METHOD ARTICLE

3D object detection combining semantic and geometric features from point clouds [version 1; peer review: awaiting peer review]

Hao Peng, Guofeng Tong, Zheng Li, Yaqi Wang, Yuyuan Shao

College of Information Science and Engineering, Northeastern University, Shenyang 110819, Liaoning, China

Abstract

Background: 3D object detection based on point clouds in road scenes has attracted much attention recently. The voxel-based methods voxelize the scene to regular grids, which can be processed with the advanced feature learning frameworks based on convolutional layers for semantic feature learning. The point-based methods can extract the geometric feature of the point due to the coordinate reservations. The combination of the two is effective for 3D object detection. However, the current methods use a voxel-based detection head with anchors for classification and localization. Although the preset anchors cover the entire scene, it is not suitable for detection tasks with larger scenes and multiple categories of objects, due to the limitation of the voxel size. Additionally, the misalignment between the predicted confidence and proposals in the Regions of the Interest (ROI) selection bring obstacles to 3D object detection.

Methods: We investigate the combination of voxel-based methods and point-based methods for 3D object detection. Additionally, a voxel-to-point module that captures semantic and geometric features is proposed in the paper. The voxel-to-point module is conducive to the detection of small-size objects and avoids the presets of anchors in the inference stage. Moreover, a confidence adjustment module with the center-boundary-aware confidence attention is proposed to solve the misalignment between the predicted confidence and proposals in the regions of the interest selection.

Results: The proposed method has achieved state-of-the-art results for 3D object detection in the Karlsruhe Institute of Technology and Toyota Technological Institute (KITTI) object detection dataset. Actually, as of September 19, 2021, our method ranked 1st in the 3D and Bird Eyes View (BEV) detection of cyclists tagged with difficulty level 'easy', and ranked 2nd in the 3D detection of cyclists tagged with 'moderate'.

Conclusions: We propose an end-to-end two-stage 3D object detector...
with voxel-to-point module and confidence adjustment module.

**Keywords**
3D object detection, point clouds, voxel-to-point, semantic, geometric.
Introduction

3D object detection is an important task for scene understanding in computer vision and autonomous driving, as it can perceive the category and location of the object of interest in the scene. Lidar can accurately reproduce 3D spatial information, making it widely used in the field of autonomous driving. In addition, 3D object detection based on point clouds has received more and more emphasis in recent years.

Point-based methods, voxel-based methods, and a combination of the two are the mainstream approaches to 3D object detection from point clouds. The point-based methods can retain the fine-grained spatial information of the scene, which is conducive to capturing the geometric characteristics of the object. In addition, many invalid detections in the space of the scene are avoided. However, the point-based method needs to perform a neighborhood search when learning the semantic features of a local area, which increases the inference time of the model. The voxel-based methods divide the entire space into regular grids, and then project them to Bird Eyes View (BEV) and use Convolutional Neural Networks (CNN) to learn the semantic features of the scene. These methods have strict requirements on the size of the grid which is an important factor for balancing the amount of calculation and accuracy. For example, although the large-size grid makes the inference speed fast, it brings obstacles to the detection of small-size target objects. Moreover, the voxel-based method is often accompanied by various types of anchors for residual regression. Due to complexity considerations, this method is not suitable for large-size scenes with multi-class objects. Currently, joint point-based and voxel-based methods have appeared in many detection frameworks. Discriminative features are learned through voxels and points in corresponding spatial locations. Whereas, most of the works based on combined methods use voxel-based detection heads. Although this improves the recall of positive samples, it also comes with some drawbacks based on the voxel-based detection head, such as invalid detection positions, and the presets for multiple types of anchors. This paper proposes a novel voxel-point-based method named SGNet, which combines semantic and geometric features. SGNet introduces some related works. The proposed SGNet model is detailed in Section III. Section IV presents experimental results and ablation studies. Finally, conclusions are presented in Section V.

Related works

3D object detection from point clouds

Generally, the current 3D object detection methods can be divided into two types: one-stage and two-stage detectors. Single-stage detectors regress bounding box and confidence directly, and two-stage detectors use a second stage to refine the first-stage predictions with region-proposal-aligned features. VoxelNet designed a Voxel Feature Encoding module (VFE) to extract and aggregate the point features in each voxel, which is then followed by Region Proposal Networks (RPN) to generate detections. SECOND (Sparsely Embedded Convolutional Detection) investigates an improved sparse convolution method based on VoxelNet, which significantly increases the speed of both training and inference. However, the 3D convolution in these methods is time-consuming and computationally expensive. PointPillars uses a novel encoder that learns features on pillars of the point cloud to predict 3D oriented boxes for objects. The two-stage detectors can usually obtain better detection results due to the refinement of the second stage. PointRCNN (Point Regional Convolutional Neural Network) uses PointNet++ as the backbone in both stages and devices an anchor-free strategy to generate 3D proposals. Part-A4 replaces the PointNet++ backbone with a sparse convolutional network, and proposes RoI-aware point cloud feature pooling in the refinement. PV-RCNN (Point-Voxel Regional Convolutional Neural Network) uses set abstraction modules to extract point features from multi-scale voxel features in the first stage to refine the region proposals. CenterPoint first detects centers of objects using a keypoint detector and regresses to other attributes. CCA-SSD (Confident Intersection over union-Aware Single-Stage object Detector) designs a
lightweight spatial-semantic feature aggregation module to adaptively fuse high-level abstract semantic features and low-level spatial features for accurate predictions of bounding boxes and classification confidence. Voxel R-CNN\(^3\) utilizes voxel RoI pooling to extract region features from 3D voxel features for further refinement after generating dense region proposals from bird-eye-view feature representations.

