# Deformable Shape-aware Point Generation for 3D Object Detection

Kai Wang[1][0009−0002−5567−6393] and Xiaowei Zhang(✉)[2][0000−0003−4854−3736]

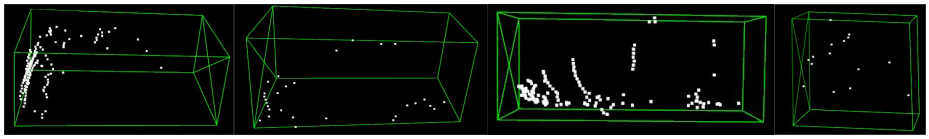
Qingdao University, Qingdao, China
2018207245@qdu.edu.cn, by1306114@buaa.edu.cn

**Abstract.** As a fundamental task of the perception system for autonomous driving, LiDAR-based 3D object detection often suffers from the absence of incomplete object shapes under long distances and occlusion. 3D object detectors, based on point cloud completion methods, significantly improve detection performance by generating pseudo-point clouds. However, due to the scarcity of guidance provided by the geometric shape and orientation information of objects, it is more challenging to recover the accurate surface shape of objects. Motivated by this, we propose Deformable Shape-aware Point Generation(DSaPG) for 3D object detection. Specifically, we design a Geometry-guided Region Proposal Network (GgRPN) including a heatmap-guided shape branch and an orientation alignment supervision, which contributes to the generation of high-quality proposals. Moreover, we design a Density-aware Deformable Point Generation(DDPG) module, which generates points by encoding probability density at each grid ball query and deformation learning to accurately recover the shape of objects. The deformable-augmented generated points can more effectively represent the shape features of objects for 3D bounding box predictions. Experiments demonstrate that the proposed DSaPG achieves competitive performance on the KITTI dataset and the Waymo Open Dataset. The code will be available at https://github.com/Wkk121/DSaPG.

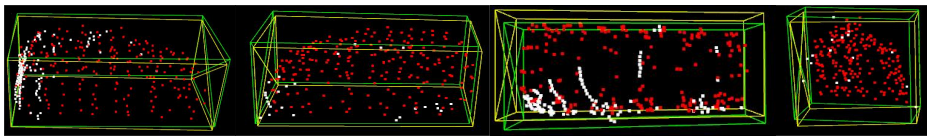
**Keywords:** 3D Object Detection · Geometry-guided RPN · Point Generation

## 1 Introduction

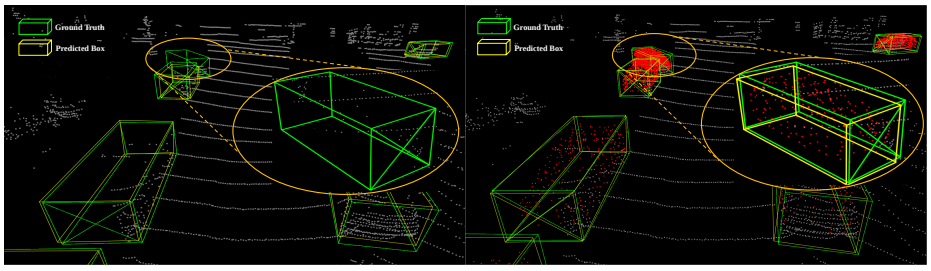
3D object detection is one of the key fundamental tasks for autonomous driving perception system. LiDAR has become one of the most popular sensors for 3D object detection [25,34,13,35], due to their ability to measure the depth of a scene [7] in the form of a point clouds, providing more geometric, shape and scale information in a variety of conditions. Despite these advantages of LIDAR sensors, due to the intrinsic sparsity of LiDAR point clouds, large-scale point cloud scenes often exhibit an imbalance in point cloud density. Objects near the sensor typically exhibit a greater number of points and a more comprehensive shape, whereas those located further away or subject to occlusion may yield



(a) Objects with raw points



(b) Objects completed by DSaPG(Ours)




(c) Missed object without point completion (d) Detected object with point completion

Fig. 1: The illustration of the effectiveness of DSaPG(Ours). The ground-truth boxes, the generated points, and the predicted boxes are shown in green, red, and yellow. The orange oval indicates that the completed object can be easily detected.

fewer points and lack complete shapes[16,32], as shown in Fig.1 (a) and (c). This situation becomes even more critical when dealing with distant objects, as it has the potential to significantly compromise the accuracy of the detector.

To address the limitations of incomplete shapes during detection, point cloud completion methods have been explored for 3D object detection to recover the desired shape within the foreground region. Existing methods can be broadly classified into three categories depending on the source of the generated points: prior knowledge based, image based, and pre-trained completion network based point cloud completion. A straightforward approach is to learn object shape from priors, the methods [33,32] predict the foreground probability in a bounding box to recover the specific shape of the object.

However, these methods are still computationally expensive and often unable to accurately recover the shape of the object. While image based point cloud completion methods[36,30,42] usually combine RGB semantic information with point clouds to achieve local or global completion. However, the dense virtual points generated from images often cause a huge computational burden, resulting in serious efficiency issues. Pre-trained completion network based methods

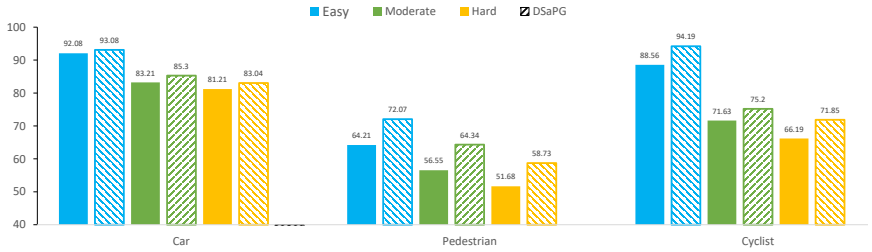



Fig. 2: This shows the performance of Voxel-RCNN[5] and DSaPG(Ours) on the KITTI val split. We show the result in three classes with three difficulty levels: Easy, Moderate, and Hard. After associating with the point generation module, the performance significantly improves.

[16,38,12] utilize a point cloud completion network pre-trained from an external dataset[2] to recover the shape of object by generating dense points in the ROIs. While these methods achieve a certain level of accuracy and efficiency, they often produce incorrect points in the wrong location and orientation due to inaccurate region suggestions and lack of geometric shape and orientation information of objects. Therefore, the recovery of an object's proper shape at a reasonable location remains a significant challenge for point cloud completion.

