


Research Article

Learning Deformable Network for 3D Object Detection on Point Clouds

Wanyi Zhang,^{1,2} Xiuhua Fu ,² and Wei Li³

¹School of Information Technology, Jilin Normal University, Siping, China

²School of Optoelectronic Engineering, Changchun University of Science and Technology, Changchun, China

³College of Physics, Jilin University, Changchun, China

Correspondence should be addressed to Xiuhua Fu; 13604435770@126.com

Received 12 June 2021; Revised 13 July 2021; Accepted 12 August 2021; Published 21 August 2021

Academic Editor: Sang-Bing Tsai

Copyright © 2021 Wanyi Zhang 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.

3D object detection based on point cloud data in the unmanned driving scene has always been a research hotspot in unmanned driving sensing technology. With the development and maturity of deep neural networks technology, the method of using neural network to detect three-dimensional object target begins to show great advantages. The experimental results show that the mismatch between anchor and training samples would affect the detection accuracy, but it has not been well solved. The contributions of this paper are as follows. For the first time, deformable convolution is introduced into the point cloud object detection network, which enhances the adaptability of the network to vehicles with different directions and shapes. Secondly, a new generation method of anchor in RPN is proposed, which can effectively prevent the mismatching between the anchor and ground truth and remove the angle classification loss in the loss function. Compared with the state-of-the-art method, the AP and AOS of the detection results are improved.

1. Introduction

The 3D object detection methods mainly solve the problem of locating and identifying targets from both 2D and 3D data, including image, LiDAR data, and point cloud data. 3D object detection acts as a more and more important role in real-world applications, such as autonomous driving cars [1, 2], housekeeping equipment [3], and augmented reality [4]. A large number of methods have been proposed to solve the problem of 3D object detection. Image-based methods: when images contain detailed information, many methods generated the 3D bounding boxes [5, 6] by firstly estimating the depth of images from monocular or stereo cameras. The accuracy of these methods was limited by the result of the low accuracy of depth estimation. Fast RCNN [7] pipeline is used to generate 3D box proposals in [8] and then apply region-based recognition. Monocular image and shape priors of cars are used to propose 3D boxes in [9], while the length and width of cars and other objects are not invariable.

Another method designs a 3D scene representation which reasons jointly about the 3D shape of multiple objects.

Methods use the RGB-D image. With the equipment and application of 3D sensors such as RGB-D cameras and LiDAR in varying reality, the problem of the inaccuracy of depth estimation is avoided [10]. Use 3D pixel-wise features to detect cars in the RGB-D images, and take advantages of the 3D car model which can obtain additional information: height, occlusion, and 3D pose. A detailed geometry representation of objects is introduced by [11]. Sliding shapes [12] proposed sliding shape on the RGB-D image to generate 3D bounding boxes by SVM classifiers on grids encoded with handcrafted features. After sliding shapes, deep sliding shapes extract geometry features through 3D convolution networks, but the cost of better geometry features is expensive computations. The 3D depth data is transported into 2D maps and CNNs are applied to localize the objects in [13]. Reduce the 3D detection work by using an efficient 2D detector and making full use of 2D information.

Similarly, F-PointNets detect objects on 2D images and then apply PointNet to the corresponding frustum point cloud to generate 3D bounding boxes. Great amount of computation was saved so this method receives a competitive high-process speed. While F-PointNets rely too much on 2D detect result; if 2D detector misses the far object or small object such as pedestrian and cyclists, F-PointNets cannot detect it either.

Multiview: Maturana and Scherer [14] show a way to represent the point cloud by volumetric and multiview. MVCNN [15] is the first method to apply multiview computer vision to 3D object perception. 2D renders of objects from 3D data of different “perspectives” is used as the original training data. The model trained by the classic convolution network of the 2D image shows a faster speed and has a better effect on the recognition and classification of 3D objects than those directly trained by 3D data. In order to make great use of image-based feature extract methods [16] project point clouds into front view, compared to LiDAR only 3D detection, Enzweiler and Gavrilu [17] make greater progress on 3D detection, especially for pedestrians or cyclists. Among them, MV3D first projects the point clouds into the bird’s eye view and then applies a CNN to generate rough 3D bounding boxes which will be projected to three views. For each view, there will be a deep fusion network which extracts region-wise features via ROI pooling to jointly object prediction and 3D box regression. However, additional needs for camera cause time synchronization and calibration problem, which limit the use and require sensors which are always in a good state.

LiDAR based: here, we try to estimate accurate 3D bounding boxes and class label of objects from point clouds. Different from RGB image data, 3D point cloud’s unique properties are very robust to the changes of view angle and illumination. They contain relative overall structural information and precise depth of each point, but it is precisely because of different data forms. unordered, sparse, and locality sensitive, that the traditional network structure is often unable to directly process point cloud data. Bariya and Nishino [18] simply extend 2D region proposal network to 3D point cloud when computation cost increases dramatically. Searching for other representations for point cloud is done [19], which yield satisfactory results when a plenty of detailed 3D structure information is provided.