Voxel-point based point cloud feature learning

Object detection based on 2D image data has been widely used in daily intelligent perception\(^3\). Voxel-based 3D object detection converts irregular point clouds into regular image-like data, and implements detection tasks with the help of convolutional networks. Because feature learning based on deep networks has developed very maturely, these methods are more conducive to the extraction of discriminative features. The unit of the voxel-based method is a voxel, which limits the acquisition of high-resolution geometric features. Additionally, the detection result is sensitive to the voxel size. The point-based methods\(^4,5\) can facilitate the learning of the geometric characteristics of the neighborhood of points. Whereas the neighborhood searching is time-consuming because it needs to traverse the candidate points. In recent years, joint point-based and voxel-based methods\(^6,7\) have become a new direction in 3D feature detection. The combination of features generated by different methods can be divided into the phased combination and synchronized combination. Actually, phased combinations most often appear in two-stage networks. Part-A\(^2\) first predicts proposals based on point features, and then uses sparse convolution to extract voxel features for two-stage bounding box refinement. The Sparse-to-Dense (STD) detector\(^6\) uses PointNet\(^6\) to learn point features in the first stage, and proposes PointsPool layer to obtain the representation of voxels in the proposal. There is no doubt that these methods based on point first and then voxel are not sufficient for semantic feature learning in the first stage. 3D IoU-Net (Intersection over Union Net)\(^8\), RangeRCNN\(^9\), and BANet (Boundary-Aware Net)\(^10\) respectively use voxel-based methods to roughly predict the bounding box, and the second stage incorporates point-based geometric features for fine-tuning. The above-mentioned methods of combining features phased can make up for the lack of features to a certain extent, but the disadvantages of the point-based or voxel-based methods in the first stage still exist. PV-RCNN\(^11\) deeply integrates both 3D voxel CNN and PointNet-based set abstraction to learn more discriminative point cloud features. Voxel R-CNN\(^11\) introduces voxel RoI pooling, which integrates spatial semantic features learned based on voxel and accelerates the PointNet\(^6\) module. SA-Det3D (Self-Attention Based Context-Aware 3D Object Detection)\(^11\) also achieved SOTA results on the KITTI dataset based on PV-RCNN framework. HP-RPN (Hybrid-Paradigm Region Proposal Network) module composed of SPCov, Auxiliary, and Keypoint branches is proposed in SIENet (Spatial Information Enhancement Network)\(^12\) to integrate point-based and voxel-based features. SPG (Semantic Point Generation)\(^13\) generates semantic points based on foreground voxels that recover the foreground regions suffering from the “missing point” problem.

The misalignment between confidence and bounding box

Most current 3D object detection frameworks have the issue that the predicted confidence does not align with the bounding box, that is, the bounding box with a high score does not necessarily have a higher IoU with the ground truth. STD\(^4\) develops an IoU estimation branch to obtain the 3D IoU between the predicted box and the corresponding ground truth, and uses it to weight the classification confidence. Part-A\(^2\) normalizes the 3D IoU between the proposal and corresponding ground truth box as the soft label for proposal confidence refinement. Similarly, PV-RCNN\(^12\) also uses 3D IoU guided scores in the confidence branch of the second stage. The attentive corner aggregation module and corner geometry encoding module are proposed in 3D IoU-Net\(^8\) to learn the IoU sensitive features from the local point cloud. And, based on the final prediction box, IoU alignment is designed to remove redundancy. CIA-SSD\(^9\) design the IoU-aware confidence rectification module for post-processing the confidence to alleviate the misalignment between the localization accuracy and classification confidence without having an additional network stage. 3D Region of Interest (RoI) alignment is used in FromVoxeltoPoint\(^1\) to crop and align the features with the proposal boxes for accurately perceiving the object position.

SGNet for 3D object detection from point clouds

In this section, an end-to-end two-stage detector named SGNet which combines semantic and geometric features is presented. The architecture of SGNet is presented in Figure 1. The point cloud encoder section introduces the encoder of the raw point clouds. Voxel-to-point module section details the voxel-to-point module proposed in this paper. The RPN-head section describes the detection task and head network structure in the first stage. We describe the confidence adjustment module in confidence adjustment module section. In refinement with the combined features section, we focus on the introduction of the refinement in second stage which includes the content of Confidence Refined RoI Pooling and RoI-head shown in Figure 1. Training loss section lists the items of training loss. Our code is provided in Software availability\(^13\).

Point cloud encoder

The point clouds retain the spatial information of the scene. The point-based features learning can capture the geometric characteristics of the scene. Whereas the voxel-based method can better capture the semantic features through the convolutional networks. So, the main purpose of the encoder in this paper is to generate input data adapted to the voxel-point-based method followed. The voxelization is adopted due to the irregularity of the raw points. In addition, less than T lidar points are reserved in each voxel. The mean value of all points in the voxel characterizes the voxel. Following PV-RCNN\(^12\), the voxel size is set to [0.05, 0.05, 0.1] on the X, Y, Z axis, respectively. Due to the small voxel size, the points after sampling still have a higher resolution, and the spatial coordinate information is preserved. We define the points after down-sampling as \(V = \{v_i = [x_i, y_i, z_i, r_i] \in \mathbb{R}^4 \}_{i=1..N} \), where N is the
number of non-empty voxels, and \( v_i \) contains XYZ mean coordinates for the \( i \)th voxel and \( r_i \) is the received mean reflectance. On the one hand, we use sparse convolution to transform the nonempty voxel representation into a high-dimensional to obtain enhanced point-wise features \( F_{p} \). On the other hand, the SPConvNet in SECOND is introduced to perform 8x downsize on the voxelized scene. The voxels are projected to BEV to generate features \( F_V \) for subsequent discriminative semantic feature learning based on 2D convolutional networks.

