STD: Sparse-to-Dense 3D Object Detector for Point Cloud

Zetong Yang[†] Yanan Sun[†] Shu Liu[†] Xiaoyong Shen[†] Jiaya Jia[†,‡]

[†]Youtu Lab, Tencent [‡]The Chinese University of Hong Kong

{tomztyang, now.syn, liushuhust, Goodshenxy}@gmail.com leojia@cse.cuhk.edu.hk

Abstract

We present a new two-stage 3D object detection framework, named sparse-to-dense 3D Object Detector (STD). The first stage is a bottom-up proposal generation network that uses raw point cloud as input to generate accurate proposals by seeding each point with a new spherical anchor. It achieves a high recall with less computation compared with prior works. Then, PointsPool is applied for generating proposal features by transforming their interior point features from sparse expression to compact representation, which saves even more computation time. In box prediction, which is the second stage, we implement a parallel intersection-over-union (IoU) branch to increase awareness of localization accuracy, resulting in further improved performance. We conduct experiments on KITTI dataset, and evaluate our method in terms of 3D object and Bird's Eye View (BEV) detection. Our method outperforms other state-of-the-arts by a large margin, especially on the hard set, with inference speed more than 10 FPS.

1. Introduction

3D scene understanding with point cloud is a very important topic in computer vision, since it benefits many applications, such as autonomous driving [8] and augmented reality [24]. In this work, we focus on one essential 3D scene recognition task, object detection based on point cloud, which predicts the 3D bounding box and class label for each object in the scene.

Compared to RGB images, LiDAR point cloud has its own unique properties. On the one hand, they provide structural and spatial information of relative location and precise depth. On the other hand, they are unordered, sparse and locality sensitive, which brings difficulties in parsing raw LiDAR point cloud.

Most existing work transforms sparse point clouds to compact representations by projecting it to images [4, 14, 9, 25, 7] or subdividing it into equally distributed voxels [23, 32, 37, 35]. CNNs can be applied for parsing the point cloud. Nevertheless, these hand-crafted representations may not be optimal. Instead of converting irregular point cloud to voxels, Qi *et al*. proposes PointNet [27, 28] to directly operate on raw LiDAR point clouds for classification and semantic segmentation. Two streams of methods tackling 3D object detection follow. One is based on voxels, *e.g*., VoxelNet [37] and SECOND [34], where voxelization is conducted on the entire point cloud, PointNet is applied to each voxel for feature extraction and CNNs are used for final bounding-box prediction. Although efficient, information loss is inevitable, degrading localization quality. The other stream is point-based, like F-PointNet [26] and PointRCNN [30]. They take raw point cloud data as input, and generate final predictions by PointNet++ [28]. These methods achieve better performance with uncontrollable receptive fields and large computation cost.

Our Contributions Different from all previous methods, we propose a two-stage 3D object detection framework. In the first stage, we take each point in the point cloud as an element, and seed them with appropriate spherical anchors, aiming to preserve accurate location information. Then a PointNet++ backbone is applied for extracting semantic context feature for each point as well as generating objectness score to filter anchors.

To generate feature for each proposal, we propose the PointsPool layer by gathering canonical coordinates and semantic features of their interior points, retaining accurate localization and context information. This layer transforms sparse and unordered point-wise expression to more compact features, enabling utilization of efficient CNNs and end-to-end training. Final prediction is achieved in the second stage. Instead of predicting the box location and class label with a simple head, we propose to augment a novel 3D IoU branch for predicting 3D IoU between predictions and ground-truth bounding boxes to alleviate inappropriate removal during post-processing.

We evaluate our model on KITTI dataset [1]. Experiments show that our model outperforms other state-of-the-arts in terms of both BEV and 3D object detection tasks, especially for difficult examples. Our primary contribution is manifold.

