sensors

Article

3D-GIoU: 3D Generalized Intersection over Union for Object Detection in Point Cloud

Jun Xu [1,*], Yanxin Ma [2,*], Songhua He [1] and Jiahua Zhu [2]

[1] College of Information Science and Engineering, Hunan University, Changsha 410082, China; hesonghua@hnu.edu.cn

[2] College of Meteorology and Oceanography, National University of Defense Technology, Changsha 410073, China; zhujiahua1019@hotmail.com

[*] Correspondence: junxu714@163.com (J.X.); mayanxin@nudt.edu.cn (Y.M.)

Received: 3 August 2019; Accepted: 18 September 2019; Published: 22 September 2019

Abstract: Three-dimensional (3D) object detection is an important research in 3D computer vision with significant applications in many fields, such as automatic driving, robotics, and human–computer interaction. However, the low precision is an urgent problem in the field of 3D object detection. To solve it, we present a framework for 3D object detection in point cloud. To be specific, a designed Backbone Network is used to make fusion of low-level features and high-level features, which makes full use of various information advantages. Moreover, the two-dimensional (2D) Generalized Intersection over Union is extended to 3D use as part of the loss function in our framework. Empirical experiments of Car, Cyclist, and Pedestrian detection have been conducted respectively on the KITTI benchmark. Experimental results with average precision (AP) have shown the effectiveness of the proposed network.

Keywords: 3D object detection; point cloud; voxel; convolutional neural networks; 3D generalized intersection over union

1. Introduction

The task of object detection is to find the objects of interest in a given scene and determine their category and location. In the past few years, deep learning has made substantial progress in many fields due to its powerful feature learning ability, ranging from object recognition [1,2] to natural language processing [3,4]. Deep learning-based object detection methods have achieved a really high detection accuracy and are widely used in various practical applications, such as autonomous driving [5–9], mobile robots [10,11], video surveillance [12–14], and so forth.

There are some three-dimensional (3D) object detection techniques based on RGB images [6,7]. However, due to the loss of sophisticated spatial structure information in the process of projecting a 3D scene into a two-dimensional (2D) image, the performance of 3D object detection in RGB image is poor. As a result, this is limited for autonomous driving and robot vision, since these applications pay more attention to the 3D positional information of an object. Compared with RGB images, point cloud contains 3D spatial and structural information of the given scene. In addition, with the rapid development of LIDAR, the acquisition of point cloud is becoming more and more convenient. Therefore, point cloud-based 3D object detection has become an important component in many 3D applications.

Currently, deep learning-based 3D object detection in point cloud algorithms has a main challenge: the low detection precision. To solve this problem, some technologies [5,15] use a 2D detection algorithm in an image to locate the object, then use bounding box regression to achieve 3D object detection. According to the detection results of a KITTI data set [16], they have achieved good results thanks to the accurate 2D detection in images. However, these methods have two problems: they are highly dependent on 2D object detection technology and have an expensive time cost. To solve these problems, we propose our solutions: 1. Only point cloud is used for 3D object detection to reduce the time cost, 2. Feature maps of different layers are aggregated to improve the expressive ability of features, 3. A new loss function, 3D Generalized Intersection over Union (GIoU) is proposed to optimize the alignment of 3D prediction and ground truth bounding box, so as to improve the precision of 3D object detection.

In the 3D object detection network, as the depth of the network increases, the size and receptive field of the feature map also change. The lower-level feature map has high resolution and contains smaller receptive field and more detailed texture information. The high-level feature map has low resolution and contains larger receptive field and more semantic information. The integration of different levels of feature maps can improve the comprehensive expression ability of feature maps. Based on this observation, this paper proposes a Backbone Network, in which the low-resolution feature map is spliced with the high-resolution feature map after the up-sampling. Features with stronger expression ability are obtained after the fusion of features of different levels.

As we know, object detection is a multitask learning problem, which consists of object classification and object localization. Actually, bounding box regression plays an important role in object localization. Many superior object detectors rely on accurate bounding box regression to locate objects, such as VoxelNet [17], SECOND [18], and Complex-YOLO [19]. Although optimizing the architecture of deep neural network is a trend to improve the accuracy of bounding box, designing a reasonable regression loss function is also an important way. Consequently, various regression loss functions have been proposed. Among the current popular object detectors, the l_1-smooth and l_2-norm are the mainstream of loss functions used to optimize bounding box, where the l_2-norm is also known as the mean square error (MSE). Nevertheless, these functions cannot represent the core evaluation criteria (i.e., Intersection over Union (IoU)). However, there is a problem with the IoU as a loss function, that is, when the two bounding boxes are completely out of coincidence, optimization cannot be performed. In addition, IoU cannot reflect the alignment of two bounding boxes. To solve these problems, Hamid et al. [20] proposed 2D Generalized Intersection over Union (GIoU) for 2D object detection, which solved the problem of unification of loss function and evaluation criteria and improved the accuracy of 2D object detection.

However, one of the drawbacks of 2D GIoU is that it only applies to 2D object detection. To break this dilemma, this paper proposes a 3D GIoU regression loss function for 3D object detection. The ablation experiments show that the use of 3D GIoU can actually improve the detection performance.

In particular, the main contributions of this paper can be summarized as follows:

A Backbone Network is designed to aggregate the low-level features and high-level features for 3D object detection, which improves the performance of detection and enhances the robustness of the network.

3D GIoU loss function is proposed to optimize bounding box regression.

The proposed detection pipeline is evaluated on a KITTI benchmark dataset, which demonstrates that the proposed network is superior to other algorithms in average precision (AP).

2. Related Work

3D object detection methods can be divided into three categories by the representations of input data, therefore, monocular image-based, point cloud-based, and multimodal fusion-based methods.

2.1. Monocular Image-Based Detection

Monocular image-based 3D detection is the most challenging for the lack of 3D space information, but there are still some studies [6,7] that have focused on it, mainly for two reasons. On the one hand, it is a fundamental problem with great impact; on the other hand, the equipment for obtaining monocular image is more convenient and much cheaper. Considering the fact that the objects should be on the ground plane, to obtain 3D proposal from monocular image, Mono3d [6] exhaustively obtained 3D proposals from several predefined 3D regions. To select the best candidates, each candidate box is scored via encoding several intuitive potentials, such as contextual information, typical object shape, size, and location priors. The expensive computing cost of 3D sliding windows in Mono3d [6] brings a serious problem of inefficiency. To solve this problem, GS3D [7] first predicted the position, category, and orientation of 2D bounding box in a monocular image. Then it estimated the size of 3D box and roughly calculated the position of 3D candidate box in camera coordinates. The resulting 3D bounding box is projected as a front view (FV), a bird's eye view (BEV), and a side view. Next, the 3D structural features extracted from projected surface regions and the texture information extracted from 2D box were merged. Finally, to improve the performance of detection, the fused features were used to refine the 3D bounding box. Compared with other monocular image-based 3D object detection methods, GS3D [7] achieves superior performance. However, the detection accuracy is far from meeting the requirements of automatic driving and other applications.

2.2. Point Cloud-Based Detection

Compared with monocular images, point cloud is regarded as an important information source for 3D object detection, since it directly reflects the real-world scenes. However, due to 3D point cloud being irregular, deep learning cannot be directly applied to object detection in point cloud. There are two popular methods to transform point cloud to regular data, and then input the transformed data to a 3D object detection network. The first method is projecting a point cloud to 2D plane to achieve 2D images [19,21,22]. To utilize the 3D data more efficiently, Complex YOLO [19] and PIXOR [21] projected point cloud to a BEV and applied 2D detection on the projected image. Although the processing method of the projection was efficient, it lost the spatial structure information of the point cloud, which led to poor detection performance.

