[27] S. Shi, C. Guo, L. Jiang, and Z. Wang, "PV-RCNN: point-voxel feature set abstraction for 3D object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10526–10535, Seattle, WA, USA, June 2020.

[28] K. He, X. Zhang, S. Ren et al., "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the International Conference on Learning Representations*, pp. 1–14, San Diego, CA, USA, May 2015.

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

[31] C. He, H. Zeng, J. Huang et al., "Structure aware single-stage 3D object detection from point cloud," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11873–11882, Seattle, WA, USA, June 2020.

[32] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020.

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

[34] P. Wang, X. Huang, X. Cheng et al., "The ApolloScape open dataset for autonomous driving and its application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, pp. 2702–2719, 2019.

[35] Y. Yan, Y. Mao, and B. Li, "Second: sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, pp. 1–17, 2018.