To tackle this challenge, we propose a novel two-stage 3D detector, namely Deformable Shape-aware Point Generation(DSaPG), which leverages the geometric shape and orientation information to complete the entire shape of objects for 3D bounding box predictions, as shown in Fig.1 (b) and (d). Specifically, we design a Geometry-guided Region Proposal Network(GgRPN) consisting of a heatmap-guided shape branch and an orientation alignment supervision. The effectiveness of heatmaps in providing information about the shape of objects is demonstrated by various methods [22,31]. Therefore, we generate a BEV shape heatmap for each scene through a pillar-wise shape occupancy network. While inspired by [39], the orientation supervision strengthens the constraints on the BEV orientation angle to obtain more accurate orientation alignment. GgRPN can generate high-quality 3D proposals and provide comprehensive priors with geometric and orientation information. Furthermore, to achieve a more reasonable recovery of the object shape and obtain more accurate fine features, we propose a density-aware deformable point generation(DDPG) module. In contrast to previous methods[16,38] that utilize raw points in ROIs or simple voxelized features as inputs, we utilize voxel centroid sampling [28] to preserve the original geometric structure information of the object. Subsequently, the estimated probability density is utilized as an additional feature in the grid ball query to encode more implicit grid features and obtain the generated points. In order to improve the spatial information features of the generated points, inspired by [1], an offset is created for each generated point based on its foreground score. The deformable-augmented generated points visually represent the predicted position

and shape of the object, thereby enhancing the detection accuracy of occluded and distant objects, illustrated in Fig.2.

Experiments demonstrate that our method achieves progressive results on KITTI datasets[6] and Waymo Open Datasets[26]. Our main contributions are summarized as follows:

- We present a deformable shape-aware point generation(DSaPG) 3D object detector, which accurately generates points at reasonable locations to complete the missing shape of the object, enhancing detection performance.
- We design a Geometry-guide RPN with shape heatmap and orientation alignment, which can generate high-quality 3D proposals and provide comprehensive priors with geometric shape information. To accurately complete the proper shape of the missing object, we propose a Density-aware Deformation Point Generation Module, which generates semantic points by encoding point probability density and learning deformation.
- Extensive experiments demonstrate that our DSaPG achieves competitive results on the KITTI Dataset and Waymo Open Dataset.

## 2 Related Work

### 2.1 LiDAR-Based 3D Object Detection

LiDAR-based object detection can be categorized into point-based and voxel-based methods depending on the feature representation learning on point clouds. Point-based methods directly learn point features for detection by sampling raw point clouds. PointRCNN [25] generates region of interest (ROI) at the point-level using the PointNet++[21] backbone and utilizes point-level features for bounding box refinement. SASA [3] enhances the sampling process by employing semantic segmentation to guide the downsampling of the point cloud to preserve points for accurate prediction. Point-based methods accurately position 3D objects but often have high computational costs due to sampling and grouping point clouds.

Voxel-based methods rasterize irregular point clouds into volumetric grids and apply 3D and 2D convolutions directly to generate predictive bounding boxes. VoxelNet [41] divides the point cloud into 3D voxels, which are further processed by the voxel feature extractor and 3D CNN encoder network. SECOND [34] introduces sparse 3D convolution which significantly improves the inference speed. PV-RCNN [24] uses the strengths of voxel-based and point-based methods to extract more discriminative features. PDV[8]uses feature coding to interpret point density directly, addressing the issue of point cloud density imbalance. However, the limited number of points captured for distant or obstructed objects continues to present a challenge to detection performance.

### 2.2 Point Cloud Completion for 3D Object Detection

To address the challenges presented by sparse and incomplete point clouds, certain methods aim to enhance the sparsity of points by generating virtual points

around the LiDAR points. Prior knowledge based point cloud completion methods: SPG [33] is an unsupervised domain adaptive method that generates semantic points in the predicted foreground region and faithfully recovers the missing parts in the foreground objects. BtcDet [32] intends to solve the problem of point cloud incompleteness due to occlusion by learning the object shape prior and estimating the complete object shape in the point cloud that is partially occluded. Image based point cloud completion methods: MVP [36] creates virtual points by completing the depth of a 2D instance point from the nearest 3D point. SFD [38] creates virtual points based on depth completion networks[10]. Pre-trained completion network based point completion methods: SIENet [16] leverages a pre-trained PCN[37] point cloud completion network to capture the abundant spatial information of ROIs. PGRCNN [12] uses a jointly trained RoI point generation module to process the contextual information of RoIs for point generation. To a certain extent, PGRCNN[12] can generate surface points with semantic characteristics and ensure the validity of generation points. However, one-stage guidance can still result in incorrect generation points and positions. The key focus of point cloud generation algorithms is to accurately recover the object's shape without introducing excessive interference. Our approach, DSaPG, fully incorporates the geometric and orientational characteristics of the object while integrating point density perception to ensure efficacy and positional accuracy of the generated points.

## 3 Methodology

### 3.1 Overview

We propose a two-stage Deformable Shape-aware Point Generation for 3D object detection composed of a geometry-guided RPN and a density-aware deformable point generation module. Fig.3 illustrates the overview of the DSaPG framework. Following many recent works[5,34], we first use a 3D voxel backbone similar to SECOND [34] to encoder point cloud, Subsequently, we utilize a geometry-guided Region Proposal Network to generate high-quality proposals. In the second stage, we propose a density-aware deformable point generation module to accurately complete the shape of the object by generating deformable points. Following this, refinement of the generated points data is conducted in order to obtain the most accurate detection output.

### 3.2 Geometry-guided Region Proposal Network

In order to obtain fine-grained geometric and orientation information guidance, we propose a Geometry-guided Region Proposal Network(GgRPN) with a heatmap-guided shape branch and an orientation alignment supervision.
**Heatmap-guided shape branch**. Firstly, in terms of accuracy and speed, we utilize [14] to extract multi-scale pillar features. Subsequently, the approximation method of [32] is adopted to generate a more comprehensive Bird's Eye View

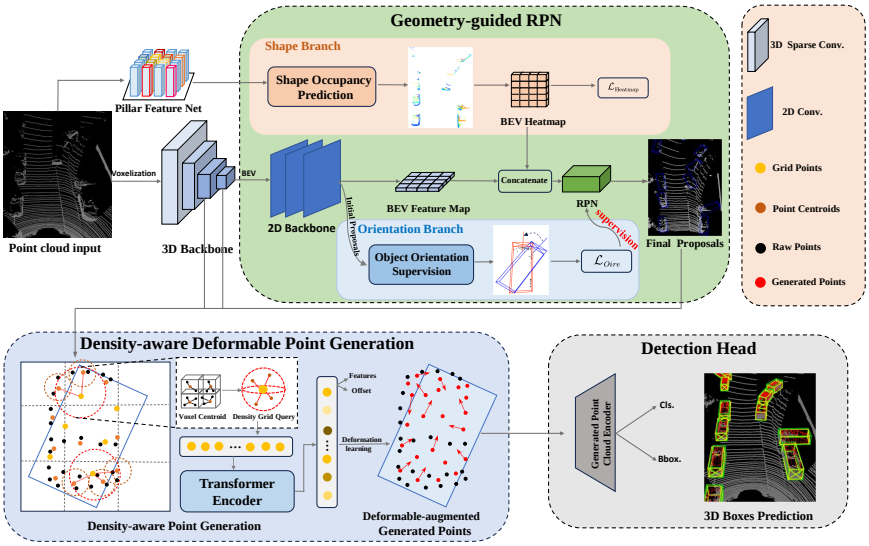



Fig. 3: DSaPG Overview. The input point cloud is first voxelized and processed through 3D sparse convolutions, a geometry-guided RPN is designed to help produce high-quality proposals. Density-aware deformable point generation module faithfully restores object missing shape. Finally, the detection head produces the final detection output using the generated points as input.