Instead of projecting point cloud to 2D planes, an alternative method is transforming point cloud to 3D voxel grids, and then processing them with 3D convolution [17,18,23]. For the VoxelNet [17] and SECOND [18], both are one-stage detectors, a simplified PointNet [24] is applied to each non-empty voxel for extracting voxel-wise feature. After that, the entire point cloud is represented as a four-dimensional (4D) tensor. Then the 4D tensor is processed by a series of 3D convolutional layers, followed by region proposal network (RPN) [25], where RPN is used to predict the classification score and the bounding box regression map. In this paper, we use voxelization to convert point clouds into regular data for 3D object detection.

2.3. Multimodal Fusion-Based Detection

Several 3D object detection techniques [26–31] used a combination of RGB image and depth map. For example, Liu et al. [27] used convolutional neural networks (CNN) to extract color features from RGB image, and used convolutional deep belief networks (CDBN) to extract geometric features from depth map. Then, advanced visual features and geometric features were extracted with deep belief networks (DBN). Next, the learned features were fused to obtain a 3D multimodality feature for object detection. Deng et al. [28] used CNN to extract the appearance and geometric features from RGB and depth images respectively, and obtained 2D detection results from RGB image. Then these 2D bounding boxes were combined with geometric features, and classification results were converted into 3D space. Finally, the bounding box regression was used to refine these 3D boxes. Slightly different from [28], Luo et al. [29] concatenated the appearance and geometric features directly, and then the concatenated features were used for determining the final detection results. These methods have a large amount of computation cost, which leads to a slow detection speed, since these methods used different branches to extract the appearance features and geometric features, respectively.

In addition, some methods [5,15,32–34] fused RGB image and point cloud for 3D object detection. Typically, MV3D [5] converted point cloud into a BEV representation. To obtain more scene information, a BEV and FV of point cloud were fed into the detection network with RGB image. 3D candidate proposals were obtained from BEV of point cloud with CNN, since BEV suffers less occlusion. Then, candidate proposals were projected to FV and RGB image. Finally, features learned from the three 2D views were fused for object classification and bounding box regression. F-PointNet [15] is a two-stage

3D object detector that used RGB image to detect in the first stage, resulting in 2D detection boxes. In the second stage, the 2D detection results were projected into point cloud to form point cloud frustum, which was segmented by PointNet [24]. Finally, the 3D bounding boxes were calculated. Obviously, the prediction of 3D bounding box in these studies relies heavily on the 2D region proposal network, although they have achieved better detection performance. Different from these approaches, we only use point cloud data to achieve higher accuracy of 3D object detection.

3. Method

In this section, the network proposed is introduced, which is named 3D-GIoU. The whole detection network is shown in Figure 1, which includes four main components: (1) Data Preprocessing; (2) Point-Voxel Feature Encoder; (3) Sparse Convolution Middle Layers; (4) Region Proposal Network, which consist of Backbone Network and Header Network.

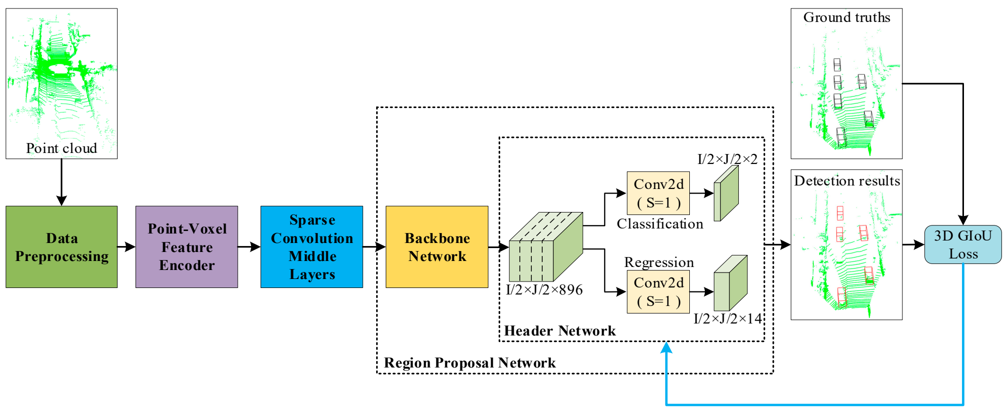


Figure 1. Three-Dimensional Generalized Intersection over Union (3D-GIoU) Architecture. The network takes point cloud as input. After the point cloud is discretized into 3D voxel grids, Point-Voxel Feature Encoder is used to learn voxel-wise features. Then, these features are processed by Sparse Convolution Middle Layers and sent to the Region Proposal Network to predict the classification score and the bounding box regression map. Last, the detection results and ground truth bounding boxes are used to calculate 3D GIoU loss, and 3D GIoU loss is used for optimizing the bounding box regression.

3.1. Data Preprocessing

Since 3D point cloud is irregular and the data input to CNN must be regular, point cloud is firstly transformed into regular data by discretizing them into 3D voxel grids. For a given point cloud, we only deal with a point cloud scene of size $L \times W \times H$ m^3 in the directions of X, Y and Z axes, and points beyond this range are discarded. In addition, according to the coordinate transformation matrix of the LIDAR to the left camera in KITTI benchmark [16], we remove points beyond the field of view of left camera. Then, the cropped point cloud is discretized into voxels of size D_x, D_y, and D_z along the three coordinate axes. Therefore, a total of $I \times J \times K$ voxels are obtained, where $I = L/D_x$, $J = W/D_y$, and $K = H/D_z$. The voxelization process of point cloud is shown in Figure 2.

A cropped 3D point cloud contains about 17,000 points and is unevenly distributed, which may bias the detection. To address this, a fixed number of points N in each voxel are obtained to alleviate the sampling deviation between voxels. Specifically, when the number of points in a voxel is greater than N, N points will be obtained by random down-sampling. On the contrary, points with all 0 are filled to the voxel when the number is less than N.

3.2. Point-Voxel Feature Encoder

Same to the previous work, PVFE [35] is used to obtain a voxel feature with high expressive ability. PVFE is composed of two full connection layers and one max-pooling layer. To be specific, each full connection layer is followed by Batch Normalization (BN) and Rectified Linear Unit (ReLU).

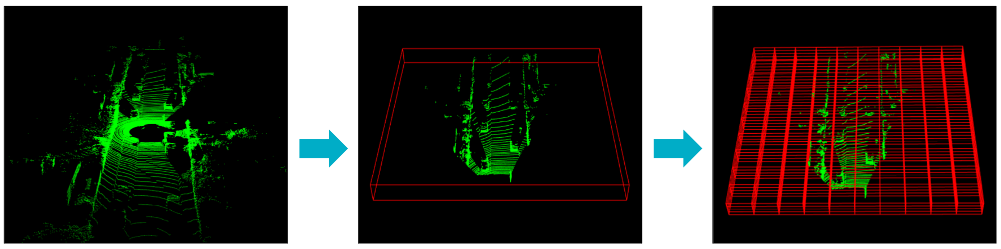


Figure 2. Voxelization of point cloud. Firstly, the original point cloud is cropped according to the fixed size $L \times W \times H\ m^3$, and then the cropped point cloud is further transformed into 3D voxel grids.

