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

Qiangeng Xu, Yiqi Zhong, Ulrich Neumann

University of Southern California
qiangengx@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.

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.

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. PartA² (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.
BtcDet estimates the shape occupancy probability $P(O_S)$ (the orangex voxels have $P(O_S) > 0.3$). When the backbone \( \Psi \) extracts detection features from the point cloud, \( P(O_S) \) 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 generates the final bounding box prediction (the red box) and the confidence score.

\[ \Theta_{MLE} = \arg\max_{\Theta} P(X, D | \{p_1, p_2, ..., p_N\}, \Theta), \] 
\[ \Theta_{MLE} = \arg\max_{\Theta} P(X, D, S_{ob}, S_{ac} | \{p_1, p_2, ..., p_N\}, \Theta). \]

\[ P(O_{OC}) \cup P(O_{SM}) \]

**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), 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, \( \{p_1, p_2, ..., p_N\} \) denote the LiDAR point cloud, \( X, D, S_{ob}, S_{ac} \) denote the estimated box center, the box dimension, the observed object 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

\[ \Theta_{MLE} = \arg\max_{\Theta} P(X, D | \{p_1, p_2, ..., p_N\}, \Theta), \] 

while structure-aware models (Shi et al. 2020; He et al. 2020; Du et al. 2020) also supervise \( S_{ob} \)'s statistic so that

\[ \Theta_{MLE} = \arg\max_{\Theta} P(X, D, S_{ob} | \{p_1, p_2, ..., p_N\}, \Theta). \]

None of the above studies explicitly model the complete object shapes \( S = S_{ob} \cup S_{ac} \), while the experiments in Sec. show the improvements if \( S \) is obtained. BtcDet estimates \( S \) by predicting the shape occupancy \( O_S \) for regions of interest. After that, BtcDet conducts object detection conditioned on the estimated probability of occupancy \( P(O_S) \). The optimization objectives can be described as follows:

\[ \arg\max_{\Theta} P(O_S | \{p_1, p_2, ..., p_N\}, R_{SM}, R_{OC}, \Theta), \] 
\[ \arg\max_{\Theta} P(X, D | \{p_1, p_2, ..., p_N\}, P(O_S), \Theta). \]

**Model overview.** As illustrated in Figure 3, BtcDet first identifies the regions of occlusion \( R_{OC} \) and signal miss \( R_{SM} \), and then, let a shape occupancy network \( \hat{O} \) estimate the probability of object shape occupancy \( P(O_S) \). 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 \( P(O_S) \) 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 \( P(O_S) \) 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 \( S \). However, we can assemble the approximated complete shapes \( \hat{S} \), 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
points against the middle section plane of the bounding box.

A heuristic $H(A, B)$ 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 $H(A, B)$ in Appendix B and visualization of assembling $\bar{S}$ in Appendix G.

**Identify $R_{OC} \cup R_{SM}$ in the spherical coordinate system.**

According to our analysis in Sec. 4, “shape miss” only exists in the occluded regions $R_{OC}$ and the regions with signal miss $R_{SM}$ (see Figure 1(c) and (d)). Therefore, we need to identify $R_{OC} \cup R_{SM}$ 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 $(r, \phi, \theta)$:

\[
\begin{align*}
    r &= \sqrt{x^2 + y^2 + z^2}, \quad \phi = \arctan2(y, x), \\
    \theta &= \arctan2(z, \sqrt{x^2 + y^2}).
\end{align*}
\]

$R_{OC}$ 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. $R_{SM}$ is formed by the spherical voxels that project to these pixels.

**Create training targets.** In $R_{OC} \cup R_{SM}$, we predict the probability $P(O_S)$ 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 $O_S = 1$ for the spherical voxels that contain $\bar{S}$, and $O_S = 0$ for the others. $O_S$ is used as the ground truth label to approximate the occupancy $O_S$ 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 $O_S$ avoids these issues after rasterization.
- The plausibility issue of point generation can be avoided.

**Estimate the shape occupancy.** In $R_{OC} \cup R_{SM}$, we encode each nonempty spherical voxel with the average properties of the points inside $(x, y, z, \text{feats})$, 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-conv (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 $P(O_S)$ is supervised by the sigmoid cross-entropy Focal Loss (Lin et al. 2017):
Shape Occupancy Probability Integration

Trained with the customized supervision, \( \Omega \) learns the shape priors of partially observed objects and generates \( \mathcal{P}(O_S) \).