(BEV) map. After that, we apply rendered Gaussian kernels to mitigate the noise introduced by shape priors, getting a high-quality ground truth label of the BEV shape heatmap as follows:

$$K_{xy} = \exp\left(-\frac{(x-p_x)^2 + (y-p_y)^2}{2\sigma_p^2}\right), \tag{1}$$

where $(x-p_x)$ and $(y-p_y)$ are the distances to the center of the object, and $\sigma_p$ is an object size-adaptive standard deviation depending on the size of the object. The pillar-wise shape occupancy network $\Psi$ is used to make shape occupancy prediction, we set shape occupancy probability $P(O_s)$=0 for pillars that do not contain shapes. The last convolutional layer of $\Psi$ generates a predicted heatmap $\hat{K}$ representing the probability of shape occupancy. After applying the sigmoid function with the threshold to emphasize the shape, we obtain the final predicted heatmap $K_{BEV} \in \mathbb{R}^{h \times w}$, where $h$ and $w$ denote the scale of the image. Then, the branch can be optimized with a focal loss [17], if $K_{xy}$= 1:

$$\mathcal{L}_{\text{Heatmap}} = -\frac{1}{N}\sum_{xy}\left(1-\hat{K}_{xy}\right)^{\alpha}\log\left(\hat{K}_{xy}\right), \tag{2}$$

otherwise:

$$\mathcal{L}_{\text{Heatmap}} = -\frac{1}{N}\sum\left(1-K_{xy}\right)^{\beta}\left(\hat{K}_{xy}\right)^{\alpha}\log\left(1-\hat{K}_{xy}\right), \tag{3}$$

where $\alpha = 2$ and $\beta = 4$ are default hyper-parameters and $N$ is the number of objects where $P(O_s) = 1$. To incorporate shape knowledge, we concatenate the $K_{BEV}$ with BEV features $\mathbf{f}^{bev}$ produced by the 2D backbone and then input them into the RPN to produce 3D proposals.

**Orientation alignment supervision**. Due to the sparse nature of the point cloud and challenges such as occlusion, it is difficult to extract enough information for accurate localization of the initial bounding box. This significantly impacts point cloud generation in the next stage. In order to alleviate the impact of imprecise proposal orientations, we fully consider the orientation constraint between the ground-truth and the initial bounding boxes. More importantly, we add an angle constraint for the predicted BEV orientation, aiming to further minimize the orientation difference between the initial and ground-truth boxes. Overall, we get the final proposals, and the orientation loss is calculated as:

$$\mathcal{L}_{Oir} = 1 - IoU\left(B_i, B_g\right) + \frac{c^2}{d^2} + \gamma(1 - |\cos(\triangle\alpha)|), \tag{4}$$

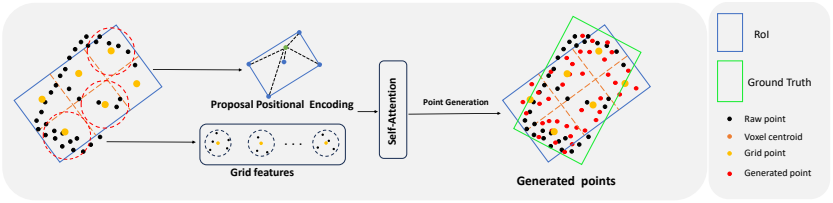
where $B_i$ and $B_g$ denote the initial and ground-truth bounding boxes, respectively, $c$ denotes the distance between the 3D centers of the two bounding boxes, $d$ denotes the diagonal length of the minimum cuboid that encloses both bounding boxes; $\triangle\alpha$ denotes the BEV orientation difference between $B_i$ and $B_g$; and $\gamma$ is a hyper-parameter weight (we set 1.25.).

### 3.3 Density-aware Deformable Point Generation

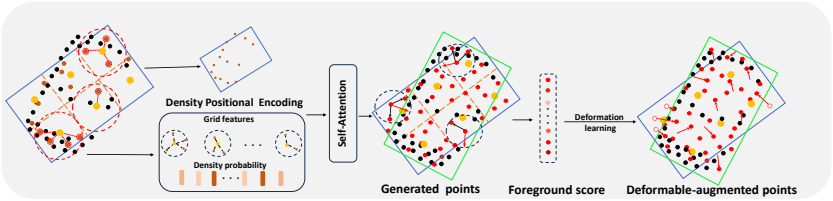
As illustrated in Fig. 4, compared with PGRCNN[12], our method incorporates density probability as an additional feature encoded into the grid query to obtain rich geometric shape information of objects. Furthermore, we also conduct deformation learning on the generated points, which helps to restore the precise shape of objects as much as possible. Specifically, we first encode the non-empty voxel information by utilizing voxel point centroids[28]. Let $\mathbf{\Gamma}_i = \{\ \mathbf{V}_i, \mathbf{f}_{\mathbf{V}_i}\}_{i=1}^{K}$ be the set of non-empty voxels, where $\mathbf{V}_i$ is the voxel index, $\mathbf{f}_{\mathbf{V}_i}$ is the non-empty voxel feature vector and $K$ is the number of non-empty voxels. The voxel point centroid $\mathbf{c}_{\mathbf{V}_i}$ is then calculated by averaging the spatial positions of all points within the same voxel $\mathbf{V}_i$:

$$\mathbf{c}_{\mathbf{V}_i} = \frac{1}{\left|\mathcal{N}\left(\mathbf{V}_i\right)\right|}\sum_{\boldsymbol{x}_i \in \mathcal{N}(\mathbf{V}_i)} \boldsymbol{x}_i, \tag{5}$$

where $\boldsymbol{x}_i$ is the spatial coordinates, $\mathcal{N}(\mathbf{V}_i)$ is the number of points within the same voxel. Next, we follow[8] to assign a voxel grid index to the centroid of each voxel point through a hash table and match the relevant voxel features.




(a) Point generation module of PG-RCNN[12]




(b) Point generation module of DSaPG(Ours)

Fig. 4: Comparison of our density-aware deformable point generation module and PGRCNN method.