**Voxel-to-point module**

The point-wise features \( F_{p} \) with high resolution and voxel-wise features \( F_{v} \) with semantic information are obtained based on the point clouds encoder. In order to augment the discrimination of features, the BEV backbone used in PointPillars and multiple Linear-BN-ReLU layers (Linear-Batch Normalization-ReLU Layers) are introduced for processing of \( F_{p} \) and \( F_{v} \) respectively. In addition, the auxiliary classification task based on anchors is attached to the voxel branch for the update of parameters in the BEV backbone. The localization task is not used as an auxiliary in this paper due to the difference between the voxel-based method and the point-based method. As shown in Figure 2, for the voxel-based method, the residual between the center of the voxel and the ground truth bounding box is regressed. Whereas the point itself instead of the center is used in the point-based method. We think this would play a negative role in the localization task. On the contrary, the commonality of classification in the two methods can assist discriminative features learning. Considering the computational cost, we did not attach more auxiliary tasks.

The enhanced point-based geometric features and voxel-based semantic features are defined as \( F_{pGeo} \) and \( F_{vSem} \), respectively. In fact, the relationship between voxels and points is a one-to-many mapping. When constructing the combination of geometric and semantic features, a position encoder is introduced to make the semantic features diverse for different points. The coordinates XYZ of the point are encoded by multiple Linear-BN-ReLU layers to generate a position vector, which is combined with voxel-based semantic features to generate point-wise semantic features \( F_{PSem} \). It is noted that the semantic features are learned through the voxel branch, while the geometric features are obtained from the point branch. Therefore, we believe that it is necessary to make soft adjustments to point-wise features due to the difference in learning methods, where one is the point-based method and another is voxel-based. As shown in Figure 1, firstly, we concatenate the point-wise semantic features and geometric features to obtain the feature \( F_{SG} = [F_{PSem}, F_{Geo}] \). So \( F_{SG} \) contains all the information of the two features. Additionally, we learn the part-attention of both semantic features and the geometric features based on multiple Linear-BN-ReLU layers and Sigmoid layers, which are denoted as \( S_{Sem} \) and \( S_{Geo} \).
respectively. Finally, the combination $F_{CSG}$ of semantic and geometric features is generated by the concatenating of weighted point-wise features, as shown in Equation (1).

$$F_{CSG} = [F_{PSem} \cdot S_{Sem}, F_{PGeo} \cdot S_{Geo}]$$ (1)

Region proposal networks head

In this paper, point-wise proposal prediction is implemented. Compared with the voxel-based method, there is no need for preset anchors. The rough confidence and localization of the proposal are regressed through multiple Linear-BN-ReLU layers, following a layer of Linear. It is noted that for the auxiliary network in the training phase, a single Linear layer is also needed for classification prediction.

Confidence adjustment module

As shown in Figure 3, lidar can only scan the surface of the object. Most of the high-quality point clouds will concentrate on the boundary of large-size objects such as cars and the center of the bounding boxes for small-size objects such as cyclists. Dense point clouds are more conducive to feature learning, that is, features are more discriminative. Actually, the classification and the localization task in the first stage are based on the features generated by the same backbone. Therefore, we can give more attention to the confidence predicted by dense points in the local neighborhood, and indirectly promote the localization task positively. This facilitates the selection of high-quality RoI in the second stage.

We design center-boundary-aware confidence attention to drive our detector to lay emphasis on the prediction based on points close to the center and the boundary. Suppose that the ground truth bounding box is $[x, y, z, l, w, h, r]$, and an inner point $P$ is $[x_p, y_p, z_p]$. The ground truth bounding box is transformed to the local coordinates $C_{loc}$ with the center point as the origin. So, the point $P$ in this coordinate system is denoted as $[x_{loc}^P, y_{loc}^P, z_{loc}^P]$. As shown in Figure 4, take the center point as the origin to construct the attention coordinate. Decreasing attention value along with both the positive and negative directions of each coordinate axis of $C_{loc}$. When the

![Figure 3. Image and point clouds representations of objects in the KITTI dataset.](image)

![Figure 4. Details of confidence attention. The center and boundary will be emphasized.](image)
The offset is \([l/2, w/2, h/2]\), it decreases to 0, and then increases. The attention value is set to 1 when the offset relative to the center point is \([l, w, h]\).

The entire confidence attention equation can be expressed algebraically as Equation (2).

\[
CA = \left(\frac{\left[ x^p_{loc}, y^p_{loc}, z^p_{loc}\right]}{[l, w, h]} + 2 - 0.5 \right) * 2
\]  

We utilize the normalized IoU in Part-A2. Additionally, the normalized confidence attention is used to the soft adjustment of confidence predicted in the first stage, as shown in Equation (3).

\[
CA_{norm} = \min(1, \max(0, 2 \times CA - 0.5))
\]  

As shown in Figure 5, the confidence attention is derived from combined features through multiple Linear-BN-ReLU layers. The weighted confidence based on the mean of \(CA_{norm}\) is used to select RoIs in the second stage. The Confidence Adjustment Module (CAM) introduces a center-boundary-aware confidence attention mechanism, which is conducive to uniformity of confidence and learning of features as CAM lays more emphasis on the dense points.

Refinement with the combined features

The point-wise combined features which encode the semantic and geometric information are used to refine the rough results produced in the first stage. We introduce Voxel RoI Pooling Layer\(^{13}\) to generate the representation of RoIs. Linear-BN-ReLU layers with shared weights transform the feature dimension into a low dimension. Then for the refinement of confidence and localization, we set two branches which consisted of multiple Linear-BN-ReLU layers to fine-tune the features, and use a Linear layer to make the final prediction. The dropout layer is adapted to prevent overfitting in the training stage. In addition, the 3D IoU between RoI and ground truth guided the refinement of confidence.

