From Points to Parts: 3D Object Detection From Point Cloud With Part-Aware and Part-Aggregation Network

Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li

Abstract—3D object detection from LiDAR point cloud is a challenging problem in 3D scene understanding and has many practical applications. In this paper, we extend our preliminary work PointRCNN to a novel and strong point-cloud-based 3D object detection framework, the part-aware and aggregation neural network (Part-A^2 net). The whole framework consists of the part-aware stage and the part-aggregation stage. First, the part-aware stage for the first time fully utilizes free-of-charge part supervisions derived from 3D ground-truth boxes to simultaneously predict high quality 3D proposals and accurate intra-object part locations. The predicted intra-object part locations within the same proposal are grouped by our new-designed RoI-aware point cloud pooling module, which results in an effective representation to encode the geometry-specific features of each 3D proposal. Then the part-aggregation stage learns to re-score the box and refine the box location by exploring the spatial relationship of the pooled intra-object part locations. Extensive experiments are conducted to demonstrate the performance improvements from each component of our proposed framework. Our Part-A^2 net outperforms all existing 3D detection methods and achieves new state-of-the-art on KITTI 3D object detection dataset by utilizing only the LiDAR point cloud data.

Index Terms—3D object detection, point cloud, part location, LiDAR, convolutional neural network, autonomous driving

1 Introduction

With the surging demand from autonomous driving and robotics, increasing attention has been paid to 3D object detection [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. Though significant achievements have been made in 2D object detection from images [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], directly extending these 2D detection methods to 3D detection might lead to inferior performance, since the point cloud data of 3D scenes has irregular data format and 3D detection with point clouds faces great challenges from the irregular data format and large search space of 6 Degrees-of-Freedom (DoF) of 3D objects.

Existing 3D object detection methods have explored several ways to tackle these challenges. Some works [6], [25], [26] utilize 2D detectors to detect 2D boxes from the image, and then adopt PointNet [27], [28] to the cropped point cloud to directly regress the parameters of 3D boxes from raw point cloud. However, these methods heavily depend on the performance of 2D object detectors and cannot take the advantages of 3D information for generating robust bounding box proposals. Some other works [1], [4], [5], [10], [11] project the point cloud from the bird view to create a 2D bird-view point density map and apply 2D Convolutional Neural Networks (CNN) to these feature maps for 3D object detection, but the hand-crafted features cannot fully exploit the 3D information of raw point cloud and may not be optimal. There are also some one-stage 3D object detectors [7], [9], [29] that divide the 3D space into regular 3D voxels and apply 3D CNN or 3D sparse convolution [30], [31] to extract 3D features and finally compress to bird-view feature map for 3D object detection. These works do not fully exploit all available information from 3D box annotations for improving the performance of 3D detection. For instance, the 3D box annotations also imply the point distributions within each 3D objects, which are beneficial for learning more discriminative features to improve the performance of 3D object detection. Also, these works are all one-stage detection frameworks which cannot utilize the RoI-pooling scheme to pool specific features of each proposal for the box refinement in a second stage.

In contrast, we propose a novel two-stage 3D object detection framework, the part-aware and aggregation neural network (i.e. Part-A^2 net), which directly operates on 3D point cloud and achieves state-of-the-art 3D detection performance by fully exploring the informative 3D box annotations from the training data. Our key observation is that, unlike object detection from 2D images, 3D objects in autonomous driving scenes are naturally and well separated by annotated 3D bounding boxes, which means the training data with 3D box annotations automatically provides free-of-charge semantic masks and even the relative location of each foreground point within the 3D ground truth bounding boxes (see Fig. 1 for illustration). In the remaining parts of this paper, the relative location of each foreground point w.r.t. the object box that it

- *S. Shi, X. Wang, and H. Li are with the Department of Electrical Engineering, The Chinese University of Hong Kong, Hong Kong SAR, China. E-mail: {ssshi, xgwang, hsli}@ee.cuhk.edu.hk.*
- *Z. Wang and J. Shi are with the SenseTime Research, Beijing, China. E-mail: {wangzhe, shijianping}@sensetime.com.*

Manuscript received 16 Aug. 2019; revised 7 Dec. 2019; accepted 18 Feb. 2020. Date of publication 28 Feb. 2020; date of current version 1 July 2021. (Corresponding author: Hongsheng Li.) Recommended for acceptance by K. Schindler. Digital Object Identifier no. 10.1109/TPAMI.2020.2977026

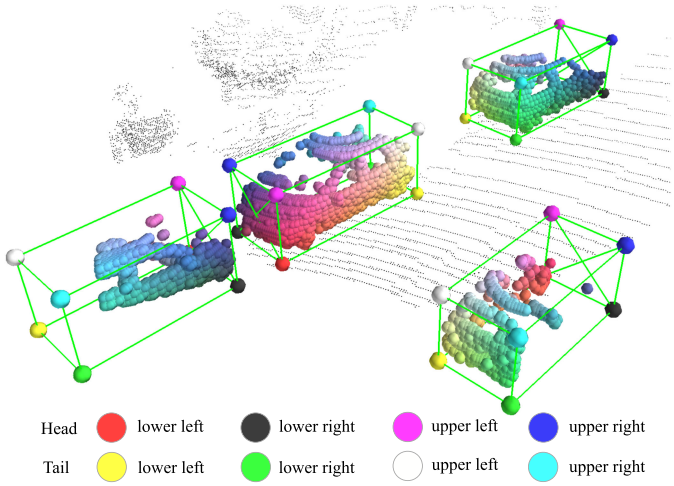


Fig. 1. Our proposed part-aware and aggregation network can accurately predict intra-object part locations even when objects are partially occluded. Such part locations can assist accurate 3D object detection. The predicted intra-object part locations by our proposed method are visualized by interpolated colors of eight corners. Best viewed in colors.

belongs to is denoted as *the intra-object part locations*. This is totally different from the box annotations in 2D images, since some parts of objects in the 2D images may be occluded. Using the ground-truth 2D bounding boxes would generate inaccurate and noisy intra-object part locations for each pixel within objects. These 3D intra-object part locations imply the 3D point distributions of 3D objects. Such 3D intra-object part locations are informative and can be obtained for free, but *were never explored in 3D object detection*.

Motivated by this observation, our proposed Part-A^2 net is designed as a novel two-stage 3D detection framework, which consists of the part-aware stage (Stage-I) for predicting accurate intra-object part locations and learning point-wise features, and the part-aggregation stage (Stage-II) for aggregating the part information to improve the quality of predicted boxes. Our approach produces 3D bounding boxes parameterized with $(x, y, z, h, w, l, \theta)$, where (x, y, z) are the box center coordinates, (h, w, l) are the height, width and length of each box respectively, and θ is the orientation angle of each box from the bird's eye view.

Specifically, in the part-aware stage-I, the network learns to segment the foreground points and estimate the intra-object part locations for all the foreground points (see Fig. 1), where the segmentation masks and ground-truth part location annotations are directly generated from the ground-truth 3D box annotations. In addition, it also generates 3D proposals from the raw point cloud simultaneously with foreground segmentation and part estimation. We investigate two strategies, i.e., anchor-free versus anchor-based strategies, for 3D proposal generation to handle to different scenarios. The anchor-free strategy is relatively light-weight and is more memory efficient, while the anchor-based strategy achieves higher recall rates with more memory and calculation costs. For the anchor-free strategy, we propose to directly generate 3D bounding box proposals in a bottom-up scheme by segmenting foreground points and generating the 3D proposals from the predicted foreground points simultaneously. Since it avoids using the large number of 3D anchor boxes in the whole 3D space as previous methods [4], [29] do, it saves much memory. For the anchor-based strategy, it generates 3D proposals from downsampled bird-view feature maps with pre-defined 3D anchor boxes at each spatial location. Since it needs to place multiple 3D anchors with different orientations and classes at each location, it needs more memory but can achieve higher object recall.

In the second stage of existing two-stage detection methods, information within 3D proposals needs to be aggregated by certain pooling operations for the following box re-scoring and location refinement. However, the previous point cloud pooling strategy (as used in our preliminary PointRCNN [32]) result in ambiguous representations, since different proposals might end up pooling the same group of points, which lose the abilities to encode the geometric information of the proposals. To tackle this problem, we propose a novel differentiable RoI-aware point cloud pooling operation, which keeps all information from both non-empty and empty voxels within the proposals, to eliminate the ambiguity of previous point cloud pooling strategy. This is vital to obtain an effective representation for box scoring and location refinement, as the empty voxels also encode the box's geometry information.

The stage-II aims to aggregate the pooled part features from stage-I by the proposed RoI-aware pooling for improving the quality of the proposals. Our stage-II network adopts the sparse convolution and sparse pooling operations to gradually aggregate the pooled part features of each 3D proposal for accurate confidence prediction and box refinement. The experiments show that the aggregated part features could improve the quality of the proposals remarkably and our overall framework achieves state-of-the-art performance on KITTI 3D detection benchmark.

Our primary contributions could be summarized into four-fold. (1) We proposed the Part-A^2 net framework for 3D object detection from point cloud, which boosts the 3D detection performance by using the free-of-charge intra-object part information to learning discriminative 3D features and by effectively aggregating the part features with RoI-aware pooling and sparse convolutions. (2) We present two strategies for 3D proposal generation to handle different scenarios. The anchor-free strategy is more memory efficient while the anchor-based strategy results in higher object recall. (3) We propose a differentiable RoI-aware point cloud region pooling operation to eliminate the ambiguity in existing point cloud region pooling operations. The experiments show that the pooled feature representation benefits box refinement stage significantly. (4) Our proposed Part-A^2 net outperforms all published methods with remarkable margins and ranks $1st$ with 14 FPS inference speed on the challenging KITTI 3D detection benchmark [33] as of August 15, 2019, which demonstrates the effectiveness of our method.

2 Related Work

3D Object Detection From 2D Images. There are several existing works on estimating the 3D bounding box from images. [34], [35] leveraged the geometry constraints between 3D and 2D bounding box to recover the 3D object pose. [36], [37], [38], [39] exploited the similarity between 3D objects and the CAD models. Chen *et al.* [40], [41] formulated the 3D geometric information of objects as an energy function to score the pre-defined 3D boxes. Ku *et al.* [42] proposed the aggregate losses

to improve the 3D localization accuracy from monocular image. Recently [43], [44] explored the stereo pair of images to improve the 3D detection performance from stereo cameras. These works can only generate coarse 3D detection results due to the lack of accurate depth information and can be substantially affected by appearance variations.

3D Object Detection From Multiple Sensors. Several existing methods have worked on fusing the information from multiple sensors (e.g., LiDAR and camera) to help 3D object detection. [1], [4] projected the point cloud to the bird view and extracted features from bird-view maps and images separately, which are then cropped and fused by projecting 3D proposals to the corresponding 2D feature maps for 3D object detection. [5] further explored the feature fusion strategy by proposing continuous fusion layer to fuse image feature to bird-view features. Different from projecting point cloud to bird-view map, [6], [25] utilized off-the-shelf 2D object detectors to detect 2D boxes first for cropping the point cloud and then applied PointNet [27], [28] to extract features from the cropped point clouds for 3D box estimation. These methods may suffers from the time synchronization problem of multiple sensors in the practical applications. Unlike these sensor fusion methods, our proposed 3D detection frameworks Part-A^2 net could achieve comparable or even better 3D detection results by using only point cloud as input.

3D Object Detection From Point Clouds Only. Zhou *et al.* [29] for the first time proposed the VoxelNet to learn discriminative features from point cloud and detect 3D object with only point cloud. [7] improved VoxelNet by introducing sparse convolution [30], [31] for efficient voxel feature extraction. [9], [10], [11] projected the point cloud to bird-view maps and applied 2D CNN on these maps. These methods do not fully exploit all available information from the informative 3D box annotations and are all one-stage 3D detection methods. In contrast, our proposed two-stage 3D detection framework Part-A^2 net explores the abundant information provided by 3D box annotations and learns to predict accurate intra-object part locations to learn the point distribution of 3D objects, and the predicted part locations are aggregated in the second stage for refining the 3D proposals, which significantly improves the performance of 3D object detection.

Point Cloud Feature Learning for 3D Object Detection. There are generally three ways of learning features from point cloud for 3D detection. (1) [1], [4], [5], [10], [11] projected point cloud to bird-view map and utilized 2D CNN for feature extraction. (2) [6], [25] conducted PointNet [27], [28] to learn the point cloud features directly from raw point cloud. (3) [29] proposed VoxelNet and [7] applied sparse convolution [30], [31] to speed up the VoxelNet for feature learning. Only the second and third methods have the potential to extract point-wise features for segmenting the foreground points and predicting the intra-object part locations in our framework. Here we design an encoder-decoder point cloud backbone network similarly with UNet [45] to extract discriminative point-wise features, which is based on the 3D sparse convolution and 3D sparse deconvolution operations since they are more efficient and effective than the point-based backbone like PointNet++ [28]. The point-based backbone and voxel-based backbone are experimented and discussed in Section 4.1.1.

3D/2D Instance Segmentation. The approaches of 3D instance segmentation are often based on point-cloud-based 3D detection methods. Several approaches are based on the 3D detection bounding boxes with an extra mask branch for predicting the object mask. Yi *et al.* [46] proposed an analysis-by-synthesis strategy to generate 3D proposals for 3D instance segmentation. Hou *et al.* [47] combined the multi-view RGB images and 3D point cloud to better generate proposals and predict object instance masks in and end-to-end manner.