To further acquire the geometric structural information of the object, we use the method of KDE[20] to encode the point density feature into each grid point feature. Specifically, we first divide each region proposal into $G \times G \times G$ regular grid points, $\mathbf{g}$=$\{$ $\mathbf{g}_1$, $\mathbf{g}_2$..., $\mathbf{g}_{G^3}\}$. The likelihood is calculated for each grid point using KDE:

$$p\left(\mathbf{c}_{\mathbf{V}_i} \mid \mathbf{g}_i\right) \approx \frac{1}{|\mathcal{N}\left(\mathbf{g}_i\right)| \, \sigma^3} \sum_{\mathbf{c}_{\mathbf{V}_i} \in \mathcal{N}(\mathbf{g}_i)} \prod_{d=1}^{3} w\left(\frac{\mathbf{c}_{\mathbf{V}_i,d} - \mathbf{c}_{\mathbf{V}_j,d}}{\sigma}\right), \tag{6}$$

where $\mathcal{N}(\mathbf{g}_i)$ is the set of voxel point centroids in a sphere of radius $r$ centered around $\mathbf{g}_i$, $\sigma$ is the bandwidth, $w$ is a non-negative kernel function, and there is an independent kernel for each xyz dimension $d$.

We aggregate the neighboring features within a radius $r$ of each grid point and incorporate the estimated probability density as an additional feature in each grid point's ball query. This provides a reliable guide to the object's geometry for the subsequent step of point generation. Once the features are appended, a PointNet multi-scale grouping module[21] is used to encoder a feature vector $\mathbf{f}_{\mathbf{g}_i}$ for each grid point $\mathbf{g}_i$ :

$$\mathbf{f}_{\mathbf{g}_i} = \text{MaxPool}\left(\{\text{FFN}([\mathbf{f}_{\mathbf{V}_i}, p(\mathbf{c}_{\mathbf{V}_i} \mid \mathbf{g}_i)])\}_{i=1}^{K}\right). \tag{7}$$

To capture the long-range dependencies between the grid points, we further process the non-empty grid point features with a transformer encoder[23]. The refined grid point feature $\tilde{\mathbf{f}}_{\mathbf{g}_i}$ is formulated as:

$$\tilde{\mathbf{f}}_{\mathbf{g}_i} = \mathcal{T}\left(\mathbf{f}_{\mathbf{g}_i}, \delta_{\mathbf{g}_i}\right) + \mathbf{f}_{\mathbf{g}_i}, \tag{8}$$

where $\mathcal{T}$ is a standard transformer encoder[29], and $\delta_{\mathbf{g}_i}$ is the positional encoding. To encode positional information, we add positional encoding to the self-attention module by using centroid offsets and the number of points in the box proposal, as described in [8]. The positional encoding for each grid feature is then calculated as:

$$\delta_{\mathbf{g}_i} = \mathrm{FFN}([(\boldsymbol{x}_{\mathbf{g}_i} - \mathbf{c}_b), \log(|\mathcal{N}(\mathbf{V}_{\mathbf{g}_i})| + \epsilon)]), \tag{9}$$

where $\boldsymbol{x}_{\mathbf{g}_i}$ is the spatial coordinate of $\mathbf{g}_i$, $\mathbf{c}_b$ is the bounding box proposal centroid, $|\mathcal{N}(\mathbf{V}_{\mathbf{g}_i})|$ is the number of points in each grid point voxel $\mathbf{V}_{\mathbf{g}_i}$, and $\epsilon$ is a constant offset.

Then, a two-layer MLP $\mathcal{A}$ is applied to the refined features to generate the offset $\mathbf{o}_i$ and the features $\mathbf{f}_{\mathbf{p}_i}$ of the generated points from the grid point:

$$[\mathbf{o}_i; \mathbf{f}_{\mathbf{p}_i}] = \mathcal{A}\left(\tilde{\mathbf{f}}_{\mathbf{g}_i}\right). \tag{10}$$

The generated point's coordinates $\boldsymbol{x}_{\mathbf{p}_i} = (x_i, y_i, z_i)$ can be calculated as $\boldsymbol{x}_{\mathbf{g}_i} + \mathbf{o}_i$. The foreground score $\mathbf{s}_i$ for each generated point is calculated by applying a linear projection and a sigmoid function to its semantic feature $\mathbf{f}_{\mathbf{p}_i}$.

In order to further obtain even and accurate generation points, we use the FPS algorithm to select $N_p$ points from the scene and add a learnable offset to each generated point based on their foreground probability score. Specially, given a generated point $\mathbf{p}_i$, the original coordinates are $\boldsymbol{x}_{\mathbf{p}_i}$ and its associated foreground score is $\mathbf{s}_i$, then we use the point coordinates and foreground score in the neighborhood to calculate an offset to get the new generated point position:

$$\Delta_{\text{offset}} = \tanh\left(W_1 \cdot \tfrac{1}{N_p} \textstyle\sum_{j=1}^{N_p} \mathrm{RELU}\left(W_2 \cdot (\mathbf{s}_i - \mathbf{s}_j) \cdot (\boldsymbol{x}_{\mathbf{p}_i} - \boldsymbol{x}_{\mathbf{p}_j})\right)\right) \tag{11}$$

where $W_1$, $W_2$ are learnable weights. The new point position $\boldsymbol{x}'_{\mathbf{p}_i}$ after learning the deformation is calculated as $\boldsymbol{x}_{\mathbf{p}_i} + \Delta_{\text{offset}}$. The deformable-augmented generated points can be adjusted to a more meaningful position, or even outside the ROI, which can produce a more accurate and comprehensive shape point cloud. This will enable the extraction of more effective spatial information features. Finally, we feed the point set with features into the PointNet++ encoder[21] to obtain the refinement feature for RoI.

### 3.4 Training Losses

The proposed DSaPG is trained in an end-to-end manner. Our overall loss includes $\mathcal{L}_{rpn}$ in stage-one, refinement loss $\mathcal{L}_{rcnn}$ in stage-two, and the point generation loss $\mathcal{L}_{\text{gen}}$:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{rpn} + \mathcal{L}_{rcnn} + \mathcal{L}_{\text{gen}}. \tag{12}$$

The RPN loss $\mathcal{L}_{rpn}$ is defined as the summation of classification loss, box regression loss, and the auxiliary supervision loss $\mathcal{L}_{Heatmap}$ in Eq. 2, it is worth noting that our $\mathcal{L}_{Oir}$ in Eq. 4 will replace RPN's box regression loss as:

$$\mathcal{L}_{rpn} = \alpha\,(1 - p_t)^{\gamma} \log(p_t) + \lambda \mathcal{L}_{Heatmap} + \omega \mathcal{L}_{Oir}, \tag{13}$$

where the coefficients $\lambda$, $\omega$ are set to 1.0 and 1.0 to balance the RPN loss. $p_t$ is the class probability of anchor and the hyper-parameters $\alpha = 0.25$ and $\gamma = 2$.