To benefit detection, \( \mathcal{P}(O_S) \) 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 \( v_{i,j,k} \). Several spherical voxels can be mapped to \( v_{i,j,k} \) by \( \cos \theta \) takes the max value of these voxels \( SV(v_{i,j,k}) \):

\[
\mathcal{P}(O_S)_{v_{i,j,k}} = \max\{\mathcal{P}(O_S)_x : x \in SV(v_{i,j,k})\} \tag{8}
\]

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

\[
f_{\Psi_i}^{\text{in}} = \left[f_{\Psi_{i-1}}^{\text{out}}, \text{maxpool}^{-1}_i(\mathcal{P}(O_S)_{\perp})\right], \tag{9}
\]

where \( f_{\Psi_i}^{\text{in}}, f_{\Psi_{i-1}}^{\text{out}} \), and \( \text{maxpool}^{-1}_i(\cdot) \) 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 \((x_p, y_p, z_p), (l_p, w_p, h_p), \theta_p, \rho_p\), 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 \( \mathcal{P}(O_S)_{\perp} \) into \( f_{geo} \) to bring awareness to the shape miss in the local regions. \( \mathcal{P}(O_S)_{\perp} \) provides two benefits for proposal refinement:

- \( \mathcal{P}(O_S)_{\perp} \) 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 \( \mathcal{P}(O_S) \), 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:

\[
w_{\text{grid}} = \mu \cdot w_p, \quad l_{\text{grid}} = \mu \cdot l_p, \quad h_{\text{grid}} = \mu \cdot h_p. \tag{10}
\]

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:

\[
\mathcal{L}_{\text{total}} = 0.3 \mathcal{L}_{\text{shape}} + \mathcal{L}_{rpn} + \mathcal{L}_{pr}. \tag{11}
\]

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.
### 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Car 3D AP&lt;sub&gt;R40&lt;/sub&gt;</th>
<th>Ped. 3D AP&lt;sub&gt;R40&lt;/sub&gt;</th>
<th>Cyc. 3D AP&lt;sub&gt;R40&lt;/sub&gt;</th>
<th>Car Mod.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Easy</td>
<td>Mod.</td>
<td>Hard</td>
<td>Easy</td>
</tr>
<tr>
<td>PointPillars (Lang et al. 2019)</td>
<td>87.75</td>
<td>78.39</td>
<td>75.18</td>
<td>57.30</td>
</tr>
<tr>
<td>SECOND (Yan et al. 2018)</td>
<td>90.97</td>
<td>79.94</td>
<td>77.09</td>
<td>58.01</td>
</tr>
<tr>
<td>SA-SSD (He et al. 2020)</td>
<td>92.23</td>
<td>84.30</td>
<td>81.36</td>
<td>-</td>
</tr>
<tr>
<td>PV-RCNN (Shi et al. 2020)</td>
<td>92.57</td>
<td>84.83</td>
<td>82.69</td>
<td>64.26</td>
</tr>
<tr>
<td>Voxel R-CNN (Deng et al. 2020)</td>
<td>92.38</td>
<td>85.29</td>
<td>82.86</td>
<td>-</td>
</tr>
<tr>
<td>BtcDet (Ours)</td>
<td><strong>93.15</strong></td>
<td><strong>86.28</strong></td>
<td><strong>83.86</strong></td>
<td><strong>69.39</strong></td>
</tr>
</tbody>
</table>

### 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Reference</th>
<th>Modality</th>
<th>Car 3D AP&lt;sub&gt;R40&lt;/sub&gt;</th>
<th>Cyc. 3D AP&lt;sub&gt;R40&lt;/sub&gt;</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Easy</td>
<td>Mod.</td>
<td>Hard</td>
<td>Easy</td>
<td>Mod.</td>
</tr>
<tr>
<td>EPNet (Huang et al. 2020)</td>
<td>89.81</td>
<td>79.28</td>
<td>74.59</td>
<td>81.23</td>
<td>-</td>
</tr>
<tr>
<td>3D-CVF (Yoo et al. 2020)</td>
<td>89.20</td>
<td>80.05</td>
<td>73.11</td>
<td>80.79</td>
<td>-</td>
</tr>
<tr>
<td>PointPillars (Lang et al. 2019)</td>
<td>88.52</td>
<td>78.32</td>
<td>74.84</td>
<td>81.02</td>
<td>68.59</td>
</tr>
<tr>
<td>STD (Yang et al. 2019)</td>
<td>87.60</td>
<td>78.31</td>
<td>73.34</td>
<td>79.75</td>
<td>71.97</td>
</tr>
<tr>
<td>HotSpotNet (Chen et al. 2020)</td>
<td>87.81</td>
<td>78.49</td>
<td>73.51</td>
<td>79.94</td>
<td>71.97</td>
</tr>
<tr>
<td>Part2 (Shi et al. 2020)</td>
<td>88.36</td>
<td>79.57</td>
<td>74.55</td>
<td>80.83</td>
<td>68.59</td>
</tr>
<tr>
<td>3DSSD (Yang et al. 2020)</td>
<td>88.75</td>
<td>79.79</td>
<td>74.16</td>
<td>80.90</td>
<td>68.59</td>
</tr>
<tr>
<td>SA-SSD (He et al. 2020)</td>
<td>85.99</td>
<td>77.40</td>
<td>70.53</td>
<td>77.97</td>
<td>68.59</td>
</tr>
<tr>
<td>Asso-3Ddet (Du et al. 2020)</td>
<td>85.99</td>
<td>77.40</td>
<td>70.53</td>
<td>77.97</td>
<td>68.59</td>
</tr>
<tr>
<td>PV-RCNN (Shi et al. 2020)</td>
<td>90.25</td>
<td>81.43</td>
<td>76.82</td>
<td>82.83</td>
<td>68.59</td>
</tr>
<tr>
<td>Voxel R-CNN (Deng et al. 2020)</td>
<td>90.90</td>
<td>81.62</td>
<td>77.06</td>
<td>83.19</td>
<td>68.59</td>
</tr>
<tr>
<td>CIA-SSD (Zheng et al. 2021)</td>
<td>89.59</td>
<td>80.28</td>
<td>72.87</td>
<td>80.91</td>
<td>-</td>
</tr>
<tr>
<td>TANet (Liu et al. 2020)</td>
<td>83.81</td>
<td>75.38</td>
<td>67.66</td>
<td>75.62</td>
<td>67.66</td>
</tr>
<tr>
<td>BtcDet (Ours)</td>
<td><strong>90.64</strong></td>
<td><strong>82.86</strong></td>
<td><strong>78.09</strong></td>
<td><strong>83.86</strong></td>
<td><strong>82.81</strong></td>
</tr>
<tr>
<td>Improvement</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td><strong>-0.26</strong></td>
<td><strong>+1.24</strong></td>
</tr>
</tbody>
</table>