Training loss

The S2Net proposed in this paper is a two-stage network. Rough predictions of the category confidence and bounding boxes of objects in the scene are obtained in the first stage. The focal loss\(^{13}\) with default parameters setting is used to overcome the class imbalance problem in classification tasks and auxiliary tasks. We follow the work of Part-A2, the residuals between ground truth and preset such as the mean size of objects, RoIs are regressed for the prediction of bounding boxes. It is noted that the corner point regularization term is introduced for localization refinement in the second stage. We use SmoothL1 loss for the regression of bounding boxes both in the first and second phases. Binary cross-entropy loss is adapted to constrain the update of the CAM network parameters, and the refinement of confidence in the second stage.

In summary, the total loss \(L\) in this paper consists of six items, as shown in Equation (4). In the first stage, the auxiliary task loss \(L_{aux}\), classification loss \(L_{cls}\), localization loss \(L_{loc}\), the CAM loss \(L_{cam}\); In the second stage, confidence refinement loss \(L_{cref}\), localization refinement loss \(L_{lref}\).

\[
L = L_{aux} + L_{cls} + L_{loc} + L_{cam} + L_{cref} + L_{lref}
\]  

Validation experiments

Datasets

The proposed detector is trained and evaluated on the widely used KITTI Object Detection Benchmark (see Underlying data). 7481 training samples and 7518 test samples in the KITTI dataset cover cars, pedestrians, and cyclists. Each object in the scene is tagged as one of the easy, moderate, and hard levels according to the size, occlusion, and truncation of the object (see the KITTI website for further information on the tagging system). Consistent with other 3D object detection algorithms (e.g. PV-RCNN, Voxel R-CNN, etc.), the training samples are split into the train set (3712 samples) and val sets (3769 samples) for evaluation. The label in the test samples is not available. We submit the result to the KITTI website for comparing the performance of the detector proposed in this paper with other SOTA methods listed in Table 1 on the test set.

Implementation details

For the KITTI dataset, the point clouds corresponding to the foreground image in the range of \([0, 70.4] \times [-40,40] \times [-3, 1]\) along the XYZ axis is used in the stage of training and inference. The size of the initial voxel is \([0.05, 0.05, 0.1]\). At most five

![Figure 5. Confidence adjustment module. (BN) Batchnorm, (GT) Ground Truth.](image)
points are reserved in each voxel. In addition, each point is characterized by a 4-dim vector of \([x, y, z, r]\), \(F_p\) and \(F_v\) generated based on point clouds encoder are 64 and 256 dimensions, respectively. The Multiple layer perceptron (MLP) for \(F_p\) is composed of two Subconv3D-BN-ReLU layers with [32, 64] dimensions. The BEV backbone is composed of convolutional blocks of 1x and 2x BEV scales, each of which consists of five Conv2d-BN-ReLU layers. The number of channels is 64, 128 respectively. Then the two scales are converted to 1x BEV size, 128-dim features through deconvolution. That is, the dimension of \(F_{\text{Geo}}\) is 256. Position encoder uses the [64, 256] Linear-BN-ReLU layers to encode the position information in a 256-dimensional vector for \(F_{\text{Geo}}\) enhancement, and then passes through the [128, 128] Linear-BN-ReLU layers to generate \(F_{\text{Geo}}\) with 128 channels. \(F_{\text{Geo}}\) is derived from \(F_v\) based on [64, 128] Linear-BN-ReLU layers. For predictions \(S_{\text{Geo}}\) and \(S_{\text{Geo}}^\ast\), two branches with [128, 64] Linear-BN-ReLU (LBR) layers are used respectively. In the first stage, each task contains the [128, 64] LBR layers. For the auxiliary task, we set a total of six anchors in each voxel, with three sizes of \([3.9, 1.6, 1.56]\), \([0.8, 0.6, 1.73]\), \([1.76, 0.6, 1.73]\) corresponding to cars, pedestrians, cyclists, and two angles of 0, \(\pi/2\). We only set the same three sizes as above for the target category in KITTI as the regression benchmark for the first stage localization, and each point only predicts one bounding box. In the RoI pooling stage, we sample 6x6x6 positions in RoI. The search range of the Voxel RoI Pooling layer is \([4, 4, 4]\) [8, 8, 8]. Other configurations in the second stage are the same as Voxel R-CNN[17].

The sparsity, irregularity of point clouds, and sample imbalance are obstacles to the training of the model for detection. We use a series of augmentations to overcome this problem. The object with points less than five is discarded in the training stage since we believe that too few points can easily add ambiguity to feature extraction. In order to make the detector robust for the irregularity of point clouds, we randomly flip the scene along the X-axis, and randomly rotate along the Z-axis a value that conforms to the uniform distribution of \([-\pi/4, \pi/4]\). Global scaling with a random factor sampled from uniformly distributed \([0.95, 1.05]\). More importantly, we follow the SECOND[4] and introduce the ground truth sampling augmentation to alleviate the insufficiency of ground truth samples. And, similar to PV-RCNN[22], road plane augmentation is also used in the training phase.

We trained our model with 50 epochs using Adam optimizer on NVIDIA RTX 2060 Super GPU graphics card. The batch size