- We propose a point-based proposal generation

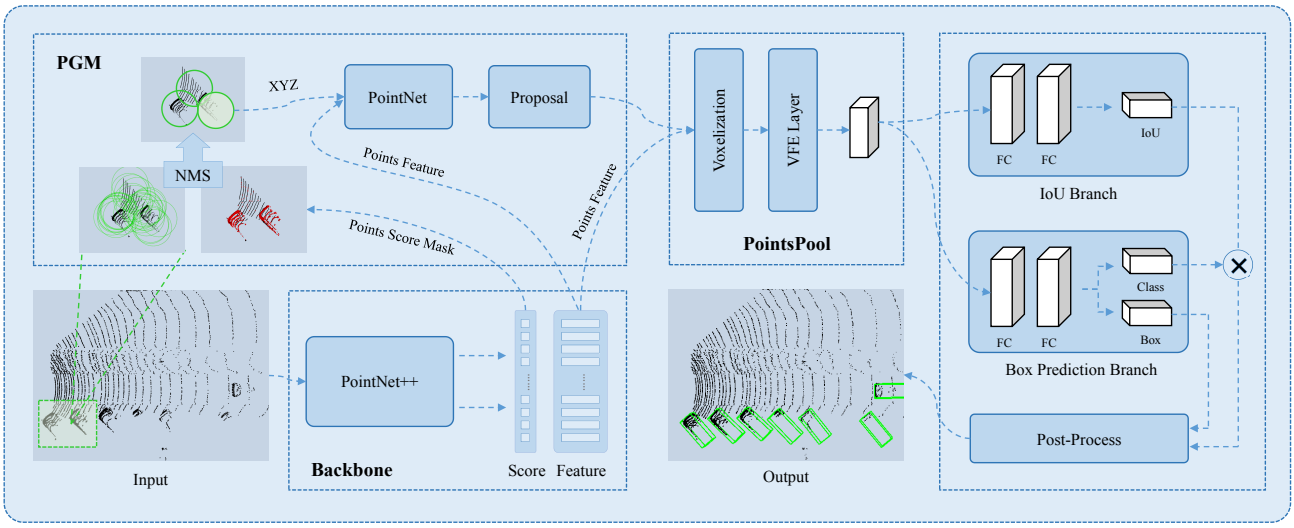


Figure 1. Illustration of our framework consisting of three different parts. The first is a proposal generation module (PGM) to generate accurate proposals from man-made point-based spherical anchors. The second part is a PointsPool layer to convert proposal features from sparse expression to compact representation. The final one is a box prediction network. It classifies and regresses proposals, and picks high-quality predictions.

paradigm for object detection on point cloud with spherical anchors. It is generic to achieve high recall.

- The proposed PointsPool layer integrates advantages of both point- and voxel-based methods, enabling efficient and effective prediction.
- Our new 3D IoU prediction branch helps alignment between classification score and localization, leading to notable improvement. Experimental results on KITTI dataset show that our framework handles many challenging cases with high occlusion and crowdedness, and achieves new state-of-the-art performance. Moreover, with our design, the good performance is achieved at a 10 FPS speed.

2. Related Work

3D Semantic Segmentation There are several approaches to tackle semantic segmentation on point cloud. In [33], a projection function converts LiDAR points to a UV map, which is then classified by 2D semantic segmentation [33, 36, 3] in pixel level. In [6, 5], a multi-view-based function produces the segmentation mask. This method fuses information from different views. Other solutions, such as [28, 27, 18, 12, 17], segment point cloud from raw LiDAR data. They directly generate features on each point while keeping original structural information. A max-pooling method gathers the global feature. It is then concatenated with local feature for processing.

3D Object Detection There are three different lines for 3D object detection. They are multi-view, voxel, and point-based methods.

For multi-view methods, MV3D [4] projects LiDAR point cloud to BEV and trains a Region Proposal Network (RPN) to generate positive proposals. It merges features from BEV, image view and front view in order to generate refined 3D bounding boxes. AVOD [14] improves MV3D by fusing image and BEV features like [20]. Unlike MV3D, which only merges features in the refinement phase, it also merges features from multiple views in the RPN phase to generate positive proposals. These methods still have the limitation when detecting small objects such as pedestrians and cyclists. They do not deal with cases with multiple objects in depth direction.

There are several LiDAR-data based 3D object detection frameworks using voxel-grid representation. In [32], each non-empty voxel is encoded with 6 statistical quantities by the points within this voxel. Binary encoding is used in [16] for each voxel grid. In PIXOR [35], each voxel grid is encoded as occupancy. All of these methods use hand-crafted representation. VoxelNet [37] instead stacks many VFE layers to generate machine-learned representation for each voxel. Compared to [37], SECOND [34] uses sparse convolution layers [10] for parsing the compact representation. PointPillars [15] uses pseudo-images as the representation after voxelization.

F-PointNet [26] is the first method of utilizing raw point cloud to predict 3D objects. It uses frustum proposals from 2D object detection as candidate boxes and regresses predictions based on interior points. Therefore, performance heavily relies on the 2D object detector. Differently, PointRCNN [30] uses the whole point cloud for proposal generation rather than 2D images. It directly uses the segmentation score of proposal's centric point for classification

considering proposal location information. Other features like size and orientation are neglected. In contrast, our design is general to utilize the strong representation power of point cloud.

3. Our Framework

Our method is a two-stage 3D object detection framework exploiting advantages of voxel- and point-based methods. To generate accurate point-based proposals, we design spherical anchors and a new strategy in assigning labels to anchors. For each generated proposal, we deploy a new PointsPool layer to convert point-based features from sparse expression to dense representation. A box prediction network is applied for final prediction. The framework is illustrated in Figure 1.

3.1. Proposal Generation Module

Existing methods of 3D object detection mainly project point cloud to different views or divide them into voxels for utilizing CNNs. We instead design a generic strategy to seed anchors based on each point independently, which is the elementary component in the point cloud. Then features of interior points of each anchor are utilized to generate proposals. With this structure, we keep sufficient context information, achieving decent recall even with a small number of proposals.

Challenge Albeit elegant, point-based frameworks inevitably face many challenges. For example, the amount of points is prohibitively huge where high redundancy exists in anchors. They cost much computation during training and inference. Also, the way to assign ground-truth labels for anchors needs to be specially designed.

Spherical Anchor The first step of proposal generation module is to reasonably seed anchors for each point. Considering that a 3D object could be with any orientations, we design spherical anchors rather than traditional cuboid anchors. For each spherical anchor, it is with a spherical receptive field parametrized by the class-specific radius (*i.e.*, 2-meter radius for car, and 1-meter radius for pedestrian and cyclist). Now the proposal predicted by each anchor is based on the points in the spherical receptive field. Each anchor is associated with a reference box for proposal generation, with pre-defined size. These anchors are located at the center of each point. Different from traditional anchor schemes, we do not pre-define the orientation of the reference box. It is instead directly predicted. As a result, the number of spherical anchors is not proportional to the number of pre-defined reference box orientation, leading to about 50% less anchors. With computation much reduced, we surprisingly achieve a much higher recall with spherical anchors than with traditional ones.