Some other approaches first estimate the semantic segmentation labels and then group the points into instances based on the learned point-wise embeddings. Wang *et al.* [48] calculated the similarities between points for grouping foreground points of each instance. Wang *et al.* [49] proposed a semantic-aware point-level instance embedding strategy to learn better features for both the semantic and instance point cloud segmentation. Lahoud *et al.* [50] proposed a mask-task learning framework to learn the feature embedding and the directional information of the instance's center for better clustering the points into instances. However, they did not utilize the free-of-charge intra-object part locations as extra supervisions as our proposed method does.

There are also some anchor-free approaches for 2D instance segmentation by clustering the pixels into instances. Brabandere *et al.* [51] adopted a discriminative loss function to cluster the pixels of the same instance in a feature space while Bai *et al.* [52] proposed to estimate a modified watershed energy landscape to separate the pixels of different instances. However, those methods only group foreground pixels/points into different instances and did not estimate the 3D bounding boxes. Different with the above methods, our proposed anchor-free approach estimates intra-object part locations and directly generates 3D bounding box proposals from individual 3D points for achieving 3D object detection.

Part Models for Object Detection. Deformable Part-based Models (DPM) [53] achieved great success on 2D object detection before the deep learning models are utilized. [54], [55], [56] extended the DPM to 3D world to reason the parts in 3D and estimate the object poses, where [54] modeled the object as a 3D cuboid with both deformable faces and deformable parts, [55] proposed a 3D DPM that generates a full 3D object model with continuous appearance representation, and [56] presented the notion of 3D part sharing with 3D CAD models to estimate the fine poses. These DPM based approaches generally adopt several part templates trained with hand-crafted features to localize the objects and estimate the object pose. In contrast, we formulate the object parts as point-wise intra-object part locations in the context of point cloud, where the training labels of part locations could be directly generated from 3D box annotations and they implicitly encode the part distribution of 3D objects. Moreover, both the estimation and aggregation of intra-object part locations are learned by the more robust deep learning networks instead of the previous hand-crafted schemes.

3 Part-A^2 Net: 3D Part-Aware and Aggregation for 3D Detection From Point Cloud

A preliminary version of this work was presented in [32], where we proposed PointRCNN for 3D object detection from raw point cloud. To make the framework more general and effective, in this paper, we extend PointRCNN to a new end-

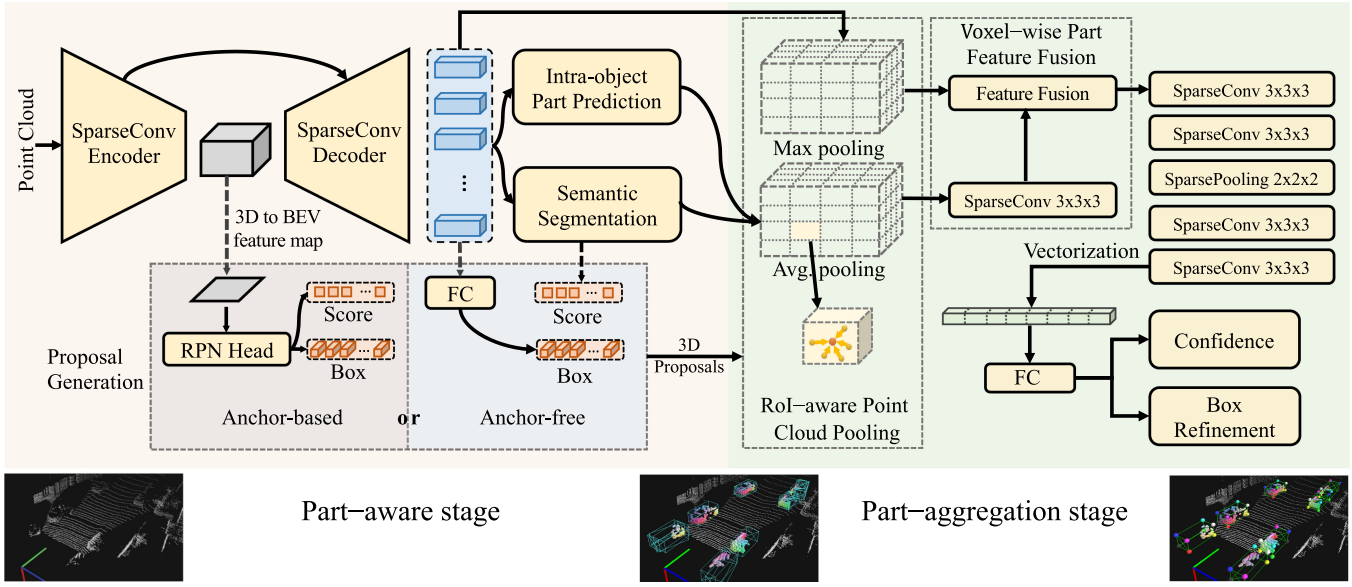


Fig. 2. The overall framework of our part-aware and aggregation neural network for 3D object detection. It consists of two stages: (a) the part-aware stage-I for the first time predicts intra-object part locations and generates 3D proposals by feeding the point cloud to our encoder-decoder network. (b) The part-aggregation stage-II conducts the proposed RoI-aware point cloud pooling operation to aggregate the part information from each 3D proposal, then the part-aggregation network is utilized to score boxes and refine locations based on the part features and information from stage-I.

to-end 3D detection framework, the part-aware and aggregation neural network, i.e. Part-A^2 net, to further boost the performance of 3D object detection from point cloud.

The key observation is that, the ground-truth boxes of 3D object detection not only automatically provide accurate segmentation mask because of the fact that 3D objects are naturally separated in 3D scenes, but also imply the relative locations for each foreground 3D point within the ground truth boxes. This is very different from 2D object detection, where 2D object boxes might only contain portion of an object due to occlusion and thus cannot provide accurate relative location for each 2D pixel. These relative locations of foreground points encode valuable information of foreground 3D points and is beneficial for 3D object detection. This is because foreground objects of the same class (like car class) generally have similar 3D shapes and point distributions. The relative locations of foreground points provide strong cues for box scoring and localization. We name the relative locations of the 3D foreground points w.r.t. to their corresponding boxes *the intra-object part locations*.

Those intra-object part locations provide rich information for learning discriminative 3D features from point cloud but were never explored in previous 3D object detection methods. With such rich supervisions, we propose a novel part-aware and aggregation 3D object detector, Part-A^2 net, for 3D object detection from point cloud. Specifically, we propose to use the free-of-charge 3D intra-object part location labels and segmentation labels as extra supervisions to learn better 3D features in the first stage. The predicted 3D intra-object part locations and point-wise 3D features within each 3D proposal are then aggregated in the second stage to score the boxes and refine their locations. The overall framework is illustrated in Fig. 2.

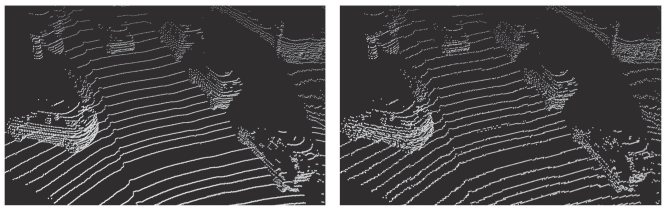
3.1 Stage-I: Part-Aware 3D Proposal Generation

The part-aware network aims to extract discriminative features from the point cloud by learning to estimate the intra-object part locations of foreground points, since these part locations implicitly encode the 3D object's shapes by indicating the relative locations of surface points of 3D objects. Also, the part-aware stage learns to estimate the intra-object part locations of foreground points and to generate 3D proposals simultaneously. Two strategies for 3D proposal generation from point clouds, anchor-free and anchor-based schemes, are proposed to handle different scenarios.

3.1.1 Point-Wise Feature Learning via Sparse Convolution

For segmenting the foreground points and estimating their intra-object part locations, we first need to learn discriminative point-wise features for describing the raw point clouds. Instead of using point-based methods like [27], [28], [57], [58], [59], [60], [61] for extracting point-wise features from the point cloud, as show in the left part of Fig. 2, we propose to utilize an encoder-decoder network with sparse convolution and deconvolution [30], [31] to learn discriminative point-wise features for foreground point segmentation and intra-object part location estimation, which is more efficient and effective than the previous PointNet++ backbone as used in our preliminary work [32].

Specifically, we voxelize the 3D space into regular voxels and extract the voxel-wise features of each non-empty voxel by stacking sparse convolutions and sparse deconvolutions, where the initial feature of each voxel is simply calculated as the mean values of point coordinates within each voxel in the LiDAR coordinate system. The center of each non-empty voxel is considered as a point to form a new point cloud with the point-wise features (i.e. the voxel-wise features), which is approximately equivalent to the raw point cloud as shown in Fig. 3, since the voxel size is much smaller (e.g., 5 cm $\times$ 5 cm $\times$ 10 cm in our method) compared to the whole 3D space ($\sim$70 m $\times$ 80 m $\times$ 4 m). For each 3D scene in the KITTI dataset [33], there are generally about 16,000 non-empty



(a) Raw point cloud (b) Voxelized point cloud

Fig. 3. Comparison of voxelized point cloud and raw point cloud in autonomous driving scenarios. The center of each non-empty voxel is considered as a point to form the voxelized point cloud. The voxelized point cloud is approximately equivalent to the raw point cloud and 3D shapes of 3D objects are well kept for 3D object detection.

voxels in the 3D space. The voxelized point cloud could not only be processed by the more efficient sparse convolution based backbone, but it also keeps approximately equivalent to the raw point cloud for 3D object detection.

Our sparse convolution based backbone is designed based on the encoder-decoder architecture. The spatial resolution of input feature volumes is 8 times downsampled by a series of sparse convolution layers with stride 2, and is then gradually upsampled to the original resolution by the sparse deconvolutions for the voxel-wise feature learning. The detailed network structure is illustrated in Section 3.5 and Fig. 7. Our newly designed 3D sparse convolution based-backbone results in better 3D box recall than the PointNet++ based backbone in our preliminary PointRCNN framework [32] (as shown by experimental results in Table 1), which demonstrates the effectiveness of this new backbone for the point-wise feature learning.

3.1.2 Estimation of Foreground Points and Intra-Object Part Locations

The segmentation masks help the network to distinguish foreground points and background, while the intra-object part locations provide rich information for the neural network to recognize and detect 3D objects. For instance, the side of a vehicle is usually a plane parallel to a side of its corresponding bounding box. By learning to estimate not only the foreground segmentation mask but also the intra-object part location of each point, the neural network develops the ability of inferring the shape and pose of the objects, which is crucial for 3D object detection.

Formulation of Intra-Object Part Location. As shown in Fig. 4, we formulate intra-object part location of each foreground point as its relative location in the 3D ground-truth bounding box that it belongs to. We denote three continuous

TABLE 1
Recall (With 100 Proposals) of the Proposal Generation Stage by Different Backbone Network and Different Proposal Generation Strategy

Method	Backbone	Proposal Scheme	Recall
PointRCNN	PointNet++	Anchor-free	74.81
Part-A^2-free	SparseConvUNet	Anchor-free	81.54
Part-A^2-anchor	SparseConvUNet	Anchors-based	85.58

The experiments are conducted on the car class at moderate difficulty of the val *split of KITTI dataset, and the evaluation metric is the 3D rotated IoU with threshold 0.7.*

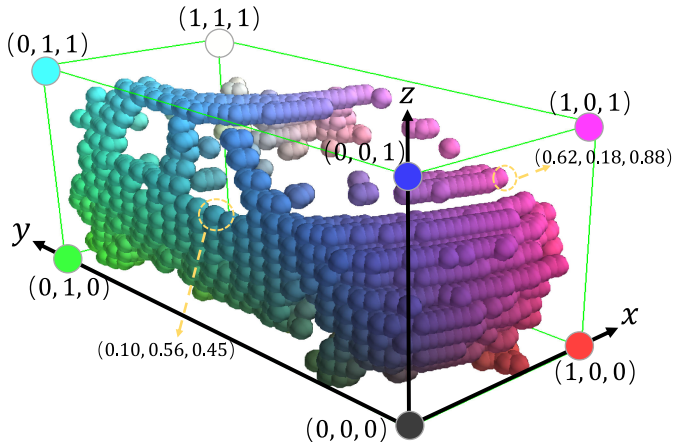


Fig. 4. Illustration of intra-object part locations for foreground points. Here we use interpolated colors to indicate the intra-object part location of each point. Best viewed in colors.

values $(x^{(part)}, y^{(part)}, z^{(part)})$ as the target intra-object part location of the foreground point $(x^{(p)}, y^{(p)}, z^{(p)})$, which can be calculated as follows

$$\begin{aligned} \begin{bmatrix} x^{(t)} & y^{(t)} \end{bmatrix} &= \begin{bmatrix} x^{(p)} - x^{(c)} & y^{(p)} - y^{(c)} \end{bmatrix} \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}, \\ x^{(part)} &= \frac{x^{(t)}}{w} + 0.5, \quad y^{(part)} = \frac{y^{(t)}}{l} + 0.5, \\ z^{(part)} &= \frac{z^{(p)} - z^{(c)}}{h} + 0.5, \end{aligned} \tag{1}$$

where $(x^{(c)}, y^{(c)}, z^{(c)})$ is the box center, (h, w, l) is the box size (height, width, length), and θ is the box orientation in bird-view. The relative part location of a foreground point $x^{(part)}, y^{(part)}, z^{(part)} \in [0, 1]$, and the part location of the object center is therefore $(0.5, 0.5, 0.5)$. Note that intra-object location coordinate system follows the similar definition of KITTI's global coordinate system, where the direction of z is perpendicular to the ground, and x and y are parallel to the horizontal plane.