| Method          | Reference     | Modality | Car(3D) Easy | Car(3D) Mod. | Car(3D) Hard | Pedestrian(3D) Easy | Pedestrian(3D) Mod. | Pedestrian(3D) Hard | Cyclist(3D) Easy | Cyclist(3D) Mod. | Cyclist(3D) Hard |
|-----------------|---------------|----------|--------------|--------------|--------------|---------------------|---------------------|---------------------|-----------------|-----------------|-----------------|
| F-PointNet[7]   | CVPR2018      | RGB+Lidar | 82.19        | 69.79        | 60.59        | 50.53               | 42.15               | 38.08               | 72.27           | 56.12           | 49.01           |
| AVOD-FPN[15]    | IROS2018      | RGB+Lidar | 83.07        | 71.76        | 65.73        | 50.46               | 42.47               | 39.04               | 63.76           | 50.55           | 44.93           |
| MMF[7]          | CVPR2019      | RGB+Lidar | 88.40        | 77.43        | 70.22        | -                   | -                   | -                   | -               | -               | -               |
| EPNet[19]       | ECCV2020      | RGB+Lidar | 89.81        | 79.28        | 74.59        | -                   | -                   | -                   | -               | -               | -               |
| SECOND[4]       | Sensors2018   | Lidar    | 83.34        | 72.55        | 65.82        | -                   | -                   | -                   | 71.33           | 52.08           | 45.83           |
| PointPillars[2] | CVPR2019      | Lidar    | 82.58        | 74.31        | 68.99        | 51.45               | 41.92               | 38.89               | 77.10           | 58.65           | 51.92           |
| PointRCNN[1]    | CVPR2019      | Lidar    | 86.96        | 75.64        | 70.70        | 47.98               | 39.37               | 36.01               | 74.96           | 58.82           | 52.53           |
| STD[16]         | ICCV2019      | Lidar    | 87.95        | 79.71        | 75.09        | 53.29               | 42.47               | 38.35               | 78.69           | 61.59           | 55.30           |
| 3D IoU-Net[25]  | Arxiv2020     | Lidar    | 87.96        | 79.03        | 72.78        | -                   | -                   | -                   | -               | -               | -               |
| SA-SSD[19]      | CVPR2020      | Lidar    | 88.75        | 79.79        | 74.16        | -                   | -                   | -                   | -               | -               | -               |
| TANet[11]       | AAAI2020      | Lidar    | 84.39        | 75.94        | 68.82        | 53.72               | 44.34               | 40.49               | 75.70           | 59.44           | 52.53           |
| 3D-SSD[18]      | CVPR2020      | Lidar    | 88.36        | 79.57        | 74.55        | 54.64               | 44.27               | 40.23               | 82.48           | 64.10           | 56.90           |
| Part-A[4]       | TPAMI2020     | Lidar    | 87.81        | 78.49        | 73.51        | 53.10               | 43.35               | 40.06               | 79.17           | 63.52           | 56.93           |
| PV-RCNN[19]     | CVPR2020      | Lidar    | 90.25        | 81.43        | 76.82        | 52.17               | 43.29               | 40.29               | 78.60           | 63.71           | 57.65           |
| VoxeIRCNN[3]    | AAAI2021      | Lidar    | 90.90        | 81.62        | 77.06        | -                   | -                   | -                   | -               | -               | -               |
| CIA-SSD[20]     | AAAI2021      | Lidar    | 89.59        | 80.28        | 72.87        | -                   | -                   | -                   | -               | -               | -               |
| SGNet(Ours)     | -             | Lidar    | 88.83        | 81.85        | 77.47        | 49.68               | 43.00               | 40.45               | 86.75           | 70.40           | 62.73           |

Table 1. Results on the KITTI test 3D detection benchmark. (Mod.) Moderate, (-) Not available.

Cobot 2022, 1:2 Last updated: 12 JAN 2022
is set to 3. We initialize the learning rate to 0.003. The learning rate decay strategy is consistent with Voxel R-CNN. 128 RoIs are reserved for each point cloud scene. We use the class-wise score threshold to prefilter some predictions, and then the 3D IoU based Non-Maximum Suppression (NMS) algorithm is adapted to remove redundant bounding boxes. Our code is provided in Software availability\(^3\).

**Results**

**Evaluation on KITTI test dataset**

The data underlying the results are available in Underlying data\(^3\). In this paper, 80% of train+val split samples are used for training and the remaining 20% for validation. Cars, pedestrians, and cyclists use score thresholds of 0.7, 0.3, 0.3 respectively in the post-processing process. We submit the results of the test set to KITTI’s official website for evaluation of our detector. The evaluations are automatically conducted by the KITTI server. We then compare the results with the SOTA algorithms evaluated by KITTI in recent years, as shown in Table 1 and Table 2. Actually, as of September 19th, our algorithm ranked 1st in 3D and BEV detection for cyclists with easy difficulty, and 2nd in the 3D detection of moderate cyclists\(^4\). In addition, SGNet proposed in this paper also achieves SOTA results on the detection of cars and pedestrians.

**Comparison of results on the 3D object detection.** SGNet has achieved SOTA results on the 3D object detection, as shown in Table 1. Compared with the best results of other methods listed in Table 1, the algorithm in this paper has achieved a very large improvement in the detection of cyclists, and achieved 4.27%, 6.3%, and 5.08% improvements in easy, moderate, and hard levels respectively. Compared with Voxel R-CNN, SGNet achieves 0.23% and 0.41% improvement in moderate and hard cars detection. SGNet has achieved a lead of 2.57% and 2.88% compared with the multi-modal algorithm EPNet. SGNet is superior to PV-RCNN, increasing 0.42%, 0.65% mean Average Precision (mAP) for the detection with the moderate and hard cars, and achieves 8.15%, 6.69%, and 5.08% large improvement on cyclists. It is noted that SGNet has achieved the best in the moderate and hard levels compared to all the car-only detectors listed in Table 1, which shows the effectiveness and ease of training of the detector in this paper.