In details, points in voxel are input to PVFE in sequence. Here, we assume that there are M points ($M \leq N$) in the voxel A, represented as point set $P = \{p_1, p_2, \cdots, p_M\} \in R^4$. For point p_i ($i = 1, 2, \cdots, M$), to obtain an expressive feature, it is necessary to comprehensively consider its own information, the spatial relationship with other points p_j ($j \neq i$), and the spatial relationship with voxel A. Therefore, the feature of each point is encoded as a (ten-dimensional) 10D vector $f = (x, y, z, r, x - \Delta x, y - \Delta y, z - \Delta z, x - x_c, y - y_c, z - z_c)$, where, (x, y, z) are the coordinates of p_i, r is the received reflectance, $(\Delta x, \Delta y, \Delta z)$ are the mean coordinates of all unfilled points in the voxel A, and (x_c, y_c, z_c) represents the center of the voxel A. Then, point features $F = \{f_1, f_2, \cdots, f_M\} \in R^{10}$ in each voxel are fed into PVFE, and then a 128-D voxel-wise feature is outputted. Consequently, the whole point cloud is mapped into a 4D tensor with a shape of $I \times J \times K \times 128$.

3.3. Sparse Convolution Middle Layers

Since the output tensor of PVFE has high dimensionality, computational efficiency becomes the major problem. To improve the efficiency of 3D CNN and make a more objective comparison with SECOND [18], we inherit the sparse convolutional middle layers (SCML) of SECOND [18]. SCML is used to process the voxel-wise features learned by PVFE, it achieves small computational cost with a certain number of parameters. SCML ensures that the output spatial structure remains unchanged while improving the data processing speed, the details of SCML can be referred to in [18].

3.4. Region Proposal Network

Region Proposal Network (RPN) [25] is an important part of the 3D object detection framework. The RPN proposed in this paper is composed of two parts: The Backbone Network and the Header Network. The Backbone Network consists of three components: top-bottom, bottom-top, and multiscale fusion. The structure of the Backbone Network is shown in Figure 3, where the size of tensors in the Figure 3 is marked according to the parameters of the car detection.

The top-bottom branch consists of three convolution blocks, which are named CB_1, CB_2, and CB_3 in turn. More specifically, CB_1 is composed of four convolutional layers, while CB_2 and CB_3 are both composed of six convolutional layers. Each convolution layer is followed by BN and ReLU. In addition, for car detection, the stride of the first convolution layer in CB_1, CB_2, and CB_3 is 2, and the stride of other convolution layers is 1. For cyclist and pedestrian, only the stride of the first convolutional layer in CB_2 and CB_3 is 2. The bottom-top branch consists of three deconvolution layers with a stride of 2, which are represented as three yellow lines in Figure 3. In addition, the 2D convolution of the blue line, the deconvolution layers indicated by the purple lines, and the concatenation of the last step constitute the multiscale fusion structure.

The input of the Backbone Network is a spatial feature with shape of $I \times J \times 128$, which is the output of SCML. The output of the Backbone Network is a multichannel feature map with size of $I/2 \times J/2 \times 896$, which incorporates multiscale features. Because the concatenated feature aggregates more detailed texture features and richer semantic information, the expression ability is stronger, which is important for predicting high-precision 3D bounding boxes. Finally, the output is fed into the Header Network to predict the classification score and bounding box regression map.

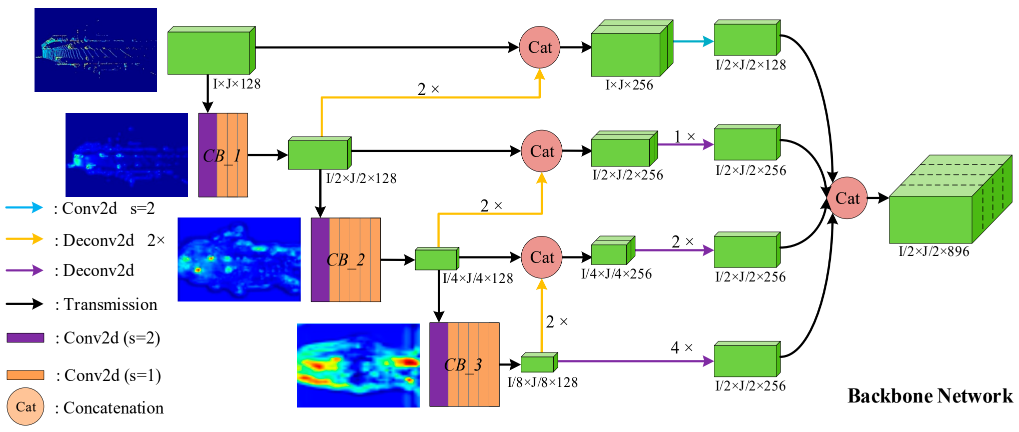


Figure 3. The architecture of Backbone Network. The meanings of lines and two-dimensional (2D) shapes with different colors in figure are given in the legend. Green 3D boxes represent feature maps with different sizes.

Different levels of feature maps examples are given in Figure 3. As shown in Figure 3, the four feature maps from top to bottom represent the input of the Backbone Network, the output of CB_1, the output of CB_2, and the output of CB_3, respectively. Obviously, the degree of abstraction of features deepens as the network hierarchy deepens, which means that the feature map contains more semantic information.

4. Loss Function

The loss of object detection pipeline proposed in this paper consists of three parts: (1) Classification loss; (2) bounding box regression loss; (3) 3D GIoU loss. In addition, to balance the relative importance, we add different weights to different parts. As shown in Equation (1), where $w_1 = 1$, $w_2 = 2$, $w_3 = 1$.

$$L = w_1 L_{cls} + w_2 L_{reg} + w_3 L_{3D_{GIoU}}, \tag{1}$$

4.1. Classification Loss

Since most of the bounding boxes predicted by RPN belong to negative samples, there is a large imbalance between positive samples and negative samples. This deviation makes the negative loss far greater than the positive loss during training, which is not conducive to the training of network. Therefore, the focal loss function proposed by Lin et al. [36] is adopted to obtain an effective pipeline, as shown in Equation (2).

$$L_{cls} = FL(p_t) = -\alpha_t (1 - p_t)^r \log(p_t), \tag{2}$$

Specifically, p_t represents the evaluation probability value of the model for the corresponding bounding box, and the scale factor is set as $\alpha_t = 0.25$, $\gamma = 2$. Essentially, the focal loss function is a dynamically scaled cross entropy loss. When the confidence of the correct class increases, the weight $(1 - p_t)^\gamma$ decays to zero. On the contrary, the weight increases.

