

# Behind the Curtain: Learning Occluded Shapes for 3D Object Detection

Qiangeng Xu, Yiqi Zhong, Ulrich Neumann

University of Southern California  
qiangenx@usc.edu, yiqizhon@usc.edu, uneumann@usc.edu

## Abstract

Advances in LiDAR sensors provide rich 3D data that supports 3D scene understanding. However, due to occlusion and signal miss, LiDAR point clouds are in practice 2.5D as they cover only partial underlying shapes, which poses a fundamental challenge to 3D perception. To tackle the challenge, we present a novel LiDAR-based 3D object detection model, dubbed *Behind the Curtain Detector (BtcDet)*, which learns the object shape priors and estimates the complete object shapes that are partially occluded (curtained) in point clouds. BtcDet first identifies the regions that are affected by occlusion and signal miss. In these regions, our model predicts the probability of occupancy that indicates if a region contains object shapes. Integrated with this probability map, BtcDet can generate high-quality 3D proposals. Finally, the probability of occupancy is also integrated into a proposal refinement module to generate the final bounding boxes. Extensive experiments on the KITTI Dataset and the Waymo Open Dataset demonstrate the effectiveness of BtcDet. Particularly, for the 3D detection of both cars and cyclists on the KITTI benchmark, BtcDet surpasses all of the published state-of-the-art methods by remarkable margins. Code is released.

## Introduction

With high-fidelity, the point clouds acquired by LiDAR sensors significantly improved autonomous agents’s ability to understand 3D scenes. LiDAR-based models achieved state-of-the-art performance on 3D object classification (Xu et al. 2020), visual odometry (Pan et al. 2021), and 3D object detection (Shi et al. 2020). Despite being widely used in these 3D applications, LiDAR frames are technically 2.5D. After hitting the first object, a laser beam will return and leave the shapes behind the occluder missing from the point cloud.

To locate a severely occluded object (e.g., the car in Figure 1(b)), a detector has to recognize the underlying object shapes even when most of its parts are missing. Since shape miss inevitably affects object perception, it is important to answer two questions:

- *What are the causes of shape miss in point clouds?*
- *What is the impact of shape miss on 3D object detection?*

Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

## Causes of Shape Miss

To answer the first question, we study the objects in KITTI (Geiger et al. 2013) and discover three causes of shape miss.

**External-occlusion.** As visualized in Figure 1(c), occluders block the laser beams from reaching the red frustums behind them. In this situation, the external-occlusion is formed, which causes the shape miss located at the red voxels.

**Signal miss.** As Figure 1(c) illustrates, certain materials and reflection angles prevent laser beams from returning to the sensor after hitting some regions of the car (blue voxels). After projected to range view, the affected blue frustums in Figure 1(c) appear as the empty pixels in Figure 1(a).

**Self-occlusion.** LiDAR data is 2.5D by nature. As shown in Figure 1(d), for a same object, its parts on the far side (the green voxels) are occluded by the parts on the near side. The shape miss resulting from self-occlusion inevitably happens to every object in LiDAR scans.

## Impact of Shape Miss

To analyze the impact of shape miss on 3D object detection, we evaluate the car detection results of the scenarios where we recover certain types of shape miss on each object by borrowing points from similar objects (see the details of finding similar objects and filling points in Sec. 3.1).

In each scenario, after resolving certain shape miss in both the *train* and *val* split of KITTI (Geiger et al. 2013), we train and evaluate a popular detector PV-RCNN (Shi et al. 2020). The four scenarios are:

- **NR:** Using the original data without shape miss recovery.
- **EO:** Recovering the shape miss caused by external-occlusion (adding the red points in Figure 2(a)).
- **EO+SM:** Recovering the shape miss caused by external-occlusion and signal miss (adding the red and blue points in Figure 2(a)).
- **EO+SM+SO:** Recovering all the shape miss (adding the red, blue and green points in Figure 2(a)).

We report detection results on cars with three occlusion levels (level labels are provided by the dataset). As shown in Figure 2(b), without recovery (NR), it is more difficult to detect objects with higher occlusion levels. Recovering shapes miss will reduce the performance gaps between objects with different levels of occlusion. If all shape miss are resolved

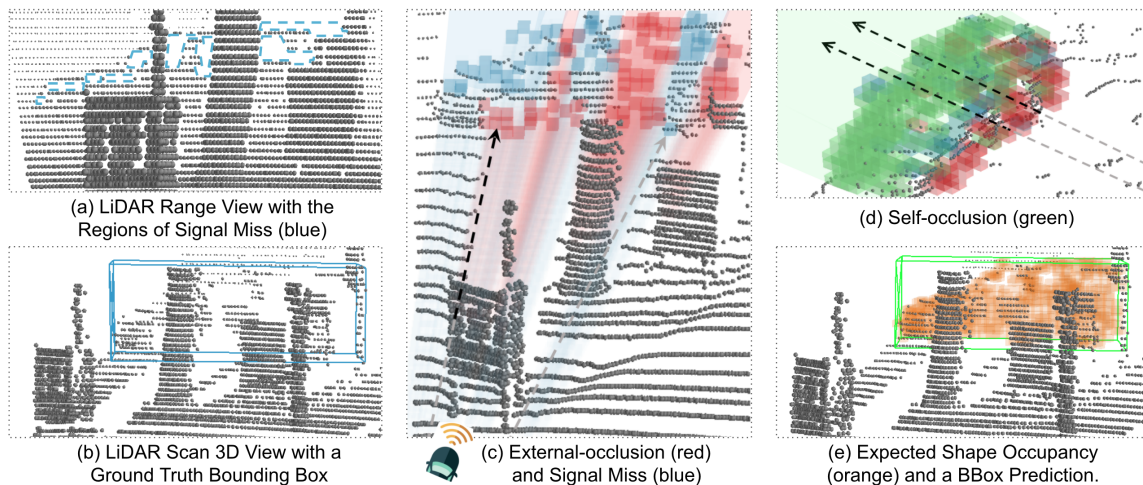
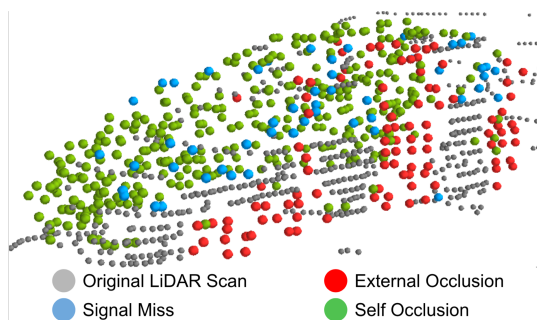
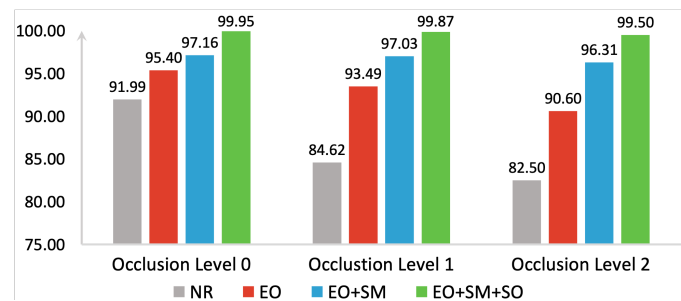


Figure 1: In a LiDAR scan (a) and (b), locating an object is difficult when its shape is largely missing. We discover three causes of shape miss: external-occlusion (red regions in (c)), signal miss (blue regions in (c)), and self-occlusion (green regions in (d)). BtcDet learns the occupancy probability of complete object shapes (e) and achieves the state-of-the-art detection performance.