This also leads to the fact that most of the current three-dimensional target detection methods need to do some preprocessing operations on the point cloud data, which becomes two-dimensional data and then sent to the network for processing. For example, Song and Xiao [20] upgrade Faster/Mask RCNN [21] to 3D, while priors only process 2D detection results. Those methods convert the irregular point clouds to regular 3D grids and apply 3D CNN detectors to realize detection task; because of the use of three-dimensional convolution, the calculation of those methods is very large. There are also some methods [22] which project point cloud data to the perspective of aerial view and carry out 2D target detection on the image after projection. However, this kind of method loses a lot of spatial detail data, so the detection results are very limited. And, some previous

approaches [23] inferred point cloud features by neural networks on the point cloud which has been divided into many grids.

In this work, we introduce a 3D detection framework that directly processes raw point cloud data and does not depend on any 2D detectors. Our detection network is inspired by recent advances in 3D neural network models for point clouds and is extended by the Hough voting network, VoteNet, for object detection.

We leverage VoteNet, a hierarchical deep network for point cloud learning. To reduce the redundant computing of converting point cloud into other forms of data or features such as voxel dose, this helps a lot with avoiding the lack of dimension and structure information in the process of project point clouds to front view or birds’ eye view because we directly process the original point cloud and then only calculate the perceived points to take advantage of the sparsity of the point cloud. Although PointNet++ has been used in object classification and semantic segmentation and achieved fascinating results, but how to use this architecture to detect 3D objects in point clouds is still a field rarely touched by researchers. There is an easy and simple idea such as what 2D object detection does, that is, whether it proposes a lot of 3D bounding boxes in point clouds which contains their features learned from network. However, it is very difficult for us to directly apply this method on the sparse original point cloud. As the surface of an object is the only thing irradiated by the depth sensor, the center of a 3D object is likely to be in an open space which is far away from any captured surface point. So, in a point cloud, there may be no perceived points in the center of the object, which is very different from that in an image. Therefore, it is difficult to gather scene context near the target center by using the point-based network. If we just expand the receiving domain of the network to capture more context information, we will also absorb many other nearby objects, which will bring more clutters and problems. A point cloud depth network with a voting mechanism similar to classical Hough voting solves this problem. The new points near the target center can not only be generated by voting but also be grouped and aggregated to generate box proposals. In [24], a powerful 3D object detector is introduced, which is pure geometry and can be directly applied to point cloud. Traditional Hoff voting has multi-independent modules, so joint optimization is difficult to carry on. However, VoteNet could be optimized end-to-end. Specifically, the point cloud is sent to the backbone network to extract features, and then points with features are sampled from the point cloud to vote and generate the object center. The new object centers come from voting which appear near the real center, and it is easy to generate 3D box proposal by any learned network. We make experiment on the 3D object detection datasets: SUN RGB-D and ScanNet [25]. Our method only uses geometry of these two datasets and gain state-of-the-art 3D object detection performance than other methods that use both RGB and geometry or even multiview RGB images. Our research shows that, in our algorithm framework, context information is more

effectively aggregated, and our method has greatly improved for the case of object center far away from the object surface.

In summary, the contributions of our work are as follows:

- (1) The Euclidean clustering method is used to realize the target detection based on the point cloud data, and the better detection effect is achieved on the Kitti dataset. According to the characteristics of Kitti dataset, the data interface is designed to preprocess the labeled target, and a suitable data enhancement method is proposed for the target detection algorithm based on neural network.
- (2) Based on the principle of LiDAR, a method of dividing space in the cylindrical coordinate system and transforming point cloud in voxel is proposed. For the first time, deformable convolution is introduced into the point cloud target detection network, which enhances the adaptability of the network to vehicles with different directions and shapes. A new generation method of anchor in RPN is proposed, which can effectively prevent the mismatch between the anchor and ground truth, and at the same time, the angle classification loss in loss function is removed. Compared with the second, the AP and AOS of detection results are improved. The improved method proposed in this paper is also suitable for other voxel-based 3D object detection algorithms.

The paper is organized as follows. Related works with more details are presented in Section 2. Then, we give a detailed introduction in Section 3, respectively. Datasets and experiments are presented in Section 4. Conclusions and outlook are presented in Section 5.

