Structure Aware Single-stage 3D Object Detection from Point Cloud

Chenhang He[1,2], Hui Zeng[1,2], Jianqiang Huang[2], Xian-Sheng Hua[2], Lei Zhang[1,2] *
[1]The Hong Kong Polytechnic University   [2]DAMO Academy, Alibaba Group
{csche, cshzeng}@comp.polyu.edu.hk, jianqiang.hjq@alibaba-inc.com
huaxiansheng@gmail.com, cslzhang@comp.polyu.edu.hk

Abstract

3D object detection from point cloud data plays an essential role in autonomous driving. Current single-stage detectors are efficient by progressively downscaling the 3D point clouds in a fully convolutional manner. However, the downscaled features inevitably lose spatial information and cannot make full use of the structure information of 3D point cloud, degrading their localization precision. In this work, we propose to improve the localization precision of single-stage detectors by explicitly leveraging the structure information of 3D point cloud. Specifically, we design an auxiliary network which converts the convolutional features in the backbone network back to point-level representations. The auxiliary network is jointly optimized, by two point-level supervisions, to guide the convolutional features in the backbone network to be aware of the object structure. The auxiliary network can be detached after training and therefore introduces no extra computation in the inference stage. Besides, considering that single-stage detectors suffer from the discordance between the predicted bounding boxes and corresponding classification confidences, we develop an efficient part-sensitive warping operation to align the confidences to the predicted bounding boxes. Our proposed detector ranks at the top of KITTI 3D/BEV detection leaderboards and runs at 25 FPS for inference.

1. Introduction

3D object detection from point cloud data is a key component in Autonomous Vehicle (AV) system. Unlike the ordinary 2D object detection which only estimates 2D bounding box from an image plane, AV requires to estimate a more informative 3D bounding box from the real world to fulfill the high-level tasks like path planning and collision avoidance. This motivates the recently emerged 3D object detection approaches which apply the convolutional neural
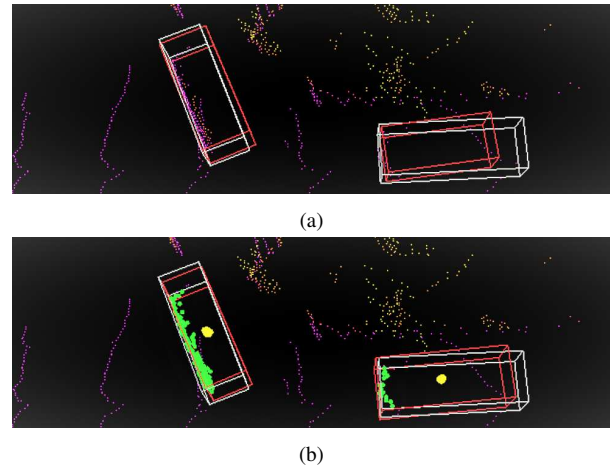
Figure 1: Predicted bounding boxes from sparse 3D point cloud by (a) the representative single-stage detector SECOND [25] and (b) our single-stage method guided by auxiliary tasks and point-level supervisions. The object points, ground-truth box, center points predicted by the auxiliary network and the final detection results are shown in green, white, yellow and red colors, respectively.

network (CNN) to process more representative point cloud data from a high-end LiDAR sensor.

Current 3D object detection from point cloud can be divided into two streams, i.e., single-stage approaches and two-stage approaches. Single-stage approaches [25, 30, 27, 8, 22, 9] parse the sparse 3D point cloud into a compact representation, such as voxel grid or birds-eye-view (BEV) image, and employ a CNN to directly predict the bounding box in a fully convolutional manner. This makes the single-stage approaches typically simple and efficient. However, the progressively downscaled feature maps inevitably lose the spatial resolution and cannot explicitly consider the structure information of point cloud data, making the single-stage detectors less accurate to process the sparse point cloud. As shown in Fig. 1(a), the single-stage detector fails to achieve accurate localization when the object contains insufficient points.

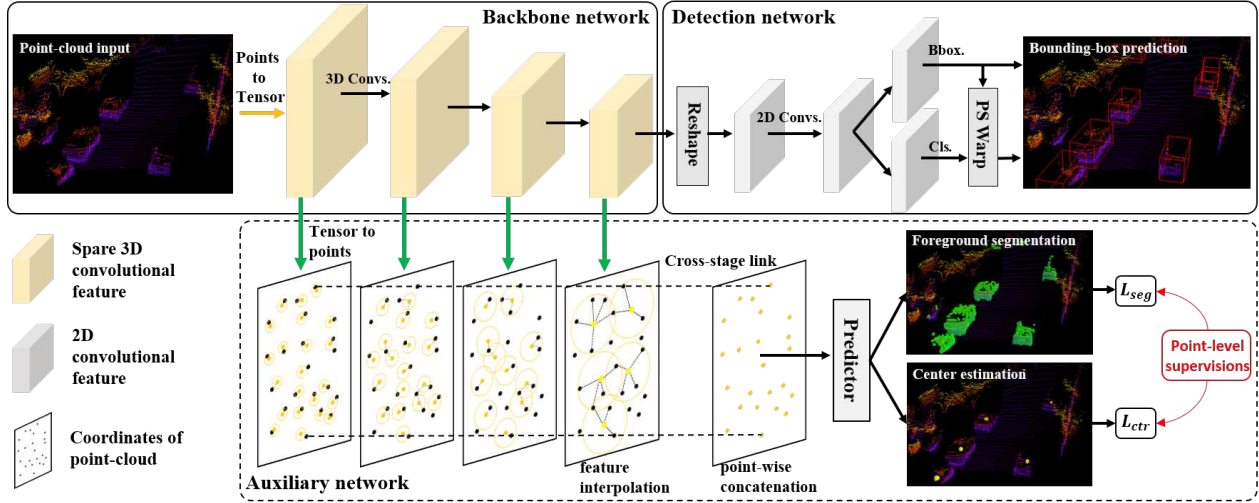Compared to single-stage approaches, two-stage meth-