This step reduces the amount of anchors to about 16K. To further compress them, we use a 3D semantic segmentation network to predict the class of each point and produce semantic feature for each point. It is followed by non-maximal suppression (NMS) to remove redundant anchors. The final score of each anchor is the segmentation score on the center point. The IoU value is calculated based on the projection of each anchor to the BEV. With these operations, we reduce the number of anchors to around only 500.

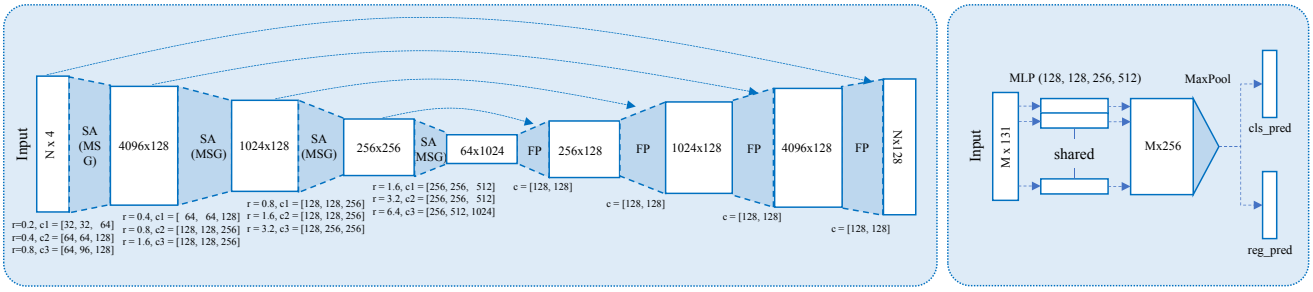
Proposal Generation Network These computed useful anchors lead to accurate proposals. Inspired by PointNet [27] in 3D classification, we gather 3D points within anchors for regression and classification. For points in an anchor, we pass their (X, Y, Z) locations, which are normalized by the anchor center coordinates, and semantic features from the segmentation network to a PointNet with several convolutional layers to predict classification scores, regression offsets and orientations. Details of the 3D segmentation networks and PointNet are illustrated in Figure 2.

Then we compute offsets regarding anchor center coordinates (A_x, A_y, A_z) and their pre-defined sizes (A_l, A_w, A_h) so as to obtain precise proposals. The pre-defined size for "car", "cyclist" and "pedestrian" are $(A_l = 3.9, A_w = 1.6, A_h = 1.6)$, $(A_l = 1.6, A_w = 0.8, A_h = 1.6)$ and $(A_l = 0.8, A_w = 0.8, A_h = 1.6)$, respectively. For angle prediction, we use a hybrid of classification and regression formulation following [26]. That is, we pre-define N_a as equally split angle bins and classify the proposal angle into different bins. Residual is regressed with respect to the bin value. N_a is set to 12 in our experiments. Finally, we apply NMS based on classification score and oriented BEV IoU to eliminate redundant proposals. Specifically, we keep up to 300 proposals during training and 100 for testing.

Assignment Strategy Given that our anchors are with spherical receptive fields rather than cubes or cuboids, it is not appropriate to assign positive or negative labels according to traditional IoU calculation [37] between the spherical receptive field and ground-truth boxes. We design a new criterion named PointsIoU to assign target labels. PointsIoU is defined as the quotient between the number of points in the intersection area of both regions and the number of points in the union area of both regions. An anchor is considered positive if its PointsIoU with a certain ground-truth box is higher than 0.55 and negative otherwise.

3.2. Proposal Feature Generation

With semantic features from the segmentation network for each point and refined proposals, we constitute compact features for each proposal.



(a) (b)

Figure 2. Illustration of networks in the proposal generation module. (a) 3D segmentation network (PointNet++). It takes a raw point cloud (x, y, z, r) as input, and generates semantic segmentation scores as well as global context features for each point by stacking SA layers and FP modules. (b) Proposal generation Network (PointNet). It treats normalized coordinates and semantic features of points within anchors as input, and produces classification and regression predictions.

Motivation For each proposal, the most straight-forward way to make final prediction is to perform PointNet++ based on interior points [30, 26]. Albeit simple, several operations such as *set abstraction* (SA) are computational expensive compared to traditional convolution or fully connected (FC) layers. As illustrated in Table 1, with 100 proposals, PointNet++ baseline takes 41ms during inference, compared with 16ms with pure FC layers. It is almost $2.5\times$ faster than the baseline, with only 0.4% performance drop. Moreover, compared to PointNet baseline, the model with FC layers yields 1.6% performance increase with only 6 extra milliseconds. It is because PointNet regression head uses less local information.

We apply a voxelization layer at this stage, named PointsPool, to compute compact proposal features that can be used in efficient FC layers for final predictions. Compared to voxelization in [37], this new layer is a gradient-conductive voxelization layer, enabling end-to-end training.