2. Related Work

Various ways have been proposed for dealing with 3D object detection problem by extracting features in image and point cloud. Several existing methods were proposed to help to detect 3D bounding boxes of objects. Xiang et al. [26] generate 3D bounding boxes just to use front view images with shape priors or project the depth data into the front view as 2D maps which is fed to CNNs to localize objects. MV3D extract features from both front view's RGB image and LiDAR point cloud which is projected to the bird eye view. 3D bounding boxes are proposed by a trained RPN on the bird eye view. However, this method is a little weak to detect the object which is far away or small such as pedestrians and cyclists, and it is difficult to deal with overlap between objects too. PointRCNN [27] directly uses row point cloud as input of deep networks to generate proposal boxes and upgrade the way to extract features from point clouds but is dragged down by the empty voxel caused by the sparsity of the point cloud. VoxelNet divides space into many voxels and puts many VFE layers to get point cloud features from each voxels. Later, these features are transported into CNNs for detection and segmentation. Similarly, SECOND [28] applies sparse convolution to get efficient

features from voxels. F-PointNet [29] uses a 2D object detector on the image to make 2D object proposal first and then get corresponding frustum point cloud as the base of regression and prediction, while the accuracy of F-PointNet relies too much on the 2D detector.

Representation learning from point clouds: there exist many deep network architectures which are proposed to deal with point clouds and gain great performance on the task of 3D object detection and object segmentation. Some of these point cloud-based 3D detection techniques introduce a way to extract features by representing the point cloud in form of voxel (divide the point clouds into many cuboids). Ashburner and Friston [30] inspire the work by applying 3D convolutional neural networks on voxels, although it is restricted by the sparsity of point clouds and high cost of 3D convolution. Every nonempty voxel is encoded in [31] by 6 statistical quantities which can be derived from the points in the voxel. Li [32] represent each voxel by fusing multiple local statistics. Song and Xiao [33] just encode the 3D voxel into the binary form. VoxelNet first sampled the point within voxel and then applied many VFE layers to extract features from each voxel to represent the whole point clouds. Then, PointNet represent point cloud data as a vector, and shape features are extracted for a FCN to finish classification before PointNet and PointNet++, and there is little work on directly obtaining the feature from the row unordered sparse point cloud. After that, PointNet is used to generate 3D objects in a frustum point cloud corresponding to a 2D object proposal in [34].

3. Architecture for 3D Object Detection

3.1. Voxelization. Our method includes the spatial-feature extractor, deformable layer, RPN layer, and final regression layer. In VoxelNet, the Cartesian coordinate system is used for voxel segmentation and point cloud clustering, as shown in Figure 1(a). For comparison, Figure 1(b) shows the voxel segmentation in the cylindrical system. Due to the limitation of LiDAR working principle, it is impossible for LiDAR to obtain the information behind the object. To overcome the shortcoming, we divide voxels based on cylindrical coordinates, compared with the division based on the Cartesian coordinate system, and the method in this paper can significantly improve the point cloud in voxels, the sparse degree of voxels, and the processing efficiency of sparse convolution algorithm.

Suppose the cylindrical coordinate system consists of three axes, ρ , θ , and z . For the given point cloud data, (x, y, z) in the Cartesian coordinate system could be converted into the cylindrical coordinate system with the coordinate $(\rho = \sqrt{x^2 + y^2}, \theta = \arctan(y/x), z_c = z)$; then, we divide the voxel space evenly. After space division, the point cloud points need to be clustered because the division in the cylindrical coordinate system is more consistent with the working principle of LiDAR than that in the Cartesian coordinate system, and more sparse point cloud could be obtained. Considering that a large number of points will lead to the consumption of computing power and the density of the grid midpoint is not uniform, the maximum number of

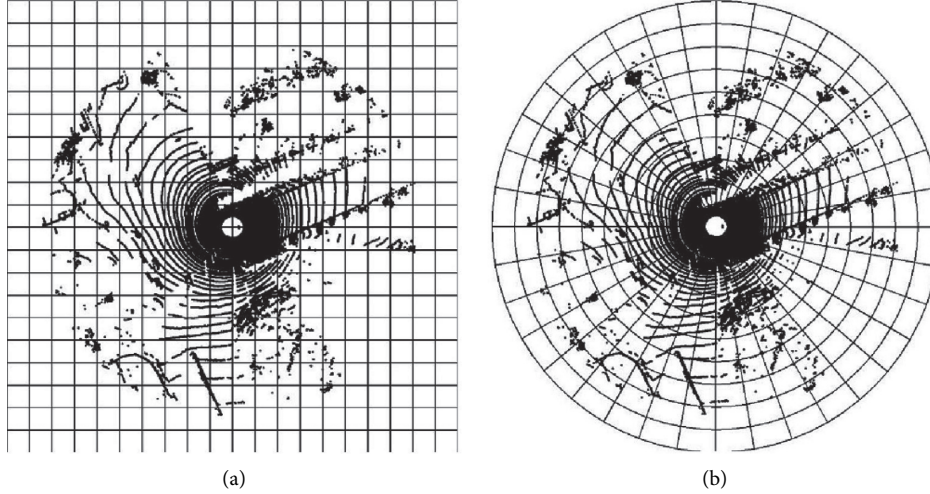


FIGURE 1: Voxel division diagram. (a) In Cartesian coordinates, according to the working principle of LiDAR, the scanning shape of laser beam on the ground is a concentric circle. When the laser irradiates the object, it will be reflected back, so it is impossible for LiDAR to obtain the information of points behind the object. (b) In cylindrical coordinates, the distribution could improve the uneven density of point cloud in voxels and the sparsity of voxels.

points in each nonempty voxel is set as T , and the redundant points are automatically discarded, and zero is added, if less than T . The voxelization algorithm is shown in Algorithm 1. We set the maximum number of voxels K and the maximum number of points T in each voxel, and a tensor with shape $K \times T \times 4$ is generated by the voxel clustering algorithm.