Figure 2: Overview of the proposed structure aware single-stage 3D object detector. Our network contains three sub-networks, a backbone network to extract the multi-stage features from point cloud, a back-end detection network to predict 3D bounding box and an auxiliary network to exploit point-wise supervisions. The yellow points in the auxiliary network represent the nonzero feature vectors in the coordinate system of the original point cloud. We also design a part-sensitive warping (PSWarp) scheme to align the classification confidences with the predicted bounding boxes.

ods [16, 19, 2, 28, 15, 20] can leverage finer spatial information at the second stage, which focuses only on the regions of interest (RoIs) predicted by the first stage, and consequently predicts more accurate bounding box. This reveals the importance of leveraging fine-grained spatial information of point cloud to achieve accurate localization. However, to operate on each point and to re-extract features for each RoI substantially increases the computational cost, making two-stage methods hard to reach real-time speed.

Motivated by the high precision of two-stage detectors, in this paper, we propose to exploit the fine-grained structure information to improve the localization accuracy and simultaneously preserve the high efficiency of single-stage approaches. We design a structure-aware single-stage 3D object detector whose framework is shown in Fig. 2. In addition to the backbone network which generates downscaled features for bounding box prediction, our detector contains an auxiliary network which guides the backbone network to learn more discriminative features with point-level supervisions. Specifically, the auxiliary network first converts the features from the backbone network back to point-wise representations and then performs two auxiliary tasks: foreground segmentation to drift the features to be sensitive to the object boundary and point-wise center estimation to make the features be aware of the intra-object relationship. The auxiliary network is jointly optimized with the backbone network in training stage and is removed after training, introducing no extra computational cost at inference stage. As shown in Fig. 1(b), our model guided by the auxiliary tasks can yield more accurate localization.

In addition, we notice that single-stage detectors usually suffer from a misalignment between the predicted bounding boxes and the corresponding classification confidences. Specifically, classification confidences are related to the current locations of employed feature maps while the predicted bounding boxes usually deviate from their current locations. This misalignment may lead to suboptimal results in NMS post-processing. Inspired by PSRoIAlign [3], we develop an efficient part-sensitive warping method (denoted by PSWarp in detection network in Fig. 2) to align the classification confidences with the predicted bounding boxes by performing spatial transformation on the classification feature maps, making our model generate more reliable confidence maps. In summary, our contribution are twofold.

(1) We propose a structure-aware single-stage 3D object detector, which employs a detachable auxiliary network to learn structure information and exhibits better localization performance without extra cost.

(2) We develop an efficient feature map warping method to mitigate the discordance between the predicted bounding boxes and corresponding classification confidences, improving the detection accuracy at negligible cost.

We evaluate our proposed detector on KITTI [4] 3D/BEV object detection benchmarks. Our detector outperforms all the previous published methods and at the same time achieves 25 FPS inference speed.

2. Related Work

Single-stage approaches. Single-stage approaches are proposed to enhance the computational efficiency by process-

ing the point cloud in a fully convolutional network with a transformed compact representation. Typical methods of this kind either apply a 2D CNN to extract features from a BEV [22] and a frontal-view panorama [24], or apply a 3D CNN to a handcrafted voxel grid [9, 27]. Zhou *et al.* [30] proposed to extracted voxel feature by a tinny PointNet [17]. Lang *et al.* [8] proposed to reduce feature dimension by stacking voxels feature along the height axis as a "pillar". Yan *et al.* [25] investigated an improved sparse convolution that optimizs the GPU usage in 3D convolutions. Our proposed method is build on top of a general single-stage architecture and we are the first one to consider using point-level supervision to enhance the representative power of convolutional feature.

Two-stage approaches. Unlike single-stage approach that directly produce 3D bounding box, two-stage approaches aim to produce more accurate detection by re-using the point cloud with full resolution in the second stage. Some image-driven methods [16, 23] have been proposed to lift a set of 3D regions of interest (RoI) from the image and then apply a PointNet to extract RoI feature by gathering the interior points with transformed canonical coordinates. Shi *et al.* [19] proposed to generate RoIs from purely point clouds by using a PointNet to segment the foreground points from the scene. Its variant work [2] generats RoIs by an efficient voxel-based CNN. Yang *et al.* [28] divided the RoI into voxels to enable using regular CNN for RoI feature extraction. Shi *et al.* [20] enriched the RoI features by performing intra-object part-aware analysis and demonstrated the effectiveness of reducing the ambiguity of bounding box. The success of two-stage approach inspired us to exploit the full-resolution spatial information encoded in the ground-truth to guide the single-stage model learning more discriminative fine-grained patterns.