(a) The points to recover different shape miss regions.



(b) The 3D Average Precisions with shape miss recovery.

Figure 2: The impact of the three types of shape miss. (b) shows PV-RCNN’s (Shi et al. 2020) car 3D detection APs with different occlusion levels on the KITTI (Geiger et al. 2013) *val* split. NR means no shape miss recovery. EO, SM, and SO indicate adding car points in the regions of external-occlusion, signal miss and self-occlusion, respectively, as visualized in (a).

(EO+SM+SO), the performance gaps are eliminated and almost all objects can be effectively detected (APs > 99%).

## The Proposed Method

The above experiment manually resolves the shape miss by filling points into the labeled bounding boxes and significantly improve the detection results. However, during test time, how do we resolve shape miss without knowing bounding box labels?

In this paper, we propose *Behind the Curtain Detector* (*BtcDet*). To the best of our knowledge, BtcDet is the first 3D object detector that targets the object shapes affected by occlusion. With the knowledge of shape priors, BtcDet estimates the probability of **shape occupancy**, *the region occupancy of complete objects as if there is no occlusion and signal miss*. After being integrated into the detection pipeline, the occupancy estimation benefits both region proposal generation and proposal refinement. Eventually, BtcDet surpasses all of the state-of-the-art methods published to date by remarkable margins.

## Related Work

**LiDAR-based 3D object detectors.** Voxel-based methods divide point clouds by voxel grids to extract features (Zhou and Tuzel 2018). Some of them also use sparse convolution to improve model efficiency, e.g., SEC-OND (Yan et al. 2018). Point-based methods such as PointRCNN (Shi et al. 2019) generate proposals directly from points. STD (Yang et al. 2019) applies sparse to dense refinement and VoteNet (Qi et al. 2019a) votes the proposal centers from point clusters. These models are supervised on the ground truth bounding boxes without explicit consideration for the object shapes.

**Learning shapes for 3D object detection.** Bounding box prediction requires models to understand object shapes. Some detectors learn the shape related statistics as an auxiliary task. PartA<sup>2</sup> (Shi et al. 2020) learns object part locations. SA-SSD and AssociateDet (He et al. 2020; Du et al. 2020) use auxiliary networks to preserve structural information. Studies (Li et al. 2021; Yan et al. 2020; Najibi et al.

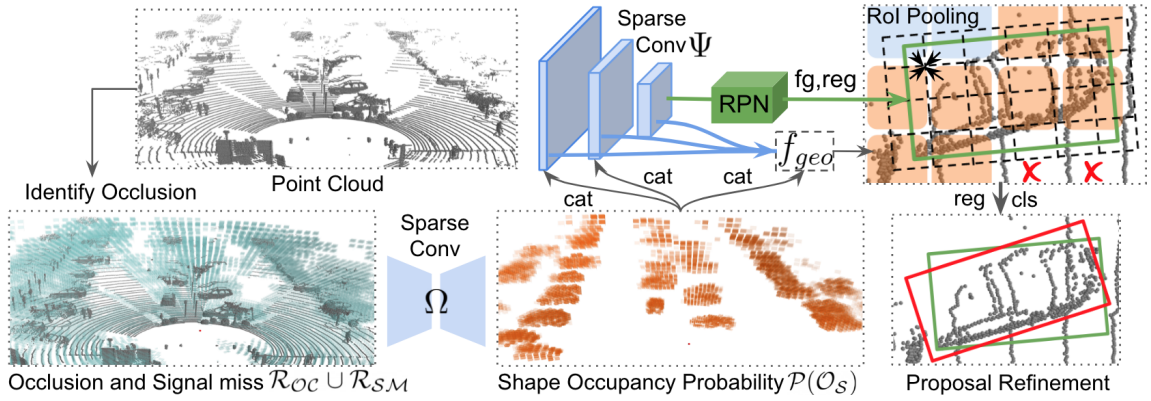


Figure 3: The detection pipeline. BtcDet first identifies the regions of occlusion and signal miss  $\mathcal{R}_{OC} \cup \mathcal{R}_{SM}$ . In these regions, BtcDet estimates the shape occupancy probability  $\mathcal{P}(\mathcal{O}_S)$  (the orange voxels have  $\mathcal{P}(\mathcal{O}_S) > 0.3$ ). When the backbone network  $\Psi$  extracts detection features from the point cloud,  $\mathcal{P}(\mathcal{O}_S)$  is concatenated with  $\Psi$ 's intermediate feature maps. Then, a RPN network takes the output and generates 3D proposals. For each proposal (e.g., the green box), BtcDet pools the local geometric features  $f_{geo}$  to the nearby grids and finally generate the final bounding box prediction (the red box) and the confidence score.

2020; Xu et al. 2021) such as SPG conduct point cloud completion to improve object detection. These models are shape-aware but overlook the impact of occlusion.

**Occlusion handling in computer vision.** The negative impact of occlusion on various computer vision tasks, including tracking (Liu et al. 2018), image-based pedestrian detection (Zhang et al. 2018), image-based car detection (Reddy et al. 2019) and semantic part detection (Saleh et al. 2021), is acknowledged. Efforts addressing occlusion include the amodal instance segmentation (Follmann et al. 2019), the Multi-Level Coding that predicts the presence of occlusion (Qi et al. 2019b). These studies, although focus on 2D images, demonstrate the benefits of modeling occlusion to solving visual tasks. Point cloud visibility is addressed in (Hu et al. 2020) and is used in multi-frame detection and data augmentation. This method, however, does not learn and explore the visibility's influence on object shapes. Our proposed BtcDet is the first 3D object detector that learns occluded shapes in point cloud data. We compare (Hu et al. 2020)'s approach with ours in Sec. .