Learning Foreground Segmentation and Intra-Object Part Location Estimation. As shown in Fig. 2, given the above sparse convolution based backbone, two branches are appended to the output features of the encoder-decoder backbone for segmenting the foreground points and predicting their intra-object part locations. Both branches utilize the sigmoid function as last non-linearity function for generating outputs. The segmentation scores of foreground points indicate the confidence of the predicted intra-object part locations since the intra-object part locations are defined and learned on foreground points only in the training stage. Since the number of foreground points is generally much smaller than that of the background points in large-scale outdoor scenes, we adopt the focal loss [21] for calculating point segmentation loss $\mathcal{L}_{\text{seg}}$ to handle the class imbalance issue

$$\begin{aligned} \mathcal{L}_{\text{seg}}(p_t) &= -\alpha_t (1 - p_t)^{\gamma} \log(p_t), \text{where } p_t \\ &= \begin{cases} p & \text{for forground points,} \\ 1 - p & \text{otherwise.} \end{cases} \end{aligned} \tag{2}$$

where p is the predicted foreground probability of a single 3D point and we keep $\alpha_t = 0.25$ and $\gamma = 2$ as the original

paper. All points inside the ground-truth boxes are utilized as positive points and others are considered as negative points for training.

For estimating the intra-object part location (denoted as $(x^{(part)}, y^{(part)}, z^{(part)})$) of each foreground point, since they are bounded between [0, 1], we apply the binary cross entropy losses to each foreground point as follows

$$\begin{aligned}\mathcal{L}_{\text{part}}(u^{(part)}) = &- u^{(part)}\log(\tilde{u}^{(part)}) \\ &- (1-u^{(part)})\log(1-\tilde{u}^{(part)}) \\ &\text{for } u \in \{x, y, z\},\end{aligned} \tag{3}$$

where $\tilde{u}^{(part)}$ is the predicted intra-object part location from the network output, and $u^{(part)}$ is the corresponding ground-truth intra-object part location. Note that the part location estimation is only conducted for foreground points.

3.1.3 3D Proposal Generation From Point Cloud

To aggregate the predicted intra-object part locations and the learned point-wise 3D features for improving the performance of 3D object detection in the second stage, we need to generate 3D proposals to group the foreground points that belong to the same object. Here we investigate two strategies for 3D proposal generation from point cloud, the anchor-free scheme and anchor-based scheme, to handle different scenarios. The anchor-free strategy is more memory efficient while the anchor-based strategy achieves higher recall with more memory cost.

Anchor-Free 3D Proposal Generation. Our model with this strategy is denoted as Part-A^2-free. We propose a novel scheme similarly to our preliminary PointRCNN [32] to generate 3D proposals in a bottom-up manner. As shown in the left part of Fig. 2, we append an extra branch to the decoder of our sparse convolution backbone to generate 3D proposal from each point that is predicted as foreground.

However, if the algorithm directly estimates object's center locations from each foreground point, the regression targets would vary in a large range. For instance, for a foreground point at the corner of an object, its relative offsets to the object center is much larger than those of a foreground point on the side of an object. If directly predicting relative offsets w.r.t. each foreground point with conventional regression losses (e.g., $L1$ or $L2$ losses), the loss would be dominated by errors of corner foreground points.

To solve the issue of large varying ranges of the regression targets, we propose the bin-based center regression loss. As shown in Fig. 5, we split the surrounding bird-view area of each foreground point into a series of discrete bins along the X and Y axes by dividing a search range $\mathcal{S}$ of each axis into bins of uniform length δ, which represents different object centers (x, y) on the X-Y plane. We observe that conducting bin-based classification with cross-entropy loss for the X and Y axes instead of direct regression with smooth-$L1$ loss [14] results in more accurate and robust center localization. To refine small localization after the assignment into each X-Y bin, a small residual is also estimated. The overall regression loss for the X or Y axes therefore consists of bin classification loss and the residual regression loss within the classified bin. For the center location z along the vertical Z axis, we directly utilize smooth-$L1$ loss for the regression since most objects' z values are generally within a small range. The object center regression targets could therefore be formulated as

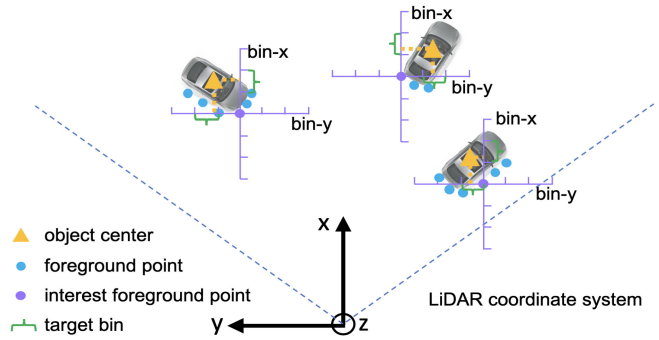


Fig. 5. Illustration of bin-based center localization. The surrounding area along X and Y axes of each foreground point is split into a series of bins to locate the object center.

$$\begin{aligned}\text{bin}_x^{(p)} &= \left\lfloor \frac{x^{(c)} - x^{(p)} + \mathcal{S}}{\delta} \right\rfloor, \ \text{bin}_y^{(p)} = \left\lfloor \frac{y^{(c)} - y^{(p)} + \mathcal{S}}{\delta} \right\rfloor, \\ \text{res}_{u\in\{x,y\}}^{(p)} &= \frac{1}{\delta}\left(u^{(c)} - u^{(p)} + \mathcal{S} - \left(\text{bin}_u^{(p)} \cdot \delta + \frac{\delta}{2}\right)\right), \\ \text{res}_z^{(p)} &= z^{(c)} - z^{(p)},\end{aligned} \tag{4}$$

where δ is the bin length, $\mathcal{S}$ is the search range, $(x^{(p)}, y^{(p)}, z^{(p)})$ is the coordinates of a foreground point of interest, $(x^{(c)}, y^{(c)}, z^{(c)})$ is the center coordinates of its corresponding object, $\text{bin}_x^{(p)}$ and $\text{bin}_y^{(p)}$ are ground-truth bin assignments and $\text{res}_x^{(p)}$ and $\text{res}_y^{(p)}$ are the ground-truth residual for further location refinement within the assigned bin.

Since our proposed bottom-up proposal generation strategy is anchor-free, it does not have an initial value for box orientation. Hence we directly divide the orientation 2π into discrete bins with bin size w, and calculate the bin classification target $\text{bin}_\theta^{(p)}$ and residual regression target $\text{res}_\theta^{(p)}$ as

$$\begin{aligned}\text{bin}_\theta^{(p)} &= \left\lfloor \frac{(\theta + \frac{w}{2}) \bmod 2\pi}{\omega} \right\rfloor, \\ \text{res}_\theta^{(p)} &= \frac{2}{\omega}\left(\left(\left(\theta + \frac{w}{2}\right) \bmod 2\pi\right) - \left(\text{bin}_\theta^{(p)} \cdot \omega + \frac{\omega}{2}\right)\right).\end{aligned} \tag{5}$$

Thus the overall 3D bounding box regression loss $\mathcal{L}_{\text{box}}$ could be formulated as

$$\begin{aligned}\mathcal{L}_{\text{bin}}^{(p)} &= \sum_{u\in\{x,y,\theta\}} \left(\mathcal{L}_{\text{ce}}(\widehat{\text{bin}}_u^{(p)}, \text{bin}_u^{(p)}) + \mathcal{L}_{\text{smooth-L1}}(\widehat{\text{res}}_u^{(p)}, \text{res}_u^{(p)})\right), \\ \mathcal{L}_{\text{res}}^{(p)} &= \sum_{v\in\{z,h,w,l\}} \mathcal{L}_{\text{smooth-L1}}(\widehat{\text{res}}_v^{(p)}, \text{res}_v^{(p)}), \\ \mathcal{L}_{\text{box}} &= \mathcal{L}_{\text{bin}}^{(p)} + \mathcal{L}_{\text{res}}^{(p)},\end{aligned} \tag{6}$$

where $\widehat{\text{bin}}_u^{(p)}$ and $\widehat{\text{bin}}_u^{(p)}$ are the predicted bin assignments and residuals of the foreground point p, $\text{bin}_u^{(p)}$ and $\text{res}_u^{(p)}$ are the ground-truth targets calculated as above, $\widehat{\text{res}}_v^{(p)}$ is the predicted center residual in vertical axis or the size residual with respect to the average object size of each class in the entire training set, $\text{res}_v^{(p)}$ is the corresponding ground-truth target of $\widehat{\text{res}}_v^{(p)}$, $\mathcal{L}_{\text{ce}}$ denotes the cross-entropy classification loss, and $\mathcal{L}_{\text{smooth-L1}}$ denotes the smooth-$L1$ loss. Note that the box regression loss $\mathcal{L}_{\text{box}}$ is only applied to the foreground points.

In the inference stage, regressed x, y and θ are obtained by first choosing the bin center with the highest predicted confidence and then adding the predicted residuals.

Based on this anchor-free strategy, our method not only fully explores the 3D information from point cloud for 3D proposal generation, but also avoids using a large set of predefined 3D anchor boxes in the 3D space by constraining the 3D proposals to be only generated by foreground points.

Anchor-Based 3D Proposal Generation. Our model with this strategy is denoted as Part-A^2-anchor. The stage-I is illustrated in Fig. 2, the sparse convolution based encoder takes a voxelized point cloud with shape $M \times N \times H$, and produces an 8-time X- and Y-axially downsampled $\frac{M}{8} \times \frac{N}{8}$ 2D bird-view feature map with $\frac{H}{16} \times D$ feature channels, where $\frac{H}{16}$ denotes the feature volume being 16-time downsampled along the Z axis, D is the feature dimension of each encoded feature voxel, and "$\times D$" denotes concatenating the features at each $x-y$ bird-view location of different heights to obtain a 1D feature vector. We then append a Region Proposal Network (RPN) head similar with [7] to the above bird-view feature map for 3D proposal generation with predefined 3D anchors. Each class has $2 \times \frac{M}{8} \times \frac{N}{8}$ predefined anchors with the specified anchor size for each class, where each pixel on the bird-view feature map has one anchor parallel to the X axis and one anchor parallel to the Y axis. Each class will have its own predefined anchors since the object sizes of different classes vary significantly. For instance, we use $(l = 3.9,\ w = 1.6, h = 1.56)$ meters for cars, $(l = 0.8, w = 0.6, h = 1.7)$ meters for pedestrians and $(l = 1.7, w = 0.6, h = 1.7)$ meters for cyclists on the KITTI dataset.

The anchors are associated with the ground-truth boxes by calculating the 2D bird-view Intersection-over-Union (IoU), where the positive IoU thresholds are empirically set as 0.6, 0.5, 0.5, and the negative IoU thresholds are 0.45, 0.35, 0.35 for cars, pedestrians and cyclists, respectively. We append two convolution layers with kernel size $1 \times 1 \times 1$ to the bird-view feature map for proposal classification and box regression. We use focal loss similarly to Eq. (2) for anchor scoring, and directly use the residual-based regression loss for the positive anchors. Here we directly adopt the commonly used smooth-$L1$ loss for regression since the center distances between anchors and their corresponding ground-truth boxes are generally within a smaller range than those of the anchor-free strategy due to the IoU thresholds. With the candidate anchor $(x^{(a)}, y^{(a)}, z^{(a)}, h^{(a)}, w^{(a)}, l^{(a)}, \theta^{(a)})$ and the target ground truth $(x^{(gt)}, y^{(gt)}, z^{(gt)}, h^{(gt)}, w^{(gt)}, l^{(gt)}, \theta^{(gt)})$, the residual-based box regression targets for center, angle and size are defined as

$$\begin{aligned}
&\Delta x^{(a)} = \frac{x^{(gt)} - x^{(a)}}{d^{(a)}},\ \Delta y^{(a)} = \frac{y^{(gt)} - y^{(a)}}{d^{(a)}},\ \Delta z^{(a)} = \frac{z^{(gt)} - z^{(a)}}{h^{(a)}},\\
&\Delta l^{(a)} = \log\left(\frac{l^{(gt)}}{l^{(a)}}\right),\ \Delta h^{(a)} = \log\left(\frac{h^{(gt)}}{h^{(a)}}\right),\ \Delta w^{(a)} = \log\left(\frac{w^{(gt)}}{w^{(a)}}\right),\\
&\Delta\theta^{(a)} = \sin\left(\theta^{(gt)} - \theta^{(a)}\right),\ \text{where } d^{(a)} = \sqrt{\left(l^{(a)}\right)^2 + \left(w^{(a)}\right)^2},
\end{aligned} \tag{7}$$

where the orientation target is encoded as $\sin(\theta^{(gt)} - \theta^{(a)})$ to eliminate the ambiguity of cyclic values of orientation. However, this method encodes two opposite directions to the same value, so we adopt an extra convolution layer with kernel size $1 \times 1 \times 1$ to the bird-view feature map as in [7] for classifying two opposite directions of orientation, where the direction target is calculated by the following approach: if $\theta^{(gt)}$ is positive, the direction target is one, otherwise the direction target is zero (Note that $\theta^{(gt)} \in [-\pi, \pi)$). We use cross entropy loss similarly to Eq. (3) for the binary classification of orientation direction, which is denoted as term $\mathcal{L}_{\text{dir}}$. Then the overall 3D bounding box regression loss $\mathcal{L}_{\text{box}}$ could be formulated as

$$\mathcal{L}_{\text{box}} = \sum_{\text{res} \in \{x,y,z,l,h,w,\theta\}} \mathcal{L}_{\text{smooth-L1}}(\widehat{\Delta\text{res}^{(a)}}, \Delta\text{res}^{(a)}) + \beta\mathcal{L}_{\text{dir}}, \tag{8}$$

where $\widehat{\Delta\text{res}^{(a)}}$ is the predicted residual for the candidate anchor, $\Delta\text{res}^{(a)}$ is the corresponding ground-truth target calculated as Eq. (7), and the loss weight $\beta = 0.1$. Note that the box regression loss $\mathcal{L}_{\text{box}}$ is only applied to the positive anchors.