**Comparison of results on the BEV detection.** As shown in Table 2, SGNet still has significant advantages on cyclists, with 2.99%, 4.99%, and 4.43% mAP ahead in easy, moderate, and hard levels, respectively. Compared with PV-RCNN, Voxel R-CNN, and CIA-SSD, the algorithm in this paper has achieved 0.4%, 0.41%, and 4.15% mAP improvement on the

---

**Table 2. Results on the KITTI test Bird Eye View (BEV) detection benchmark.** (Mod) Moderate, (\(\_\)) Not available.

| Method      | Reference     | Modality | Car(BEV) | Pedestrian(BEV) | Cyclist(BEV) |
|-------------|---------------|----------|----------|-----------------|--------------|
|             |               |          | Easy     | Mod. | Hard | Easy     | Mod. | Hard | Easy | Mod. | Hard |
| F-PointNet\(^4\) | CVPR2018       | RGB+Lidar | 91.17 | 84.67 | 74.77 | 57.13 | 49.57 | 45.48 | 77.26 | 61.37 | 53.78 |
| AVOD-FPN\(^5\)  | IROS2018      | RGB+Lidar | 90.99 | 84.82 | 79.62 | 58.49 | 50.32 | 46.98 | 69.39 | 57.12 | 51.09 |
| MMF\(^6\)       | CVPR2019      | RGB+Lidar | 93.67 | 88.21 | 81.99 | -    | -    | -    | -    | -    |
| EPNet\(^9\)     | ECCV2020      | RGB+Lidar | 94.22 | 88.47 | 83.69 | -    | -    | -    | -    | -    |
| SECOND\(^4\)    | Sensors2018   | Lidar    | 89.39 | 83.77 | 78.59 | -    | -    | -    | 76.50 | 56.05 | 49.45 |
| PointPillars\(^3\) | CVPR2019     | Lidar    | 90.07 | 86.56 | 82.81 | 57.60 | 48.64 | 45.78 | 79.90 | 62.73 | 55.58 |
| PointRCNN\(^1\) | CVPR2019     | Lidar    | 92.13 | 87.39 | 82.72 | 54.77 | 46.13 | 42.84 | 82.56 | 67.24 | 60.28 |
| STD\(^10\)      | ICCV2019      | Lidar    | 94.74 | 89.19 | 86.42 | 60.02 | 48.72 | 44.55 | 81.36 | 67.23 | 59.35 |
| 3D IoU-Net\(^1\) | Arxiv2020     | Lidar    | 94.76 | 88.38 | 81.93 | -    | -    | -    | -    | -    |
| SA-SSD\(^12\)   | CVPR2020     | Lidar    | 95.03 | 91.03 | 85.96 | -    | -    | -    | -    | -    |
| TANet\(^11\)    | AAAI2020     | Lidar    | 91.58 | 86.54 | 81.19 | 60.85 | 51.38 | 47.54 | 79.16 | 63.77 | 56.21 |
| 3D-SSD\(^13\)   | CVPR2020     | Lidar    | 92.66 | 89.02 | 85.86 | 60.54 | 49.94 | 45.73 | 85.04 | 67.62 | 61.14 |
| Part-A2\(^4\)   | TPAMI2020    | Lidar    | 91.70 | 87.79 | 84.61 | 59.04 | 49.81 | 45.92 | 83.43 | 68.73 | 61.85 |
| PV-RCNN\(^14\)  | CVPR2020     | Lidar    | 94.98 | 90.65 | 86.14 | 59.86 | 50.57 | 46.74 | 82.49 | 68.89 | 62.41 |
| VoxelRCNN\(^13\) | AAAI2021     | Lidar    | 94.85 | 88.83 | 86.13 | -    | -    | -    | -    | -    |
| CIA-SSD\(^20\)  | AAAI2021     | Lidar    | 93.74 | 89.84 | 82.39 | -    | -    | -    | -    | -    |
| SGNet(Ours)     | -            | Lidar    | 93.04 | 89.14 | **86.54** | 53.84 | 47.29 | 44.10 | **88.03** | **73.88** | **66.84** |

---

Page 9 of 13
detection of difficult vehicles, respectively. SGNet is superior to EPNet with a 0.67%, 2.85% increasing on moderate, hard cars respectively.

**Evaluation on KITTI val dataset**

We use the *train* set for SGNet training and take evaluation on the *val* set. As shown in Table 3, the algorithm in this paper has achieved very competitive results calculated by 11 recall positions. Compared with the other methods listed in Table 3, SGNet has achieved the best results in moderate difficulty level detection, which is ahead of Voxel R-CNN and PV-RCNN by 0.58% and 1.41% mAP respectively. Compared to CIA-SSD, our algorithm achieved a 5.29% mAP improvement on the moderate difficulty level. We also calculated mAP with 40 recall positions. And the comparison with EPNet, PV-RCNN, Voxel R-CNN is presented in Table 4. SGNet is superior to the other three methods in Table 4 in the hard difficulty level with slightly behind on easy, moderate difficulty levels.

**Qualitative analysis**

The representative detection results of SGNet proposed in this paper on the *val* split and *test* split are shown in Figure 6. Our model achieves good performance, and many unlabeled objects in KITTI dataset are still perceived.

**Ablation studies**

*Confidence adjustment module.* In order to verify the contribution of the CAM module proposed in this paper to the final mAP of 3D detection, we conducted a comparative experiment between the algorithm in this paper and the algorithm

---

### Table 3. Performance comparison on KITTI val set with average precision 11 for cars. (Mod) Moderate.

| Method         | Reference      | Modality       | Car(3D)   |
|----------------|----------------|----------------|-----------|
|                |                |                | Easy | Mod. | Hard |
| F-PointNet     | CVPR2018       | RGB+Lidar      | 83.76 | 70.92 | 63.65 |
| SECOND         | Sensors2018    | Lidar          | 88.61 | 76.62 | 77.22 |
| PointPillars   | CVPR2019       | Lidar          | 86.62 | 76.06 | 68.91 |
| PointRCNN      | CVPR2019       | Lidar          | 88.88 | 78.63 | 77.38 |
| STD            | ICCV2019       | Lidar          | 89.70 | 79.30 | 74.80 |
| 3D IoU-Net     | Arxiv2020      | Lidar          | 89.31 | 79.26 | 76.85 |
| SA-SSD         | CVPR2020       | Lidar          | 90.15 | 79.91 | 78.78 |
| TANet          | AAAI2020       | Lidar          | 87.52 | 76.64 | 73.86 |
| 3D-SSD         | CVPR2020       | Lidar          | 89.71 | 79.45 | 78.67 |
| Part-A         | TPAMI2020      | Lidar          | 89.47 | 79.47 | 78.54 |
| PV-RCNN        | CVPR2020       | Lidar          | 89.35 | 83.69 | 78.70 |
| Voxel R-CNN    | AAAI2021       | Lidar          | 89.41 | 84.52 | 78.93 |
| CIA-SSD        | AAAI2021       | Lidar          | 90.04 | 79.81 | 78.80 |
| SGNet(Ours)    | -              | Lidar          | 89.08 | 85.10 | 78.79 |