Auxiliary task learning. Learning from a group of side tasks to enhance the main task's performance has been revisited in autonomous driving applications. Yang *et al.* [26] proposed to estimate the ground-level to enhance the detector's awareness of road geometry. Liang *et al.*[10] proposed to enhance the fusion of cross-modality inputs from multi-task predictions. Our method is closely connected to them by employing point-wise prediction tasks to improve 3D object detection, meanwhile differing from them by regarding the predictions as auxiliary tasks, which is detachable in the time of inference. Auxiliary tasks can be heterogeneous to the main task, providing a multi-fold regularization effect in the optimization. For example, Zhao *et al.* [29] facilitated the crowd segmentation task to learn a more attentive density map for crowd counting. Mordan *et al.*[14] facilitated the depth estimation task to learn the scene-aware features, improving the robustness of detecting occluded objects. By taking advantage of auxiliary tasks, we can maintain the efficiency of single-stage network at inference.

3. Structure-Aware 3D Object Detection via Auxiliary Network Learning

In this section, we develop an efficient structure-aware single-stage network for point cloud detection. Sec. 3.1 introduces our backbone and detection networks. Sec.3.2 introduces the proposed auxiliary network to enrich the hidden features of the backbone network via two special auxiliary tasks. Sec. 3.3 introduces a part-sensitive warping operation to generate a more accurate confidence map. Sec. 3.4 presents the loss functions for training.

3.1. Backbone and detection networks

Input data representation. Previous works normally encode the point cloud into a 3D sparse tensor by dividing it into a voxel grid and representing each voxel feature as the nonzero entry of the input tensor. However, the voxelization is a time-consuming pre-processing method. For simplicity, we directly represent each point as a nonzero entry of the input tensor by quantizing the point's coordinates to tensor indices. Let $\{p_i = (x_i, y_i, z_i) : i = 1, ...N\}$ be the coordinates of point cloud and $d = [d_x, d_y, d_z] \in \mathbb{R}^3$ be the quantization step of input tensor. The tensor indices can then be represented as $\{\bar{p}_i = (\lfloor \frac{x_i}{d_x} \rfloor, \lfloor \frac{y_i}{d_y} \rfloor, \lfloor \frac{z_i}{d_z} \rfloor) : i = 1, ..., N\}$, where $\lfloor \cdot \rfloor$ is the floor function. We iteratively assign each point to the entry of input tensor according to the associated index. If multiple points share the same index, we overwrite the entry with the latest point. We find that this pre-processing method is efficient and adequate to achieve reasonable performance with a finer quantization step $d = [0.05\text{m}, 0.05\text{m}, 0.1\text{m}]$.

Network architecture. As shown in Fig. 2, we employ the commonly used backbone network [30, 25, 2] as our feature extractor. The network contains four convolutional blocks, each of which is composed of sub-manifold convolutions with kernel size of 3. The last three blocks contain an additional sparse convolution with a stride of 2. Each convolution is followed by a batch normalization [5] and ReLU non-linearity. As a result, the backbone network produces multi-stage feature maps at different spatial resolutions. The detection network reshapes the feature map from the backbone output to a BEV representation by concatenating the feature vectors along the depth dimension into one channel. Then it applies six standard 3×3 convolutions with non-linearity to further abstract the feature. Two sibling 1×1 convolutions without non-linearity is applied to generate task-specific pixel-wise predictions: a set of part-sensitive classification maps and a regression map that encode the anchor-offsets for the oriented 3D objects.

3.2. Detachable auxiliary network

As discussed in Sec. 1, we propose to learn a detachable auxiliary network with point-wise supervisions, which
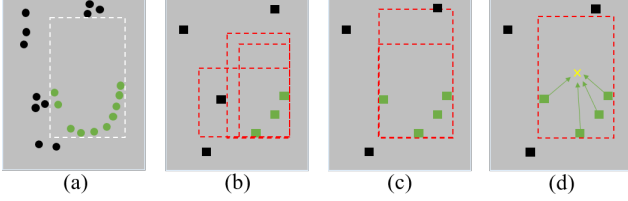
Figure 3: (a) A vanilla example of predicting bounding box from 2D point cloud. The foreground points, background points and the ground-truth bounding box are shown in green, black and white colors respectively. (b) The predictions from the convolutional features, the black and green squares denote the nonzero feature vectors. (c) The predictions from boundary-sensitive convolutional features. (d) The prediction from structure-aware convolutional features. The yellow cross denotes the estimated object center.

helps the feature extracted by backbone network be aware of the structure information of 3D point cloud.

Motivation. Generally, the downsampled convolutional features extracted from point cloud will inevitably lose structure details which are vital to generate accurate localization. A vanilla example of detecting objects from 2D point set is illustrated in Fig. 3. As shown in Fig. 3(a), only a few points from the object are detected and there are some background points close to its boundary. This case is very common in real scenario where the object is far from the sensor and occluded with others of no interest. As the CNN progressively decreases the spatial resolution of the point cloud, some object points may be submerged in the background points, resulting in the feature at the object boundary being misclassified in the low-resolution feature space, as shown in Fig. 3(b). Consequently, the model is misled and produces low-quality bounding boxes.

Our solution is to build an auxiliary network with point-level supervisions to guide the intermediate features from different stages of backbone CNN to learn the fine-grained structure of the point cloud. To achieve this goal, we first need to convert the extracted CNN features back to point-wise representations.