3.2. Network Architecture. Point cloud feature selection: $V = \{p_i = (\rho_i, \theta_i, z_i, r_i)^T \in R^4\}_{i=1, \dots, t}$ is defined as a nonempty voxel, where ρ_i, θ_i, z_i are the 3D coordinates of point and r_i is the reflection intensity. First, the average value $V_m = (v_\rho, v_\theta, v_z)$ of all points in each voxel and the center of each voxel $V_c = (c_\rho, c_\theta, c_z)$ are estimated. Then, the average distance and the distance from each point to the center of the voxel are added into the feature; finally, a tensor with shape $K \times T \times 10$ is generated as $p_i = (\rho_i, \theta_i, z_i, r_i, \rho_i - v_\rho, \theta_i - v_\theta, z_i - v_z, \rho_i - c_\rho, \theta_i - c_\theta, z_i - c_z)$ $i = 1, \dots, t$.

Voxel feature extraction layer: voxel feature extraction used VFE proposed by VoxelNet as the main structure; the VFE layer takes clustered voxels as input and uses the fully connected layer, batch norm, and ReLU layer to extract the feature.

Assuming that the number of the output nodes of VFE is $C = 2 \times n, n = Z_+^*$, the VFE layer takes $T \times 10$ points corresponding to the same voxel and is fully connected to $C/2$ outputs, obtains feature with dimension $K \times C/2$, then uses the maximum pooling to obtain the local aggregation feature of each voxel with dimension $K \times 1$, and finally copies it to the point-by-point feature and obtains the final voxel feature with shape $K \times C$. The single VFE layer is shown in Figure 2.

Spatial-feature extraction layer: we use submanifold sparse convolution and common sparse convolution to extract spatial feature. By gradually increasing the features between the voxels of receptive field, more context information is added for shape description. At the same time, the spatial-feature extraction layer can extract the information

about z-axis and transform the sparse 3D data into dense 2D pseudoimages. The spatial-feature extraction layer takes the features obtained from the voxel feature extraction layer as input and converts nonempty voxels into sparse 4D tensors according to the coordinates of each voxel in the voxel grid; then, the spatial features are extracted.

The spatial-feature extraction layer consists of two modules; the first module contains a submanifold convolution layer and a normal 3D sparse convolution layer; the second module consists of two submanifold convolutions and a normal 3D sparse convolution layer. The two modules only carry out downsampling on the z-axis without changing the length and width of the feature map. After two modules' processing, the dimension of z-axis is downsampled to two levels, and the sparse data is transformed into dense feature mapping. Then, the 4D tensors of the two layers are readjusted to 3D tensors similar to images. The structure of spatial-feature extraction layer is shown in Figure 3.

Deformable convolution layer: because the shape of convolution kernel used in convolution layer is fixed, the receptive field is still square after many convolutions, which results in the limited ability of network for deformation modeling. Deformable convolution and deformable ROI pooling are proposed by deformable convolution networks (DCN). DCN is based on the adaptive deformation of receptive field by adjusting the position of input sampling of convolution check. The standard sampling in ordinary convolution makes it difficult for the network to adapt to geometric deformation. In order to weaken this limitation, an offset variable is added to the position of each sampling point in the convolution kernel. The sampling position of convolution kernel changes adaptively according to the shape of the sample, instead of being limited to the standard lattice point. In this paper, DCN is introduced into point cloud object detection for the first time. As the direction of vehicles in point cloud object detection algorithm is distributed in 360 degrees, while ordinary convolution can only

```

(1) input points  $P = \{x_i, y_i, z_i\}, i = 1, 2, \dots, n$ 
(2) //convert to cylindrical coordinate system.
(3) for each  $i \in [1, n]$  do
(4)    $\rho_i = \sqrt{x^2 + y^2}, \theta_i = \arctan(y/x), z_{ic} = z_i$ 
(5) end for
(6) //clustering
(7) for each  $i \in [1, n]$  do
(8)   calculate the voxel  $v_i$  of  $p_i$ 
(9)   if  $v_i$  in hash table then
(10)    if number of voxel  $v\_count > T$  then
(11)     continue to next point.
(12)    else
(13)      $v\_count ++$ 
(14)     if number of nonzero voxel  $nz\_count > K$  then
(15)      return nonzero voxel, points in each voxel, and grid coordinates.
(16)    else
(17)     if the last point then
(18)      return nonzero voxel points in each voxel, and grid coordinates
(19)    else
(20)     continue to next point.
(21)    end if
(22)  end if
(23) end if
(24) else
(25)  if  $nz\_count > K$  then
(26)   return nonzero voxel, points in each voxel, and grid coordinates.
(27)  else
(28)   add new voxel in hash table.
(29)  end if
(30) end if
(31) end for

```

ALGORITHM 1: Voxelization algorithm.