### Table 4. Performance comparison on KITTI val set with average precision 40 for cars. (Mod) Moderate, (BEV) Bird Eye View.

| Method         | Reference      | Modality       | Car(3D)   | Car(BEV)  |
|----------------|----------------|----------------|-----------|-----------|
|                |                |                | Easy | Mod. | Hard | Easy | Mod. | Hard |
| EPNet          | ECCV2020       | RGB+Lidar      | 92.28 | 82.59 | 80.14 | 95.51 | 88.76 | 88.36 |
| PV-RCNN        | CVPR2020       | Lidar          | 92.57 | 84.83 | 82.69 | 95.76 | 91.11 | 88.93 |
| Voxel R-CNN    | AAAI2021       | Lidar          | 92.38 | 85.29 | 82.86 | 95.52 | 91.25 | 88.99 |
| SGNet(Ours)    | -              | Lidar          | 92.16 | 85.01 | 82.99 | 95.54 | 91.22 | 89.26 |
Figure 6. Qualitative analysis of KITTI results. The 3D detection results of our model on val split (top 4 rows) and test split (last 4 rows). The ground truth 3D bounding boxes are in red while cyan implies the result predicted by SGNet.

without CAM. We calculate mAP by 40 recall positions. The results are presented in Table 5. As shown in Table 5, that the CAM promote the performance of our algorithm on easy, moderate, hard cars detection with 0.18%, 1.8%, 1.99% mAP improvement respectively. The CAM module proposed in this paper adjusts the confidence and aligns it with the localization task. Actually, the CAM module can effectively perceive the center and boundary of the object, and perfectly predicts the confidence attention. Some instances of point cloud objects with confidence attention colored can be seen in Figure 7.

Position encoder. We also explored the impact of the position encoder module in the VTPM on 3D detection. As shown in Table 6, in order to verify the efficiency of the position encoder module, the comparison experiments which include the position encoder and another one do not are set, respectively.
Table 5. Performance of Confidence Adjustment Module (CAM) with average precision 40 for cars. (Mod)

| CAM   | Car(3D) | Easy | Mod. | Hard |
|-------|---------|------|------|------|
| ✓     | 91.98   | 83.21| 81.00|
|       | 92.16   | 85.01| 82.99|

Moderate.

Table 6. Performance of position encoder with average precision 40 for cars. (Mod)

| Position encoder | Car(3D) | Easy | Mod. | Hard |
|------------------|---------|------|------|------|
| ✓                | 92.63   | 83.25| 82.86|
|                  | 92.16   | 85.01| 82.99|

Moderate.

The results indicate that the position encoder enhances the mAP of 3D cars detection, and the promotion of 1.76%, 0.13% is obtained in moderate, hard difficulty levels, correspondingly. This also verifies our ideas, that is, increase the diversity of features by position encoding mechanisms, thereby driving the voxel-point-based 3D object detection.

Conclusions

We propose a novel end-to-end two-stage 3D object detector named SGNet that combines semantic and geometric features. The voxel-to-point module is used to construct semantic-geometric features of point clouds. In addition, we design the CAM based on the center-boundary-aware confidence attention for alignment between predicted confidence and proposals. The SGNet proposed in this paper has achieved SOTA performance in the KITTI dataset. However, our algorithm is not particularly strong on pedestrian detection, which may be that there are many objects similar to those in point clouds data format, such as poles. In subsequent work, we will try to resolve this issue.

Data availability

Underlying data

The authors used the KITTI dataset in validating the algorithm of the present work. The KITTI dataset is open and freely available here: http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_val_split=3d. There are no requirements for accessing or downloading the data.

This dataset was not generated nor is it owned by the authors of this article; the listed owners are unknown. Therefore, neither the authors nor Cobot are responsible for the content of this dataset and cannot provide information about data collection. As this dataset contains potentially identifying images/information, caution is advised when using this dataset in future research.

Figshare: Data of 3D Object Detection Combining Semantic and Geometric Features from Point Clouds. https://doi.org/10.6084/m9.figshare.16859404.v4.

This project contains the following underlying data:

- KITTI-val.txt (raw data underlying Tables 3 and 4).
- nocam.txt (raw data underlying Table 5).
- nposcorder.txt (raw data underlying Table 6).

Extended data

Figshare: Data of 3D Object Detection Combining Semantic and Geometric Features from Point Clouds. https://doi.org/10.6084/m9.figshare.16859404.v4.

This project contains the following extended data:

- Table1.xlsx (results on the KITTI test 3D detection benchmark).
- Table2.xlsx (results on the KITTI test bird eye view detection benchmark).
- Table3.xlsx (performance comparison on KITTI val set, with average precision 11 for cars).
- Table4.xlsx (performance comparison on KITTI val set, with average precision 40 for cars).
- Table5.xlsx (performance of CAM, with average precision 40 for cars).
- Table6.xlsx (performance of position encoder, with average precision 40 for cars).
- Qualitative.png (qualitative analysis of KITTI results).
- cam-instances.png (instances of point cloud objects with confidence attention colored).
- KITTI-test.png (additional data associated with Table 1 and Table 2).

Data are available under the terms of the Creative Commons Attribution 4.0 International License (CC-BY 4.0).