## Behind the Curtain Detector

Let  $\Theta$  denote the parameters of a detector,  $\{p_1, p_2, \dots, p_N\}$  denote the LiDAR point cloud,  $\mathcal{X}, \mathcal{D}, \mathcal{S}_{ob}, \mathcal{S}_{oc}$  denote the estimated box center, the box dimension, the observed objects shapes and the occluded object shapes, respectively. Most LiDAR-based 3D object detectors (Yi et al. 2020; Chen et al. 2020; Shi and Rajkumar 2020) only supervise the bounding box prediction. These models have

$$\Theta_{MLE} = \underset{\Theta}{\operatorname{argmax}} P(\mathcal{X}, \mathcal{D} | \{p_1, p_2, \dots, p_N\}, \Theta), \quad (1)$$

while structure-aware models (Shi et al. 2020; He et al. 2020; Du et al. 2020) also supervise  $\mathcal{S}_{ob}$ 's statistics so that

$$\Theta_{MLE} = \underset{\Theta}{\operatorname{argmax}} P(\mathcal{X}, \mathcal{D}, \mathcal{S}_{ob} | \{p_1, p_2, \dots, p_N\}, \Theta). \quad (2)$$

None of the above studies explicitly model the complete object shapes  $\mathcal{S} = \mathcal{S}_{ob} \cup \mathcal{S}_{oc}$ , while the experiments in Sec. show the improvements if  $\mathcal{S}$  is obtained. BtcDet estimates  $\mathcal{S}$

by predicting the shape occupancy  $\mathcal{O}_S$  for regions of interest. After that, BtcDet conducts object detection conditioned on the estimated probability of occupancy  $\mathcal{P}(\mathcal{O}_S)$ . The optimization objectives can be described as follows:

$$\underset{\Theta}{\operatorname{argmax}} P(\mathcal{O}_S | \{p_1, p_2, \dots, p_N\}, \mathcal{R}_{SM}, \mathcal{R}_{OC}, \Theta), \quad (3)$$

$$\underset{\Theta}{\operatorname{argmax}} P(\mathcal{X}, \mathcal{D} | \{p_1, p_2, \dots, p_N\}, \mathcal{P}(\mathcal{O}_S), \Theta). \quad (4)$$

**Model overview.** As illustrated in Figure 3, BtcDet first identifies the regions of occlusion  $\mathcal{R}_{OC}$  and signal miss  $\mathcal{R}_{SM}$ , and then, let a shape occupancy network  $\Omega$  estimate the probability of object shape occupancy  $\mathcal{P}(\mathcal{O}_S)$ . The training process is described in Sec. .

Next, BtcDet extracts the point cloud 3D features by a backbone network  $\Psi$ . The features are sent to a Region Proposal Network (RPN) to generate 3D proposals. To leverage the occupancy estimation, the sparse tensor  $\mathcal{P}(\mathcal{O}_S)$  is concatenated with the feature maps of  $\Psi$ . (See Sec. .)

Finally, BtcDet applies the proposal refinement. The local geometric features  $f_{geo}$  are composed of  $\mathcal{P}(\mathcal{O}_S)$  and the multi-scale features from  $\Psi$ . For each region proposal, we construct local grids covering the proposal box. BtcDet pools the local geometric features  $f_{geo}$  onto the local grids, aggregates the grid features, and generates the final bounding box predictions. (See Sec. .)

## Learning Shapes in Occlusion

**Approximate the complete object shapes for ground truth labels.** Occlusion and preclude the knowledge of the complete object shapes  $\mathcal{S}$ . However, we can assemble the approximated complete shapes  $\bar{\mathcal{S}}$ , based on two assumptions:

- Most foreground objects resemble a limited number of shape prototypes, e.g., pedestrians share a few body types.
- Foreground objects, especially vehicles and cyclists, are roughly symmetric.

We use the labeled bounding boxes to query points belonging to the objects. For cars and cyclists, we mirror the object

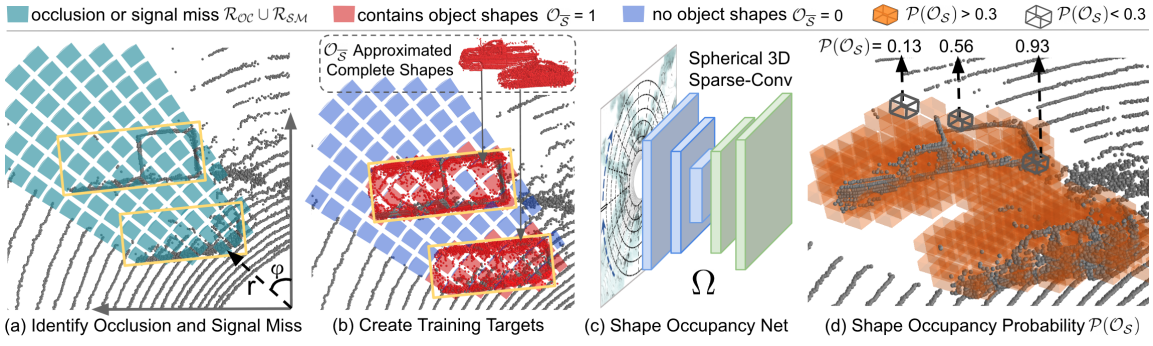


Figure 4: Learning Ocluded Shapes. (a) The regions of occlusion or signal miss  $\mathcal{R}_{OC} \cup \mathcal{R}_{SM}$  can be identified after the spherical voxelization for the point cloud. (b) To label the occupancy  $\mathcal{O}_{\bar{S}}$  (1 or 0), We place the approximated complete object shapes  $\bar{S}$  (red points) in the corresponding boxes. (c) A shape occupancy network  $\Omega$  predicts the shape occupancy probability  $\mathcal{P}(\mathcal{O}_S)$  for voxels in  $\mathcal{R}_{OC} \cup \mathcal{R}_{SM}$ , supervised by  $\mathcal{O}_{\bar{S}}$ . (d) Voxels are colored orange if it has a prediction  $\mathcal{P}(\mathcal{O}_S) > 0.3$ .

points against the middle section plane of the bounding box.

A heuristic  $\mathcal{H}(A, B)$  is created to evaluate if a source object  $B$  covers most parts of a target object  $A$  and provides points that can fill  $A$ 's shape miss. To approximate  $A$ 's complete shape, we select the top 3 source objects  $B_1, B_2, B_3$  with the best scores. The final approximation  $\bar{S}$  consists of  $A$ 's original points and the points of  $B_1, B_2, B_3$  that fill  $A$ 's shape miss. The target objects are object in the current training frame, while the source objects come from other frames of the training set. Please find details of  $\mathcal{H}(A, B)$  in Appendix B and visualization of assembling  $\bar{S}$  in Appendix G.

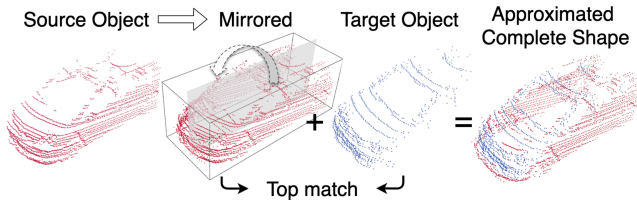


Figure 5: Assemble the approximated complete shape  $\bar{S}$  for an object (blue) by using points from top match objects.