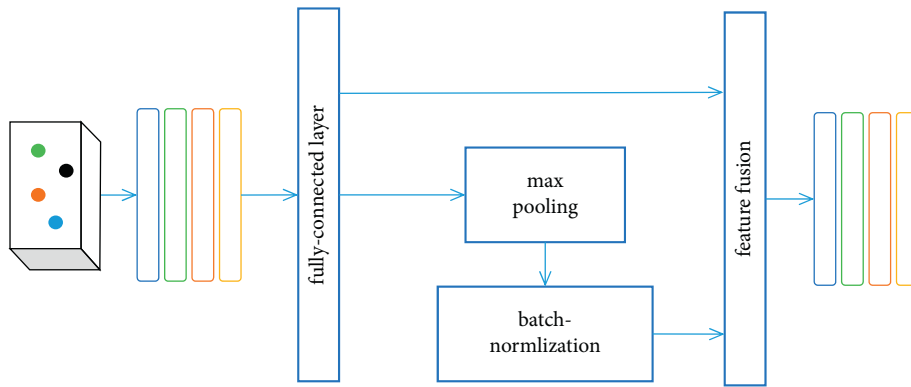


FIGURE 2: Single VFE layer. The VFE layer receives the $K \times T \times 10$ tensor as input and extracts the voxel feature with shape $K \times 128$.

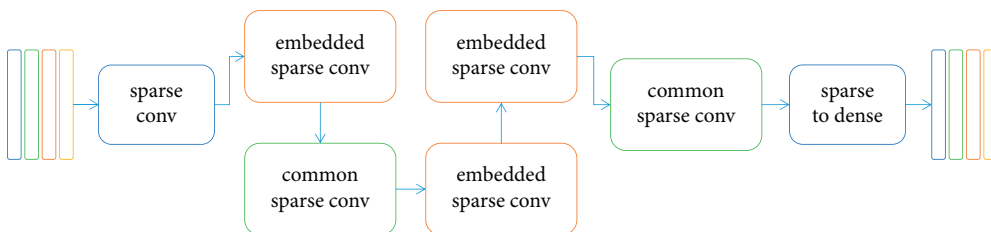


FIGURE 3: Spatial-feature extractor layer.

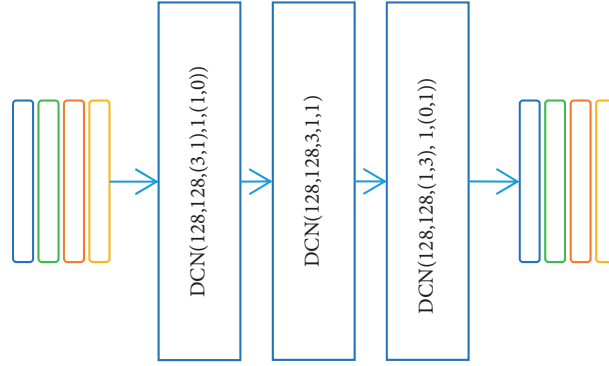


FIGURE 4: Deformable layer.

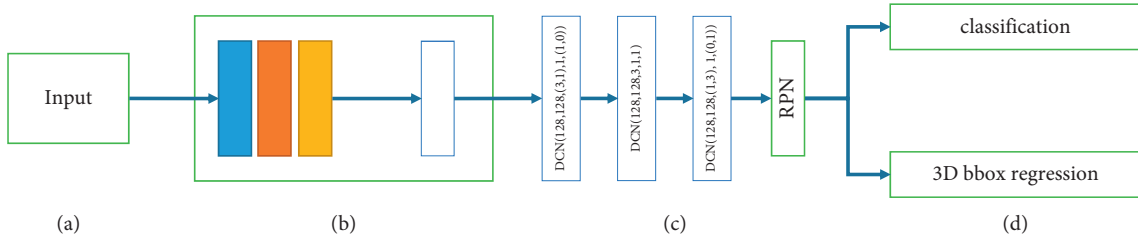


FIGURE 5: Processing of the proposed pipeline. (a) Input, (b) spatial-feature extractor, (c) deformable layer, and (d) results.

form a square receptive field, the introduction of deformable convolution can better adapt to the changes of different directions of vehicles. After the deformable convolution is applied to the spatial-feature extraction layer, the deformable convolution is expressed as $DCN(c_o, c_i, k, s, p)$, c_i and c_o are the channels of input and output tensor, k is the size of the convolution kernel, and s and p represent step and padding, respectively. The specific structure is shown in Figure 4.