PointsPool Layer PointsPool layer is composed of three steps. In the first step, we randomly choose N interior points for each proposal with their canonical coordinates and semantic features as initial feature. For each proposal, we obtain point canonical locations by subtracting the proposal center (X, Y, Z) values and rotating them to the proposal predicted orientation. These canonized coordinates enable the model to be robust under geometrical transformation, and be aware of inner points' relative locations for better performance than only using semantic features.

The second step is using the voxelization layer to subdivide each proposal into equally spaced voxels as [37]. Specifically, we partition each proposal to $(d_l = 6, d_w = 6, d_h = 6)$ voxels. $N_r = 35$ points are randomly sampled for each voxel. Concatenated features of canonical coordinates and semantic features of these points are used for each voxel. Compared to voxelization in [37], this layer has gradient representation, making it possible for end-to-end training. While passing gradients, we only need to pass gradients of these randomly selected points. Finally, we apply a Voxel Feature Encoding (VFE) layer with channels $(128, 128, 256)$ [37] for extracting features of each voxel, so as to generate features of proposals with shapes $(d_l \times d_w \times d_h \times 256)$. After getting features of each proposal, we flatten them for following FC layers in the box prediction head.

Methods	Proposal No.	Inference Time	Moderate
PointNet (4 conv layers)	100	10 ms	77.1
PointNet++ (3 SA)	100	41 ms	79.1
PointsPool + 2FC	100	16 ms	78.7

Table 1. 3D object detection AP on KITTI val moderate set. We compare inference time and AP among different box regression network architectures.

Score-NMS	IoU-NMS	Easy	Moderate	Hard
√	-	88.8	78.7	78.2
-	√	90.9	90.9	90.6

Table 2. 3D object detection AP on KITTI val set. We conduct experiments to show importance of the post process. "Score-NMS" means using classification scores as NMS sorting scores. "IoU-NMS" means for each prediction, we use its largest IoU among all ground-truth boxes as the sorting score.

3.3. Box Prediction Network

Our box prediction network has two branches for box estimation and IoU estimation respectively.

Box Estimation Branch In this branch, we use 2 FC layers with channels $(512, 512)$ to extract features of each proposal. Then another 2 FC layers are applied for classification and regression respectively. We directly regress offsets between the ground-truth box and proposals, parametrized by (t_l, t_w, t_h). We further predict the shift (t_x, t_y, t_z) from proposal center to the ground-truth. As for angle prediction, we still use a hybrid of classification and regression formulation, same as the one described in Section 3.1.

IoU Estimation Branch In previous work [15, 34, 37, 14, 30], NMS is applied to results of box estimation to remove duplicate predictions. The classification score is used for ranking during NMS. Noted in [11, 22, 29], the classification scores of boxes are not highly correlated with the localization quality. Similarly, weak correlation between classification score and box quality affects point-based object detection tasks. Given that LiDAR for autonomous driving is usually gathered at a fixed angle, and objects are partially covered, localization accuracy is extremely sensitive to relative position between visible part and its full view while the classification branch cannot provide enough information. As shown in Table 2, if we feed the oracle IoU value of each predicted box instead of the classification score to NMS for duplicate removal, the performance increases by around 12.6%.

Based on this fact, we develop an IoU estimation branch for predicting 3D IoU between boxes and corresponding ground-truth. Then, we multiply each box's classification score with its 3D IoU as a new sorting criterion. This design relieves the discrepancy between localization accuracy and classification score, effectively improving the final performance. Moreover, this IoU estimation branch is general and can be applied to other 3D object detectors. We expect similar performance improvement on other frameworks.

3.4. Loss Function

We use a multi-task loss to train our network. Our total loss is composed of proposal generation loss L_{prop} and box prediction loss L_{box} as

$$L_{total} = L_{prop} + L_{box}. \tag{1}$$

The proposal generation loss is the summation of 3D semantic segmentation loss and proposal prediction loss. We use focal loss [21] as segmentation loss L_{seg}, keeping the original parameters $\alpha_t = 0.25$ and $\gamma = 2$. The prediction loss consists of proposal classification loss and regression loss. The overall proposal generation loss is defined as Eq. (2). s_i and u_i are the predicted classification score and ground-truth label for anchor i, respectively. N_{cls} and N_{pos} are the number of anchors and positive samples.

$$\begin{aligned} L_{prop} = L_{seg} &+ \frac{1}{N_{cls}} \sum_i L_{cls}(s_i, u_i) \\ &+ \lambda \frac{1}{N_{pos}} \sum_i [u_i \geq 1](L_{loc} + L_{ang}), \end{aligned} \tag{2}$$

where the Iverson bracket indicator function $[u_i \geq 1]$ reaches 1 when $u_i \geq 1$ and 0 otherwise. L_{cls} is simply the softmax cross-entropy loss. We parameterize an anchor A by its center (A_x, A_y, A_z) and size (A_l, A_w, A_h). Its ground truth box G is with (G_x, G_y, G_z) and (G_l, G_w, G_h). Location regression loss is composed of center residual prediction loss and size residual prediction loss, formulated as

$$L_{loc} = L_{dis}(A_{ctr}, G_{ctr}) + L_{dis}(A_{size}, G_{size}), \tag{3}$$

where L_{dis} is the smooth-l_1 loss. A_{ctr} and A_{size} are predicted center residual and size residual by the proposal generation network, while G_{ctr} and G_{size} are targets for them. The target of our network is defined as

$$\begin{cases} G_{ctr} &= G_j - A_j, & j \in (x,\ y,\ z) \\ G_{size} &= (G_j - A_j)/A_j, & j \in (l,\ w,\ h). \end{cases} \tag{4}$$

Angle loss includes orientation classification loss and residual prediction loss as

$$L_{angle} = L_{cls}(t_{a-cls}, v_{a-cls}) + L_{dis}(t_{a-res}, v_{a-res}), \tag{5}$$

where t_{a-cls} and t_{a-res} are predicted angle class and residual while v_{a-cls} and v_{a-res} are their targets.