**KITTIP 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.

**KITTITest 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 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.

**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 2.90% 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.
### Table 4: Ablation studies on the learned features (Sec.) and the features fused into Ψ and \( f_{geo} \) (Sec.). BtcDet\(_2\) directly uses a binary map that labels \( R_{OC} \bigcup R_{SM} \) and \( \bot \) to indicate the spherical and the Cartesian coordinate. The \( \bot \) operator converts float values to binary codes with a threshold of 0.5. All variants share the same architecture.

<table>
<thead>
<tr>
<th>Model</th>
<th>Learned Features</th>
<th>Integrated Features</th>
<th>3D AP(<em>{R</em>{11}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>BtcDet(_1) (base)</td>
<td>–</td>
<td>–</td>
<td>83.71</td>
</tr>
<tr>
<td>BtcDet(_2)</td>
<td>( P(O_\Sigma) ) ( \perp ) ( R_{OC} \bigcup R_{SM} )</td>
<td>( P(O_\Sigma) ) ( \perp ) ( R_{OC} \bigcup R_{SM} )</td>
<td>84.01</td>
</tr>
<tr>
<td>BtcDet(_3)</td>
<td>( P(O_\Sigma) ) ( \perp ) ( 1(P(O_\Sigma) \geq 0.5) )</td>
<td>( P(O_\Sigma) ) ( \perp ) ( 1(P(O_\Sigma) \geq 0.5) )</td>
<td>85.59</td>
</tr>
<tr>
<td>BtcDet(_4)</td>
<td>( P(O_\Sigma) ) ( \perp ) ( 1(P(O_\Sigma) \geq 0.5) )</td>
<td>( P(O_\Sigma) ) ( \perp ) ( 1(P(O_\Sigma) \geq 0.5) )</td>
<td>85.59</td>
</tr>
</tbody>
</table>

Table 4: Ablation studies on the learned features (Sec.) and the features fused into Ψ and \( f_{geo} \) (Sec.). BtcDet\(_2\) directly uses a binary map that labels \( R_{OC} \bigcup R_{SM} \) and \( \bot \) to indicate the spherical and the Cartesian coordinate. The \( \bot \) 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\(_2\) directly fuses the binary map of \( R_{OC} \bigcup R_{SM} \) into the detection pipeline. Although the binary map provides the information of occlusion, the improvement is limited since the regions with code 1 are mostly background regions and less informative.

BtcDet\(_3\) learns \( P(O_\Sigma) \) \( \perp \) 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\(_4\) 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\(_1\).

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

In Table 5, the experiment on BtcDet\(_5\) shows that we can improve the final prediction AP by 0.8 if we only integrate \( P(O_\Sigma) \) \( \perp \) for proposal refinement. On the other hand, the experiment on BtcDet\(_6\) shows the integration with \( \Psi \) alone can improve the AP by 1.2 for proposal box and final bounding box prediction AP by 2.0 over the baseline.

The comparisons of BtcDet\(_7\), BtcDet\(_8\), and BtcDet (main) demonstrates integrating \( P(O_\Sigma) \) \( \perp \) with \( \Psi \)'s first two layers is the best choice. Since \( P(O_\Sigma) \) is a low level feature while the third layer of \( \Psi \) would contain high level features, we observe a regression when BtcDet\(_8\) also concatenates \( P(O_\Sigma) \) \( \perp \) 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