The proposal refinement loss $\mathcal{L}_{rcnn}$ includes the IoU-guided confidence prediction loss [24] and box refinement loss as:

$$\mathcal{L}_{rcnn} = \mathcal{L}_{iou} + \sum_{r_2} \mathcal{L}_{\text{smooth-L1}}\ (\Delta r_2)\,, \tag{14}$$

where $\Delta r_2$ is the residual between the predicted box and proposal target which are encoded similarly to $\Delta r_1$.

$\mathcal{L}_{\text{gen}}$ is an auxiliary loss term that specifically supervises point generation, calculated with the point generation module outputs, same as PGRCNN[12]:

$$\mathcal{L}_{\text{gen}} = \mathcal{L}_{\text{seg}}\ + \mathcal{L}_{\text{shape}}\ , \tag{15}$$

where $\mathcal{L}_{\text{seg}}$ is a segmentation loss that controls the foreground probability of the generated point, and $\mathcal{L}_{\text{shape}}$ supervises the shape of the generation point cloud.

## 4 Experiments

### 4.1 Dataset and Evaluation

**KITTI Dataset**.The KITTI dataset[6] includes 7481 LiDAR frames for training and 7518 LiDAR frames for testing. We follow [34] to divide the training data into a train split of 3712 frames and val split of 3769 frames. We report the results on the validation set using 3D AP at 40 recall thresholds. The IoU thresholds used for cars, pedestrians, and cyclists are 0.7, 0.5, and 0.5 respectively.
**Waymo Open Dataset(WOD)**. The WOD [26] consists of 1000 sequences, including 798 training sequences and 202 validation sequences. The dataset considers two difficulty levels, LEVEL_1(boxes with at least five LiDAR points) and LEVEL_2 (boxes with at least one LiDAR point). The official evaluation metric for 3D detection is the mean Average Precision (mAP), with an IoU threshold of 0.7 for measuring the detection performance of the vehicle category.

### 4.2 Implementation Details

In this work, we implement our work on OpenPCDet[27]. We used a NVIDIA RTX 3090 GPU to train our network, the batch size is set as 2 with 80 epochs with an ADAM optimizer[11]. The initial learning rate is set to 0.01 and is updated as training progresses by cosine annealing strategy. For voxelization, we clip the range of point clouds into [0, 70.4]$m$ for the X-axis, [-40, 40]$m$ for the Y-axis, and [-3, 1]$m$ for the Z-axis on KITTI dataset[6]. The input voxel size is set as ($0.05m$, $0.05m$, $0.1m$). We follow the data augmentation approaches mentioned in [34] and [24]. Density ball query uses a set of radii r = [[0.8, 1.2], [1.2, 2.4]] to aggregate voxel point centroid features at each layer with two 32-layer FFNs. We follow Pointformer [18] for the FFN size for point density positional encoding with the added density feature from the number of points in each RoI grid voxel. $N_p$, the number of points sampling for deformation learning, is set to 2048. More details are given in the supplementary material.

Table 1: Comparison with state-of-the-art methods on the KITTI val set. The best performance value is in bold, second-best is underlined.

| Method | Reference | Modality | Car 3D $AP_{R40}$ | | | Ped. 3D $AP_{R40}$ | | | Cyc. 3D $AP_{R40}$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Easy | Mod. | Hard | Easy | Mod. | Hard | Easy | Mod. | Hard |
| SECOND [34] | Sensors 2018 | L | 88.61 | 78.62 | 77.22 | 56.55 | 52.98 | 47.73 | 80.58 | 67.15 | 63.10 |
| PointRCNN [25] | CVPR 2019 | L | 88.72 | 78.61 | 77.82 | 62.72 | 53.85 | 50.25 | 86.84 | 71.62 | 65.59 |
| EPNet[9] | ECCV 2020 | L+R | 88.76 | 78.65 | 78.32 | 66.74 | 59.29 | 54.82 | 83.88 | 65.60 | 62.70 |
| CLOCs[19] | IROS 2020 | L+R | 89.49 | 79.31 | 77.36 | 62.88 | 56.20 | 50.10 | 87.57 | 67.62 | 63.67 |
| PV-RCNN[24] | CVPR 2020 | L | 92.10 | 84.36 | 82.48 | 64.26 | 56.67 | 51.91 | 88.88 | 71.95 | 66.78 |
| PCRGNN[38] | AAAI 2021 | L | 90.94 | 81.43 | 80.45 | - | - | - | - | - | - |
| VoxelRCNN [5] | AAAI 2021 | L | 92.38 | 85.29 | 82.86 | - | - | - | - | - | - |
| SIENet[16] | PR 2022 | L | 92.49 | 85.43 | 83.05 | - | - | - | - | - | - |
| BtcDet [32] | AAAI 2022 | L | **93.15** | **86.28** | 83.86 | 69.39 | 61.19 | 55.86 | 91.45 | 74.70 | 70.08 |
| PDV [8] | CVPR 2022 | L | 92.56 | 85.29 | 83.05 | 66.90 | 60.80 | 55.85 | 92.72 | 74.23 | 69.60 |
| LoGoNet [15] | CVPR 2023 | L+R | 92.04 | 85.04 | **84.31** | 70.20 | 63.72 | 59.46 | 91.74 | **75.35** | 72.42 |
| PGRCNN [12] | ICCV 2023 | L | 92.73 | 85.26 | 82.83 | 68.44 | 60.63 | 55.36 | 93.84 | 74.85 | 70.15 |
| DSaPG(Ours) | - | L | 93.08 | 85.30 | 83.24 | **72.07** | **64.34** | **59.75** | **94.19** | 75.20 | **72.65** |

### 4.3 KITTI Dataset Results

Table 1 shows the results on the KITTI val set. Corresponding to the KITTI protocol [6], we calculate the AP results under 40 recall thresholds (R40). As shown in Table 1, our DSaPG achieves outstanding performance for cyclist and pedestrian detection. Compared to the state-of-the-art method PGRCNN[12], DSaPG gains by 3.71% and 0.35% in AP metric on the moderate level of the pedestrian and cyclist categories, respectively. Experiment results reveal that the advantages of our DSaPG are more prominent for the pedestrian category. For instance, compared to state-of-the-art multi-modal method LoGoNet[15], our DSaPG improves by 1.87%, 0.62% and 0.29% percentage points in AP under easy, moderate, and hard levels respectively. And our DSaPG demonstrates 0.52% AP performance gains over PDV[8] of the car at the hard level. The experimental results on the KITTI test set are shown in Table 2, our proposed DSaPG shows superior performance across different levels for the cyclist category. Specifically, our DSaPG outperforms PGRCNN [12] 0.91%, 0.16%, and 1.10% under three levels for cyclist. DSaPG also outperforms recent methods like OcTr[40] by 0.12% and 0.16% AP for car at moderate and hard levels. These experiments demonstrate the effectiveness of our proposed method to solve the problem of missing shapes due to occlusion and distance. Moreover, DSaPG achieves highly competitive inference speed compared to the classical two-stage 3D detector PVRCNN [24] and the same type completion method BtcDet[32]. Qualitative results on the KITTI validation set are illustrated in Fig. 5.