4.2. 3D Bounding Box Regression Loss

For the bounding box regression, the 3D ground truth bounding box is parameterized as $(x_g, y_g, z_g, l_g, w_g, h_g, \theta_g)$ while the matching anchor is $(x_a, y_a, z_a, l_a, w_a, h_a, \theta_a)$, where (x, y, z) denote the center coordinate, (l, w, h) is the length, width, and height of the 3D box, and θ is the yaw rotation around Z axis. Then, we define vector $r^* \in R^7$, which encodes the regression targets. Finally, r^* is computed as:

$$\begin{aligned} &x_r = \frac{x_g - x_a}{d_a},\ y_r = \frac{y_g - y_a}{d_a},\ z_r = \frac{z_g - z_a}{h_a}, \\ &l_r = \log\left(\frac{l_g}{l_a}\right),\ w_r = \log\left(\frac{w_g}{w_a}\right),\ h_r = \log\left(\frac{h_g}{h_a}\right), \\ &\theta_r = \theta_g - \theta_a, \end{aligned} \tag{3}$$

Then, bounding box regression loss is defined as following:

$$L_{reg} = \sum\nolimits_{e \in (x,y,z,l,w,h)} \text{SmoothL1}(e_r) + \text{SmoothL1}(\sin\theta_r), \tag{4}$$

4.3. 3D GIoU Loss

Currently, the regression loss of the bounding box (mean squared error loss, l_1-smooth loss) is the mainstream method to optimize the bounding box in the object detection. However, IoU is the most commonly used metric for comparing the similarity between two arbitrary shapes, which is also known as Jaccard index. In fact, two shapes can overlap in different ways to get the same l_1 or l_2-norms values, but when they overlap in different ways, the IoU value is different [20], which indicates that the l_1 and the l_2-norms cannot effectively reflect the detection effect. However, IoU not only reflects the distance between the predicted and ground truth bounding box, but also has scale invariance. Therefore, some object detection techniques [37,38] adopt IoU loss to optimize the bounding box. Here, given two arbitrary shapes $A, B \in R^n$, the IoU and IoU loss are defined follows:

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|}, \tag{5}$$

$$L_{IoU} = 1 - \text{IoU}, \tag{6}$$

It cannot be neglected that there are two shortcomings of IoU loss in optimizing bounding box:

(1) When the predicted and ground truth bounding box do not coincide completely, the gradient of loss function is 0, which makes it impossible to optimize;
(2) Two shapes can overlap in different ways to get the same IoU value, that is, the IoU does not reflect how overlap between two objects occurs (see Figure 4).

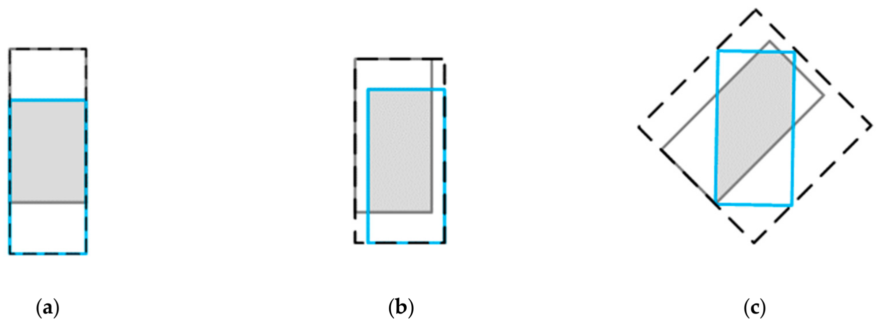


Figure 4. Three different ways of overlap between two rectangles with the exact same IoU values, (i.e., IoU = 0.50), but different GIoU values (i.e., from the left to right GIoU = 0.50, 0.45, and 0.09, respectively). For the case with better aligned orientation, the GIoU value will be higher.

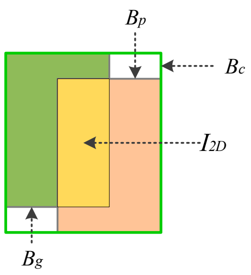
To solve these issues, Hamid et al. proposed a 2D GIoU loss for optimizing the bounding box in 2D object detection [20]. Motivated by [20], we propose a 3D GIoU loss function for 3D object detection, which contributes to align 3D predicted and ground truth bounding boxes.

In this paper, the optimization of the bounding box adopts two losses of l_1- smooth and 3D GIoU. In particular, l_1-smooth is firstly used to optimize all 3D bounding boxes, and then the 3D GIoU is used to optimize those bounding boxes that are judged to be positive samples. The algorithm of 3D GIoU loss is defined as Algorithm 1.

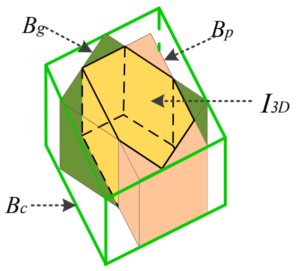
Algorithm 1: 3D Generalized Intersection Over Union Loss

```
Input: The information of the predicted $B_p$ and ground truth $B_g$ bounding box:
$B_p = (x_p, y_p, z_p, l_p, w_p, h_p, \theta_p)$,    $B_g = (x_g, y_g, z_g, l_g, w_g, h_g, \theta_g)$
Output: $L_{3D\_GIoU}$
1.  Calculating projections $B'_p$ and $B'_g$ of box $B_p$ and $B_g$ on the bird's eye view, respectively.
    $B'_p = (x^1_p, y^1_p, x^2_p, y^2_p, \theta'_p)$,    $B'_g = (x^1_g, y^1_g, x^2_g, y^2_g, \theta'_g)$
2.  $B'_c = (x^1_c, y^1_c, x^2_c, y^2_c, \theta'_c) \leftarrow$ the information of smallest enclosing box;
3.  $A_p \leftarrow$ the area of the 2D box $B'_p$;
4.  $A_g \leftarrow$ the area of the 2D box $B'_g$;
5.  $A_c \leftarrow$ the area of the 2D box $B'_c$;
6.  $I_{2D} \leftarrow$ intersection between $B'_p$ and $B'_g$;
7.  $U_{2D} \leftarrow$ union between $B'_p$ and $B'_g$;
8.  $I_h \leftarrow$ the height of the intersection of $B_p$ and $B_g$;
9.  $U_h \leftarrow$ the height of the union of $B_p$ and $B_g$;
10. $V_p \leftarrow$ the volume of the 3D box $B_p$;
11. $V_g \leftarrow$ the volume of the 3D box $B_g$;
12. $V_c \leftarrow$ the volume of the 3D box $B_c$, where $B_c$ represents the smallest 3D enclosing box;
13. Calculating intersection $I_v$ of $B_p$ and $B_g$:
    if $I_{2D} \leq 0$ :
        $I_v = 0$;
    else:
    if $I_h \leq 0$:
        $I_v = 0$;
          else:
              $I_v = I_{2D} \times I_h$;
14. $IoU_{3D} = \frac{I_v}{U_v}$, where $U_v = V_p + V_g - I_v$;
15. $GIoU_{3D} = IoU_{3D} - \frac{(V_c - U_v)}{V_c}$;
16. $L_{3D\_GIoU} = 1 - GIoU_{3D}$;
```

To better understand the calculation of the smallest enclosing box B_c in 3D GIoU, we give 2D and 3D examples in (a) and (b) of Figure 5.



(**a**) 2D



(**b**) 3D

Figure 5. Different ways of overlap between bounding boxes in case of 2D and 3D, respectively. For (**a**) and (**b**), cyan and pink represent the predicted bounding box B_p and ground truth B_g, respectively, and yellow represents the intersection of them. In addition, the green bounding box represents the smallest enclosing box B_c.

5. Experiments

The KITTI benchmark dataset [16] was employed to evaluate our proposed method. It contains 7481 training and 7518 testing point clouds, including three categories: car, cyclist, and pedestrian. The training dataset was divided into a training set (3712) and a validation set (3769), since the ground truth of the testing dataset is not publicly available.