### Identify $\mathcal{R}_{OC} \cup \mathcal{R}_{SM}$ in the spherical coordinate system.

According to our analysis in Sec. , ‘‘shape miss’’ only exists in the occluded regions  $\mathcal{R}_{OC}$  and the regions with signal miss  $\mathcal{R}_{SM}$  (see Figure 1(c) and (d)). Therefore, we need to identify  $\mathcal{R}_{OC} \cup \mathcal{R}_{SM}$  before learning to estimate shapes.

In real-world scenarios, there exists at most one point in the tetrahedron frustum of a range image pixel. When the laser is stopped at a point, the entire frustum behind the point is occluded. We propose to voxelize the point cloud using an evenly spaced spherical grid so that the occluded regions can be accurately formed by the spherical voxels behind any LiDAR point. As shown in Figure 4(a), each point  $(x, y, z)$  is transformed to the spherical coordinate system as  $(r, \phi, \theta)$ :

$$\begin{aligned} r &= \sqrt{x^2 + y^2 + z^2}, \quad \phi = \arctan2(y, x), \quad (5) \\ \theta &= \arctan2(z, \sqrt{x^2 + y^2}). \end{aligned}$$

$\mathcal{R}_{OC}$  includes nonempty spherical voxels and the empty voxels behind these voxels. In Figure 1(a), the dashed lines mark the potential areas of signal miss. In range view, we

can find pixels on the borders between the areas having LiDAR signals and the areas of no signal.  $\mathcal{R}_{SM}$  is formed by the spherical voxels that project to these pixels.

**Create training targets.** In  $\mathcal{R}_{OC} \cup \mathcal{R}_{SM}$ , we predict the probability  $\mathcal{P}(\mathcal{O}_S)$  for voxels if they contain points of  $\bar{S}$ . As illustrated in 4(b),  $\bar{S}$  are placed at the locations of the corresponding objects. We set  $\mathcal{O}_{\bar{S}} = 1$  for the spherical voxels that contain  $\bar{S}$ , and  $\mathcal{O}_{\bar{S}} = 0$  for the others.  $\mathcal{O}_{\bar{S}}$  is used as the ground truth label to approximate the occupancy  $\mathcal{O}_S$  of the complete object shape. Estimating occupancy has two advantages over generating points:

- $\bar{S}$  is assembled by multiple objects. The shape details approximated by the borrowed points are inaccurate and the point density of different objects is inconsistent. The occupancy  $\mathcal{O}_{\bar{S}}$  avoids these issues after rasterization.
- The plausibility issue of point generation can be avoided.

**Estimate the shape occupancy.** In  $\mathcal{R}_{OC} \cup \mathcal{R}_{SM}$ , we encode each nonempty spherical voxel with the average properties of the points inside  $(x, y, z, feats)$ , then, send them to a shape occupancy network  $\Omega$ . The network consists of two down-sampling sparse-conv layers and two up-sampling inverse-convolution layers. Each layer also includes several sub-manifold sparse-convs (Graham and van der Maaten 2017) (see Appendix D). The spherical sparse 3D convolutions are similar to the ones in the Cartesian coordinate, except that the voxels are indexed along  $(r, \phi, \theta)$ . The output  $\mathcal{P}(\mathcal{O}_S)$  is supervised by the sigmoid cross-entropy Focal Loss (Lin et al. 2017):

$$\mathcal{L}_{focal}(p_v) = -(1 - p_v)^\gamma \log(p_v), \quad (6)$$

$$\text{where } p_v = \begin{cases} \mathcal{P}(\mathcal{O}_S) & \text{if } \mathcal{O}_{\bar{S}} = 1 \text{ at voxel } v \\ 1 - \mathcal{P}(\mathcal{O}_S) & \text{otherwise,} \end{cases}$$

$$\mathcal{L}_{shape} = \frac{\sum_{v \in \mathcal{R}_{OC} \cup \mathcal{R}_{SM}} w_v \cdot \mathcal{L}_{focal}(p_v)}{|\mathcal{R}_{OC} \cup \mathcal{R}_{SM}|}, \quad (7)$$

$$\text{where } w_v = \begin{cases} \delta & \text{if } v \in \text{regions of shape miss} \\ 1 & \text{otherwise.} \end{cases}$$

Since  $\bar{S}$  borrows points from other objects in the shape miss regions, we assign them a weighting factor  $\delta$ , where  $\delta < 1$ .

## Shape Occupancy Probability Integration

Trained with the customized supervision,  $\Omega$  learns the shape priors of partially observed objects and generates  $\mathcal{P}(\mathcal{O}_S)$ . To benefit detection,  $\mathcal{P}(\mathcal{O}_S)$  is transformed from the spherical coordinate to the Cartesian coordinate and fused with  $\Psi$ , a sparse 3D convolutional network that extracts detection features in the Cartesian coordinate.

For example, a spherical voxel has a center  $(r, \phi, \theta)$  which is transformed as  $x = r\cos\theta\cos\phi$ ,  $y = r\cos\theta\sin\phi$ ,  $z = r\sin\theta$ . Assume  $x, y, z$  is inside a Cartesian voxel  $v^{i,j,k}$ . Since several spherical voxels can be mapped to  $v^{i,j,k}$ ,  $v^{i,j,k}$  takes the max value of these voxels  $SV(v^{i,j,k})$ :

$$\mathcal{P}(\mathcal{O}_S)_{v^{i,j,k}} = \max\{\mathcal{P}(\mathcal{O}_S)_{sv} : sv \in SV(v^{i,j,k})\}. \quad (8)$$

The occupancy probability of these Cartesian voxels forms a sparse tensor map  $\mathcal{P}(\mathcal{O}_S)_\perp = \{\mathcal{P}(\mathcal{O}_S)_v\}$ , which is, then, down-sampled by max-poolings into multiple scales and concatenated with  $\Psi$ 's intermediate feature maps:

$$f_{\Psi_i}^{in} = [f_{\Psi_{i-1}}^{out}, \maxpool_{\times 2}^{i-1}(\mathcal{P}(\mathcal{O}_S)_\perp)], \quad (9)$$

where  $f_{\Psi_i}^{in}$ ,  $f_{\Psi_{i-1}}^{out}$  and  $\maxpool_{\times 2}^{i-1}(\cdot)$  denote the input features of  $\Psi$ 's  $i$ th layer, the output features of  $\Psi$ 's  $i-1$ th layer, and applying stride-2 maxpooling  $i-1$  times, respectively.

The Region Proposal Network (RPN) takes the output features of  $\Psi$  and generates 3D proposals. Each proposal includes  $(x_p, y_p, z_p), (l_p, w_p, h_p), \theta_p, p_p$ , namely, center location, proposal box size, heading and proposal confidence.