Software availability
Source code available from: https://github.com/penghao1990/SGNet/tree/v1.0.2

Archived source code at time of publication: https://doi.org/10.5281/zenodo.5769873.

License: Apache-2.0 License

References

1. Geiger A, Lenz P, Stiller C, et al.: Vision meets robotics: The KITTI dataset. Int J Rob Res. 2013; 32(11): 1231–1237. Publisher Full Text
2. Qi CR, Liu W, Wu C, et al.: Frustum PointNets for 3D Object Detection from RGB-D Data. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2018; 918–927. Publisher Full Text
3. Shi S, Wang X, Li H: PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2019; 770–779. Publisher Full Text
4. Shi S, Wang Z, Shi J, et al.: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2017; 77–85. Publisher Full Text
5. Charles RQ, Su H, Kaichun M, et al.: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2017; 77–85. Publisher Full Text
6. Charles RQ, Su H, Yi L, et al.: PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. arXiv preprint. 2017. Reference Source
7. Zhou Y, Tuzel O: VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2018; 4490–4499. Publisher Full Text
8. Yan Y, Mao Y, Li B: SECOND: Sparsely Embedded Convolutional Detection. Sensors (Basel). 2018; 18(10): 3337. Publisher Abstract | Publisher Full Text | Free Full Text
9. Lang AH, Vora S, Caesar H, et al.: PointPillars: Fast Encoders for Object Detection From Point Clouds. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2019; 12689–12697. Publisher Full Text
10. He C, Zeng H, Huang J, et al.: Structure Aware Single-Stage 3D Object Detection From Point Cloud. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2020; 11870–11879. Publisher Full Text
11. Liu Z, Zhao X, Huang T, et al.: TANet: Robust 3D Object Detection from Point Clouds with Triple Attention. Proceedings of the AAAI Conference on Artificial Intelligence. 2020; 34(7): 11677–11684. Publisher Full Text
12. Shi S, Guo C, Jiang L, et al.: PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2020; 10526–10535. Publisher Full Text
13. Deng J, Shi S, Li P, et al.: Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection. AAAI, 2021. Reference Source
14. Zhou D, Fang J, Song X, et al.: Joint 3D Instance Segmentation and Object Detection for Autonomous Driving. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2020; 1836–1846. Publisher Full Text
15. Ku J, Mozaffari M, Lee J, et al.: Joint 3D Proposal Generation and Object Detection from View Aggregation. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2018; 1–8. Publisher Full Text
16. Yang Z, Sun Y, Liu S, et al.: STD: Sparse-to-Dense 3D Object Detector for Point Cloud. 2019 IEEE/CVF International Conference on Computer Vision (ICCV); 2019; 1951–1960. Publisher Full Text
17. Liang M, Yang B, Chen Y, et al.: Multi-Task Multi-Sensor Fusion for 3D Object Detection. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2019; 7337–7345. Publisher Full Text
18. Yang Z, Sun Y, Liu S, et al.: 3DSSD: Point-Based 3D Single Stage Object Detector. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2020; 11037–11045. Publisher Full Text
19. Huang T, Liu Z, Chen X, et al.: EPNet: Enhancing Point Features with Image Semantics for 3D Object Detection. ECCV, 2020. Publisher Full Text
20. Zheng W, Tang W, Chen S, et al.: CAA-SSD: Confident IoU-Aware Single-Stage Object Detector From Point Cloud. AAAI, 2021. Reference Source
21. Graham B: Sparse 3d convolutional neural networks. BMVC, 2015. Publisher Full Text
22. Graham B, van der Maaten L: Submanifold sparse convolutional networks. CoRR, 2017. Reference Source
23. Graham B, Engelcke M, van der Maaten L: 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2018; 9224–9232. Publisher Full Text
24. Yin T, Zhou X, Krähenbühl P: Center-based 3D Object Detection and Tracking. CVPR, 2021. Reference Source
25. Li J, Luo S, Zhu Z, et al.: 3D IoU-Net: IoU Guided 3D Object Detector for Point Clouds. arXiv preprint. 2020. Reference Source
26. Liang Z, Zhang M, Zhang Z, et al.: RangeRCNN: Towards Fast and Accurate 3D Object Detection with Range Image Representation. 2020. Reference Source
27. Qian R, Lai X, Li X: Boundary-Aware 3D Object Detection from Point Clouds. arXiv, 2021. Reference Source
28. Bhattacharyya P, Huang C, Czarnecki K: SA-Det3D: Self-Attention Based Context-Aware 3D Object Detection. 2021. Reference Source
29. Li Z, Yao Y, Quan Z, et al.: SIENet: Spatial Information Enhancement Network for 3D Object Detection from Point Cloud. 2021. Reference Source
30. Xu Q, Zhou Y, Wang W, et al.: SPG: Unsupervised Domain Adaptation for 3D Object Detection via Semantic Point Generation. Proceedings of the IEEE conference on computer vision and pattern recognition (ICCV). 2021. Reference Source
31. Li J, Dai H, Shao L, et al.: From Voxel to Point: IoU-guided 3D Object Detection for Point Cloud with Voxel-to-Point Decoder. MM '21: The 29th ACM International Conference on Multimedia (ACM MM). 2021. Reference Source
32. penghao1990: penghao1990/SGNet:v1.0.2 (v1.0.2). Zenodo. 2021. http://www.doi.org/10.5281/zenodo.5769873.
33. Lin TY, Goyal P, Girshick R, et al.: Focal Loss for Dense Object Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2020; 42(2): 318–327. Publisher Full Text
34. Peng H: Data of 3D Object Detection Combining Semantic and Geometric Features From Point Clouds. figshare. Dataset. 2021. http://www.doi.org/10.6084/m9.figshare.16859404.v4
35. 3D Object Detection Evaluation. 2021; (accessed 19 September 2021). Reference Source