Our proposed network is shown in Figure 5. The proposed network uses a backbone extractor to extract the spatial-feature extractor, and then, a deformable layer and RPN layer are listed to add more discriminative ability; finally, 3D object regression is predicted.

RPN is often used to determine whether there is a target to be detected in the local part of the feature map. The structure of RPN is shown in Figure 6. The network consists of three modules; firstly, the convolution layer is used to downsample the input tensor three times to get feature map x_1, y_2, y_3 ; then, we deconvolute and upsample x_1, y_2, y_3 to get up_1, up_2, up_3 ; finally, we splice up_1, up_2, up_3 together to obtain the final feature map. A 1×1 convolution is applied to predict the category the object regression.

4. Experiments

Data augmentation: the training process of neural network is the adjustment process of mapping model parameters. In the process of deep learning network model training, the amount of data has a great impact on the convergence speed and generalization performance of the network. **Random sampling:** we use the training dataset to generate a point cloud dataset which contains all target categories and target

3D frames. In the training process, we randomly extract n ground targets from the dataset and insert them into the current training data. This strategy greatly increases the number of real targets in each frame point cloud. **Random disturbance:** considering the influence of noise on network performance, the similar method used in VoxelNet is used. For each frame, the ground truth and its point cloud are transformed independently and randomly, instead of converting all point clouds with the same parameters. **Global rotation and scaling:** we apply the global scaling and rotation to the whole point clouds.

Implementation: we implement our network with PyTorch framework and use Adam as the optimizer. The initial learning rate is set as 0.0002 and learning rate decay strategy is used during training. The whole experiment trained 160 epochs in total and took about 22 hours to train the network on GTX2080TI. We use AP (Average Precision) and AOS (Average Orientation Similarity) as the evaluation method.

Evaluate results in KITTI dataset: we evaluate the vehicle detection performance on KITTI dataset of the proposed framework. We divide the Kitti dataset according to the difficulty of detection. Our method exceeds the SECOND by 1.24%, 1.10%, and 3.19% in the AP in the simple, medium, and hard datasets, respectively, and 0.84%, 1.21%, and 1.47% in AOS. The detailed results are shown in Tables 1 and 2. The “SECOND” is the previous state-of-the-art result, “Deformable” means we only add deformable layer for 3d final regression, “Cartesian” means we perform the voxelization on the Cartesian Coordinate system, and “ours” means our overall performance.

Comparison of different voxel partition methods: the voxel partition method in the cylindrical coordinate system is more suitable for the working principle of LiDAR. Firstly, we calculate the variance of the number of nonempty voxels