## Occlusion-Aware Proposal Refinement

**Local geometry features.** BtcDet's refinement module further exploits the benefit of the shape occupancy. To obtain accurate final bounding boxes, BtcDet needs to look at the local geometries around the proposals. Therefore, we construct a local feature map  $f_{geo}$  by fusing multiple levels of  $\Psi$ 's features. In addition, we also fuse  $\mathcal{P}(\mathcal{O}_S)_\perp$  into  $f_{geo}$  to bring awareness to the shape miss in the local regions.  $\mathcal{P}(\mathcal{O}_S)_\perp$  provides two benefits for proposal refinement:

- $\mathcal{P}(\mathcal{O}_S)_\perp$  only has values in  $\mathcal{R}_{OC} \cup \mathcal{R}_{SM}$  so that it can help the box regression avoid the regions outside  $\mathcal{R}_{OC} \cup \mathcal{R}_{SM}$ , e.g., the regions with cross marks in Figure 3.
- The estimated occupancy indicates the existence of unobserved object shapes, especially for empty regions with high  $\mathcal{P}(\mathcal{O}_S)$ , e.g., some orange regions in Figure 3.

$f_{geo}$  is a sparse 3D tensor map with spatial resolution of  $400 \times 352 \times 5$ . The process for producing  $f_{geo}$  is described in Appendix D.

**RoI pooling.** On each proposal, we construct local grids which have the same heading of the proposal. To expand the receptive field, we set a size factor  $\mu$  so that:

$$w_{grid} = \mu \cdot w_p, \quad l_{grid} = \mu \cdot l_p, \quad h_{grid} = \mu \cdot h_p. \quad (10)$$

The grid has a dimension of  $12 \times 4 \times 2$ . We pool the nearby features  $f_{geo}$  onto the nearby grids through trilinear-interpolation (see Figure 3) and aggregates them by sparse 3D convolutions. After that, the refinement module predicts an IoU-related class confidence score and the residues between the 3D proposal boxes and the ground truth bounding boxes, following (Yan et al. 2018; Shi et al. 2020).

## Total Loss

The RPN loss  $\mathcal{L}_{rpn}$  and the proposal refinement loss  $\mathcal{L}_{pr}$  follow the most popular design among detectors (Shi et al. 2020; Yan et al. 2018). The total loss is:

$$\mathcal{L}_{total} = 0.3\mathcal{L}_{shape} + \mathcal{L}_{rpn} + \mathcal{L}_{pr}. \quad (11)$$

More details of the losses and the network architectures can be found in Appendix C and D.

## Experiments

In this section, we describe the implementation details of BtcDet and compare BtcDet with state-of-the-art detectors on two datasets: the KITTI Dataset (Geiger et al. 2013) and the Waymo Open Dataset (Sun et al. 2019). We also conduct ablation studies to demonstrate the effectiveness of the shape occupancy and the feature integration strategies. More detection results can be found in the Appendix F. The quantitative and qualitative evaluations of the occupancy estimation can be found in the Appendix E and H.

**Datasets.** The *KITTI Dataset* includes 7481 LiDAR frames for training and 7518 LiDAR frames for testing. We follow (Chen et al. 2017) to divide the training data into a *train* split of 3712 frames and a *val* split of 3769 frames. The *Waymo Open Dataset* (WOD) consists of 798 segments of 158361 LiDAR frames for training and 202 segments of 40077 LiDAR frames for validation. The KITTI Dataset only provides LiDAR point clouds in 3D, while the WOD also provides LiDAR range images.

**Implementation and training details.** BtcDet transforms the point locations  $(x, y, z)$  to  $(r, \phi, \theta)$  for the KITTI Dataset, while directly extracting  $(r, \phi, \theta)$  from the range images for the WOD. On the KITTI Dataset, we use a spherical voxel size of  $(0.32m, 0.52^\circ, 0.42^\circ)$  within the range  $[2.24m, 70.72m]$  for  $r$ ,  $[-40.69^\circ, 40.69^\circ]$  for  $\phi$  and  $[-16.60^\circ, 4.00^\circ]$  for  $\theta$ . On the WOD, we use a spherical voxel size of  $(0.32m, 0.81^\circ, 0.31^\circ)$  within the range  $[2.94m, 74.00m]$  for  $r$ ,  $[-180^\circ, 180^\circ]$  for  $\phi$  and  $[-33.80^\circ, 6.00^\circ]$  for  $\theta$ . Determined by grid search, we set  $\gamma = 2$  in Eq.6,  $\delta = 0.2$  in Eq.7 and  $\mu = 1.05$  in Eq.10.

In all of our experiments, we train our models with a batch size of 8 on 4 GTX 1080 Ti GPUs. On the KITTI Dataset, we train BtcDet for 40 epochs, while on the WOD, we train BtcDet for 30 epochs. The BtcDet is end-to-end optimized by the ADAM optimizer (Kingma and Ba 2014) from scratch. We applies the widely adopted data augmentations (Shi et al. 2020; Deng et al. 2020; Lang et al. 2019; Yang et al. 2020; Ye et al. 2020), which includes flipping, scaling, rotation and the ground-truth augmentation.

## Evaluation on the KITTI Dataset

We evaluate BtcDet on the KITTI *val* split after training it on the *train* split. To evaluate the model on the KITTI test set, we train BtcDet on 80% of all *train+val* data and hold out the remaining 20% data for validation. Following the protocol in (Geiger et al. 2013), results are evaluated by the Average Precision (AP) with an IoU threshold of 0.7 for cars and 0.5 for pedestrians and cyclists.

Method	Car 3D $AP_{R40}$			Ped. 3D $AP_{R40}$			Cyc. 3D $AP_{R40}$			3D $AP_{R11}$
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Car Mod.
PointPillars (Lang et al. 2019)	87.75	78.39	75.18	57.30	51.41	46.87	81.57	62.94	58.98	77.28
SECOND (Yan et al. 2018)	90.97	79.94	77.09	58.01	51.88	47.05	78.50	56.74	52.83	76.48
SA-SSD (He et al. 2020)	92.23	84.30	81.36	-	-	-	-	-	-	79.91
PV-RCNN (Shi et al. 2020)	92.57	84.83	82.69	64.26	56.67	51.91	88.88	71.95	66.78	83.90
Voxel R-CNN (Deng et al. 2020)	92.38	85.29	82.86	-	-	-	-	-	-	84.52
BtcDet (Ours)	<b>93.15</b>	<b>86.28</b>	<b>83.86</b>	<b>69.39</b>	<b>61.19</b>	<b>55.86</b>	<b>91.45</b>	<b>74.70</b>	<b>70.08</b>	<b>86.57</b>

Table 1: Comparison on the KITTI *val* set, evaluated by the 3D Average Precision (AP) under 40 recall thresholds (R40). The 3D APs on under 11 recall thresholds are also reported for the moderate car objects.