Discussion of the Two 3D Proposal Generation Strategies. Both of these two 3D proposal generation strategies have their advantages and limitations. The proposed anchor-free strategy is generally light-weight and memory efficient because it does not requires evaluating a large number of anchors at each spatial location in the 3D space. The efficiency is more obvious for multi-class object detection since different classes in 3D object detection generally require different anchor boxes, while the anchor-free scheme can share the point-wise feature for generating proposals for multiple classes. The second anchor-based proposal generation strategy achieves slightly higher recall by covering the whole bird-view feature map with its predefined anchors for each class, but has more parameters and requires more GPU memory. The detailed experiments and comparison are discussed in Section 4.1.4.

3.2 RoI-Aware Point Cloud Feature Pooling

Given the predicted intra-object part locations and the 3D proposals, we aim to conduct box scoring and proposal refinement by aggregating the part information and learned point-wise features of all the points within the same proposal. In this subsection, we first introduce the canonical transformation to reduce the effects from the rotation and location variations of different 3D proposals, then we propose the RoI-aware point cloud feature pooling module to eliminate the ambiguity of previous point cloud pooling operation and to encode the position-specific features of 3D proposals for box refinement.

Canonical Transformation. We observe that if the box refinement targets are normalized in a canonical coordinate system, it can be better estimated by the following box refinement stage. We transform the pooled points belonging to each proposal to individual canonical coordinate systems of the corresponding 3D proposals. The canonical coordinate system for one 3D proposal denotes that (1) the origin is located at the center of the box proposal; (2) the local X' and Y' axes are approximately parallel to the ground plane with X' pointing towards the head direction of proposal and the other Y' axis perpendicular to X'; (3) the Z' axis remains the same as that of the global coordinate system. All pooled points' coordinates p of the box proposal should be transformed to the canonical coordinate system as $\tilde{p}$ by proper rotation and translation. The positive 3D proposals and their corresponding ground-truth 3D boxes are also transformed to the canonical coordinate system to calculate the residual regression targets for box refinement.

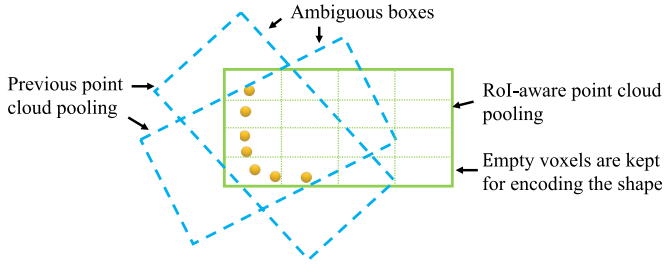


Fig. 6. Illustration of the proposed RoI-aware point cloud feature pooling. The previous point cloud pooling approach could not effectively encode the proposal's geometric information (blue dashed boxes). Our proposed RoI-aware point cloud pooling method could encode the box's geometric information (the green box) by keeping the empty voxels, which could be efficiently processed by following sparse convolution.

The proposed canonical coordinate system substantially eliminates much rotation and location variations of different 3D proposals and improves the efficiency of feature learning for later box location refinement. Although this transformation may cause the losing of occlusion information, we observe that the advantage out-weights the disadvantage and the deep neural network has large model capacity to handle different occlusion cases.

RoI-Aware Point Cloud Feature Pooling. The point cloud pooling operation in our preliminary work PointRCNN [32] simply pools the point-wise features from the 3D proposals whose corresponding point locations are inside the 3D proposal. All the inside points' features are aggregated by the PointNet++ encoder for refining the proposal in the second stage. However, we observe that this operation loses much 3D geometric information and introduces ambiguity between different 3D proposals. This phenomenon is illustrated in Fig. 6, where different proposals result in the same pooled points. The same pooled features introduces adverse effects to the following refinement stage.

Therefore, we propose the RoI-aware point cloud pooling module to evenly divide each 3D proposal into regular voxels with a fixed spatial shape $(L_x \times L_y \times L_z)$, where L_x, L_y, L_z are the integer hyperparameters of the pooling resolution in each dimension of the 3D proposals (e.g., $14 \times 14 \times 14$ is adopted in our framework) and independent of different 3D proposal sizes.

Specifically, let $\mathcal{F} = \{f_i \in \mathbb{R}^{C_0}, i \in 1, \ldots, n\}$ denote the point-wise features of all the inside points in a 3D proposal $\mathbf{b}$, and they are scattered in the divided voxels of the 3D proposal according to their local canonical coordinates $\mathcal{X} = \{(x_i^{(ct)}, y_i^{(ct)}, z_i^{(ct)}) \in \mathbb{R}^3, i \in 1, \ldots, n\}$, where n is the number of inside points. Then the RoI-aware voxel-wise max pooling and average pooling operations could be denoted as

$$Q = \text{RoIAwareMaxPool}(\mathcal{X},\ \mathcal{F},\ \mathbf{b}), \quad Q \in \mathbb{R}^{L_x \times L_y \times L_z \times C_0}, \tag{9}$$

$$Q = \text{RoIAwareAvgPool}(\mathcal{X},\ \mathcal{F},\ \mathbf{b}), \quad Q \in \mathbb{R}^{L_x \times L_y \times L_z \times C_0}, \tag{10}$$

where Q is the pooled 3D feature volumes of proposal $\mathbf{b}$. Specifically, the feature vector at the kth voxel Q_k of the voxel-wise max pooling and average pooling could be computed as

$$Q_k = \begin{cases} \max\{f_i \in \mathcal{N}_k\} & \text{if } |\mathcal{N}_k| > 0, \\ 0 & \text{if } |\mathcal{N}_k| = 0, \end{cases}$$
$$\text{or} \tag{11}$$
$$Q_k = \begin{cases} \frac{1}{|\mathcal{N}_k|}\sum_i f_i,\ f_i \in \mathcal{N}_k & \text{if } |\mathcal{N}_k| > 0, \\ 0 & \text{if } |\mathcal{N}_k| = 0, \end{cases}$$

where $\mathcal{N}_k$ is the set of points belonging to the kth voxel and $k \in \{1, \ldots, L_x \times L_y \times L_z\}$. Note that here the features of empty voxels ($|\mathcal{N}_k| = 0$) would be set to zeros and marked as empty for the following sparse convolution based feature aggregation.

The proposed RoI-aware point cloud pooling module encodes different 3D proposals with the same local spatial coordinates, where each voxel encodes the features of a corresponding fixed grid in the 3D proposal. This position-specific feature pooling better captures the geometry of the 3D proposal and results in an effective representation for the follow-up box scoring and location refinement. Moreover, the proposed RoI-aware point cloud pooling module is differentiable, which enables the whole framework to be end-to-end trainable.

3.3 Stage-II: Part Location Aggregation for Confidence Prediction and 3D Box Refinement

By considering the spatial distribution of the predicted intra-object part locations and the learned point-wise part features in a 3D box propsoal from stage-I, it is reasonable to aggregate all the information within a proposal for box proposal scoring and refinement. Based on the pooled 3D features, we train a sub-network to robustly aggregate information to score box proposals and refine their locations.

Fusion of Predicted Part Locations and Semantic Part Features. As shown in the right of Fig. 2, we adopt the proposed RoI-aware point cloud pooling module to obtain discriminative features of each 3D proposal. Let $\mathbf{b}$ denote a single 3D proposal, and for all of its inside points (with canonical coordinate $\mathcal{X} = \{(x_i^{(ct)}, y_i^{(ct)}, z_i^{(ct)}) \in \mathbb{R}^3, i \in 1, \ldots, n\}$), we denote $\mathcal{F}_1 = \{(x_i^{(part)}, y_i^{(part)}, z_i^{(part)}, s_i) \in \mathbb{R}^4, i \in 1, \ldots, n\}$ as their predicted point-wise part locations and semantic scores from stage-I, and denote $\mathcal{F}_2 = \{f_i^{(sem)} \in \mathbb{R}^C, i \in 1, \ldots, n\}$ as their point-wise semantic features learned by backbone network. Here n is the total number of inside points of proposal $\mathbf{b}$. Then the part feature encoding of proposal $\mathbf{b}$ could be formulated as follows

$$\begin{aligned} Q^{(part)} &= \text{RoIAwareAvgPool}(\mathcal{X},\ \mathcal{F}_1,\ \mathbf{b}), \\ Q^{(sem)} &= \text{RoIAwareMaxPool}(\mathcal{X},\ \mathcal{F}_2,\ \mathbf{b}), \\ Q_k^{(roi)} &= \left[G(Q_k^{(part)}),\ Q_k^{(sem)}\right], k \in \{1, \ldots, L_x \times L_y \times L_z\}, \end{aligned} \tag{12}$$

where G denotes a submanifold sparse convolution layer to transform the pooled part locations to the same feature dimensions C to match $Q^{(sem)}$, and $[\cdot, \cdot]$ denotes feature concatenation. Here $Q^{(part)}, Q^{(sem)}$ and $Q^{(roi)}$ have the same spatial shape $(14 \times 14 \times 14$ by default). The fused features $Q^{(roi)}$ encode both geometric and semantic information of the box proposals by the backbone network. Note that here we use the average pooling for pooling the predicted intra-object part

locations $\mathcal{F}_1$ to obtain representative predicted part location of each voxel of the proposal, while we use the max pooling for pooling the semantic part features $\mathcal{F}_2$.

Sparse Convolution for Part Information Aggregation. For each 3D proposal, we need to aggregate fused features $Q^{(roi)}$ from all inside spatial locations of this proposal for robust box scoring and refinement. As shown in the right part of Fig. 2, we stack several 3D sparse convolutional layers with kernel size $3 \times 3 \times 3$ to aggregate all part features of a proposal as the receptive filed increases. Here we also insert a sparse max-pooling with kernel size $2 \times 2 \times 2$ and stride 2 between the sparse convolutional layers to down-sample the feature volume to $7 \times 7 \times 7$ for saving the computation cost and parameters. Finally we vectorize it to a feature vector (empty voxels are kept as zeros) and feed it into two branches for box scoring and location refinement.

Compared with the naive method to directly vectorize the pooled 3D feature volume to a feature vector, our sparse convolution based part aggregation strategy could learn the spatial distribution of the predicted part locations effectively by aggregating features from local to global scales. The sparse convolution strategy also enables a larger $14 \times 14 \times 14$ pooling size by saving much computations, parameters and GPU memory.

3D IoU Guided Box Scoring. For the box scoring branch of stage-II, inspired by [35], [62], we normalize the 3D Intersectoin-over-Union (IoU) between 3D proposal and its corresponding ground truth box as the soft label for proposal quality evaluation. The proposal quality $q^{(a)}$ is defined as

$$q^{(a)} = \begin{cases} 1 & if \ \ \text{IoU} > 0.75, \\ 0 & if \ \ \text{IoU} < 0.25, \\ 2\text{IoU} - 0.5 & \text{otherwise}, \end{cases} \tag{13}$$

which is also supervised by a binary cross entropy loss $\mathcal{L}_{\text{score}}$ defined similarly to Eq. (3). Our experiments in Section 4.1.7 show that comparing with the traditional classification based box scoring, the IoU guided box scoring leads to slightly better performance.

3.4 Overall Loss

Our whole network is end-to-end trainable and the overall loss function is consist of the part-aware loss and part-aggregation loss.

Losses of Part-Aware Stage-I. For the part-aware stage-I, the loss function consists of three terms with equal loss weights, including focal loss for foreground point segmentation, binary cross entropy loss for the regression of part locations and smooth-$L1$ loss for 3D proposal generation,

$$\mathcal{L}_{\text{aware}} = \mathcal{L}_{\text{seg}} + \frac{1}{N_{\text{pos}}} \mathcal{L}_{\text{part}} + \lambda \frac{1}{M_{\text{pos}}} \mathcal{L}_{\text{box}}, \tag{14}$$

where loss weight $\lambda = 2.0$, N_{pos} is the total number of foreground points, $M_{\text{pos}} = N_{\text{pos}}$ for Part-A^2-free model and M_{pos} is the total number of positive anchors for Part-A^2-anchor model. For $\mathcal{L}_{\text{box}}$ loss, as mentioned in Section 3.1.3, we adopt the bin-based box generation loss for Part-A^2-free, and adopt the residual-based box regression loss for Part-A^2-anchor model.

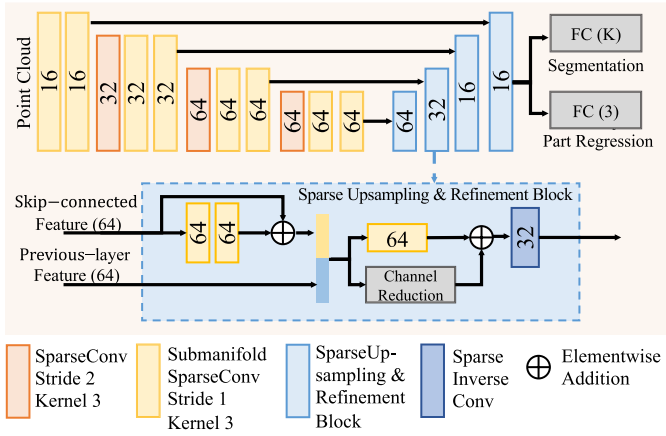


Fig. 7. The sparse convolution based encoder-decoder backbone network of part-aware stage-I of the Part-A^2-anchor model.