The box prediction loss is almost the same as the proposal prediction loss, mentioned above, with two extra losses, which are 3D IoU loss and corner loss. When training IoU branch, we use 3D IoU between proposals and corresponding ground-truth boxes as ground-truth, and smooth-l_1 loss as the loss function. Corner loss is the distance between the predicted 8 corners and assigned ground-truth, expressed as

$$L_{corner} = \sum_{k=1}^{8} \|P_k - G_k\|, \tag{6}$$

where P_k and G_k are the location of ground-truth and prediction for point k.

4. Experiments

We evaluate our method on the widely used KITTI Object Detection Benchmark [1]. There are 7,481 training images / point clouds and 7,518 test images / point clouds with three categories of Car, Pedestrian and Cyclist. We use average precision (AP) metric to compare with different methods. During evaluation, we follow the official KITTI evaluation protocol – that is, the IoU threshold is 0.7 for class Car and 0.5 for Pedestrian and Cyclist.

4.1. Implementation Details

Following former works [37, 15, 14, 34], in order to avoid IoU misalignment in KITTI evaluation protocol on Car, Pedestrian and Cyclist, we train two networks, one for car and the other for both pedestrian and cyclist.

Network Architecture To align network input, we randomly choose 16K points from the entire point cloud for each scene. Our 3D semantic segmentation network is

Class	Method	Modality	$AP_{BEV}(\%)$			$AP_{3D}(\%)$		
			Easy	Moderate	Hard	Easy	Moderate	Hard
Car	MV3D [4]	RGB + LiDAR	86.02	76.90	68.49	71.09	62.35	55.12
	AVOD [14]		86.80	85.44	77.73	73.59	65.78	58.38
	F-PointNet [26]		88.70	84.00	75.33	81.20	70.39	62.19
	AVOD-FPN [14]		88.53	83.79	77.90	81.94	71.88	66.38
	RoarNet [31]		88.75	86.08	78.80	83.95	75.79	67.88
	UberATG-MMF [19]		89.49	87.47	79.10	**86.81**	76.75	68.41
	VoxelNet [37]	LiDAR	89.35	79.26	77.39	77.47	65.11	57.73
	SECOND [34]		88.07	79.37	77.95	83.13	73.66	66.20
	PointPillars [15]		88.35	86.10	79.83	79.05	74.99	68.30
	PointRCNN [30]		89.47	85.68	79.10	85.94	75.76	68.32
	Ours		**89.66**	**87.76**	**86.89**	86.61	**77.63**	**76.06**
Pedestrian	AVOD [14]	RGB + LiDAR	42.51	35.24	33.97	38.28	31.51	26.98
	F-PointNet [26]		58.09	50.22	47.20	51.21	**44.89**	40.23
	AVOD-FPN [14]		58.75	51.05	**47.54**	50.80	42.81	40.88
	VoxelNet [37]	LiDAR	46.13	40.74	38.11	39.48	33.69	31.51
	SECOND [34]		55.10	46.27	44.76	51.07	42.56	37.29
	PointPillars [15]		58.66	50.23	47.19	52.08	43.53	41.49
	Ours		**60.99**	**51.39**	45.89	**53.08**	44.24	**41.97**
Cyclist	AVOD [14]	RGB + LiDAR	63.66	47.74	46.55	60.11	44.90	38.80
	F-PointNet [26]		75.38	61.96	54.68	71.96	56.77	50.39
	AVOD-FPN [14]		68.09	57.48	50.77	64.00	52.18	46.61
	VoxelNet [37]	LiDAR	66.70	54.76	50.55	61.22	48.36	44.37
	SECOND [34]		73.67	56.04	48.78	70.51	53.85	46.90
	PointPillars [15]		79.14	62.25	56.00	75.78	59.07	52.92
	Ours		**81.04**	**65.32**	**57.85**	**78.89**	**62.53**	**55.77**

Table 3. Performance on KITTI test set for both Car, Pedestrian and Cyclists.

based on PointNet++ with four SA levels and four *feature propagation* (FP) layers. The proposal generation subnetwork is a multi-layer perception consisting of four hidden layers with channels $(128, 128, 256, 512)$, followed by a PointsPool layer where we randomly sample $N = 512$ interior points per proposal as its initial input. These representations are then passed to the box regression network. Both the box estimation and IoU estimation branches consist of 2 fully connected layers with 512 channels.