Point-wise feature representation. The auxiliary network is shown in Fig. 2. It first converts each nonzero index of backbone feature to a real-world coordinates based on the quantization step of current stage, so that each backbone feature can be represented in a point-wise form. We denote it by $\{(f_j, p_j) : j = 1, ..., M\}$ with f being the feature vector and p being the point coordinates. To generate full-resolution point-wise features, we employ a feature propagation layer [18] at each stage to interpolate backbone features at the coordinates of original point cloud $\{p_i : i = 1, ..., N\}$. For interpolation, we use the inverse distance weighted average among all the points in a neighboring region. Let $\{(\hat{f}_i, p_i) : i = 1, ..., N\}$ be the interpo-

lated (propagated) feature, the feature vector at each point can be calculated by:

$$\hat{f}_i = \frac{\sum_{j=1}^{M} w_j(p_i)f_j}{\sum_{j=1}^{M} w_j(p_i)}, \quad (1)$$

where

$$w_j(p_i) = \begin{cases} \frac{1}{||p_i - p_j||_2} & \text{if } p_j \in \mathcal{N}(p_i) \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

$\mathcal{N}(p_i)$ denotes a ball region, which has a radius of 0.05m, 0.1m, 0.2m, and 0.4m in each stage, respectively. We concatenate these point-wise features by a cross-stage link and apply a shallow predictor to generate task-specific outputs. The predictor is realized by a shared multi-layer perceptron with neuron sizes of (64, 64, 64) and two task-specific outputs are generated by unit point convolutions.

Auxiliary tasks. We first introduce a point-wise foreground segmentation task to guide the backbone CNN learn more discriminative patterns in the object boundary. Specifically, we employ a sigmoid function to the segmentation branch to predict the foreground/background probability of each point, denoted by $\tilde{s}_i$. Let s_i be a binary label to indicate whether a point falls into a ground-truth bounding box. The foreground segmentation task can be optimized with a focal loss [12], i.e.,

$$\mathcal{L}_{\text{seg}} = \frac{1}{N_{pos}} \sum_{i}^{N} -\alpha(1 - \hat{s}_i)^{\gamma} \log(\hat{s}_i), \quad (3)$$

where

$$\hat{s}_i = \begin{cases} \tilde{s}_i & \text{if } s_i = 1 \\ 1 - \tilde{s}_i & \text{otherwise.} \end{cases} \quad (4)$$

α and γ are the hyper-parameters and we use the empirical values 0.25 and 2 as specified in the original paper [12].

The above segmentation task enables the backbone network to more precisely detect the object boundary, as shown in Fig. 3(c). With more precise feature maps, the model can generate more accurate bounding boxes. However, even if the boundary points are precisely detected, there is still ambiguity in determining the scale and shape of bounding boxes since the feature maps are very sparse. To further improve the localization accuracy, we employ another auxiliary task to learn the relative position of each object point to the object center. As shown in Fig. 3(d), this intra-object relationship can help determine the scale and shape of the object, resulting in more precise localization.

Let $\Delta\tilde{\mathbf{p}} \in \mathbb{R}^{N \times 3}$ be the output of center estimation branch and $\Delta\mathbf{p}$ be the target offsets from object points to the corresponding center. The center estimation task can be optimized with the following Smooth-l_1 [13] loss:

$$\mathcal{L}_{\text{ctr}} = \frac{1}{N_{pos}} \sum_{i}^{N} \text{Smooth-}l_1(\Delta\tilde{\mathbf{p}} - \Delta\mathbf{p}) \cdot \mathbb{1}[s_i = 1], \quad (5)$$

where N_{pos} is the number of foreground points and $\mathbb{1}[\cdot]$ is a indicator function.

Combining the foreground segmentation and center estimation tasks enables the backbone network to learn structure-aware features. As will be seen in Sec. 4.4, employing these two auxiliary tasks significantly improve the localization accuracy of the backbone network. Besides, the auxiliary network is only employed in the training stage, introducing no extra computational cost for inference.

3.3. Part-sensitive warping

To solve the misalignment between the predicted bounding boxes and corresponding confidence maps, we propose a part-sensitive warping operation, namely PSWarp, as an efficient variant of PSRoIAlign [3], to align the classification confidences with the predicted bounding boxes by performing spatial transformation on the feature maps.

Similar to PSRoIAlign, we first modify the last classification layers to generate K part-sensitive classification maps, denoted by $\{\mathcal{X}^k : k = 1, 2, ..., K\}$, each of which encodes information of a certain part of the object, for example, {upper-left, upper-right, bottom-left, bottom-right} in the case of $K = 4$. At the same time, we divide the predicted bounding box at each feature-map position into K sub-windows and select the center position of each sub-window as the sample point. In this way, we can generate K sampling grids $\{\mathcal{S}^k : k = 1, 2, ..., K\}$ and each of which associates with a classification map. Our PSWarp is composed by a feature map sampler [6], as shown in Fig. 4, which takes the classification maps and sampling grids as input, producing output maps sampled from the input at grid points. The final confidence map $\mathcal{C}$ is computed by taking the average among K sampled classification maps. Given a predicted bounding box p and its corresponding sample points $\{(u^k, v^k) = \mathcal{S}_p^k : k = 1, 2, ..., K\}$, the final confidence of this bounding box can be calculated by:

$$\mathcal{C}_p = \frac{1}{K} \sum_{k=1}^{K} \sum_{\substack{i \in \{\lfloor u^k \rfloor \lfloor u^k + 1 \rfloor\} \\ j \in \{\lfloor v^k \rfloor, \lfloor v^k + 1 \rfloor\}}} \mathcal{X}_{ij}^k \times b(i, j, u^k, v^k), \quad (6)$$

where b is a bilinear sampling kernel which has a form of $b(i, j, u, v) = max(1 - |i - u|, 0) \times max(1 - |j - v|, 0)$.