5.1. Network Details

5.1.1. Car Detection

For the car detection task, the range of point cloud taken into consideration was $L \times W \times H = [0, 70.4] \times [-40, 40] \times [-3, 1]$ m^3 along X, Y, and Z axis, respectively. The 3D voxel dimension was set to be $D_x \times D_y \times D_z = 0.2 \times 0.2 \times 0.4$ m^3, which led to $I \times J \times K = 352 \times 400 \times 10$. In addition, N = 35 was set as the maximum number of points for random down-sampling within the voxel. Following the SECOND [18], the set of anchors was a 3D box with measurement of $l \times w \times h = 3.9 \times 1.6 \times 1.56$ m^3, which is the mean size of car and centered at $z = -1$m. As to the orientation, $\theta = 0^{\circ}$ or $\theta = 90^{\circ}$ was considered in our experiments.

5.1.2. Cyclist and Pedestrian Detection

For cyclist and pedestrian detection, the range of point cloud was set to $L \times W \times H = [0, 48] \times [-20, 20] \times [-2.5, 0.5]$ m^3, and the size of 3D voxel was $D_x \times D_y \times D_z = 0.2 \times 0.2 \times 0.3$ m^3, which led to $I \times J \times K = 240 \times 200 \times 10$. As with car detection, N = 35 was taken for random down-sampling of points in voxels. For the detection of cyclist, the set of anchors was a 3D box with size $l \times w \times h = 1.76 \times 0.6 \times 1.73$ m^3, while the size was $l \times w \times h = 0.8 \times 0.6 \times 1.73$ m^3 for pedestrian. Besides, all the anchors were centered at $z = -0.6$ m.

5.2. Training

In the experiments of this paper, there were only 3712 point clouds in the training set, which would inevitably lead to the overfitting of our network. To solve this problem, we introduced three different forms of data augmentation in SECOND [18]: (1) Motion; (2) global scaling and rotation; (3) sample ground truths from the database. The proposed framework was trained for 200k iterations using the Adam optimizer [39]. The initial learning rate was 0.002, the exponential decay rate was 0.8, and there was a decay every 18,750 iterations. For the detection of car, cyclist, and pedestrian, the batch size of 3 was used, distributed on a GTX 2080 Ti GPU, and the whole network took about 22 h to train.

5.3. Comparisons on the KITTI Validation Set

The 3D detection performance of our network on the KITTI verification set is shown in Table 1. In order to demonstrate the superior performance of the proposed detector, we compared it with other detectors, such as the MV3D [5], AVOD [33] and F-PointNet [15] which used both RGB image and point cloud, and VoxelNet [17], SECOND [18], PointPillars [40], and PVFE [35] which only used point cloud. Besides, the performance of our method for BEV object localization is given in Table 2.

Compared with 2D object detection, 3D object detection is more challenging, since it requires higher localization accuracy of 3D bounding box in space. As shown in Table 1, we can see that 3D-GIoU proposed in this paper is more suitable for 3D object detection. Specifically, for car and cyclist, 3D-GIoU significantly outperformed other approaches across all difficulty levels. Moreover, in the cyclist detection, our method achieved an AP of 63.51% in the Hard level, which is 7.72% higher than the result of SECOND [18]. In addition, in the performance for BEV object localization, which is shown in Table 2, 3D-GIoU achieved better results with respect to AP compared with other methods, although it was slightly inferior in pedestrian detection task.

In addition, as shown in Figure 6, it is easy to find that the AP of 3D-GIoU was significantly higher than that of other methods across three difficulty levels. In addition, the AP of 3D-GIoU decreased more slowly with the difficulty level from Easy to Moderate, then to Hard, which further demonstrates that our network has better robustness.

Table 1. 3D detection performance: Average precision (AP) (%) for 3D box in the KITTI valuation set.

Method	**Modality**	**Car**			**Cyclist**			**Pedestrian**		
		Easy	**Mod.**	**Hard**	**Easy**	**Mod.**	**Hard**	**Easy**	**Mod.**	**Hard**
MV3D	Img. & Lidar	71.09	62.35	55.12	N/A	N/A	N/A	N/A	N/A	N/A
AVOD	Img. & Lidar	81.94	71.88	66.38	64.00	52.18	46.61	50.80	42.81	40.88
F-PointNet	Img. & Lidar	81.20	70.39	62.19	71.96	56.77	50.39	51.21	44.89	51.21
VoxelNet	Lidar	77.47	65.11	57.73	61.22	48.36	44.37	39.48	33.69	31.50
PointPillars	Lidar	86.96	76.35	70.19	77.75	58.55	54.85	67.07	58.74	55.97
PVFE	Lidar	87.32	77.12	68.87	81.58	62.41	56.33	58.48	51.74	45.09
SECOND	Lidar	85.99	75.51	68.25	80.47	57.02	55.79	56.99	50.22	43.59
3D-GIoU	Lidar	87.83	77.91	75.55	83.32	64.69	63.51	67.23	59.58	52.69

Table 2. Bird's Eye View (BEV) detection performance: AP (%) for BEV box in the KITTI valuation set.

Method	**Modality**	**Car**			**Cyclist**			**Pedestrian**		
		Easy	**Mod.**	**Hard**	**Easy**	**Mod.**	**Hard**	**Easy**	**Mod.**	**Hard**
MV3D	Img & Lidar	86.02	76.90	68.48	N/A	N/A	N/A	N/A	N/A	N/A
AVOD	Img & Lidar	88.53	83.79	77.90	68.09	57.48	50.77	58.75	51.05	47.54
F-PointNet	Img & Lidar	88.07	84.00	75.33	75.38	61.96	54.68	58.09	50.22	47.02
PIXOR	Lidar	89.38	83.70	77.97	N/A	N/A	N/A	N/A	N/A	N/A
VoxelNet	Lidar	89.35	79.26	77.39	66.07	54.76	50.55	46.13	40.74	38.11
PointPillars	Lidar	90.12	86.67	84.53	80.89	61.54	58.63	73.08	68.20	63.20
PVFE	Lidar	89.98	87.03	79.31	84.30	64.72	58.42	61.93	54.88	51.93
SECOND	Lidar	89.23	86.25	78.95	82.88	63.46	57.63	60.81	53.67	51.10
3D-GIoU	Lidar	90.16	87.92	86.55	85.35	66.91	65.06	70.16	62.57	55.52