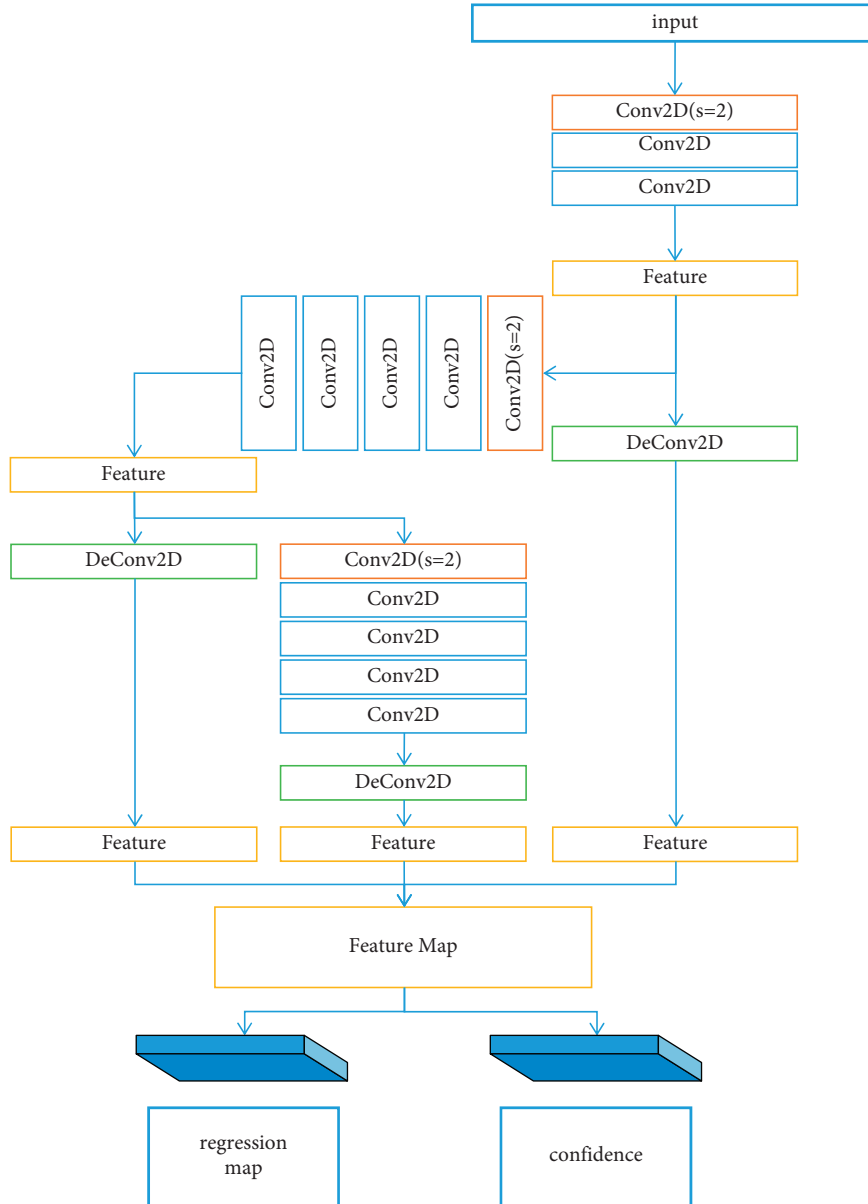


FIGURE 6: RPN layer.

TABLE 1: Evaluation of AP methods.

	Simple	Medium	Hard
SECOND	87.43	76.48	69.10
Ours	88.67	77.58	72.29
Cartesian	88.06	77.13	71.45
Deformable	88.36	77.67	71.64

TABLE 2: Evaluation of AOS methods.

	Simple	Medium	Hard
SECOND	90.45	88.31	86.98
Ours	91.29	89.52	88.45
Cartesian	90.97	89.14	88.24
Deformable	90.40	89.06	88.11

and the number of voxel points obtained by different partition methods. It is obvious that the number of nonempty voxels can be reduced effectively by using the cylindrical coordinate system to divide the group point cloud, and the uniformity of the voxel points can be improved. The detailed statistical results are shown in Table 3.

Evaluation of deformable convolution module: deformable convolution can adaptively adjust the sampling

TABLE 3: Evaluation of voxel partition methods.

Division method	Voxel num	Variance
Cartesian	6500	13.831
Cylindrical	5000	10.154

position according to the shape of the target. In this paper, deformable convolution is introduced into the target algorithm of 3D target detection to adapt to the target with any



FIGURE 7: Detecting result.

angle between 0 and 360 degrees. We compared the influence of deformable convolution on 3D object detection through experiments in Table 1.

Visualization results: the detection result is shown in Figure 7. The above figure shows the visualization results of the 3D bounding box of the detection results projected on the aerial view. The following figure shows the road conditions in front of the vehicle photographed by the camera. In the aerial view, the red box is a car, the blue box is a pedestrian, and the green box is a rider. The algorithm can accurately detect the cars, pedestrians, and riders around the vehicle.

5. Conclusions

3D object detection algorithm based on point cloud data in driverless scene has always been a research hotspot in driverless perception technology. With the development and maturity of deep neural network technology, the method of 3D target detection using neural network began to show great advantages. Based on the point cloud data collected by vehicle 64 line LiDAR and using the Kitti dataset as the evaluation sample, this paper studies how to detect the position, size, and direction of obstacles in the environment quickly and accurately based on the point cloud data, so as to provide reliable information for vehicle tracking and path planning.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

- [1] M. Cho, D. Shin, and J.-J. Lee, "Position detection of a scattering 3D object by use of the axially distributed image sensing technique," *Journal of the Optical Society of Korea*, vol. 18, no. 4, pp. 414–418, 2014.
- [2] T. Southey, "Improving object detection using 3D spatial relationships," *The EMBO Journal*, vol. 19, no. 11, pp. 2558–2568, 2013.
- [3] M. Ma, F. Guo, Z. Cao, and K. Wang, "Development of an artificial compound eye system for three-dimensional object detection," *Applied Optics*, vol. 53, no. 6, pp. 1166–1172, 2014.
- [4] L. Yan, K. Liu, E. Belyaev, and M. Duan, "RTL3D: real-time LIDAR-based 3D object detection with sparse CNN," *IET Computer Vision*, vol. 14, no. 5, pp. 224–232, 2020.
- [5] J. Chen, Y. Fang, and Y. K. Cho, "Real-time 3D crane workspace update using a hybrid visualization approach," *Journal of Computing in Civil Engineering*, vol. 31, no. 5, pp. 04017049.1–04017049.15, 2017.
- [6] F. Pomerleau, M. Liu, F. Colas, and R. Siegwart, "Challenging data sets for point cloud registration algorithms," *The International Journal of Robotics Research*, vol. 31, no. 14, pp. 1705–1711, 2012.
- [7] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, Santiago, Chile, December 2015.
- [8] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3D object detection from RGB-D data," 2018, <https://arxiv.org/abs/1711.08488>.
- [9] Y. Mae, J. Choi, H. Takahashi, K. Ohara, T. Takubo, and T. Arai, "Interoperable vision component for object detection and 3D pose estimation for modularized robot control," *Mechatronics*, vol. 21, no. 6, pp. 983–992, 2011.
- [10] Z. Zhao and X. Chen, "Building 3D semantic maps for mobile robots using RGB-D camera," *Intelligent Service Robotics*, vol. 9, no. 4, pp. 1–13, 2016.
- [11] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D mapping with an RGB-D camera," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 177–187, 2014.