Compared to PSRoIAlign and other RoI-based methods, PSWarp is more efficient since it mitigates the needs of generating RoIs from a dense feature map with NMS. It considers only one pixel in each sub-window therefore has the same computational complexity as a standard convolution. In Sec. 4.4, we show that PSWarp can achieve comparable performance to PSRoIAlign with only 1/10 of time.

3.4. Loss functions

We apply the common anchor-based setting [30, 25, 8] to optimize the primary network. Let $\mathcal{L}_{box}$ and $\mathcal{L}_{cls}$ be the
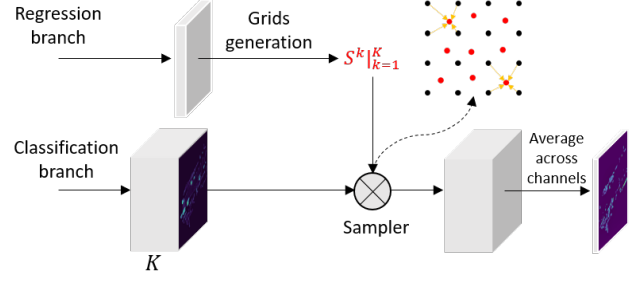

Figure 4: Part-sensitive warping. We generate K sampling grids from the dense bounding boxes predicted by regression branch. Each grid is used to sample the feature points from the classification map using a bilinear interpolation kernel. The final confidence map is computed by taking average among K sampled maps.

two losses that impose on the regression branch and classification branch, respectively. $\mathcal{L}_{box}$ is a Smooth-l_1 loss [13] and $\mathcal{L}_{cls}$ is a focal loss [12]. We jointly optimize the detection task and the auxiliary tasks by applying a gradient descent method to minimize the weighted sum of the following losses:

$$\mathcal{L} = \mathcal{L}_{cls} + \omega\mathcal{L}_{box} + \mu\mathcal{L}_{seg} + \lambda\mathcal{L}_{ctr}, \quad (7)$$

where ω is empirically set to 2 according to [30, 25], μ and λ are the hyper-parameters to balance the auxiliary tasks from detection tasks. We will conduct experiments to properly select them in Sec. 4.2.

4. Experiments

We evaluate our proposed structure-aware single-stage detector (SA-SSD) on the KITTI 3D/BEV object detection benchmark [4]. The dataset contains 7,481 training samples and 7,518 testing samples. We further divide the training data into a training set with 3,712 samples and a validation set with 3,769 samples following the common protocol. We conduct experiments on the most commonly used *car* category and use average precision (AP) with an (IoU) threshold 0.7 as evaluation metric. The benchmark consider three levels of difficulties: *easy*, *moderate*, and *hard* based on the object size, occlusion state, and truncation level. The Average precision (AP) is calculated using 40 recall positions[1]. The source code is available at https://github.com/skyhehe123/SA-SSD.

4.1. Implementation details

Training details. We use the common setting in [30, 25] by selecting the interest LiDAR points that lie between the range (0m, 70.4m), (-40m, 40m), (-3m, 1m) along X, Y, Z axes, respectively, and dropping those points that are invisible in the image view. In training, we use the matching

---

[1]On 08.10.2019, KITTI changed its evaluation setting by using 40 recall positions as suggested in [21].

Method	Modality	BEV			3D			FPS
		Easy	Moderate	Hard	Easy	Moderate	Hard	
Two-stage:								
MV3D[1]	LiDAR+RGB	86.49	78.98	72.23	74.97	63.63	54.00	2.8
F-PointNet[16]	LiDAR+RGB	91.17	84.67	74.77	82.19	69.79	60.59	5.9
AVOD[7]	LiDAR+RGB	89.75	84.95	78.32	76.39	66.47	60.23	10
PointRCNN[19]	LiDAR	92.13	87.39	82.72	86.96	75.64	70.70	-
F-ConvNet[23]	LiDAR+RGB	91.51	85.84	76.11	87.36	76.39	66.69	2.1
Fast PointRCNN[2]	LiDAR	90.87	87.84	80.52	85.29	77.40	70.24	15.4
MMF[10]	LiDAR+RGB	93.67	88.21	81.99	88.40	77.43	70.22	12.5
STD[28]	LiDAR	94.74	89.19	**86.42**	87.95	79.71	**75.09**	10
One-stage:								
VoxelNet[30]	LiDAR	87.95	78.39	71.29	77.82	64.17	57.51	4.4
ContFuse[11]	LiDAR+RGB	94.07	85.35	75.88	83.68	68.78	61.67	16.7
SECOND[25]	LiDAR	89.39	83.77	78.59	83.34	72.55	65.82	20
PointPillars[8]	LiDAR	90.07	86.56	82.81	82.58	74.31	68.99	**42**[2]
SA-SSD (ours)	LiDAR	**95.03**	**91.03**	85.96	**88.75**	**79.79**	74.16	25

Table 1: Performance comparison with previous methods on KITTI test server. BEV and 3D object detection metric are used, reported by the Average Precision(AP) with IoU threshold 0.7. The bold value indicates the top performance.

thresholds for the positive and negative anchors of 0.6 and 0.45, respectively. The matching IoU between the bounding boxes and anchors is calculated by their nearest horizontal rectangles in BEV. The anchor for detecting the car has a size of 1.6m (width), 3.9m (length) and 1.56m (height). All the anchors that contain no points are ignored.