### 4.4 Waymo Open Dataset Results

We also report the multi-class Waymo Open Dataset results on the validation set in Table 3. DSaPG achieves state-of-the-art results on all classes on both LEVEL _1 and LEVEL_2 mAP/mAPH metrics. Our method leads a boost of 0.28% on vehicle LEVEL_2 3D mAP. DaSPG performs well on the other

Table 2: Comparison with state-of-the-art methods on the KITTI test set. The best performance value is in bold, second-best is underlined.

| Method | Time | Car 3D $AP_{R40}$ | | | Car BEV $AP_{R40}$ | | | Cyc. 3D $AP_{R40}$ | | | Cyc. BEV $AP_{R40}$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ms | Easy | Mod. | Hard | Easy | Mod. | Hard | Easy | Mod. | Hard | Easy | Mod. | Hard |
| SECOND [34] | 50 | 83.34 | 72.55 | 65.82 | 89.39 | 83.77 | 78.59 | 71.33 | 52.08 | 45.83 | 76.50 | 56.05 | 49.45 |
| PointRCNN [25] | 100 | 86.96 | 75.64 | 70.70 | 92.13 | 87.39 | 82.72 | 74.96 | 58.82 | 52.53 | 82.56 | 67.24 | 60.28 |
| EPNet[9] | - | 89.81 | 79.28 | 74.59 | 94.22 | 88.47 | 83.69 | - | - | - | - | - | - |
| CLOCs[19] | 100 | 88.94 | 80.67 | 77.15 | 93.05 | 89.80 | 86.57 | - | - | - | - | - | - |
| PV-RCNN[24] | 80 | 90.25 | 81.43 | 76.82 | **94.98** | **90.65** | 86.14 | 78.60 | 63.71 | 57.65 | 82.49 | 68.89 | 62.41 |
| PCRGNN[38] | - | 89.13 | 79.90 | 75.54 | 94.91 | 89.62 | 86.57 | - | - | - | - | - | - |
| CT3D [23] | 70 | 87.83 | 81.77 | 77.16 | 92.36 | 88.83 | 84.07 | - | - | - | - | - | - |
| VoxelRCNN [5] | **40** | **90.90** | 81.62 | 77.06 | 94.85 | 88.83 | 86.13 | - | - | - | - | - | - |
| SIENet [16] | - | 88.22 | 81.71 | 77.22 | 92.38 | 88.65 | 86.03 | - | - | - | - | - | - |
| BtcDet [32] | 90 | 90.64 | **82.86** | **78.09** | 92.81 | 89.34 | 84.55 | 82.81 | **68.68** | 61.81 | 84.48 | **71.76** | 64.70 |
| PDV [8] | - | 90.43 | 81.86 | 77.36 | 94.56 | 90.48 | 86.23 | 83.04 | 67.81 | 60.46 | **85.54** | 71.31 | 64.40 |
| OcTr[40] | - | 90.88 | 82.64 | 77.77 | - | - | - | - | - | - | - | - | - |
| PGRCNN [12] | 60 | 89.38 | 82.13 | 77.23 | 93.39 | 89.46 | 86.54 | 82.77 | 67.82 | 61.25 | 84.94 | 70.65 | 64.03 |
| DSaPG(Ours) | 70 | 90.61 | 82.76 | 77.93 | 94.92 | 90.59 | **86.85** | **83.68** | 67.98 | **62.35** | 85.03 | 71.35 | **64.79** |

Table 3: Performance comparison on the validation set of Waymo Open Dataset. 20 % of training data was used for this experiment. †denotes the re-implemented model in the same environment.

| Method | Vehicle | | | | Pedestrian | | | | Cyclist | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LEVEL_1 | | LEVEL_2 | | LEVEL_1 | | LEVEL_2 | | LEVEL_1 | | LEVEL_2 | |
| | mAP | mAPH | mAP | mAPH | mAP | mAPH | mAP | mAPH | mAP | mAPH | mAP | mAPH |
| SECOND [34] | 72.27 | 71.69 | 63.85 | 63.33 | 68.70 | 58.18 | 60.72 | 51.31 | 60.62 | 59.28 | 58.34 | 57.05 |
| PV-RCNN[24] | 75.17 | 74.60 | 66.35 | 65.84 | 72.65 | 63.52 | 63.42 | 55.29 | 66.26 | 64.82 | 63.88 | 62.48 |
| CT3D [23] | 76.30 | - | 69.04 | - | - | - | - | - | - | - | - | - |
| VoxelRCNN [5] | 75.59 | 74.23 | 66.59 | 65.72 | 66.79 | 58.56 | 58.98 | 50.62 | 67.23 | 66.98 | 66.01 | 63.25 |
| PGRCNN† [12] | 75.26 | 74.36 | 68.19 | 67.84 | 73.81 | 66.56 | 66.01 | 61.14 | 68.69 | 68.02 | 66.16 | 65.10 |
| PDV [8] | **76.85** | 76.33 | 69.30 | 68.81 | 74.19 | 65.96 | 65.85 | 58.28 | 68.71 | 67.55 | 66.49 | 65.36 |
| DSaPG(Ours) | 76.45 | **76.59** | **69.58** | **68.86** | **74.56** | **66.72** | **66.26** | **62.32** | **69.56** | **68.29** | **67.23** | **65.66** |
| *Improvement* | ***-0.40*** | ***+0.26*** | ***+0.28*** | ***+0.05*** | ***+ 0.37*** | ***+0.76*** | ***+0.25*** | ***+1.18*** | ***+0.85*** | ***+0.27*** | ***+0.74*** | ***+0.30*** |

classes as well, increasing performance by 1.18% and 0.30% on pedestrian and cyclist LEVEL_2 3D mAPH, respectively. We also outperform PDV[8] and PGRCNN[12] by 4.04%, and 1.18% on pedestrian, and cyclist LEVEL_2 3D mAPH, respectively. It is worth noting that our method does not perform well on vehicle, possibly because it requires complete point clouds of objects to supervise point generation. The Waymo Open Dataset has significantly more object instances than KITTI, which makes it difficult to collect points for all instances.