Training Parameters Our model is trained stage-by-stage to save GPU memory. The first stage consists of 3D semantic segmentation and proposal generation, while the second is for box prediction. For the first stage, we use ADAM [13] optimizer with an initial learning rate 0.001 for the first 80 epochs and then decay it to 0.0001 for the last 20 epochs. Each batch consists of 16 point clouds evenly distributed on 4 GPU cards. For the second stage, we train 50 epochs with batch size 1. The learning rate is initialized as 0.001 for first 40 epochs and is then decayed by 0.1 in every 5 epochs. For each input point cloud, we sample 256 proposals, with ratio 1:1 for positives and negatives. Our implementation is based on Tensorflow [2]. For the box prediction network, a proposal is considered positive if its maximum 3D IoU with all ground-truth boxes is higher than 0.55 and negative if its maximum 3D IoU is below 0.45 during training the car model. The positive and negative 3D IoU thresholds are 0.5 and 0.4 for the pedestrian and cyclist models. Besides, for the IoU branch, we only train on positive proposals.

Data Augmentation Data augmentation is important to prevent overfitting. First, similar to that of [34], we randomly add several ground-truth boxes with their interior points from other scenes to current point cloud in order to simulate objects with various environments. Then, for each bounding box, we randomly rotate it following a uniform distribution $\Delta\theta_1 \in [-\pi/4, +\pi/4]$ and randomly add a translation $(\Delta x, \Delta y, \Delta z)$. Third, each point cloud is flipped along the x-axis in camera coordinate with probability 0.5. We also randomly rotate each point cloud around z-axis (up axis) by a uniformly distributed random variable $\Delta\theta_2 \in [-\pi/4, +\pi/4]$. Finally, we apply a global scaling to point cloud with a random variable drawn from the uniform distribution $[0.9, 1.1]$.

4.2. Main Results

For evaluation on the test set, we train the model on split train/val sets at ratio 4:1. The performance of our method and comparison with previous methods are listed in Table 3. Our model outperforms other methods by a large margin on Car and Cyclist classes, especially on the hard set. Compared to multi-view methods that use other sensors as extra information, our method still achieves higher AP with input of only raw point cloud. Compared to UberATG-MMF [19], which is the best multi-sensor detector, STD outperforms it by 0.88% on the moderate level on 3D detection of Cars. Also large increase 7.65% on the hard set is obtained, man-

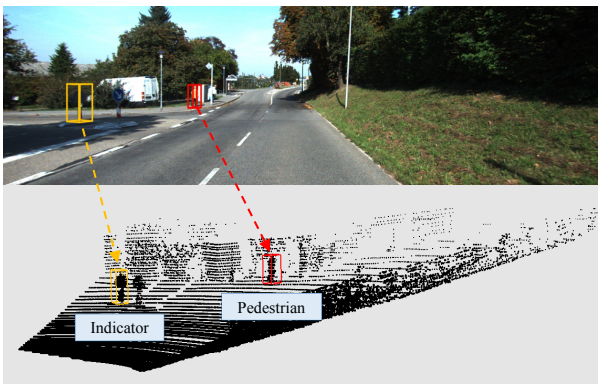


Figure 3. Small objects such as indicators are easy to detect on RGB images, but not on LiDAR data.

Class	Easy	Moderate	Hard
Car (BEV)	90.5	88.5	88.1
Car (3D)	89.7	79.8	79.3
Pedestrian (BEV)	75.9	69.9	66.0
Pedestrian (3D)	73.9	66.6	62.9
Cyclist (BEV)	89.6	76.0	72.7
Cyclist (3D)	88.5	72.8	67.9

Table 4. 3D and BEV detection AP on KITTI val set.

Method	Easy	Moderate	Hard
MV3D [4]	71.29	62.68	56.56
AVOD [14]	84.41	74.44	68.65
VoxelNet [37]	81.97	65.46	62.85
SECOND [34]	87.43	76.48	69.10
F-PointNet [26]	83.76	70.92	63.65
PointRCNN [30]	88.88	78.63	77.38
Ours (without IoU branch)	88.8	78.7	78.2
Ours (IoU branch)	**89.7**	**79.8**	**79.3**

Table 5. 3D detection AP on KITTI val set of our model for "Car" compared to other state-of-the-art methods.

Methods	Proposals num	IoU threshold	Recall
AVOD [14]	50	0.5	91.0
Ours	50	0.5	**96.3**
PointRCNN [30]	100	0.7	74.8
Ours	100	0.7	**76.8**

Table 6. Recall of proposals on KITTI val set compared to other methods with the same proposal number and IoU threshold.

shape	anchors amount	proposals num	Recall(IoU=0.7)
cuboid	$1\times$	100	74.2
cuboid	$2\times$	100	75.7
sphere	$1\times$	100	**76.8**

Table 7. Recall of generated proposals on KITTI val set.

ifesting the effectiveness of our proposal-generation module and IoU branch.

Note that on Pedestrian class, STD is still the best among LiDAR-only detectors. Multi-sensor detectors work better because there are very few 3D points on pedestrians, making it difficult to distinguish them from other small objects like indicator or telegraph pole, as shown in Figure 3. Extra information of RGB would help in these cases.

Compared to LiDAR-only detectors, and voxel or point methods, our method works best on all three classes. Specifically, on Car detection, STD achieves a better AP by 1.87%, 2.64% and 3.97% compared to PointRCNN [30], PointPillars [15] and SECOND [34] respectively on the moderate set. The improvement on the hard set is more significant – 7.74%, 7.76% and 9.86% increase respectively. We present several qualitative results in Figure 4.