Losses of Part-Aggregation Stage-II. For the part-aggregation stage-II, the loss function includes a binary cross entropy loss term for box quality regression and a smooth-$L1$ loss term for 3D box proposal refinement,

$$\mathcal{L}_{\text{aggregation}} = \mathcal{L}_{\text{score}} + \frac{1}{T_{\text{pos}}} \mathcal{L}_{\text{box\_refine}}, \tag{15}$$

where T_{pos} is the number of positive proposals, and we conducted the residual-based box regression loss for $\mathcal{L}_{\text{box\_refine}}$ as used in Eq. (7), which includes box center refinement loss, size refinement loss and angle refinement loss. Besides that, we also add the corner regularization loss $\mathcal{L}_{\text{corner}}$ as used in [6], and the final box refinement loss is as follows

$$\mathcal{L}_{\text{box\_refine}} = \sum_{\text{res} \in \{x,y,z,l,h,w,\theta\}} \mathcal{L}_{\text{smooth-L1}}(\widehat{\Delta \text{res}^{(r)}}, \Delta \text{res}^{(r)}) + \mathcal{L}_{\text{corner}}, \tag{16}$$

where $\widehat{\Delta \text{res}^{(r)}}$ is the predicted residual for the 3D proposal, $\Delta \text{res}^{(r)}$ is the corresponding ground-truth target calculated similarly to Eq. (7), and all losses here have the same loss weights. Note that here the angle refinement target is directly encoded as $\Delta\theta^{(r)} = (\theta^{(gt)} - \theta^{(r)})$ since the angle difference between proposals and their corresponding ground-truth boxes are within a small range due to the IoU constraint for positive proposals.

Overall Loss. Hence the overall loss function of our Part-A^2 net for end-to-end training is calculated as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{aware}} + \mathcal{L}_{\text{aggregation}}, \tag{17}$$

where the losses of these two stages have equal loss weights.

3.5 Implementation Details

We design a UNet-like architecture [45] for learning point-wise feature representations with 3D sparse convolution and 3D sparse deconvolution on the obtained sparse voxels. The spatial resolution is downsampled 8 times by three sparse convolutions of stride 2, each of which is followed by several submanifold sparse convolutions. As illustrated in Fig. 7, we also design a similar up-sampling block as that in [63] based on sparse operations for refining the fused features.

Network Details. As shown in Fig. 7, for the part-aware stage-I of Part-A^2-anchor model, the spatial feature volumes have four scales with feature dimensions 16-32-64-64, and we use three 3D sparse convolution layers with kernel size $3 \times 3 \times 3$ and stride 2 to downsample the spatial resolution by 8 times. We stack two submanifold convolution layers in each level with kernel size $3 \times 3 \times 3$ and stride 1. There are four sparse up-sampling blocks (see Fig. 7 for network details) in the decoder to gradually increase feature dimensions as 64-64-32-16. Note that the stride of the last up-sampling block is 1 and the stride of other three up-sampling blocks is 2. For the Part-A^2-free net, we increase the feature dimensions of decoder to 128 in each scale and use simple concatenation to fuse the features from the same level of encoder and the previous layer of decoder, since the learned point-wise features of decoder should encode more discriminative features for the bottom-up 3D proposal generation.

For the part-aggregation stage, as shown in Fig. 2, the pooling size of RoI-aware point cloud pooling module is $14 \times 14 \times 14$, which is downsampled to $7 \times 7 \times 7$ after processed by the sparse convolutions and max-pooling with feature dimensions 128. We vectorize the downsampled feature volumes to a single feature vector for the final box scoring and location refinement.

Training and Inference Details. We train the entire network end-to-end with the ADAM optimizer and a batch size 6 for 50 epochs. The cosine annealing learning rate strategy is used with an initial learning rate 0.001. We randomly select 128 proposals from each scene for training stage-II with $1:1$ positive and negative proposals, where the positive proposals for box refinement have 3D IoU with their corresponding ground truth box of at least 0.55 for all classes, otherwise they are negative proposals, and the scoring target is defined as Eq. (13) for the confidence prediction.

We conduct common data augmentation during training, including random flipping, global scaling with scaling factor uniformly sampled from $[0.95, 1.05]$, global rotation around the vertical axis by an angle uniformly sampled from $[-\frac{\pi}{4}, \frac{\pi}{4}]$. In order to simulate objects with various environment as [7], we also randomly "copy" several ground-truth boxes with their inside points from other scenes and "paste" them to current training scenes. The whole training process of our proposed part-aware and part-aggregation networks takes about 17 hours on a single NVIDIA Tesla V100 GPU.

For inference, only 100 proposals are kept from part-aware stage-I with NMS threshold 0.85 for Part-A^2-free and 0.7 for Part-A^2-anchor, which are then scored and refined by the following part-aggregation stage-II. We finally apply the rotated NMS with threshold 0.01 to remove redundant boxes and generate the final 3D detection results. The overall inference time is about 70ms on a single Tesla V100 GPU card.

3.6 Pros and Cons

Our proposed 3D object detection framework has some advantages and disadvantages under different situations.

Compared with the previous 3D object detection methods [1], [4], [5], [6], [7], [26], [29], [64], (1) our proposed method for the first time introduces the learning of intra-object part locations to improve the performance of 3D object detection from point cloud, where the predicted intra-object part locations effectively encode the point distribution of 3D objects to benefit the 3D object detection. (2) The proposed RoI-aware feature pooling module eliminates the ambiguity of previous point cloud pooling operations and transforms the sparse point-wise features to the regular voxel features to encode the position-specific geometry and semantic features of 3D proposals, which effectively bridges the proposal generation network and the proposal refinement network, and results in high detection accuracy. (4) Besides, the learning process of part locations could also be adopted to other tasks for learning more discriminative point-wise features, such as the instance segmentation of point cloud. The proposed RoI-aware pooling module could also be flexibly utilized on transforming the point-wise features from point-based networks (such as PointNet++) to the sparse voxel-wise features, that could be processed by more efficient sparse convolution networks.

On the other hand, our method also has some limitations. Since our method aims at high performing 3D object detection in autonomous driving scenarios, some parts of our method could not be well applied for the 3D object detection of indoor scenes. This is because the 3D bounding boxes in indoor scenes may overlap with each other (such as chairs under the table), therefore the 3D bounding box annotations of indoor scenes could not provide the accurate point-wise segmentation labels. Also, there are some categories whose orientation is not well-defined (such as the round tables), hence we could not generate accurate labels of the proposed intra-object part locations.

However, our proposed anchor-free proposal generation strategy still shows great potential on the 3D proposal generation of indoor scenes since the indoor objects do not always stay on the ground and our anchor-free strategy avoids to set 3D anchors in the whole 3D space.

4 Experiments

In this section, we evaluate our proposed method with extensive experiments on the challenging 3D detection benchmark of KITTI [33] dataset. In Section 4.1, we present extensive ablation studies and analysis to investigate individual components of our models. In Section 4.2, we demonstrate the main results of our methods by comparing with state-of-the-art 3D detection methods. Finally we visualize some qualitative results of our proposed 3D detection model in Section 4.3.

Dataset. There are 7481 training samples and 7518 test samples in the dataset of KITTI 3D detection benchmark. The training samples are divided into the *train* split (3712 samples) and *val* split (3769 samples) as the frequently used partition of KITTI dataset. All models are only trained on the *train* split, and evaluated on the *val* and *test* splits.

Models. There are three main models in our experiments, i.e., Part-A^2-free model, Part-A^2-anchor model and our preliminary PointRCNN model [32]. The network details of Part-A^2-free and Part-A^2-anchor models have been demonstrated in Section 3, and the whole framework is illustrated in Fig. 2. As discussed in Section 3.1.3, the key differences between these two versions of Part-A^2 models is that Part-A^2-free generates 3D proposals in the bottom-up (anchor-free) manner, while the Part-A^2-anchor net generates 3D proposals with the proposed anchor-based scheme. PointRCNN is in the preliminary version of this work [32]. It utilizes PointNet++ [28] to

TABLE 2
Effects of RoI-Aware Point Cloud Pooling by Replacing the RoI-Aware Pooling or the Sparse Convolution, and the Pooling Sizes of all the Settings are $14 \times 14 \times 14$

Pooling Scheme	Stage-II	AP_{Easy}	$AP_{Mod.}$	AP_{Hard}
RoI fixed-sized pool	sparse conv	88.78	78.61	78.05
RoI-aware pool	FCs	89.46	79.32	**78.77**
RoI-aware pool	sparse conv	**89.47**	**79.47**	78.54

The results are the 3D detection performance of car class on the val *split of KITTI dataset.*

TABLE 3
Effects of Using Different RoI-Aware Pooling Sizes in Our Part-Aggregation Stage

RoI Pooling Size	AP_{Easy}	$AP_{Mod.}$	AP_{Hard}
$6 \times 6 \times 6$	89.02	78.85	78.04
$8 \times 8 \times 8$	89.09	78.97	78.15
$10 \times 10 \times 10$	89.44	79.15	78.42
$12 \times 12 \times 12$	**89.61**	79.35	78.50
$14 \times 14 \times 14$	89.47	**79.47**	78.54
$16 \times 16 \times 16$	89.52	79.45	**78.56**

The results are the 3D detection performance of car class on the val *split of KITTI dataset.*

extract point-wise features, which are used to generate 3D proposals in a bottom-up manner via segmenting the foreground points as demonstrated in Section 3.1.3. Furthermore, in the stage-II of PointRCNN, we pool the inside points and their point-wise features for each 3D proposals, which are then fed to a second PointNet++ encoder to extract features of each 3D proposal for proposal confidence prediction and 3D box proposal refinement.

4.1 From Points to Parts: Ablation Studies for Part-A^2 Net

In this section, we provide extensive ablation experiments and analysis to investigate the individual components of our proposed Part-A^2 net models.

4.1.1 SparseConvUNet versus PointNet++ Backbones for 3D Point-Wise Feature Learning

As mentioned in Section 3.1, instead of utilizing PointNet++ as the backbone network, we design a sparse convolution based UNet (denoted as SparseConvUNet) for point-wise feature learning, and the network details are illustrated in Fig. 7. We first compare the PointRCNN with PointNet++ backbone and Part-A^2-free with SparseConvUNet backbone with the same loss functions to test these two different backbone networks.

Table 1 shows that our SparseConvUNet based Part-A^2-free ($2nd$ row) achieves 81.54 percent recall with 3D IoU threshold 0.7, which is 6.73 percent higher than the recall of PointNet++ based PointRCNN ($1st$ row), and it demonstrates that our new designed SparseConvUNet could learn more discriminative point-wise features from the point cloud for the 3D proposal generation. As shown in Table 5, we also provide the recall values of different number of proposals for these two backbones. We could find the recall of the sparse convolution based backbone consistently outperforms the recall of the PointNet++ based backbone, which further validates that the sparse convolution based backbone is better than the PointNet++ based backbone for point-wise feature learning and 3D proposal generation.

Tables 1 and 5 also show that our Part-A^2-anchor model achieves higher recall than the Part-A^2-free model. Therefore, in our remaining experimental sections, we mainly adopt the Part-A^2-anchor model for ablation studies and experimental comparison unless specified otherwise.

4.1.2 Ablation Studies for RoI-Aware Point Cloud Pooling

In this section, we designed ablation experiments to validate the effectiveness of our proposed RoI-aware point cloud pooling module with the Part-A^2-anchor model, and we also explored more pooling sizes to investigate the trend of performance when increasing the RoI pooling size.

Effects of RoI-Aware Point Cloud Region Pooling. As discussed in Section 3.2, the proposed RoI-aware point cloud pooling module normalizes different 3D proposals to the same coordinate system to encode geometric information of proposals. It solves the ambiguous encoding by previous 3D point cloud pooling schemes as shown in Fig. 6. The 3D proposals are divided into regular voxels to encode the position-specific features for each 3D proposal.

To validate the effects of the RoI-aware pooling module, we conduct the following comparison experiments. (a) We replace RoI-aware pooling by fixed-sized RoI pooling, i.e. pooling all 3D proposals with the same fixed-size ($l = 3.9$, $w = 1.6, h = 1.56$ meters for car) 3D box calculated from the mean object size of the training set with $14 \times 14 \times 14$ grids. The center and orientation of the 3D grid are set as its corresponding 3D proposal's center and orientation, respectively. This is very similar to the pooling scheme used in PointRCNN, where not all geometric information is well preserved during pooling. (b) We replace sparse convolutions of stage-II with several FC layers. As shown in Table 2, removing RoI-aware pooling substantially decreases detection accuracy, while replacing sparse convolutions of stage-II with FC layers achieves similar performance, which proves the effectiveness of our proposed RoI-aware pooling but not the sparse convolution contributes to the main improvements.

Effects of RoI Pooling Size. The 14×14 pooling size was very commonly chosen for 2D object detection, and we follow the same setting to use $14 \times 14 \times 14$ as the 3D RoI-aware pooling size. We also test different RoI pooling sizes as shown in Table 3. The pooling size shows robust performance for different 3D objects. Similar performances can be observed if the pooling sizes are greater than $12 \times 12 \times 12$.