### 4.5 Ablation Studies

**Effect of Components.** Table 4 details how each proposed module influences the accuracy of our DSaPG. The results are evaluated with $AP_{R40}$ and $AP_{R11}$ of moderate level for car, cyclist, and pedestrian. Method(a) is the two-stage voxel-based baseline. Method(b) extends (a) with a density-aware deformation point generation. The DDPG module leads to a boost of 1.59%, 1.56%, 2.23% AP, which verifies that DDPG can significantly improve detection performance.

Table 4: Performance comparison of different components on KITTI val set.

| Methods | GgRPN | DDPG | 3D $AP_{R40}$(Mod.) | | | 3D $AP_{R11}$(Mod.) | | |
|---|---|---|---|---|---|---|---|---|
| | | | Car | Cyc. | Ped. | Car | Cyc. | Ped. |
| a | - | - | 83.53 | 71.68 | 61.25 | 83.24 | 69.49 | 58.37 |
| b | - | ✓ | 85.12 | 73.24 | 63.48 | 84.98 | 71.59 | 61.24 |
| c | ✓ | - | 84.75 | 73.98 | 63.25 | 83.87 | 71.66 | 60.53 |
| d | ✓ | ✓ | **85.30** | **75.20** | **64.34** | **84.31** | **73.69** | **62.45** |

Table 5: Performance comparison of Geometry-guided RPN on KITTI val set.

| Methods | $L_{Heatmap}$ | $L_{Oir}$ | 3D $AP_{R40}$( Mod.) | | |
|---|---|---|---|---|---|
| | | | Car | Cyc. | Ped. |
| a | - | - | 83.53 | 71.68 | 61.25 |
| b | ✓ | - | 84.23 | **74.23** | 62.23 |
| c | - | ✓ | 83.62 | 72.69 | 62.86 |
| d | ✓ | ✓ | **84.75** | 73.98 | **63.25** |

Method(c) replaces the traditional RPN with a Geometry-guided RPN Module, which makes 1.22%, 2.30%, 2.00% AP improvement, the optimization ability of the GgRPN for the whole backbone is proved. Method (d) is the proposed DSaPG, which achieves state-of-the-art accuracy of 1.07%, 4.2%, and 4.08% under $AP_{R11}$. The two modules collaborate to enhance detection performance.

**Effect of Geometry-guide RPN Module.** In Table 5, we analyze the effect of Geometry-guide RPN Module. To investigate the impact of the shape branch and the orientation supervision on object detection performance, we ablated each loss term and compared the 3D detection performances of three classes on the moderate level. Lines (a) and (b) denote the performance boost of the shape branch, the heatmap shape improves 0.70%, 2.55%, 0.98% AP at a moderate level. BEV heatmap provides shape knowledge guidance, which aids in the detection of minute objects. Lines (a) and (c) show the effectiveness of orientation alignment supervision with an overall increase in performance of 0.09%, 1.01%, 1.61%. The orientation branch can effectively align the initial boxes with the GT boxes. Comparing lines (b), (c), and (d) with each other, it can be indicated that effective optimization of integrated detectors by joint supervision loss.

**Effect of Density-aware Deformable Point Generation.** Table 6 demonstrates the effectiveness of Density-aware Deformable Point Generation. Method (b) incorporates a density transformer into encoder grid features, significantly enhancing detection performance, particularly for cyclists by 1.67%. In method (b), we apply deformation learning on generated points, yielding improvements of 0.34%, 0.64%, and 0.78% across three classes. Method (c) involves a point generation module that extracts ROI features and creates virtual points to complete shapes, resulting in a 2.50% improvement at the moderate level for pedestrian. Method (f), representing the full DDPG framework, further boosts detection results with improvements of 1.59%, 1.56%, and 2.23% over method (a). These

Table 6: Performance comparison of Density-aware Deformable Point Generation on KITTI val set. D.T., D.L., and P.G. stand for the density-aware transformer, deformation learning, and point generation respectively.

| Methods | D.T. | D.L | P.G. | 3D $AP_{R40}$(Mod.) | | |
|---|---|---|---|---|---|---|
| | | | | Car | Cyc. | Ped. |
| a | - | - | - | 83.53 | 71.68 | 61.25 |
| b | ✓ | - | - | 84.56 | **73.35** | 62.95 |
| c | - | ✓ | - | 83.87 | 72.32 | 62.03 |
| d | - | - | ✓ | 84.63 | 73.16 | **63.75** |
| e | ✓ | ✓ | - | 84.66 | 73.08 | 63.24 |
| f | ✓ | ✓ | ✓ | **85.12** | 73.24 | 63.48 |

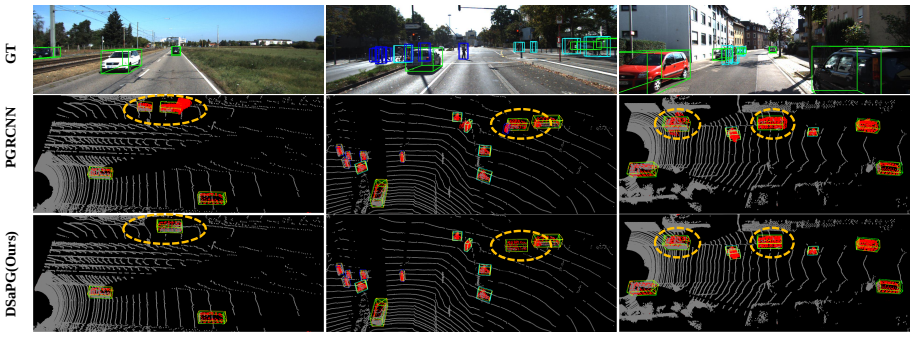



Fig. 5: Qualitative results performed on the KITTI training split. The ground truth for the car, pedestrian, and cyclist are shown in green, blue, and cyan, respectively. Generated points and predicted boxes are marked in red and yellow.

ablation experiments confirm that our DDPG effectively enhances detection performance.

## 5 Conclusion

In this paper, we present a deformable shape-aware point generation(DSaPG) 3D object detector, which accurately generates points at reasonable locations to complete the missing shape of the object. Specially, in the first stage, we design a geometry-guided RPN with BEV shape heatmap and orientation alignment which contributes to generating high quality 3D proposals. In the second stage, we propose a density-aware deformable point generation module to accurately complete the shape of the object by generating points based on encoding point probability density and learning deformation. DSaPG achieves highly competitive performance on the KITTI dataset and Waymo Open Dataset.

**Acknowledgements** This work was supported in part by the National Natural Science Foundation of China (Grant No.61902204), in part by the Natural Science Foundation of Shandong Province of China (Grant No.ZR2019BF028).