The network is trained for 50 epochs using an SGD optimizer. The batch size, learning rate, and weight decay are st to 2, 0.01 and 0.001, respectively. The learning rate is decayed with a cosine annealing strategy. At inference stage, we filter out the low-confidence bounding box by a threshold of 0.3. The IoU threshold for non-maximum suppression (NMS) is 0.1.

Data augmentation. We perform a common cut-and-paste strategy [30, 25, 2] for data augmentation. Specifically, we collect all the ground-truth boxes and the associated points that fall into these boxes as a pool of instance. For each example, we randomly draw at most 10 instances from the pool and place them into the current point cloud. Each placement is followed by a collision test to avoid the violation of the physical rule. All ground-truth boxes are individually augmented. Each box is randomly rotated and translated. The noise for the rotation is uniformly drawn from $[-\pi/15, \pi/15]$ and the noise for the translation is drawn from $\mathcal{N}(0, 0, 0.25)$. In addition, we apply random flipping, global rotation and global scaling to the whole point cloud. The noise for global rotation is uniformly drawn from $[-\pi/15, \pi/15]$ and the scaling factor is uniformly drawn from $[0.95, 1.05]$.

---

[2]PointPillars can run at 62 FPS by using TensorRT for GPU acceleration. Here we show the run-time using its PyTorch pipeline, which is 42 FPS according to [8], for fair comparison.

(a) μ for foreground segmentation task ($\lambda = 0$).
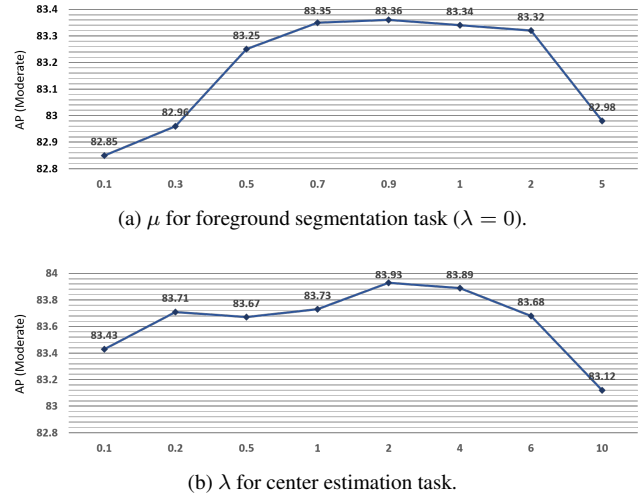

(b) λ for center estimation task.

Figure 5: The selection weights for auxiliary tasks.

4.2. Weight selection of auxiliary tasks

The weights μ and λ in Eq. 7 determine the influence of each auxiliary task on the main task, which is the key hyper-parameters in our method. To find their optimal values, we conduct experiments by first tuning μ with λ fixed to 0, and then tuning λ with the selected μ. As shown in Fig. 5, when the weight μ for the foreground segmentation task lies within a certain range, the detection performance (reported by AP in the moderate subset) can be obviously improved. A too small weight is difficult to contribute to the main task, and a too large weight degrades the performance by alienating the feature representation. A similar tendency can be observed in the selection of λ. In the following experiments, we use $\mu = 0.9$ and $\lambda = 2$.
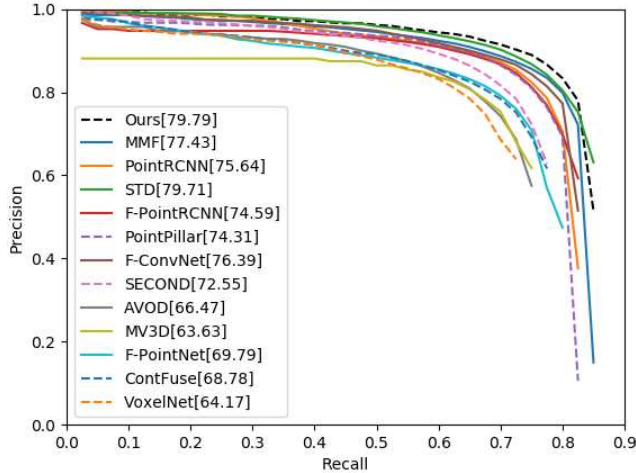
Figure 6: Evaluation results of different methods on KITTI 3D object detection test set. For each method, we plot precision-recall curve, the AP on most important moderate subset are reported. The single-stage methods are shown in dotted line.

Legend in figure:
- Ours[79.79]
- MMF[77.43]
- PointRCNN[75.64]
- STD[79.71]
- F-PointRCNN[74.59]
- PointPillar[74.31]
- F-ConvNet[76.39]
- SECOND[72.55]
- AVOD[66.47]
- MV3D[63.63]
- F-PointNet[69.79]
- ContFuse[68.78]
- VoxelNet[64.17]

4.3. Comparison with state-of-the-arts

We compare our SA-SSD 3D point cloud detector with other state-of-the-art approaches by submitting the detection results to the KITTI server for evaluation. As shown in Table 1, our approach achieves the best performance among all competitors in both 3D and BEV detection tasks. By the time of submission, our method ranked the first on KITTI 3D/BEV object detection leaderboards in the most important car category. In addition, our method is 2.5 times faster than the second top method STD [28]. In BEV detection, we achieve a significant improvement (1.8%) on AP compared to the STD method on the most important moderate subset. In the context of single-stage detector, our method outperforms all competitors by a large margin. Specifically, our model leads the PointPillars [8] by $(6.2\%, 5.5\%, 5.2\%)$ in 3D detection and $(5.0\%, 4.5\%, 3.1\%)$ in BEV detection. Our backbone network is build on top of the architecture of SECOND [25] while achieves $(5.4\%, 7.2\%, 8.3\%)$ improvement over the baseline. This large improvement mainly comes from the enriched hidden feature from the auxiliary tasks. Owing to the single-stage architecture and voxel-free pre-processing, our SA-SSD can run at 25 FPS which is faster than most of the methods. Fig. 6 further shows that our method outperforms state-of-the-art approaches with different recall settings, indicating that our method achieves better detection coverage as well as accuracy. We also show some prediction results in Fig. 7 and we project the 3D bounding boxes detected from LiDAR to the RGB images for better visualization. As observed, our method can produce high-quality 3D bounding boxes in different kinds of scenes.