4.1.3 Sparse Convolution versus Fully-Connected Layers for Part Aggregation

In our Part-A^2 net, after applying the RoI-aware point cloud pooling module, there are several ways to implement the part-aggregation stage. The simplest strategy is to directly vectorize the pooled feature volumes to a feature vector followed by several fully-connected layers for box scoring and refinement. From the $1st$ row of Table 4, we could see that this naive way already achieved promising results, which are benefited from the effective representations of our RoI-aware point cloud pooling since each position of the feature

TABLE 4
Comparison of Several Different Part-Aggregation Network Structures

Pooling Size	Stage-II	Downsampling in Stage-II	AP_{Easy}	$AP_{Mod.}$	AP_{Hard}
$7 \times 7 \times 7$	FCs		89.17	79.11	78.03
$7 \times 7 \times 7$	sparse conv		89.24	79.21	78.11
$14 \times 14 \times 14$	FCs		89.46	79.32	**78.77**
$14 \times 14 \times 14$	sparse conv	✓	**89.47**	**79.47**	78.54

The results are the 3D detection performance of car class on the val *split of KITTI dataset.*

TABLE 6
3D Object Detection Results of Part-A^2-Free Net and Part-A^2-Anchor Net on the KITTI *val* Split Set

Method	Class	IoU Thresh	AP_{Easy}	$AP_{Mod.}$	AP_{Hard}
Part-A^2-free	Car	0.7	88.48	78.96	78.36
Part-A^2-anchor	Car	0.7	**89.47**	**79.47**	**78.54**
Part-A^2-free	Cyclist	0.5	88.18	**73.35**	**70.75**
Part-A^2-anchor	Cyclist	0.5	**88.31**	73.07	70.20
Part-A^2-free	Pedestrian	0.5	**70.73**	**64.13**	57.45
Part-A^2-anchor	Pedestrian	0.5	70.37	63.84	**57.48**

vector encodes a specific intra-object position of the object of interest to help learn the shape of the box better. In the $2nd$ row of Table 4, we further investigate using sparse convolution with kernel size $3 \times 3 \times 3$ to aggregate the features from local to global scales gradually, which achieves slightly better results with the same pooling size $7 \times 7 \times 7$. The $3rd$ row shows that fully-connected layers with larger pooling size $14 \times 14 \times 14$ achieves improved performance, but this design consumes much calculations and GPU memory. As we mentioned in Section 3.3, our proposed part-aggregation network adopts a large pooling size $14 \times 14 \times 14$ to capture details and then use sparse max-pooling to downsample the feature volumes for feature encoding, which achieves the best performance in the easy and moderate difficulty levels as shown in Table 4 with lower computation and GPU memory cost than fully-connected layers.

4.1.4 Ablation Studies for 3D Proposal Generation

We investigate the two strategies for 3D proposal generation from point cloud, one is the anchor-based strategy and the other is our proposed anchor-free strategy, in Part-A^2-anchor and Part-A^2-free models. In this section, we first experiment and discuss two proposal generation strategies in details to provide a reference to choose better strategy for different settings of 3D proposal generation. Then we compare the performance of different center regression looses for the two strategies.

Anchor-Free versus Anchor-Based 3D Proposal Generation. We validate the effectiveness of our proposal generation strategies with state-of-the-art two-stage 3D detection methods. As shown in Table 5, our preliminary PointRCNN with anchor-free proposal generation and PointNet++ backbone already achieve significantly higher recall than previous methods. With only 50 proposals, PointRCNN obtains 96.01 percent recall at IoU threshold 0.5, which outperforms recall 91 percent of AVOD [4] by 5.01 percent at the same number of proposals, note that the latter method uses both 2D image and point cloud for proposal generation while we only use point cloud as input.

TABLE 5
Recall of Generated Proposals by Compared Methods With Different Numbers of RoIs and 3D IoU Thresholds for the Car Class at Moderate Difficulty of the *Val* Split

RoIs #	Recall (IoU=0.5)			Recall (IoU=0.7)		
	MV3D	AVOD	PointRCNN	PointRCNN	Part-A^2 -free	Part-A^2-anchor
10	-	86.00	**86.66**	29.87	66.31	**80.68**
20	-	-	**91.83**	32.55	74.46	**81.64**
30	-	-	**93.31**	32.76	76.47	**82.90**
40	-	-	**95.55**	40.04	76.88	**83.05**
50	-	91.00	**96.01**	40.28	77.12	**83.71**
100	-	-	**96.79**	74.81	81.54	**85.58**
200	-	-	**98.03**	76.29	84.93	**89.32**
300	91.00	-	**98.21**	82.29	86.03	**91.64**

Note that only MV3D [1] and AVOD [4] of previous methods reported the recall rates of proposals.

We also report the recall of 3D bounding box at IoU threshold 0.7 by our anchor-free and anchor-based strategies in Table 5. Part-A^2-free model (with anchor-free proposal generation strategy) achieves 77.12 percent recall at IoU threshold 0.7 with only 50 proposals, which is much higher than the recall of our preliminary work PointRCNN since Part-A^2-free model adopts better sparse convolution based backbone. Our Part-A^2-anchor model (with anchor-based proposal generation strategy) further improves the recall to 83.71 percent at IoU threshold 0.7 with 50 proposals. This is because the anchor-based strategy has a large number of anchors to more comprehensively cover the entire 3D space to achieve a higher recall. However, the improvement comes with sacrifices, as it needs different sets of anchors for different classes at each spatial location. For instance, the anchor size of pedestrians is $(l = 0.8m, w = 0.6m, h = 1.7m)$ while the anchor size of cars is $(l = 3.9m, w = 1.6m, h = 1.56m)$. They are unlikely to share the same anchor. In contrast, our anchor-free strategy still generates a single 3D proposal from each segmented foreground point even for many classes, since we only need to calculate the 3D size residual with respect to the corresponding average object size based on its semantic label.

The 3D detection results of the Part-A^2-free and Part-A^2-anchor models on cars, cyclists and pedestrians are reported in Table 6. We could see that the 3D detection results of cyclist and pedestrian by Part-A^2-free are comparable to those by Part-A^2-anchor model, while the results of cars by Part-A^2-free are lower than those by Part-A^2-anchor on the moderate and easy difficulties. Hence the bottom-up Part-A^2-free model has better potential on multi-class 3D detection on small-size objects (such as cyclists and pedestrians) with lower memory cost, while the anchor-based Part-A^2-anchor model may achieve a slightly better performance on 3D detection of large-size objects such as cars. That is because the predefined anchors are closer to the center locations of objects with large sizes, while the bottom-up proposal generation strategy suffers from difficulty of regressing large residuals from the object surface points to object centers.

Center Regression Losses of 3D Bounding Box Generation. We compare different center regression losses on our Part-A^2-free net and our Part-A^2-anchor net, including the proposed

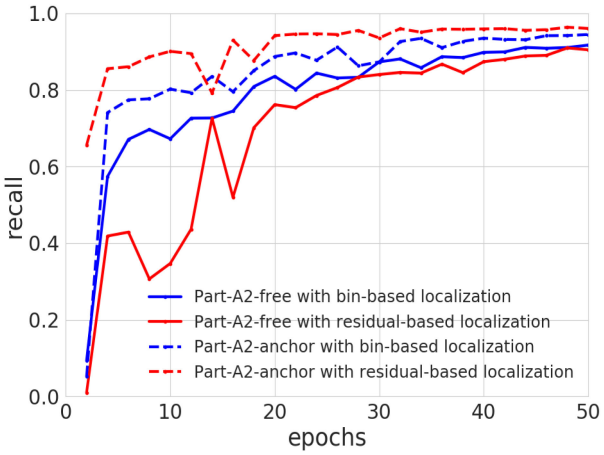


Fig. 8. Recall versus training iterations for different center regression losses of Part-A^2-free net and Part-A^2-anchor net. The results are generated according to the generated proposals of Part-A^2-free net and Part-A^2-anchor net on the car class of the *val* split with IoU threshold 0.5.

bin-based regression loss (Eq. (4)) and the residual-based regression loss (first row of Eq. (7)). As shown in Fig. 8, for the Part-A^2-free net with anchor-free proposal generation strategy, the bin-based regression loss (solid blue line) converges faster than the residual-based regression loss (solid red line). In contrast, for the Part-A^2-anchor net with anchor-based proposal generation scheme, the residual-based regression loss (dashed red line) converges faster and better than the bin-based regression loss (dashed blue line). It demonstrates that the proposed bin-based center regression loss is more suitable with the anchor-free proposal generation strategy to achieve better performance, since the center regression targets of anchor-free scheme (generally from a surface point to the object center) vary a lot and bin-based localization could better constrain the regression targets and make the convergence faster and more stable. Fig. 8 shows that the Part-A^2-anchor net achieves better recall with the residual-based center regression loss and we also adopt residual-based center localization loss for the Part-A^2-anchor net as mentioned in Section 3.1.3.

4.1.5 Benefits of Intra-Object Part Location Prediction for 3D Object Detection

To validate the effectiveness of utilizing the free-of-charge intra-object part locations for 3D detection, we test removing the part location supervisions from our Part-A^2-anchor model. In the backbone network, we only remove the branch for predicting intra-object part locations and keep other modules unchanged. The point-wise part locations for RoI-aware pooling are replaced with the canonical coordinates of each point.

TABLE 7
Effects of Intra-Object Part Location Supervisions and Stage-II Refinement Module, and the Evaluation Metrics are the Recall and Average Precision with 3D Rotated IoU Threshold 0.7

Stage-I	Stage-II	Part Prediction	Recall box#100	AP_{Easy}	$AP_{Mod.}$	AP_{Hard}
✓			80.90	88.48	77.97	75.84
✓		✓	80.99	88.90	78.54	76.44
✓	✓		82.92	89.23	79.00	77.66
✓	✓	✓	**84.33**	**89.47**	**79.47**	**78.54**

The results are the 3D detection performance of car class on the val *split of KITTI dataset, and the detection results of stage-I are generated by directly applying NMS to the box proposals from stage-I.*

TABLE 8
Effects of 3D IoU Guided Box Scoring for Ranking the Quality of the Predicted 3D Boxes

NMS Ranking Score	AP_{Easy}	$AP_{Mod.}$	AP_{Hard}
classification	89.13	78.81	77.95
3D IoU guided scoring	**89.47**	**79.47**	**78.54**

The results are the 3D detection performance of car class on the val *split of KITTI dataset.*

As shown in Table 7, compared with the model trained without intra-object part location supervisions (the 3rd versus the 4th rows), the models with part location supervisions achieves better recall and average precision on all difficulty levels of the *val* split of the car class. The remarkable improvements on recall and precision indicate that the network learned better 3D features for scoring box and refining locations for detection with detailed and accurate supervisions of the intra-object part locations.

4.1.6 One-Stage versus Two-Stage 3D Object Detection

Table 7 shows that without the stage-II for box scoring and refinement, the proposal recalls of our first proposal stage are comparable (80.90 versus 80.99). However, the performance improves significantly (82.92 versus 84.33) after the 100 proposals are refined by the part-aggregation stage. It demonstrates that the predicted intra-object part locations are beneficial for stage-II, and our part-aggregation stage-II could effectively aggregate the predicted intra-object part locations to improve the quality of the predicted 3D boxes. The performance gaps between stage-I and stage-II ($1st$ row versus $3rd$ row, $2nd$ row versus 4th row) also demonstrate that our stage-II improves the 3D detection performance significantly by re-scoring the box proposals and refining their box locations.

4.1.7 Effects of IoU Guided Box Scoring

As mentioned in Section 3.3, we apply the normalized 3D IoU to estimate the quality of the predicted 3D boxes, which is used as the ranking score in the final NMS (non-maximum-suppression) operation to remove the redundant boxes. Table 8 shows that comparing with the traditional classification score for NMS, our 3D IoU guided scoring method increases the performance marginally in all difficulty levels, which validates the effectiveness of using normalized 3D IoU to indicate the quality of predicted 3D boxes.

4.1.8 Memory Cost of Anchor-Free and Anchor-Based Proposal Generation

As shown in Table 9, we compare the model complexity of the anchor-free and anchor-based proposal generation strategies by calculating the number of parameters and the number of generated boxes with different number of object classes. Part-A^2-free model (with anchor-free proposal generation strategy) consistently generates ~16k proposals (i.e., the number

TABLE 9
The Number of Parameters on Proposal Generation Head, and the Number of Generated Boxes With Different Number of Classes for Part-A^2-Free Model and Part-A^2-Anchor Model

Number of classes	Part-A^2-free		Part-A^2-anchor	
	Number of parameters	Number of generated boxes	Number of parameters	Number of anchors
1	1775269	~16k	4648588	70.4k
3	1775527	~16k	4662952	211.2k

The parameters of (5, 10, 100) are counted by setting faked number of classes and the number of generated boxes are for the KITTI scene.

of points of the point cloud), which is independent with the number of classes, while the number of generated boxes (i.e., the predefined anchors) of Part-A^2-anchor model (with anchor-based proposal generation), increases linearly with the number of classes since each class has its own anchors with specified object sizes for each class. The number of anchors of Part-A^2-anchor model achieves 211.2k for detecting objects of 3 classes, which shows that our anchor-free proposal generation strategy is a relatively light-weight and memory efficient strategy especially for multiple classes.

We also report the inference GPU memory cost for three classes detection (car, pedestrian and cyclist) on KITTI [33] dataset. The inference is conducted by PyTorch framework on a single NVIDIA TITAN Xp GPU card. For the inference of a single scene, Part-A^2-free model consumes about 1.16 GB GPU memory while Part-A^2-anchor model consumes 1.63 GB GPU memory. For the inference with six scenes simultaneously, Part-A^2-free model consumes about 3.42 GB GPU memory while Part-A^2-anchor model consumes 5.46 GB GPU memory. It demonstrates that the Part-A^2-free model (with anchor-free proposal generation strategy) is more memory efficient than Part-A^2-anchor model (with anchor-based proposal generation).