Method	Reference	Modality	Car 3D $AP_{R40}$				Cyc. 3D $AP_{R40}$			
			Easy	Mod.	Hard	mAP	Easy	Mod.	Hard	mAP
EPNet (Huang et al. 2020)	ECCV 2020	LiDAR+RGB	89.81	79.28	74.59	81.23	-	-	-	-
3D-CVF (Yoo et al. 2020)	ECCV 2020	LiDAR+RGB	89.20	80.05	73.11	80.79	-	-	-	-
PointPillars (Lang et al. 2019)	CVPR 2019	LiDAR	82.58	74.31	68.99	75.29	77.10	58.65	51.92	62.56
STD (Yang et al. 2019)	ICCV 2019	LiDAR	87.95	79.71	75.09	80.92	78.69	61.59	55.30	65.19
HotSpotNet (Chen et al. 2020)	ECCV 2020	LiDAR	87.60	78.31	73.34	79.75	82.59	65.95	59.00	69.18
PartA <sup>2</sup> (Shi et al. 2020)	TPAMI 2020	LiDAR	87.81	78.49	73.51	79.94	79.17	63.52	56.93	66.54
3DSSD (Yang et al. 2020)	CVPR 2020	LiDAR	88.36	79.57	74.55	80.83	82.48	64.10	56.90	67.83
SA-SSD (He et al. 2020)	CVPR 2020	LiDAR	88.75	79.79	74.16	80.90	-	-	-	-
Asso-3Ddet (Du et al. 2020)	CVPR 2020	LiDAR	85.99	77.40	70.53	77.97	-	-	-	-
PV-RCNN (Shi et al. 2020)	CVPR 2020	LiDAR	90.25	81.43	76.82	82.83	78.60	63.71	57.65	66.65
Voxel R-CNN (Deng et al. 2020)	AAAI 2021	LiDAR	<b>90.90</b>	81.62	77.06	83.19	-	-	-	-
CIA-SSD (Zheng et al. 2021)	AAAI 2021	LiDAR	89.59	80.28	72.87	80.91	-	-	-	-
TANet (Liu et al. 2020)	AAAI 2021	LiDAR	83.81	75.38	67.66	75.62	73.84	59.86	53.46	62.39
BtcDet (Ours)	-	LiDAR	90.64	<b>82.86</b>	<b>78.09</b>	<b>83.86</b>	<b>82.81</b>	<b>68.68</b>	<b>61.81</b>	<b>71.10</b>
<i>Improvement</i>	-	-	-0.26	+1.24	+0.94	+0.67	+0.33	+2.73	+2.81	+1.92

Table 2: Comparison on the KITTI *test* set, evaluated by the 3D Average Precision (AP) of 40 recall thresholds (R40) on the KITTI server. BtcDet surpasses all the leader board front runners that are associated with publications released before our submission. The mAPs are averaged over the APs of easy, moderate, and hard objects. Please find more results in Appendix F.

**KITTI validation set.** As summarized in Table 1, we compare BtcDet with the state-of-the-art LiDAR-based 3D object detectors on cars, pedestrians and cyclists using the AP under 40 recall thresholds (R40). We reference the R40 APs of SA-SSD, PV-RCNN and Voxel R-CNN to their papers, the R40 APs of SECOND to (Pang et al. 2020) and the R40 APs of PointRCNN and PointPillars to the results of the officially released code. We also report the published 3D APs under 11 recall thresholds (R11) for the moderate car objects. On all object classes and difficulty levels, BtcDet outperforms models that only supervise bounding boxes (Eq.1) as well as structure-aware models (Eq.2). Specifically, BtcDet outperforms other models by 2.05% 3D R11 AP on the moderate car objects, which makes it the first detector that reaches above 86% on this primary metric.

**KITTI test set.** As shown in Table 2, we compare BtcDet with the front runners on the KITTI test leader board. Besides the official metrics, we also report the mAPs that average over the APs of easy, moderate, and hard objects. As of May. 4th, 2021, compared with all the models associated with publications, BtcDet **surpasses** them on car and cyclist detection by big margins. Those methods include the models that take inputs of both LiDAR and RGB images and the ones taking LiDAR input only. We also list more comparisons and the results in Appendix F.

## Evaluation on the Waymo Open Dataset

We also compare BtcDet with other models on the Waymo Open Dataset (WOD). We report both 3D mean Average Precision (mAP) and 3D mAP weighted by Heading (mAPH) for vehicle detection. The official metrics also include separate mAPs for objects belonging to different distance ranges. Two difficulty levels are also introduced, where the LEVEL\_1 mAP calculates for objects that have more than 5 points and the LEVEL\_2 mAP calculates for objects that have more than 1 point.

As shown in Table 3, BtcDet outperforms these state-of-the-art detectors on all distance ranges and all difficulty levels by big margins. BtcDet outperforms other detectors on the LEVEL\_1 3D mAP by 2.99% and the LEVEL\_2 3D mAP by 3.51%. In general, BtcDet brings more improvement on the LEVEL\_2 objects, since objects with fewer points usually suffer more from occlusion and signal miss. These strong results on WOD, one of the largest published LiDAR datasets, manifest BtcDet’s ability to generalize.

## Ablation Studies

We conduct ablation studies to demonstrate the effectiveness of the shape occupancy and the feature integration strategies. All model variants are trained on the KITTI *train* split and evaluated on the *val* split.

Method	LEVEL_1 3D mAP				mAPH	LEVEL_2 3D mAP				mAPH
	Overall	0-30m	30-50m	50m-Inf		Overall	0-30m	30-50m	50m-Inf	
PointPillar (Lang et al. 2019)	56.62	81.01	51.75	27.94	-	-	-	-	-	-
MVF (Zhou et al. 2020)	62.93	86.30	60.02	36.02	-	-	-	-	-	-
SECOND (Yan et al. 2018)	72.27	-	-	-	71.69	63.85	-	-	-	63.33
Pillar-OD (Wang et al. 2020)	69.80	88.53	66.50	42.93	-	-	-	-	-	-
AFDet (Ge et al. 2020)	63.69	87.38	62.19	29.27	-	-	-	-	-	-
PV-RCNN (Shi et al. 2020)	70.30	91.92	69.21	42.17	69.69	65.36	91.58	65.13	36.46	64.79
Voxel R-CNN (Deng et al. 2020)	75.59	92.49	74.09	53.15	-	66.59	91.74	67.89	40.80	-
BtcDet (ours)	<b>78.58</b>	<b>96.11</b>	<b>77.64</b>	<b>54.45</b>	<b>78.06</b>	<b>70.10</b>	<b>95.99</b>	<b>70.56</b>	<b>43.87</b>	<b>69.61</b>

Table 3: Comparison for vehicle detection on the Waymo Open Dataset validation set.