	Moderate	Easy	Hard
MV3D [1]	62.68	71.29	56.56
AVOD [7]	74.44	84.41	68.65
VoxelNet [30]	65.46	81.97	62.85
F-PointNet [16]	70.92	83.76	63.65
SECOND [25]	76.48	87.43	69.10
PointRCNN [19]	78.63	88.88	77.38
STD [6]	79.8	89.7	**79.3**
SA-SSD (ours)	**79.91**	**90.15**	78.78

Table 2: 3D detection AP on KITTI val set of our model for "Car" compared with other state-of-the-art methods. The AP is calculated with 11 recall positions.

Segment	Center	PSWarp	Moderate	Easy	Hard
			82.88	92.10	79.96
✓			83.36	92.53	80.13
✓	✓		83.93	92.86	80.91
✓	✓	✓	84.30	93.23	81.36

Table 3: Performance of proposed method with different configurations. The average precision on 3D object for easy, moderate, and hard subsets on KITTI val split are reported.

	2×3	3×5	4×7	Time
PSWarp	84.05	84.13	84.30	∼0.4ms
PSRoIAlign [3]	84.18	84.22	84.35	∼ 4ms

Table 4: Comparison between our PSWarp and PSRoIAlign. The AP value on moderate subsets are reported.

	Data	Input	Net	NMS	Overall
SECOND[25]	1.5	6.6	37.5	0.7	46.3
SA-SSD (ours)	1.5	<0.01	37.9	0.7	40.1

Table 5: Runtime (in millisecond) analysis of different steps during inference.

4.4. Ablation study

In this section, we conduct a comprehensive analysis of the effectiveness of different proposed modules in our detector. We first evaluate our method on the validation set and report AP with 11 recall positions to compare with the results from previous arts. As shown in Table 2, SA-SSD outperforms previous state-of-the art methods in the both moderate and easy subsets. Then we study the effects of the auxiliary tasks and PSWarp on our model. Results are shown in Table 3.

Effect of segmentation tasks. As shown in Table 3, leveraging segmentation auxiliary task contributes to a performance improvement of around $(0.5\%, 0.4\%, 0.2\%)$ on each subset. It can be observed that the performance gains on easy and moderate subsets are higher than that on the hard subset. This is because the objects from hard subset usually contain only a few points, providing very limited useful in-
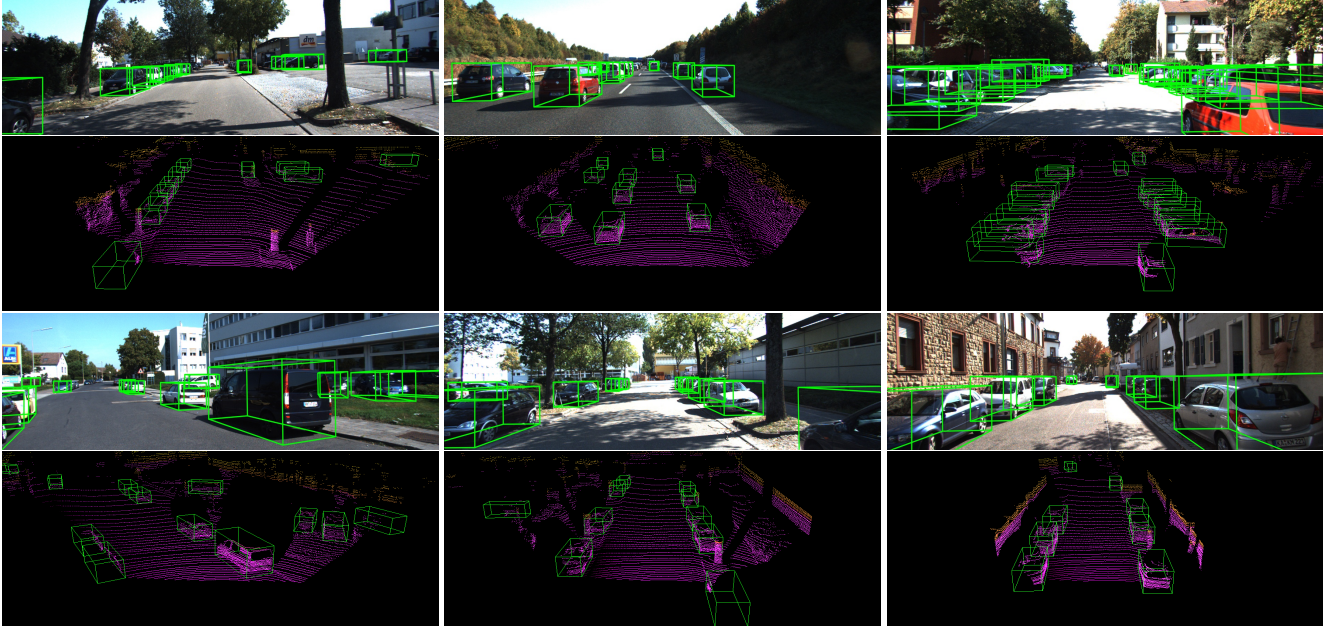
Figure 7: Qualitative results on KITTI test set. The predicted bounding boxes are shown in green. The predictions are projected onto the RGB images (upper row) for better visualization. Best viewed with color.

formation for the segmentation task.