4.1.9 Analysis of False Positive Samples

Fig. 9 shows the ratios of false positives of our best performance Part-A^2-anchor model on the KITTI validation dataset with different score thresholds, which are caused by confusion with background, poor localization, and confusion with objects from other categories. It can be seen that, the majority of false positives are from background and poor localization. The confusion of background mainly comes from the fact that the sparse point cloud could not provide enough semantic information for some background like the flower terrace. The LiDAR-only 3D detection methods may mistakenly recognize them as foreground objects like car since they have similar geometry shape in the point cloud. The ratio of false positives from poor localization increases significantly with the increasing score threshold. This is because the evaluation requirement of 3D rotated IoU constraint for 3D detection is more strict than the evaluation metric of 2D detection.

4.2 Main Results and Comparison With State-of-the-Arts on KITTI Benchmark

In this section, we report the comparison results with state-of-the-art 3D detection methods on the KITTI benchmark.

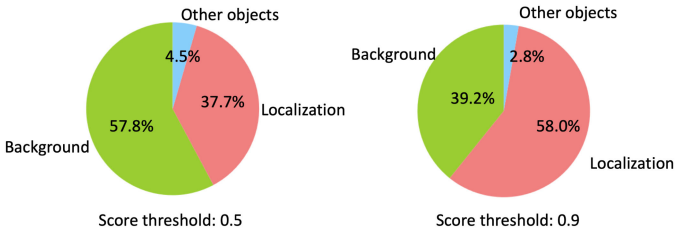


Fig. 9. Ratios of high-scored false positives for the car class on the *val* split of KITTI dataset that are due to poor localization, confusion with other objects, or confusion with background or unlabeled objects.

We mainly report the performance of our Part-A^2-anchor model as it is able to reach higher accuracy in our ablation studies.

Comparison With State-of-the-art 3D Detection Methods. We evaluate our methods on the 3D detection benchmark and the bird's eye view detection benchmark of the KITTI test split, whose results are evaluated on KITTI's offical test server. The results are shown in Table 10.

For the 3D object detection benchmark, by only using LiDAR point clouds, our proposed Part-A^2 net outperforms all previous peer-reviewed LiDAR only methods on all difficulty levels for all the three classes, and outperforms all previous multi-sensor methods on the most important "moderate" difficulty level for both car and cyclist classes. For the bird's view detection of car, pedestrian and cyclist, our method outperforms previous state-of-the-art methods by large margins on almost all the difficulty levels. As of August 15, 2019, our proposed Part-A^2-anchor net ranks $1st$ place among all methods on the most important car class of 3D object detection leaderboard of KITTI 3D Object Detection Benchmark [65], while our method also ranks $1st$ among all LiDAR-only methods on the cyclist class.

Results on Validation Set. For the most important car category, our methods are compared with state-of-the-art methods on KITTI *val* split including both 3D object detection (shown in Table 11) and 3D object localization (shown in Table 12). We could see that on the most important "moderate" difficulty level, our Part-A^2 net outperforms state-of-the-art methods on both two tasks with large margins by using only the point clouds as inputs. In addition, our Part-A^2 net achieves new state-of-the-art performance on all difficulty levels of the KITTI 3D object detection *val* split, which demonstrates the effectiveness of our proposed methods for 3D object detection.

As shown in Table 13, we also report the performance of our methods for cyclist and pedestrian on the validation set for reference. Note that compared with PointRCNN, our latest method Part-A^2-anchor net improves the performance of cyclist significantly while achieves comparable results on pedestrian. The reason for slightly inferior performance on pedestrians might be that the orientation of pedestrian is hard to be recognized from the sparse point cloud, which is harmful for the prediction of part locations in our Part-A^2-anchor net. Multi-sensor methods that integrate RGB images would have advantages for detecting small objects like pedestrians.

Evaluation of Part-A^2-Anchor Net for Predicting Intra-Object Part Locations. The intra-object part locations predicted by

TABLE 10
Performance Evaluation on KITTI Official Test Server (*Test* Split)

Method	Modality	3D Detection (Car)			BEV Detection (Car)			3D Detection (Ped.)			BEV Detection (Ped.)			3D Detection (Cyc.)			BEV Detection (Cyc.)		
		Mod.	Easy	Hard	Mod.	Easy	Hard	Mod.	Easy	Hard	Mod.	Easy	Hard	Mod.	Easy	Hard	Mod.	Easy	Hard
MV3D [1]	RGB + LiDAR	62.35	71.09	55.12	76.90	86.02	68.49	-	-	-	-	-	-	-	-	-	-	-	-
ContFuse [5]	RGB + LiDAR	66.22	82.54	64.04	85.83	88.81	77.33	-	-	-	-	-	-	-	-	-	-	-	-
AVOD-FPN [4]	RGB + LiDAR	71.88	81.94	66.38	83.79	88.53	77.90	42.81	50.80	40.88	51.05	58.75	47.54	52.18	64.00	46.61	57.48	68.09	50.77
F-PointNet [6]	RGB + LiDAR	70.39	81.20	62.19	84.00	88.70	75.33	44.89	51.21	40.23	50.22	58.09	47.20	56.77	71.96	50.39	61.96	75.38	54.68
PC-CNN-V2 [8]	RGB + LiDAR	73.80	84.33	64.83	86.10	88.49	77.26	-	-	-	-	-	-	-	-	-	-	-	-
UberATG-MMF [64]	RGB + LiDAR	76.75	86.81	68.41	87.47	89.49	79.10	-	-	-	-	-	-	-	-	-	-	-	-
VoxelNet [29]	LiDAR only	65.11	77.47	57.73	79.26	89.35	77.39	33.69	39.48	31.51	40.74	46.13	38.11	48.36	61.22	44.37	54.76	66.70	50.55
SECOND [7]	LiDAR only	73.66	83.13	66.20	79.37	88.07	77.95	42.56	51.07	37.29	46.27	55.10	44.76	53.85	70.51	40.90	56.04	73.67	48.78
PointPillars [9]	LiDAR only	74.99	79.05	68.30	**86.10**	88.35	79.83	43.53	52.08	41.49	50.23	58.66	47.19	59.07	75.78	52.92	62.25	79.14	56.00
PointRCNN (Ours)	LiDAR only	75.76	**85.94**	68.32	85.68	89.47	79.10	41.78	49.43	38.63	47.53	55.92	44.67	59.60	73.93	53.59	66.77	81.52	60.78
Part-A^2-anchor (Ours)	LiDAR only	**77.86**	**85.94**	**72.00**	84.76	**89.52**	**81.47**	**44.50**	**54.49**	**42.36**	**51.12**	**59.72**	**48.04**	**62.73**	**78.58**	**57.74**	**68.12**	**81.91**	**61.92**

The 3D object detection and bird's eye view detection are evaluated by mean average precision with 11 recall positions. The rotated IoU threshold is 0.7 for car and 0.5 for pedestrian/cyclist.

our part-aware stage-I are crucial for the part-aggregation stage-II to accurately score the box and refine the box location. Here we evaluate the accuracy of predicted intra-object part locations by the following metric:

$$\text{AbsError}_u = \frac{1}{||\mathcal{G}||}\sum_{i\in\mathcal{G}} |\tilde{u}_i^{(part)} - u_i^{(part)}|, u \in \{x, y, z\}, \tag{18}$$

where $\tilde{u}_i^{(part)}$ is the predicted part location, $u_i^{(part)}$ is the ground truth part location, and $\mathcal{G}$ is the set of foreground points for each sample. The final mAbsError_u is the mean value of AbsError_u for all samples.

As shown in Table 14, for the most important car category, the mean error of our predicted intra-object part locations is 6.28 percent, which shows that the part-aware network accurately predicts the intra-object part locations since the average error is only ± 6.28 cm per meter for the cars. Based on this accurate intra-object part locations, our part-aggregation stage-II could better score the boxes and refine the box locations by utilizing the predicted geometric information. Here we also report the detailed error statistics of predicted intra-object part locations on different difficulty levels of the KITTI *val* split in Fig. 10 for reference.

We further analyze the correlations between the errors of the predicted intra-object part locations and the errors of the predicted 3D bounding boxes by calculating the Pearson correlation coefficient, which is $[-1, 1]$ with 1 denotes fully positive linear correlation and -1 is fully negative linear correlation. Here we utilize $1 - \text{IoU}$ to indicate the errors of the predicted 3D bounding boxes, where

TABLE 11
Performance Comparison of 3D Object Detection on the Car Class of the KITTI *val* Split Set

Method	Reference	Modality	AP (IoU=0.7)		
			Mod.	Easy	Hard
MV3D [1]	CVPR 2017	RGB & LiDAR	62.68	71.29	56.56
ContFuse [5]	ECCV 2018	RGB & LiDAR	73.25	86.32	67.81
AVOD-FPN [4]	IROS 2018	RGB & LiDAR	74.44	84.41	68.65
F-PointNet [6]	CVPR 2018	RGB & LiDAR	70.92	83.76	63.65
VoxelNet [29]	CVPR 2018	LiDAR only	65.46	81.98	62.85
SECOND [7]	Sensors 2018	LiDAR only	76.48	87.43	69.10
PointRCNN (Ours)	CVPR 2019	LiDAR only	78.63	88.88	77.38
Part-A^2-free (Ours)	-	LiDAR only	78.96	88.48	78.36
Part-A^2-anchor (Ours)	-	LiDAR only	**79.47**	**89.47**	**78.54**

TABLE 12
Performance Comparison of Bird-View Object Detection on the Car Class of the KITTI *val* Split Set

Method	Reference	Modality	AP (IoU=0.7)		
			Mod.	Easy	Hard
MV3D [1]	CVPR 2017	RGB & LiDAR	78.10	86.55	76.67
F-PointNet [6]	CVPR 2018	RGB & LiDAR	84.02	88.16	76.44
ContFuse [5]	ECCV 2018	RGB & LiDAR	87.34	95.44	82.43
VoxelNet [29]	CVPR 2018	LiDAR only	84.81	89.60	78.57
SECOND [7]	Sensors 2018	LiDAR only	87.07	89.96	79.66
PointRCNN (Ours)	CVPR 2019	LiDAR only	87.89	90.21	85.51
Part-A^2-free (Ours)	-	LiDAR only	88.05	90.23	85.85
Part-A^2-anchor (Ours)	-	LiDAR only	**88.61**	**90.42**	**87.31**

TABLE 13
3D Object Detection Results of Cyclist and Pedestrian of Different Models on the KITTI *val* Split Set

Method	Class	AP (IoU=0.5)		
		Mod.	Easy	Hard
PointRCNN	Cyclist	69.70	86.13	65.40
Part-A^2-free	Cyclist	**73.35**	88.18	**70.75**
Part-A^2-anchor	Cyclist	73.07	**88.31**	70.20
PointRCNN	Pedestrian	63.70	69.43	**58.13**
Part-A^2-free	Pedestrian	**64.13**	**70.73**	57.45
Part-A^2-anchor	Pedestrian	63.84	70.37	57.48

TABLE 14
Mean Distance Error of Predicted Intra-Object Part Locations by Part-Aware Stage for the Car Class of KITTI *val* Split

	mAbsError_x	mAbsError_y	mAbsError_z	mAbsError
Overall	7.24%	6.42%	5.17%	6.28%
False Positives	12.97%	12.09%	7.71%	10.92%

As shown in Fig. 4, here x, y, z are along the direction of width, length and height of the object, respectively. Note that here the false positives denotes the false positives samples caused by inaccurate localizations.

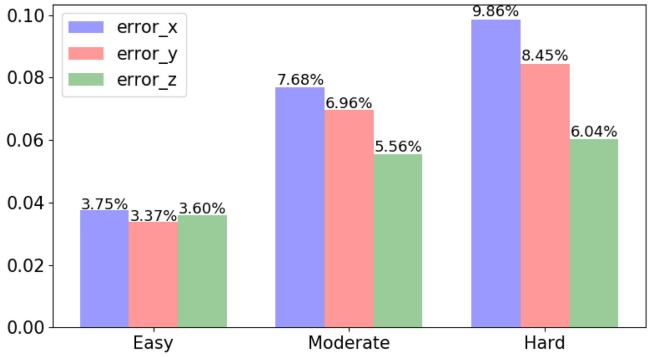


Fig. 10. Statistics of predicted intra-object part location errors for the car class on the *val* split of KITTI dataset.

IoU is the 3D Intersection-over-Union (IoU) between the predicted 3D bounding box and its best matched ground-truth box. As shown in Table 15, we could see that the errors of intra-object part locations have obviously positive correlation with the errors of the predicted 3D bounding boxes. The overall correlation coefficient is 0.531 and the most correlated axis is the z-axis in the height direction where the correlation coefficient achieves 0.552, which demonstrates that accurate intra-object part locations are beneficial for predicting more accurate 3D bounding boxes.

TABLE 15
Pearson Correlation Coefficient Between the Errors of the Predicted Intra-Object Part Locations and the Errors of the Predicted 3D Bounding Boxes

x-axis	y-axis	z-axis	Overall
0.468	0.442	0.552	0.531

We also report the errors of intra-object part locations on the false positive samples which are caused by inaccurate localization (see row 2 of Table. 14), and we could see that the predicted part location errors increase significantly in all three axes, which indicate that inaccurately predicted intra-object part locations may lead to unsatisfactory 3D object localization and decrease the performance of 3D object detection.

4.3 Qualitative Results

We present some representative results generated by our proposed Part-A^2-anchor net on the *test* split of KITTI dataset in Fig. 11. From the figure we could see that our proposed part-aware network could estimate accurate intra-object part locations by using only point cloud as inputs, which are aggregated by our designed part-aggregation network to generate accurate 3D bounding boxes.