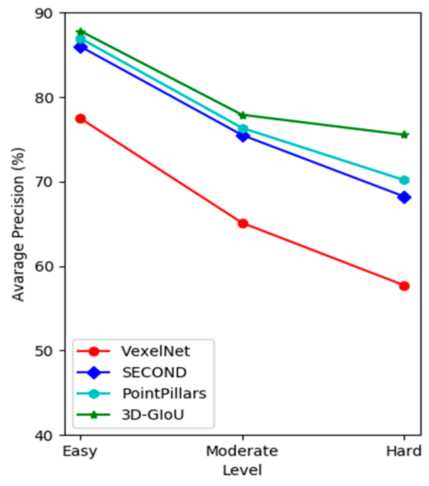
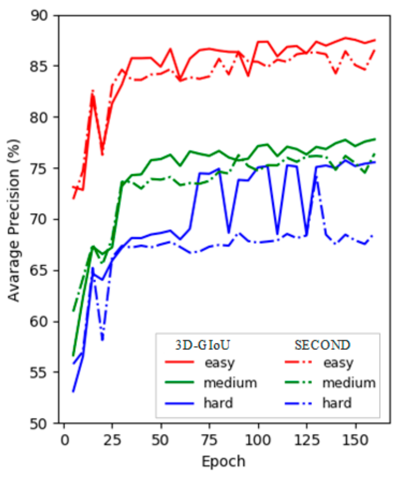
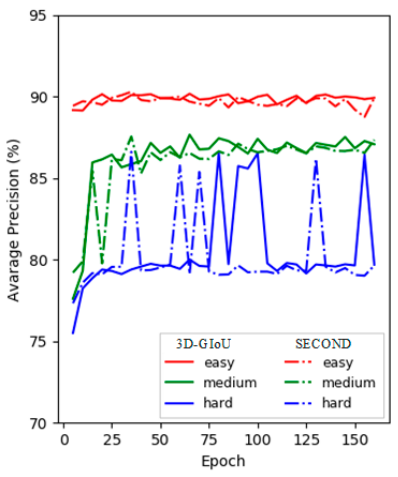


Figure 6. The AP of different methods on KITTI validation set with the different difficulty level (car detection).

In order to compare the performance between our structure and the basic network SECOND [18] more intuitively, the training detection results on the KITTI verification set with 3D and BEV are shown in Figure 7. As shown in Figure 7, the 3D detection performance of 3D-GIoU significantly outperformed that of SECOND [18], although the BEV detection performance was not visually different from SECOND [18].



(**a**) 3D



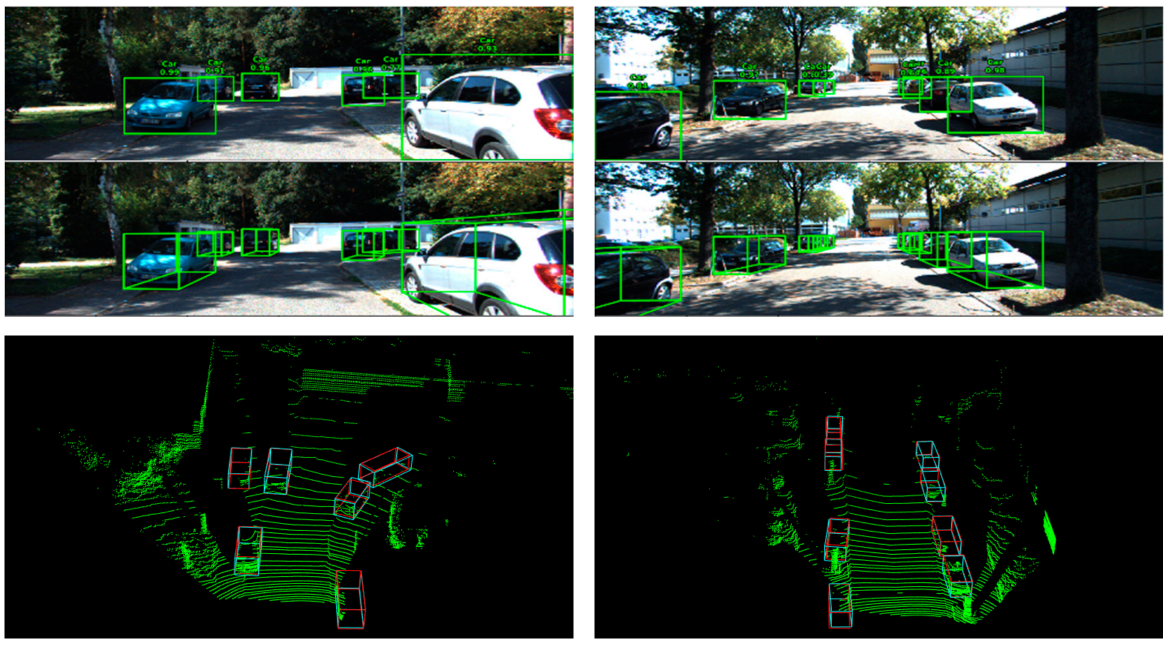
(**b**) BEV

Figure 7. 3D-GIoU vs. SECOND for the detection performance of 3D and BEV evaluation on the KITTI validation set across three difficulty levels (i.e., Easy, Moderate and Hard). In (**a**) and (**b**), the solid line represents 3D-GIoU, while the dotted line represents SECOND.

Additionally, we can see from Figure 7 that the AP of our architecture was significantly lower than SECOND [18] at the beginning of training. However, after training about 15 epochs, the performance of our network reached the level of SECOND [18] with both of 3D and BEV, and then far exceeded SECOND [18]. Obviously, the results fully demonstrate that our structure is easier to train.

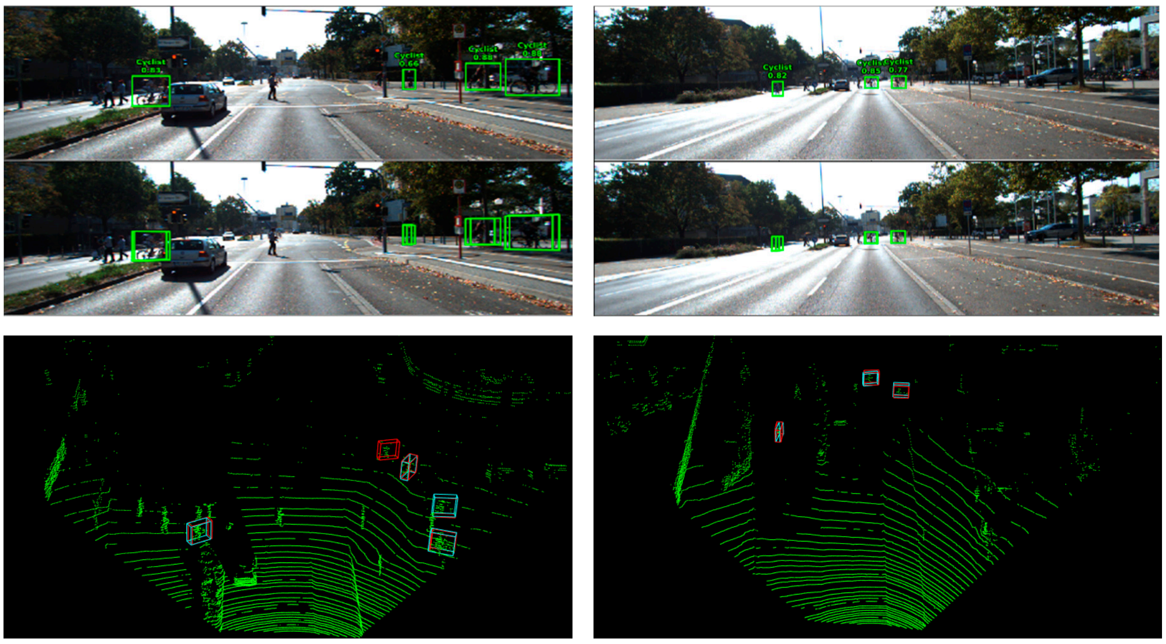
5.4. Analysis of the Detection Results

Some detection results on the KITTI validation set of our network are shown in Figure 8. As shown in the RGB images in Figure 6, 3D bounding boxes were also projected into corresponding images of the point cloud, resulting in 2D bounding boxes and 3D bounding boxes on the image.