4.3. Ablation studies

For ablation studies, we follow VoxelNet [37] to split the official training set into a *train* set of 3,717 images/scenes and a *val* set of 3,769 images/scenes. Images in train/val set belong to different video clips. Following [37], all ablation studies are conducted on the `car` class due to the relatively large amount of data to make system run stably.

Results On Validation Set We first report the performance on KITTI val set in Table 4. Our results on validation set compared to other methods are listed in Table 5. Unlike voxel-based methods of [37] and [34], our model preserves more structure and appearance details, leading to better performance. Compared to point-based methods, the proposal generation module and IoU branch keep more accurate proposals and high-quality predictions, which brings higher AP especially on the hard set. We compare recall among different 2-stage object detectors in Table 6, demonstrating the powerful of our proposal generation module.

Effect of Anchors Receptive Field Given that anchors play an important role, it is important to make anchors cover as much as possible ground-truth area while not consuming much computation. We use spherical receptive field that has only one radius for each detection model. In order to justify the effectiveness of this design, we conduct experiments varying the shape and size of receptive fields. Average Recall (AR) with IoU threshold 0.7 is the metric. The result is shown in Table 7. First, "cuboid" shape of receptive field needs more than one angles, *i.e.* $(0, \pi/2)$, because of the disproportion between length and width, leading to $2\times$ more data and accordingly more computation. "Cuboid" with only one orientation causes 1.5% decrease in terms of AR. Moreover, spherical receptive field brings additional context information, which benefits anchor classification and regression.

Effect of Proposal Feature Our proposal features are with canonical coordinates and 3D semantic features. We quantify their benefits using original points coordinates as our baseline. As shown in Table 8, using 3D segmentation features results in around 36.5% performance boost on moderate set in terms of AP. It means global context information enhances model capability greatly. With canonical

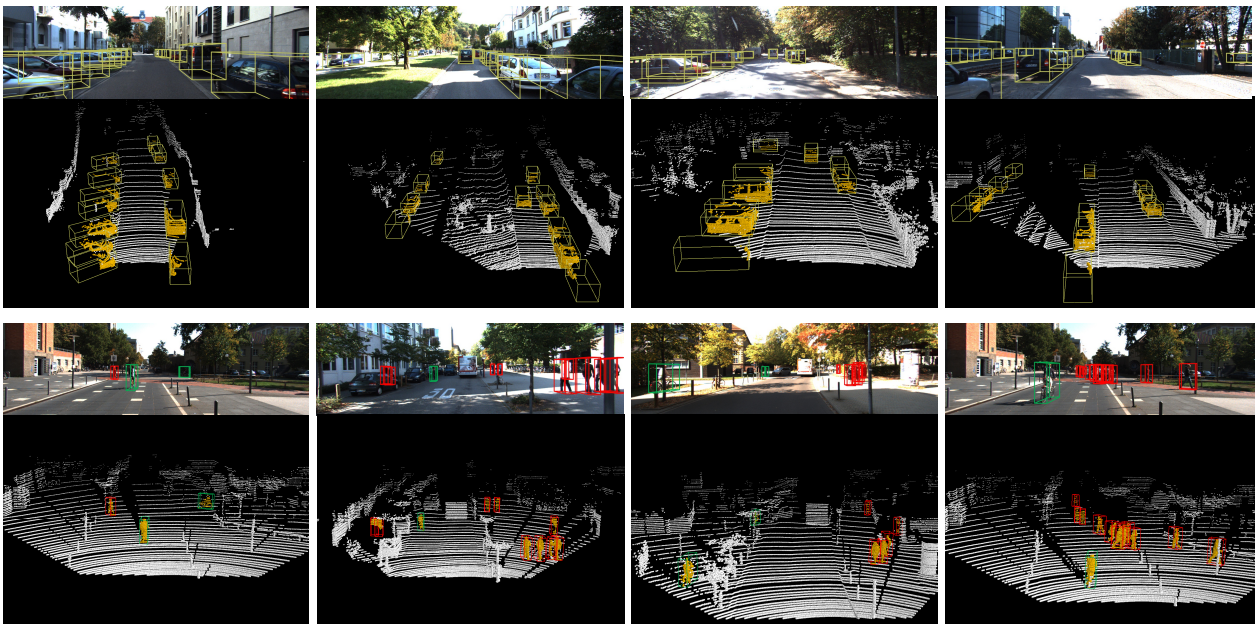


Figure 4. Visualization of our results on KITTI test set. Cars, pedestrians and cyclists are highlighted in yellow, red and green respectively. The upper row in each image is the 3D object detection result projected onto the RGB image. The other is the result in the LiDAR phase.

semantic	canonized	Easy	Moderate	Hard
-	-	38.7	31.1	26.0
√	-	82.5	67.6	67.2
√	√	**88.8**	**78.7**	**78.2**

Table 8. 3D object detection AP on KITTI val set. A tick in canonized item means using canonical coordinates rather than original coordinates as part of feature. A tick in "semantic" means using points feature from 3D semantic segmentation backbone in proposal feature.