- [12] Y. Furukawa and J. Ponce, “Accurate, dense, and robust multiview stereopsis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 8, pp. 1362–1376, 2010.
- [13] R.-F. Wang, W.-Z. Chen, S.-Y. Zhang, Y. Zhang, and X.-Z. Ye, “Similarity-based denoising of point-sampled surfaces,” *Journal of Zhejiang University—Science*, vol. 9, no. 6, pp. 807–815, 2008.
- [14] D. Maturana and S. A. Scherer, “VoxNet: A 3D convolutional neural network for real-time object recognition,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 922–928, Hamburg, Germany, September 2015.
- [15] H. Su, S. Maji, and E. Kalogerakis, “Multi-view convolutional neural networks for 3D shape recognition,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 945–953, Santiago, Chile, December 2015.
- [16] C. Premebida, J. Carreira, J. Batista, and U. Nunes, “Pedestrian detection combining RGB and dense LIDAR data,” in *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4112–4117, Chicago, IL, USA, September 2014.
- [17] M. Enzweiler and D. M. Gavrilu, “A multilevel mixture-of-experts framework for pedestrian classification,” *IEEE Transactions on Image Processing*, vol. 20, no. 10, pp. 2967–2979, 2011.
- [18] A. S. Mian, M. Bennamoun, and R. A. Owens, “On the repeatability and quality of keypoints for local feature-based 3D object retrieval from cluttered scenes,” *International Journal of Computer Vision*, vol. 89, no. 2, pp. 348–361, 2011.
- [19] P. Bariya and K. Nishino, “Scale-hierarchical 3D object recognition in cluttered scenes,” in *Proceedings of the Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1657–1664, San Francisco, CA, USA, June 2010.
- [20] S. Song and J. Xiao, “Deep sliding shapes for amodal 3D object detection in RGB-D images,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 808–816, Las Vegas, NV, USA, June 2016.
- [21] 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.
- [22] S. Du, B. Liu, Y. Liu, and J. Liu, “Global-local articulation pattern-based pedestrian detection using 3D lidar data,” *Remote Sensing Letters*, vol. 7, no. 7, pp. 681–690, 2016.
- [23] H. Kuang, B. Wang, J. An, M. Zhang, and Z. Zhang, “Voxel-FPN: multi-scale voxel feature aggregation for 3D object detection from LIDAR point clouds,” *Sensors*, vol. 20, no. 3, p. 704, 2020.
- [24] S. Qie and J. Cheng, S. Wang, C. Xu, and G. Xiangyang, “Point-selection and multi-level-point-feature fusion-based 3d point cloud classification,” *Electronics Letters*, vol. 56, no. 6, pp. 290–293, 2020.
- [25] A. Dai, A. X. Chang, M. Savva, M. Halber, T. A. Funkhouser, and M. Nießner, “ScanNet richly-annotated 3D reconstructions of in-door scenes,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2432–2443, Honolulu, HI, USA, July 2017.
- [26] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, “Asymmetric fingerprinting based on 1-out-of-n oblivious transfer,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1903–1911, Boston, MA, USA, June 2015.
- [27] S. Shi, X. Wang, and H. Li, “3D object proposal generation and detection from point cloud,” 2019, <https://arxiv.org/abs/1812.04244>.
- [28] Y. Yan, Y. Mao, and B. Li, “SECOND: sparsely embedded convolutional detection,” *Sensors*, vol. 18, no. 10, p. 3337, 2018.
- [29] Q. Zhu and Z. Mu, “PointNet++ and three layers of features fusion for occlusion three-dimensional ear recognition based on one sample per person,” *Symmetry*, vol. 12, no. 1, p. 78, 2020.
- [30] J. Ashburner and K. J. Friston, “Voxel-based morphometry—the methods,” *NeuroImage*, vol. 11, no. 6, pp. 805–821, 2000.
- [31] D. Z. Wang and I. Posner, “Voting for voting in online point cloud object detection,” in *Proceedings of the Robotics: Science and Systems XI*, Rome, Italy, March 2015.
- [32] B. Li, “3D fully convolutional network for vehicle detection in point cloud,” in *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 1513–1518, Vancouver, BC, Canada, September 2017.
- [33] S. Song and J. Xiao, “Sliding shapes for 3D object detection in depth images,” in *Computer Vision—ECCV 2014—13th European Conference*, pp. 634–651, Springer, Berlin, Germany, 2014.
- [34] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: deep hierarchical feature learning on point sets in a metric space,” in *Proceedings of the Annual Conference on Neural Information Processing Systems*, pp. 5099–5108, Long Beach, CA, USA, June 2017.