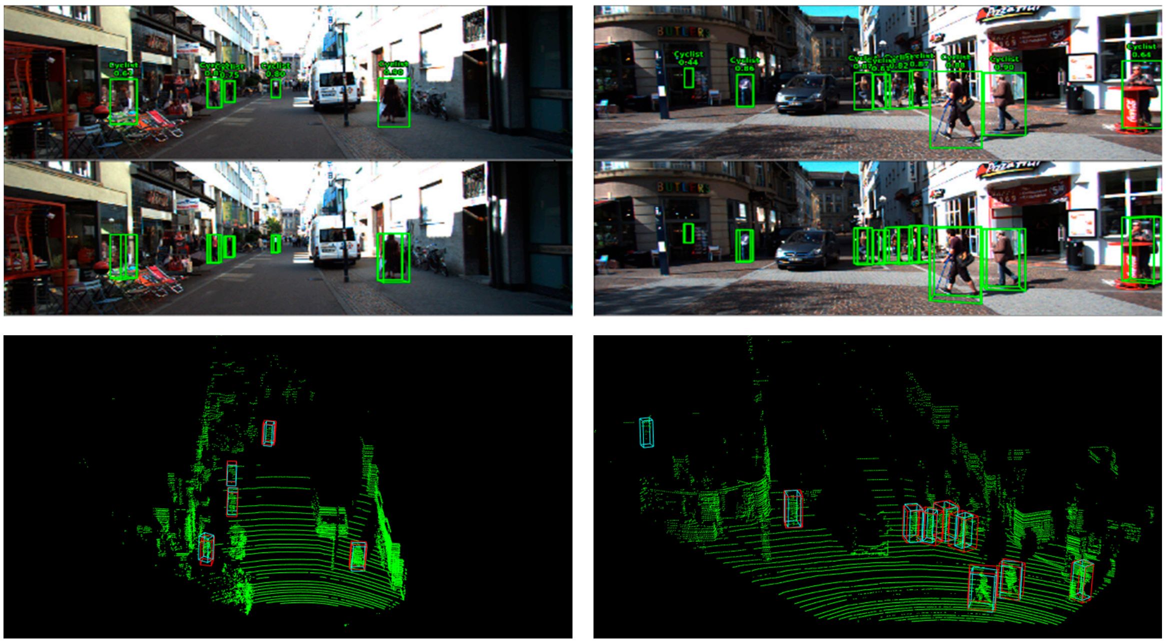


(**a**) Car

Figure 8. *Cont.*



(**b**) Cyclist



(**c**) Pedestrian

Figure 8. Several 3D detection results on the KITTI validation set. In each RGB image, all the 2D and 3D bounding boxes represent the detection results. The digit and word beside each 2D box represent the instance score and class. In the point clouds, teal 3D boxes indicate detection results, and 3D red boxes represent ground truths.

5.4.1. Car Detection

The four images and the associated point clouds in Figure 8a are shown as typical car detection examples. Whether it is a long- or close-distance car, our network can achieve superior detection results, even if the available points belonging to a long-distance car are few. Furthermore, the proposed network can successfully detect highly occluded cars, which is a great challenge task for other networks. Consequently, these results show that the proposed network is suitable for 3D car detection.

5.4.2. Cyclist and Pedestrian Detection

The images and the associated point clouds in Figure 8b and c show the detection results of cyclists and pedestrians, respectively. It is easy to find that there were more detection errors than cars. The causes of these errors can be summarized into three points. Firstly, compared with cars, there were relatively few instances of cyclists and pedestrians in the training set, which led to insufficient training of the network. Secondly, the size of cyclists and pedestrians was smaller, that is to say, each instance contained fewer points, which made it easier to confuse with other objects with similar size. Thirdly, the positioning of the 3D bounding box of some successfully detected objects was not precise enough, which was mainly reflected in the rotation angle. Therefore, how to filter out the unrelated points, improve the object recall rate, and give more accurate 3D bounding box is a research focus of cyclist and pedestrian detection.

5.5. Ablation Studies

To prove the effectiveness of the 3D GIoU loss and Backbone Network proposed in this paper, we have done some ablation experiments on the KITTI validation set, and the results are summarized in Table 3. In particular, Baseline 1 represents the corrected SECOND [18], which adds the 3D GIoU loss. Correspondingly, Baseline 2 represents replacing the RPN in SECOND [18] with the RPN proposed in this article, which is composed of the Backbone Network and the Header Network.

Table 3. 3D and BEV detection performance: AP (%) on the KITTI valuation set.

Method	**Method**	**Car**			**Cyclist**			**Pedestrian**		
		Easy	**Mod.**	**Hard**	**Easy**	**Mod.**	**Hard**	**Easy**	**Mod.**	**Hard**
3D	SECOND	85.99	75.51	68.25	80.47	57.02	55.79	56.99	50.22	43.59
	Baseline 1	87.20	76.80	74.65	82.84	62.34	56.66	58.16	51.42	44.74
	Baseline 2	87.62	77.37	75.53	83.89	64.27	62.75	59.37	52.42	49.78
	3D-GIoU	87.83	77.91	75.55	83.32	64.69	63.51	67.23	59.58	52.69
BEV	SECOND	89.23	86.25	78.95	82.88	63.46	57.63	60.81	53.67	51.10
	Baseline 1	89.99	86.82	86.03	84.83	64.56	58.55	62.34	59.35	52.70
	Baseline 2	89.80	87.13	86.31	85.42	65.78	64.45	66.40	59.40	52.56
	3D-GIoU	90.16	87.92	86.55	85.35	66.91	65.06	70.16	62.57	55.52

According to Table 3, we can make the following comparison and get the corresponding conclusion:

(1) Comparing Baseline 1 with SECOND [18], it is easy to find that the proposed 3D GIoU loss can improve detection performance. In particular, the AP of the Hard level was increased by 6.4%.
(2) Comparing Baseline 2 with SECOND [18], we can find that the use of the proposed Backbone Network improved the detection performance in Hard level by 7.28%.
(3) By comparing 3D-GIoU with Baseline 1, Baseline 2, and SECOND [18], it is not difficult to find that when the 3D GIoU loss and Backbone Network are used simultaneously, the performance of 3D object detection is greatly improved.

6. Conclusions

In this paper, a Backbone Network and a 3D GIoU loss function are proposed for 3D object detection in point cloud. Backbone Network can effectively combine detail texture features in low-level feature maps with semantic features in high-level feature maps, and 3D GIoU loss can significantly improve the localization accuracy of 3D box. A large number of experiments have been carried out on the public KITTI benchmark, and our module has achieved excellent results, which fully demonstrate that the proposed structure is suitable for 3D object detection in point cloud.

Author Contributions: Supervision, S.H.; Writing—original draft, J.X.; Writing—review & editing, Y.M. and J.Z.

Funding: This work was supported by the National Natural Science Foundation of China (No. 61471371).