## References


1. Bhattacharyya, P., Czarnecki, K.: Deformable pv-rcnn: Improving 3d object detection with learned deformations. arXiv preprint arXiv:2008.08766 (2020)
2. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
3. Chen, C., Chen, Z., Zhang, J., Tao, D.: Sasa: Semantics-augmented set abstraction for point-based 3d object detection. In: AAAI. vol. 36, pp. 221–229 (2022)
4. Chen, X., Kundu, K., Zhu, Y., Berneshawi, A.G., Ma, H., Fidler, S., Urtasun, R.: 3d object proposals for accurate object class detection. Adv. Neural Inform. Process. Syst. **28** (2015)
5. Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., Li, H.: Voxel r-cnn: Towards high performance voxel-based 3d object detection. In: AAAI. vol. 35, pp. 1201–1209 (2021)
6. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 3354–3361. IEEE (2012)
7. Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., Bennamoun, M.: Deep learning for 3d point clouds: A survey. IEEE Trans. Pattern Anal. Mach. Intell. **43**(12), 4338–4364 (2020)
8. Hu, J.S., Kuai, T., Waslander, S.L.: Point density-aware voxels for lidar 3d object detection. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 8469–8478 (2022)
9. Huang, T., Liu, Z., Chen, X., Bai, X.: Epnet: Enhancing point features with image semantics for 3d object detection. In: Eur. Conf. Comput. Vis. pp. 35–52. Springer (2020)
10. Imran, S., Liu, X., Morris, D.: Depth completion with twin surface extrapolation at occlusion boundaries. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 2583–2592 (2021)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
12. Koo, I., Lee, I., Kim, S.H., Kim, H.S., Jeon, W.j., Kim, C.: Pg-rcnn: Semantic surface point generation for 3d object detection. In: Int. Conf. Comput. Vis. pp. 18142–18151 (2023)
13. Ku, J., Mozifian, M., Lee, J., Harakeh, A., Waslander, S.L.: Joint 3d proposal generation and object detection from view aggregation. In: IEEE Int. Conf. Intell. Rob. Syst. pp. 1–8. IEEE (2018)
14. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 12697–12705 (2019)
15. Li, X., Ma, T., Hou, Y., Shi, B., Yang, Y., Liu, Y., Wu, X., Chen, Q., Li, Y., Qiao, Y., et al.: Logonet: Towards accurate 3d object detection with local-to-global cross-modal fusion. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 17524–17534 (2023)
16. Li, Z., Yao, Y., Quan, Z., Xie, J., Yang, W.: Spatial information enhancement network for 3d object detection from point cloud. Pattern Recognition **128**, 108684 (2022)
17. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 2980–2988 (2017)
18. Pan, X., Xia, Z., Song, S., Li, L.E., Huang, G.: 3d object detection with pointformer. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 7463–7472 (2021)

19. Pang, S., Morris, D., Radha, H.: Clocs: Camera-lidar object candidates fusion for 3d object detection. In: IEEE Int. Conf. Intell. Rob. Syst. pp. 10386–10393. IEEE (2020)
20. Parzen, E.: On estimation of a probability density function and mode. The annals of mathematical statistics **33**(3), 1065–1076 (1962)
21. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Adv. Neural Inform. Process. Syst. **30** (2017)
22. Shen, Y., Zhang, Y., Wu, Y., Wang, Z., Yang, L., Coleman, S., Kerr, D.: Bsh-det3d: Improving 3d object detection with bev shape heatmap. In: IEEE Int. Conf. Intell. Rob. Syst. pp. 5730–5737. IEEE (2023)
23. Sheng, H., Cai, S., Liu, Y., Deng, B., Huang, J., Hua, X.S., Zhao, M.J.: Improving 3d object detection with channel-wise transformer. In: Int. Conf. Comput. Vis. pp. 2743–2752 (2021)
24. Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., Li, H.: Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 10529–10538 (2020)
25. Shi, S., Wang, X., Li, H.: Pointrcnn: 3d object proposal generation and detection from point cloud. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 770–779 (2019)
26. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al.: Scalability in perception for autonomous driving: Waymo open dataset. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 2446–2454 (2020)
27. Team, O.D.: Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet (2020)
28. Thomas, H., Goulette, F., Deschaud, J.E., Marcotegui, B., LeGall, Y.: Semantic classification of 3d point clouds with multiscale spherical neighborhoods. In: 2018 International conference on 3D vision (3DV). pp. 390–398. IEEE (2018)
29. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Adv. Neural Inform. Process. Syst. **30** (2017)
30. Wu, X., Peng, L., Yang, H., Xie, L., Huang, C., Deng, C., Liu, H., Cai, D.: Sparse fuse dense: Towards high quality 3d detection with depth completion. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 5418–5427 (2022)
31. Xia, Q., Chen, Y., Cai, G., Chen, G., Xie, D., Su, J., Wang, Z.: 3-d hanet: A flexible 3-d heatmap auxiliary network for object detection. IEEE Trans. Geosci. Remote Sens. **61**, 1–13 (2023)
32. Xu, Q., Zhong, Y., Neumann, U.: Behind the curtain: Learning occluded shapes for 3d object detection. In: AAAI. vol. 36, pp. 2893–2901 (2022)
33. Xu, Q., Zhou, Y., Wang, W., Qi, C.R., Anguelov, D.: Spg: Unsupervised domain adaptation for 3d object detection via semantic point generation. In: Int. Conf. Comput. Vis. pp. 15446–15456 (2021)
34. Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors **18**(10), 3337 (2018)
35. Yang, H., Wang, W., Chen, M., Lin, B., He, T., Chen, H., He, X., Ouyang, W.: Pvt-ssd: Single-stage 3d object detector with point-voxel transformer. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 13476–13487 (2023)
36. Yin, T., Zhou, X., Krähenbühl, P.: Multimodal virtual point 3d detection. Adv. Neural Inform. Process. Syst. **34**, 16494–16507 (2021)
37. Yuan, W., Khot, T., Held, D., Mertz, C., Hebert, M.: Pcn: Point completion network. In: 2018 international conference on 3D vision (3DV). pp. 728–737. IEEE (2018)

38. Zhang, Y., Huang, D., Wang, Y.: Pc-rgnn: Point cloud completion and graph neural network for 3d object detection. In: AAAI. vol. 35, pp. 3430–3437 (2021)
39. Zheng, W., Tang, W., Jiang, L., Fu, C.W.: Se-ssd: Self-ensembling single-stage object detector from point cloud. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 14494–14503 (2021)
40. Zhou, C., Zhang, Y., Chen, J., Huang, D.: Octr: Octree-based transformer for 3d object detection. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 5166–5175 (2023)
41. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 4490–4499 (2018)
42. Wang, Y., Chao, W.L., Garg, D., Hariharan, B., Campbell, M., Weinberger, K.Q.: Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 8445–8453 (2019)