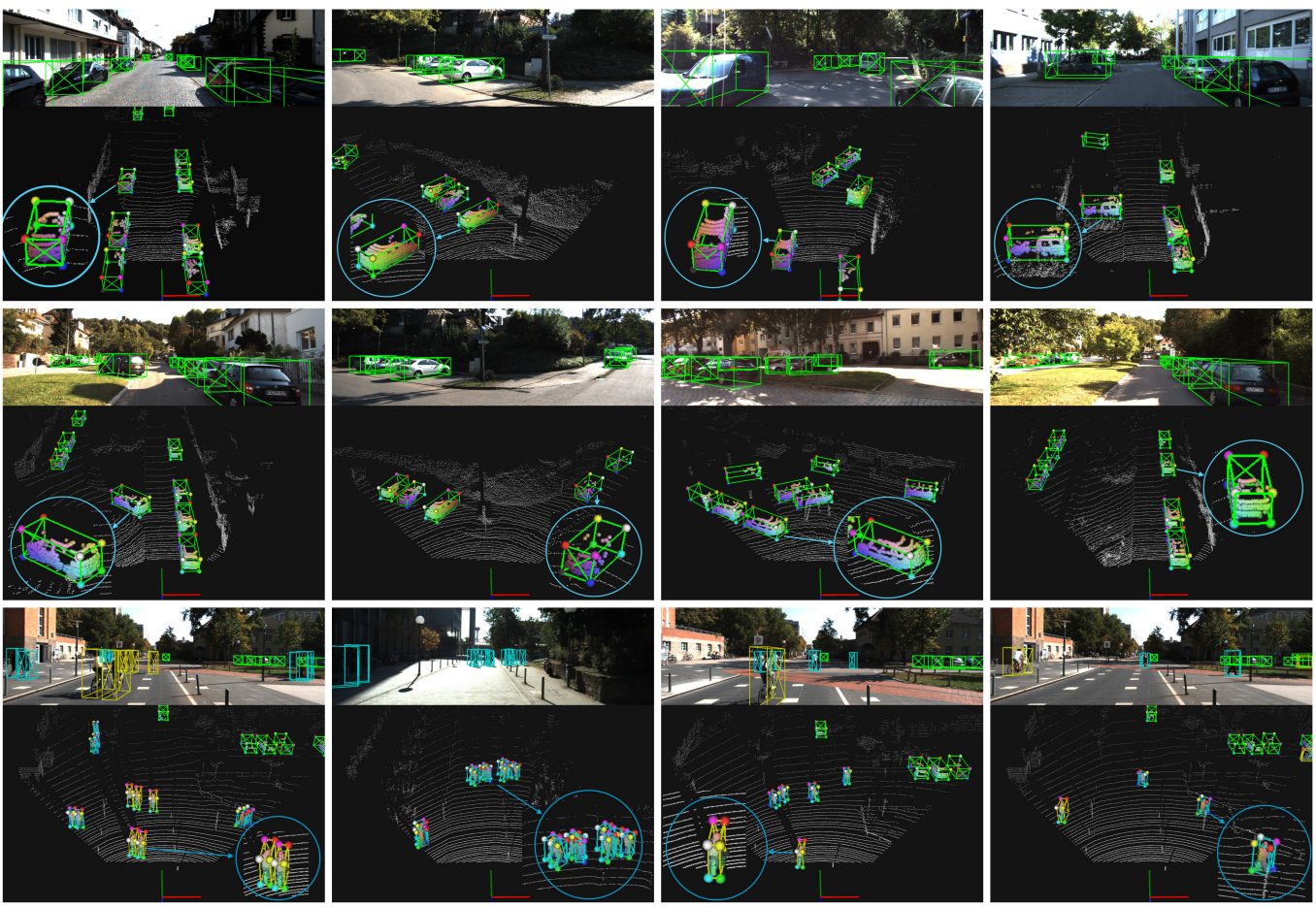


Fig. 11. Qualitative results of Part-A^2-anchor Net on the KITTI *test* split. The predicted 3D boxes are drawn with green 3D bounding boxes, and the estimated intra-object part locations are visualized with interpolated colors as shown in Fig. 4. Best viewed in colors.

5 Conclusion

In this paper, we extend our preliminary work PointRCNN to a novel 3D detection framework, the part-aware and aggregation neural network (Part-A^2 net), for detecting 3D objects from raw point clouds. Our part-aware stage-I learns to estimate the accurate intra-object part locations by using the free-of-charge intra-object location labels and foreground labels from the ground-truth 3D box annotations. Meanwhile, the 3D proposals are generated by two alternative strategies, anchor-free scheme and anchor-based scheme. The predicted intra-object part locations of each object are pooled by the novel RoI-aware point cloud pooling scheme. The following part-aggregation stage-II can better capture the geometric information of object parts to accurately score the boxes and refine their locations.

Our approach significantly outperforms existing 3D detection methods and achieves new state-of-the-art performance on the challenging KITTI 3D detection benchmark. Extensive experiments were carefully designed and conducted to investigate the individual components of our proposed framework.

References

[1] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6526–6534.

[2] S. Song and J. Xiao, "Sliding shapes for 3D object detection in depth images," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 634–651.

[3] S. Song and J. Xiao, "Deep sliding shapes for amodal 3D object detection in RGB-D images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 808–816.

[4] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander, "Joint 3D proposal generation and object detection from view aggregation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–8.

[5] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3D object detection," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 641–656.

[6] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3D object detection from RGB-D data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 918–927.

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

[8] X. Du, M. H. Ang, S. Karaman, and D. Rus, "A general pipeline for 3d detection of vehicles," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2018, pp. 3194–3200.

[9] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12689–12697.

[10] B. Yang, M. Liang, and R. Urtasun, "HDNET: Exploiting HD maps for 3D object detection," in *Proc. 2nd Conf. Robot Learn.*, 2018, pp. 146–155.

[11] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3D object detection from point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7652–7660.

[12] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3D detection, tracking and motion forecasting with a single convolutional net," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3569–3577.

[13] M. Simony, S. Milzy, K. Amendey, and H.-M. Gross, "Complex-YOLO: An euler-region-proposal for real-time 3D object detection on point clouds," in *Proc. Eur. Conf. Comput. Vis.*, 2018, p. 0.

[14] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.

[15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[16] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.

[18] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 7263–7271.

[19] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2117–2125.

[20] J. Dai *et al.*, "Deformable convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 764–773.

[21] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, Feb. 2020.

[22] H. Law and J. Deng, "CornerNet: Detecting objects as paired keypoints," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 734–750.

[23] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2961–2969.

[24] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into high quality object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6154–6162.

[25] D. Xu, D. Anguelov, and A. Jain, "Pointfusion: Deep sensor fusion for 3D bounding box estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 244–253.

[26] Z. Wang and K. Jia, "Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3D object detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 1742–1749.

[27] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.

[28] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Advances Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.

[29] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4490–4499.

[30] B. Graham and L. van der Maaten, "Submanifold sparse convolutional networks," 2017, *arXiv: 1706.01307*. [Online]. Available: https://dblp.org/rec/journals/corr/GrahamM17.bib

[31] B. Graham, M. Engelcke, and L. van der Maaten, "3D semantic segmentation with submanifold sparse convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9224–9232.

[32] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 770–779.

[33] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITI vision benchmark suite," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.

[34] A. Mousavian, D. Anguelov, J. Flynn, and J. Košecká, "3D bounding box estimation using deep learning and geometry," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5632–5640.

[35] B. Li, W. Ouyang, L. Sheng, X. Zeng, and X. Wang, "GS3D: An efficient 3D object detection framework for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1019–1028.

[36] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, and T. Chateau, "Deep MANTA: A coarse-to-fine many-task network for joint 2D and 3D vehicle analysis from monocular image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2040–2049.

[37] M. Zhu *et al.*, "Single image 3D object detection and pose estimation for grasping," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 3936–3943.

[38] R. Mottaghi, Y. Xiang, and S. Savarese, "A coarse-to-fine model for 3D pose estimation and sub-category recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 418–426.

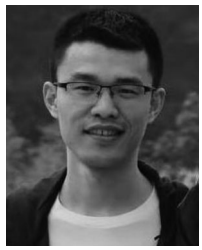
[39] F. Manhardt, W. Kehl, and A. Gaidon, "ROI-10D: Monocular lifting of 2D detection to 6D pose and metric shape," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2064–2073.

[40] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3D object detection for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2147–2156.

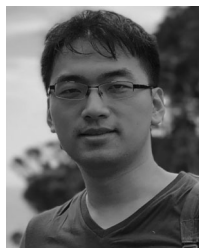
[41] X. Chen *et al.*, "3D object proposals for accurate object class detection," in *Proc. Advances Neural Inf. Process. Syst.*, 2015, pp. 424–432.

[42] J. Ku, A. D. Pon, and S. L. Waslander, "Monocular 3D object detection leveraging accurate proposals and shape reconstruction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 11859–11868.

[43] P. Li, X. Chen, and S. Shen, "Stereo R-CNN based 3D object detection for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7636–7644.
[44] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Weinberger, "Pseudo-liDAR from visual depth estimation: Bridging the gap in 3D object detection for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8437–8445.
[45] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Medical Image Comput. Comput.-Assisted Intervention*, 2015, pp. 234–241.
[46] L. Yi, W. Zhao, H. Wang, M. Sung, and L. J. Guibas, "GSPN: Generative shape proposal network for 3D instance segmentation in point cloud," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3947–3956.
[47] J. Hou, A. Dai, and M. Nießner, "3D-SIS: 3D semantic instance segmentation of RGB-D scans," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4421–4430.
[48] W. Wang, R. Yu, Q. Huang, and U. Neumann, "SGPN: Similarity group proposal network for 3D point cloud instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2569–2578.
[49] X. Wang, S. Liu, X. Shen, C. Shen, and J. Jia, "Associatively segmenting instances and semantics in point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4096–4105.
[50] J. Lahoud, B. Ghanem, M. Pollefeys, and M. R. Oswald, "3D instance segmentation via multi-task metric learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 9256–9266.
[51] B. D. Brabandere, D. Neven, and L. V. Gool, "Semantic instance segmentation with a discriminative loss function," *CoRR*, vol. abs/1708.02551, 2017. [Online]. Available: http://arxiv.org/abs/1708.02551
[52] M. Bai and R. Urtasun, "Deep watershed transform for instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5221–5229.
[53] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.
[54] S. Fidler, S. Dickinson, and R. Urtasun, "3D object detection and viewpoint estimation with a deformable 3D cuboid model," in *Proc. Advances Neural Inf. Process. Syst.*, 2012, pp. 611–619.
[55] B. Pepik, P. Gehler, M. Stark, and B. Schiele, "3D 2 pm–3D deformable part models," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 356–370.
[56] J. J. Lim, A. Khosla, and A. Torralba, "FPM: Fine pose parts-based model with 3D cad models," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 478–493.
[57] Q. Huang, W. Wang, and U. Neumann, "Recurrent slice networks for 3D segmentation of point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2626–2635.
[58] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on X-transformed points," in *Proc. Advances Neural Inf. Process. Syst.*, 2018, pp. 820–830.
[59] S. Wang, S. Suo, W.-C. Ma, A. Pokrovsky, and R. Urtasun, "Deep parametric continuous convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2589–2597.
[60] H. Zhao, L. Jiang, C.-W. Fu, and J. Jia, "PointWeb: Enhancing local neighborhood features for point cloud processing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5565–5573.
[61] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep convolutional networks on 3D point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9621–9630.
[62] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang, "Acquisition of localization confidence for accurate object detection," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 784–799.
[63] S. Sun, J. Pang, J. Shi, S. Yi, and W. Ouyang, "FishNet: A versatile backbone for image, region, and pixel level prediction," in *Proc. Advances Neural Inf. Process. Syst.*, 2018, pp. 762–772.
[64] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3D object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7337–7345.
[65] KITTI 3D object detection benchmark leader board, Accessed: Aug. 15, 2019. [Online]. Available: http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d



Shaoshuai Shi received the BS degree in computer science and technology from the Harbin Institute of Technology, China, in 2017. He is currently working toward the PhD degree in the Department of Electronic Engineering, The Chinese University of Hong Kong. His research interests include computer vision, deep learning, and 3-D scene understanding.



Zhe Wang received the BS degree from the Optical Engineering of Zhejiang University, in 2012, and the PhD degree from the Department of Electronic Engineering, The Chinese University of Hong Kong, in 2017. He is current a research vice director of SenseTime. His research interests include computer vision and deep learning.

Jianping Shi received the BS degree in computer science and engineering from Zhejiang University, China, in 2011, and the PhD degree from the Computer Science and Engineering Department, The Chinese University of Hong Kong, in 2015. She is currently the executive research director of SenseTime. Her research interests include computer vision and deep learning.

Xiaogang Wang received the BS degree from the University of Science and Technology of China, in 2001, the MS degree from The Chinese University of Hong Kong, in 2003, and the PhD degree from the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, in 2009. He is currently an associate professor with the Department of Electronic Engineering, The Chinese University of Hong Kong. His research interests include computer vision and machine learning.

Hongsheng Li received the bachelor's degree in automation from the East China University of Science and Technology, in 2006, and the master''s and doctorate degrees in computer science from Lehigh University, Pennsylvania, in 2010, and 2012, respectively. He is currently an assistant professor with the Department of Electronic Engineering, The Chinese University of Hong Kong. His research interests include computer vision, medical image analysis, and machine learning.