Effect of center estimation task. The center estimation task brings substantial improvements $(0.6\%, 0.3\%, 0.8\%)$ on all the three subsets. The improvement is especially significant on hard subset. This is consistent with our expectation that learning the internal structure of object is essential for determining the object scale and shape when the data points are relatively sparse.

Effect of part-sensitive warping. The proposed PSWarp can further improve the performance by $(0.4\%, 0.3\%, 0.5\%)$ on the three subsets, validating the effectiveness of refining the classification confidences to the predicted bounding boxes. We also compare PSWarp with its counterpart PSRoIAlign with only re-scoring functionality. Three spatial resolutions for bounding-box alignment, 2×3, 3×5 and 4×7, are evaluated. As can be seen from Table 4, PSWarp exhibits comparable perfomance with PSRoIAlign in high-resolution cases and is only slightly worse that it in low-resolution cases. This is because the predicted bounding boxes typically occupy only a few pixels in the final feature map, thus the center point of each sub-window is representative enough. However, PSWarp avoids enumerating RoIs from a dense feature map, taking only around 1/10 runtime of PSRoIAlign.

4.5. Runtime analysis

We evaluate the runtime of each step of our framework during inference. We run our program as well as the base-line single-stage counterpart SECOND [25] in a moderate desktop equipped with an Intel i7 CPU and a 2080ti GPU. The overall pipeline has the following steps: 1) read the data from LiDAR file and remove the points that out of the range and image scope (Data), 2) encode the point clouds into a input tensor (Input), 3) process the encoded tensor by a neural network (Net), 4) remove the duplicated predictions (NMS). As shown in Table 5, our method has a total runtime of 40.1 ms. Compared to the baseline method SEC-OND [25], using a voxel-free encoding pre-process can help to save ~6.6 ms and PSWarp brings negligible extra cost (only 0.4 ms).

5. Conclusion

In this work, we studied the limitations of current single-stage 3D object detectors and proposed a new detector, namely structure-aware single-stage detector, for 3D point cloud detection. We first proposed to learn an auxiliary network using two point-level supervisions to guide the features learned in backbone network to be aware of the structure information of 3D objects. This significantly improved the detection accuracy without introducing additional computational cost at inference stage. We further developed a part-sensitive warping operation to mitigate the discordance between the predicted bounding boxes and their corresponding confidences in NMS post-processing. Experiments on the KITTI 3D/BEV detection benchmark showed that the proposed method achieved state-of-the-art performance with high efficiency.

References

[1] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017. 6, 7

[2] Y. Chen, S. Liu, X. Shen, and J. Jia. Fast point r-cnn. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2019. 2, 3, 6

[3] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016. 2, 5, 7

[4] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 2, 5

[5] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 3

[6] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015. 5, 7

[7] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018. 6, 7

[8] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 1, 3, 5, 6, 7

[9] B. Li. 3d fully convolutional network for vehicle detection in point cloud. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1513–1518. IEEE, 2017. 1, 3

[10] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7345–7353, 2019. 3, 6

[11] M. Liang, B. Yang, S. Wang, and R. Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 641–656, 2018. 6

[12] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 4, 5

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 4, 5

[14] T. Mordan, N. Thome, G. Henaff, and M. Cord. Revisiting multi-task learning with rock: a deep residual auxiliary block for visual detection. In *Advances in Neural Information Processing Systems*, pages 1310–1322, 2018. 3

[15] C. R. Qi, O. Litany, K. He, and L. J. Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2019. 2

[16] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018. 2, 3, 6, 7

[17] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 3

[18] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 4

[19] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019. 2, 3, 6, 7

[20] S. Shi, Z. Wang, X. Wang, and H. Li. Part-aˆ 2 net: 3d part-aware and aggregation neural network for object detection from point cloud. *arXiv preprint arXiv:1907.03670*, 2019. 2, 3

[21] A. Simonelli, S. R. R. Bulò, L. Porzi, M. López-Antequera, and P. Kontschieder. Disentangling monocular 3d object detection. *arXiv preprint arXiv:1905.12365*, 2019. 5

[22] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018. 1, 3

[23] Z. Wang and K. Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2019. 3, 6

[24] B. Wu, A. Wan, X. Yue, and K. Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018. 3

[25] Y. Yan, Y. Mao, and B. Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 1, 3, 5, 6, 7, 8

[26] B. Yang, M. Liang, and R. Urtasun. Hdnet: Exploiting hd maps for 3d object detection. In *Conference on Robot Learning*, pages 146–155, 2018. 3

[27] B. Yang, W. Luo, and R. Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018. 1, 3

[28] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia. STD: sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2019. 2, 3, 6, 7

[29] M. Zhao, J. Zhang, C. Zhang, and W. Zhang. Leveraging heterogeneous auxiliary tasks to assist crowd counting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12736–12745, 2019. 3

[30] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018. 1, 3, 5, 6, 7