Acknowledgments: The authors would like to thank the support of the laboratory, university and government.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chao, M.; Yulan, G.; Yinjie, L.; Wei, A. Binary Volumetric Convolutional Neural Networks for 3-D Object Recognition. *IEEE Trans. Instrum. Meas.* **2019**, *68*, 38–48.
2. Chao, M.; Yulan, G.; Jungang, Y.; Wei, A. Learning Multi-view Representation with LSTM for 3D Shape Recognition and Retrieval. *IEEE Trans. Multimed.* **2019**, *21*, 1169–1182.
3. Ankit, K.; Ozan, I.; Peter, O.; Mohit, I.; James, B.; Ishaan, G.; Victor, Z.; Romain, P.; Richard, S. Ask Me Anything Dynamic Memory Networks for Natural Language Processing. *arXiv* **2015**, arXiv:1506.07285.
4. Alexis, C.; Holger, S.; Yann Le, C.; Loïc, B. Deep Convolutional Networks for Natural Language Processing. *arXiv* **2018**, arXiv:1805.09843.
5. Chen, X.; Ma, H.; Wan, J.; Li, B.; Xia, T. Multi-View 3D Object Detection Network for Autonomous Driving. In Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; Volume 1, pp. 1907–1915.
6. Chen, X.; Kundu, K.; Zhang, Z.; Ma, H.; Fidler, S.; Urtasun, R. Monocular 3d Object Detection for Autonomous Driving. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 2147–2156.
7. Li, B.; Ouyang, W.; Sheng, L.; Zeng, X.; Wang, X. GS3D: An Efficient 3D Object Detection Framework for Autonomous Driving. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long beach, CA, USA, 15–21 June 2019; pp. 1019–1028.
8. Guan, P.; Ulrich, N. 3D Point Cloud Object Detection with Multi-View Convolutional Neural Network. In Proceedings of the IEEE Conference on International Conference on Pattern Recognition (ICPR), Cancun, Mexico, 4–8 December 2016; pp. 2040–2049.
9. Zeng, Y.; Hu, Y.; Liu, S.; Ye, J.; Han, Y.; Li, X.; Sun, N. RT3D: Real-Time 3-D Vehicle Detection in LiDAR Point Cloud for Autonomous Driving. *IEEE Robot. Autom. Lett.* **2018**, *3*, 3434–3440. [CrossRef]
10. François, P.; Francis, C.; Roland, S. *A Review of Point Cloud Registration Algorithms for Mobile Robotics*; Foundations and Trends®in Robotics, Mike Casey: Boston, MA, USA, 2015; Volume 4, pp. 1–104.
11. Boyoon, J.; Sukhatme, G.S. Detecting Moving Objects Using a Single Camera on a Mobile Robot in an Outdoor Environment. In Proceedings of the 8th Conference on Intelligent Autonomous Systems, Amsterdam, The Netherlands, 10–13 March 2004; pp. 980–987.
12. Lavanya, S.; Nirvikar, L.; Dileep, K.Y. A Study of Challenging Issues on Video Surveillance System for Object Detection. *J. Basic Appl. Eng. Res.* **2017**, *4*, 313–318.
13. Khan, M.; Jamil, A.; Zhihan, L.; Paolo, B.; Po, Y.; Sung, W. Efficient Deep CNN-Based Fire Detection and Localization in Video Surveillance Applications. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *49*, 1419–1434.
14. Cheng-bin, J.; Shengzhe, L.; Trung, D.D.; Hakil, K. Real-Time Human Action Recognition Using CNN Over Temporal Images for Static Video Surveillance Cameras. In *Advances in Multimedia Information Processing—PCM 2015*; Springer: Cham, Switzerland, 2015; pp. 330–339.
15. Qi, C.R.; Liu, W.; Wu, C.; Su, H.; Guibas, L.J. Frustum PointNets for 3D Object Detection from RGB-D Data. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 918–927.
16. Kitti 3D Object Detection Benchmark Leader Board. Available online: http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d (accessed on 28 April 2018).
17. Zhou, Y.; Tuzel, O. 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), Salt Lake City, UT, USA, 18–22 June 2018; pp. 4490–4499.
18. Yan, Y.; Mao, Y.; Li, B. SECOND: Sparsely Embedded Convolutional Detection. *Sensors* **2018**, *18*, 3337. [CrossRef] [PubMed]
19. Simon, M.; Milz, S.; Amende, K.; Gross, H.M. Complex-YOLO: An Euler-Region-Proposal for Real-Time 3D Object Detection on Point Clouds. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 197–209.

20. Hamid, R.; Nathan, T.; JunYoung, G.; Amir, S.; Ian, R.; Silvio, S. Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 658–666.
21. Yang, B.; Luo, W.; Urtasun, R. PIXOR: Real-time 3D Object Detection from Point Clouds. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7652–7660.
22. Li, B.; Zhang, T.; Xia, T. Vehicle detection from 3D lidar using fully convolutional network. *arXiv* **2016**, arXiv:1608.07916.
23. Engelcke, M.; Rao, D.; Wang, D.Z.; Tong, C.H.; Posner, I. Vote3deep: Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 1355–1361.
24. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 77–85.
25. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 24–27 June 2014; pp. 580–587.
26. Kiwoo, S.; Youngwook Paul, K.; Masayoshi, T. RoarNet: A Robust 3D Object Detection based on RegiOn Approximation Refinement. *arXiv* **2018**, arXiv:1811.03818.
27. Liu, W.; Ji, R.; Li, S. Towards 3D Object Detection with Bimodal Deep Boltzmann Machines over RGBD Imagery. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 3013–3021.
28. Zhuo, D.; Londin, J.L. Amodal Detection of 3D Objects: Inferring 3D Bounding Boxes from 2D Ones in RGB-Depth Images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 5762–5770.
29. Qianhui, L.; Huifang, M.; Yue, W.; Li, T.; Rong, X. 3D-SSD: Learning Hierarchical Features from RGB-D Images for Amodal 3D Object Detection. *arXiv* **2015**, arXiv:1711.00238.
30. Song, S.; Xiao, J. Deep Sliding Shapes for Amodal 3D Object Detection in Rgb-d Images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 808–816.
31. Ling, M.; Yang, B.; Wang, S.; Raquel, U. Deep Continuous Fusion for Multi-Sensor 3D Object Detection. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 641–656.
32. Huitl, R.; Schroth, G.; Hilsenbeck, S.; Schweiger, F.; Steinbach, E. TUMindoor: An Extensive Image and Point Cloud Dataset for Visual Indoor Localization and Mapping. In Proceedings of the IEEE International Conference on Image Processing, Orlando, FL, USA, 30 September–3 October 2012.
33. Ku, J.; Mozifian, M.; Lee, J.; Harakeh, A.; Waslander, S. Joint 3D Proposal Generation and Object Detection from View Aggregation. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Madrid, Spain, 1–5 October 2018; pp. 1–8.
34. Li, M.; Hu, Y.; Zhao, N.; Qian, Q. One-Stage Multi-Sensor Data Fusion Convolutional Neural Network for 3D Object Detection. *Sensors* **2019**, *19*, 1434. [CrossRef] [PubMed]
35. Xu, J.; Ma, Y.; He, S.; Zhu, J.; Xiao, Y.; Zhang, J. PVFE: Point-Voxel Feature Encoders for 3D Object Detection. In Proceedings of the IEEE International Conference on Signal, Information and Data Processing, Chongqing, China, 11–13 December 2019. accepted.
36. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollar, P. Focal Loss for Dense Object Detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2999–3007.
37. Everingham, M.L.; Van Gool, C.; Williams, K.I.; Winn, J.; Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
38. Lin, T.-Y.; Dollar, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 936–944.

39. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
40. Alex, H.; Sourabh, V.; Holger, C.; Zhou, L.; Jiong, Y.; Oscar, B. PointPillars: Fast Encoders for Object Detection from Point Clouds. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 12697–12705.