Model Variant	Learned Features	Integrated Features	3D $AP_{R11}$ Car Mod.
BtcDet <sub>1</sub> (base)	-	-	83.71
BtcDet <sub>2</sub>	-	$\mathcal{R}_{OC} \cup \mathcal{R}_{SM}$	84.01
BtcDet <sub>3</sub>	$\mathcal{P}(\mathcal{O}_S)_\perp$	$\mathcal{P}(\mathcal{O}_S)_\perp$	86.03
BtcDet <sub>4</sub>	$\mathcal{P}(\mathcal{O}_S)_\odot$	$\mathbf{1}(\mathcal{P}(\mathcal{O}_S)_\perp \geq 0.5)$	85.59
<b>BtcDet (main)</b>	$\mathcal{P}(\mathcal{O}_S)_\odot$	$\mathcal{P}(\mathcal{O}_S)_\perp$	<b>86.57</b>

Table 4: Ablation studies on the learned features (Sec. ) and the features fused into  $\Psi$  and  $f_{geo}$  (Sec. ). BtcDet<sub>2</sub> directly use a binary map that labels  $\mathcal{R}_{OC} \cup \mathcal{R}_{SM}$ .  $\odot$  and  $\perp$  indicate the spherical and the Cartesian coordinate. The “ $\mathbf{1}$ ” operator converts float values to binary codes with a threshold of 0.5. All variants share the same architecture.

**Shape Features.** As shown in Table 4, we conduct ablation studies by controlling the shape features learned by  $\Omega$  and the features used in the integration. All the model variants share the same architecture and integration strategies.

Similarly to (Hu et al. 2020), BtcDet<sub>2</sub> directly fuses the binary map of.  $\mathcal{R}_{OC} \cup \mathcal{R}_{SM}$  into the detection pipeline. Although the binary map provides the information of occlusion, the improvement is limited since that the regions with code 1 are mostly background regions and less informative.

BtcDet<sub>3</sub> learns  $\mathcal{P}(\mathcal{O}_S)_\perp$  directly. The network  $\Omega$  predicts probability for Cartesian voxels. One Cartesian voxel will cover multiple spherical voxels when being close to the sensor, and will cover a small portion of a spherical voxel when being located at a remote distance. Therefore, the occlusion regions are misrepresented in the Cartesian coordinate.

BtcDet<sub>4</sub> convert the probability to hard occupancy, which cannot inform the downstream branch if a region is less likely or more likely to contain object shapes.

These experiments demonstrate the effectiveness of our choices for shape features, which help the main model improve 2.86 AP over the baseline BtcDet<sub>1</sub>.

**Integration strategies.** We conduct ablation studies by choosing different layers of  $\Psi$  to concatenate with  $\mathcal{P}(\mathcal{O}_S)_\perp$  and whether to use  $\mathcal{P}(\mathcal{O}_S)_\perp$  to form  $f_{geo}$ . The former mostly affects the proposal generation, while the latter affects proposal refinement.

In Table 5, the experiment on BtcDet<sub>5</sub> shows that we can improve the final prediction AP by 0.8 if we only integrate  $\mathcal{P}(\mathcal{O}_S)_\perp$  for proposal refinement. On the other hand, the experiment on BtcDet<sub>6</sub> shows the integration with  $\Psi$  alone can improve the AP by 1.2 for proposal box and final bounding

Model Variant	Integrate Layers of $\Psi$	Integrate $f_{geo}$	Proposal bbox 3D $AP_{R11}$	Final 3D $AP_{R11}$
BtcDet <sub>1</sub> (base)	-	-	77.75	83.71
BtcDet <sub>5</sub>	-	$\checkmark$	77.73	84.50
BtcDet <sub>6</sub>	1,2	-	<b>78.97</b>	85.72
BtcDet <sub>7</sub>	1	$\checkmark$	78.54	85.73
BtcDet <sub>8</sub>	1,2,3	$\checkmark$	78.76	86.11
<b>BtcDet (main)</b>	1,2	$\checkmark$	78.93	<b>86.57</b>

Table 5: Ablation studies on which layers of  $\Psi$  are fused with  $\mathcal{P}(\mathcal{O}_S)_\perp$  (Eq. 9) and whether to fuse  $\mathcal{P}(\mathcal{O}_S)_\perp$  into  $f_{geo}$ . We evaluate on the KITTI’s moderate car objects and show the 3D  $AP_{R11}$  of the proposal and final bounding box.

box prediction AP by 2.0 over the baseline.

The comparisons of BtcDet<sub>7</sub>, BtcDet<sub>8</sub> and BtcDet (main) demonstrates integrating  $\mathcal{P}(\mathcal{O}_S)_\perp$  with  $\Psi$ ’s first two layers is the best choice. Since  $\mathcal{P}(\mathcal{O}_S)$  is a low level feature while the third layer of  $\Psi$  would contain high level features, we observe a regression when BtcDet<sub>8</sub> also concatenates  $\mathcal{P}(\mathcal{O}_S)_\perp$  with  $\Psi$ ’s third layer.

These experiments demonstrate both the integration with  $\Psi$  and the integration to form  $f_{geo}$  can bring improvement independently. When working together, two integrations finally help BtcDet surpass all the state-of-the-art models.

## Conclusion and Future Work

In this paper, we analyze shape miss on 3D object detection, which is attributed to occlusion and signal miss in point cloud data. To solve this problem, we propose Behind the Curtain Detector (BtcDet), the first 3D object detector that targets this fundamental challenge. A training method is designed to learn the underlying shape priors. BtcDet can faithfully estimate the complete object shape occupancy for regions affected by occlusion and signal miss. After the integration with the probability estimation, both the proposal generation and refinement are significantly improved. In the experiments on the KITTI Dataset and the Waymo Open Dataset, BtcDet surpasses all the published state-of-the-art methods by remarkable margins. Ablation studies further manifest the effectiveness of the shape features and the integration strategies. Although our work successfully demonstrates the benefits of learning occluded shapes, there is still room to improve the model efficiency. Designing models that expedite occlusion identification and shape learning can be a promising future direction.