NMS	Soft-NMS	3D	Easy	Moderate	Hard
√	-	-	88.8	78.7	78.2
-	√	-	88.9	79.0	78.4
-	-	√	**89.7**	**79.8**	**79.3**

Table 9. 3D object detection AP on KITTI val moderate set. Our experiments analyze influence of our 3D IoU branch. "3D" means using 3D IoU branch for post-processing.

NMS sorting score	Easy	Moderate	Hard
3D-IoU	89.0	79.1	78.7
cls-score × 3D-IoU	**89.7**	**79.8**	**79.3**

Table 10. 3D object detection AP on KITTI val moderate set. Our experiments analyze influence of different ways to use 3D IoU branch. "3D-IoU" means only using 3D IoU as NMS sorting score. "cls-score × 3D-IoU" indicates the way we describe in Section 3.3.

transformation, AP increases by 11.1% on moderate set.

Effect of IoU Branch Our 3D IoU prediction branch estimates the localization quality to finally increase performance. As illustrated in Table 9, our 3D IoU guided NMS outperforms traditional methods of NMS and soft-NMS by 1.1% and 0.8% on moderate set respectively, manifesting the usefulness of this branch. We note directly taking predicted 3D IoU as the NMS sorting criterion, as shown in Table 10, performs not well. The reason is that only positive proposals are considered in IoU branch, while classification score can tell positive predictions from negative ones. Accordingly, combination of classification score and predicted IoU is very effective.

Inference Time The total inference time of STD is 80ms on a TitanV GPU where the PointNet++ backbone takes 54ms, the proposal generation module including PointNet and NMS takes 10ms, PointsPool layer takes about 6ms, and the second stage with two branches takes 10ms. STD is the fastest model among all point-based methods and multi-view methods, manifesting the reasonable design of STD. Note that, we merge batch normalization into convolution layers, and split the input point cloud of first SA level $(16K)$ in PointNet++ to (32×512) for parallel computation so as to shorten inference time, resulting in 25ms and 50ms speedup respectively with performance unchanged.

5. Conclusion

We have proposed a new two-stage 3D object detection framework that takes advantages of both voxel- and point-based methods. We introduce spherical anchors based on points and refine them for accurate proposal generation without loss of localization information in the first stage. Then a PointsPool layer is applied for generating compact representations for proposals which is beneficial to reduce inference time. The second stage reduces incorrect removal

in post-process which further improves performance. Our model works decently on 3D detection, especially on the hard set.

References

[1] "kitti 3d object detection benchmark". http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d, 2019.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, 2016.

[3] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018.

[4] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 2017.

[5] A. Dai, A. X. Chang, M. Savva, M. Halber, T. A. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017.

[6] A. Dai and M. Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. In *ECCV*, 2018.

[7] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *ICRA*, 2017.

[8] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012.

[9] A. González, G. Villalonga, J. Xu, D. Vázquez, J. Amores, and A. M. López. Multiview random forest of local experts combining RGB and LIDAR data for pedestrian detection. In *IV*, 2015.

[10] B. Graham, M. Engelcke, and L. van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018.

[11] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang. Acquisition of localization confidence for accurate object detection. In *ECCV*, 2018.

[12] M. Jiang, Y. Wu, and C. Lu. Pointsift: A sift-like network module for 3d point cloud semantic segmentation. *CoRR*, 2018.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, 2014.

[14] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander. Joint 3d proposal generation and object detection from view aggregation. *CoRR*, 2017.

[15] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. *CVPR*, 2019.

[16] B. Li. 3d fully convolutional network for vehicle detection in point cloud. In *IROS*, 2017.

[17] J. Li, B. M. Chen, and G. H. Lee. So-net: Self-organizing network for point cloud analysis. *CoRR*, 2018.

[18] Y. Li, R. Bu, M. Sun, and B. Chen. Pointcnn. *CoRR*, 2018.

[19] M. Liang*, B. Yang*, Y. Chen, R. Hu, and R. Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *CVPR*, 2019.

[20] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[21] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[22] S. Liu, C. Lu, and J. Jia. Box aggregation for proposal decimation: Last mile of object detection. In *ICCV*, 2015.

[23] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, 2015.

[24] Y. Park, V. Lepetit, and W. Woo. Multiple 3d object tracking for augmented reality. In *ISMAR*, 2008.

[25] C. Premebida, J. Carreira, J. Batista, and U. Nunes. Pedestrian detection combining RGB and dense LIDAR data. In *ICoR*, 2014.

[26] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from RGB-D data. *CoRR*, 2017.

[27] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[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 *NIPS*, 2017.

[29] L. Qi, S. Liu, J. Shi, and J. Jia. Sequential context encoding for duplicate removal. In *NIPS*, 2018.

[30] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019.

[31] K. Shin, Y. Kwon, and M. Tomizuka. Roarnet: A robust 3d object detection based on region approximation refinement. *arXiv preprint arXiv:1811.03818*, 2018.

[32] D. Z. Wang and I. Posner. Voting for voting in online point cloud object detection. In *Robotics: Science and Systems XI*, 2015.

[33] 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 *ICRA*, 2018.

[34] Y. Yan, Y. Mao, and B. Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018.

[35] B. Yang, W. Luo, and R. Urtasun. PIXOR: real-time 3d object detection from point clouds. In *CVPR*, 2018.

[36] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[37] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *CoRR*, 2017.