## References

- Chen, Q.; Sun, L.; Wang, Z.; Jia, K.; and Yuille, A. 2020. Object as hotspots: An anchor-free 3d object detection approach via firing of hotspots. In *European Conference on Computer Vision*, 68–84. Springer.
- Chen, X.; Ma, H.; Wan, J.; Li, B.; and Xia, T. 2017. Multi-view 3D Object Detection Network for Autonomous Driving. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6526–6534.
- Deng, J.; Shi, S.; Li, P.; Zhou, W.; Zhang, Y.; and Li, H. 2020. Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection. *arXiv:2012.15712*.
- Du, L.; Ye, X.; Tan, X.; Feng, J.; Xu, Z.; Ding, E.; and Wen, S. 2020. Associate-3Ddet: Perceptual-to-Conceptual Association for 3D Point Cloud Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13329–13338.
- Follmann, P.; König, R.; Härtinger, P.; Klostermann, M.; and Böttger, T. 2019. Learning to see the invisible: End-to-end trainable amodal instance segmentation. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1328–1336. IEEE.
- Ge, R.; Ding, Z.; Hu, Y.; Wang, Y.; Chen, S.; Huang, L.; and Li, Y. 2020. Afdet: Anchor free one stage 3d object detection. *arXiv preprint arXiv:2006.12671*.
- Geiger, A.; Lenz, P.; Stiller, C.; and Urtasun, R. 2013. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11): 1231–1237.
- Graham, B.; and van der Maaten, L. 2017. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*.
- He, C.; Zeng, H.; Huang, J.; Hua, X.-S.; and Zhang, L. 2020. Structure Aware Single-stage 3D Object Detection from Point Cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
- Hu, P.; Ziglar, J.; Held, D.; and Ramanan, D. 2020. What You See is What You Get: Exploiting Visibility for 3D Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Huang, T.; Liu, Z.; Chen, X.; and Bai, X. 2020. Epnet: Enhancing point features with image semantics for 3d object detection. In *European Conference on Computer Vision*, 35–52. Springer.
- Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Lang, A. H.; Vora, S.; Caesar, H.; Zhou, L.; Yang, J.; and Beijbom, O. 2019. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 12697–12705.
- Li, Z.; Yao, Y.; Quan, Z.; Yang, W.; and Xie, J. 2021. SIENet: Spatial Information Enhancement Network for 3D Object Detection from Point Cloud. *arXiv preprint arXiv:2103.15396*.
- Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2980–2988.
- Liu, Y.; Jing, X.-Y.; Nie, J.; Gao, H.; Liu, J.; and Jiang, G.-P. 2018. Context-aware three-dimensional mean-shift with occlusion handling for robust object tracking in RGB-D videos. *IEEE Transactions on Multimedia*, 21(3): 664–677.
- Liu, Z.; Zhao, X.; Huang, T.; Hu, R.; Zhou, Y.; and Bai, X. 2020. Tanet: Robust 3d object detection from point clouds with triple attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 11677–11684.
- Najibi, M.; Lai, G.; Kundu, A.; Lu, Z.; Rathod, V.; Funkhouser, T.; Pantofaru, C.; Ross, D.; Davis, L. S.; and Fathi, A. 2020. Dops: Learning to detect 3d objects and predict their 3d shapes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 11913–11922.
- Pan, Y.; Xiao, P.; He, Y.; Shao, Z.; and Li, Z. 2021. MULLS: Versatile LiDAR SLAM via Multi-metric Linear Least Square. *arXiv preprint arXiv:2102.03771*.
- Pang, S.; Morris, D.; and Radha, H. 2020. CLOCs: Camera-LiDAR Object Candidates Fusion for 3D Object Detection. *arXiv preprint arXiv:2009.00784*.
- Qi, C. R.; Litany, O.; He, K.; and Guibas, L. J. 2019a. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9277–9286.
- Qi, L.; Jiang, L.; Liu, S.; Shen, X.; and Jia, J. 2019b. Amodal instance segmentation with kins dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3014–3023.
- Reddy, N. D.; Vo, M.; and Narasimhan, S. G. 2019. Occlusion-net: 2d/3d occluded keypoint localization using graph networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7326–7335.
- Saleh, K.; Szénási, S.; and Vámosy, Z. 2021. Occlusion Handling in Generic Object Detection: A Review. In *2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 000477–000484. IEEE.
- Shi, S.; Guo, C.; Jiang, L.; Wang, Z.; Shi, J.; Wang, X.; and Li, H. 2020. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10529–10538.
- Shi, S.; Wang, X.; and Li, H. 2019. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–779.
- Shi, S.; Wang, Z.; Shi, J.; Wang, X.; and Li, H. 2020. From Points to Parts: 3D Object Detection from Point Cloud with Part-aware and Part-aggregation Network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1.



- Shi, W.; and Rajkumar, R. 2020. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1711–1719.
- Sun, P.; Kretzschmar, H.; Dotiwalla, X.; Chouard, A.; Patnaik, V.; Tsui, P.; Guo, J.; Zhou, Y.; Chai, Y.; Caine, B.; Vasudevan, V.; Han, W.; Ngiam, J.; Zhao, H.; Timofeev, A.; Ettinger, S.; Krivokon, M.; Gao, A.; Joshi, A.; Zhang, Y.; Shlens, J.; Chen, Z.; and Anguelov, D. 2019. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. *arXiv:1912.04838*.
- Wang, Y.; Fathi, A.; Kundu, A.; Ross, D.; Pantofaru, C.; Funkhouser, T.; and Solomon, J. 2020. Pillar-based object detection for autonomous driving. *arXiv preprint arXiv:2007.10323*.
- Xu, Q.; Sun, X.; Wu, C.-Y.; Wang, P.; and Neumann, U. 2020. Grid-gcn for fast and scalable point cloud learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5661–5670.
- Xu, Q.; Zhou, Y.; Wang, W.; Qi, C. R.; and Anguelov, D. 2021. Spg: Unsupervised domain adaptation for 3d object detection via semantic point generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 15446–15456.
- Yan, X.; Gao, J.; Li, J.; Zhang, R.; Li, Z.; Huang, R.; and Cui, S. 2020. Sparse single sweep lidar point cloud segmentation via learning contextual shape priors from scene completion. *arXiv preprint arXiv:2012.03762*.
- Yan, Y.; Mao, Y.; and Li, B. 2018. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10): 3337.
- Yang, Z.; Sun, Y.; Liu, S.; and Jia, J. 2020. 3dssd: Point-based 3d single stage object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11040–11048.
- Yang, Z.; Sun, Y.; Liu, S.; Shen, X.; and Jia, J. 2019. Std: Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE International Conference on Computer Vision*, 1951–1960.
- Ye, M.; Xu, S.; and Cao, T. 2020. Hvnet: Hybrid voxel network for lidar based 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 1631–1640.
- Yi, H.; Shi, S.; Ding, M.; Sun, J.; Xu, K.; Zhou, H.; Wang, Z.; Li, S.; and Wang, G. 2020. Segvoxelnet: Exploring semantic context and depth-aware features for 3d vehicle detection from point cloud. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2274–2280. IEEE.
- Yoo, J. H.; Kim, Y.; Kim, J. S.; and Choi, J. W. 2020. 3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection. *arXiv preprint arXiv:2004.12636*, 3.
- Zhang, S.; Wen, L.; Bian, X.; Lei, Z.; and Li, S. Z. 2018. Occlusion-aware R-CNN: detecting pedestrians in a crowd. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 637–653.
- Zheng, W.; Tang, W.; Chen, S.; Jiang, L.; and Fu, C.-W. 2021. CIA-SSD: Confident IoU-Aware Single-Stage Object Detector From Point Cloud. In *AAAI*.
- Zhou, Y.; Sun, P.; Zhang, Y.; Anguelov, D.; Gao, J.; Ouyang, T.; Guo, J.; Ngiam, J.; and Vasudevan, V. 2020. End-to-end multi-view fusion for 3d object detection in lidar point clouds. In *Conference on Robot Learning*, 923–932.
- Zhou, Y.; and Tuzel, O. 2018. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4490–4499.