ImpDet: Exploring Implicit Fields for 3D Object Detection

Xuelin Qian
Fudan University
xlqian@fudan.edu.cn

Li Wang
Agora
wangli@agora.io

Yi Zhu*
Amazon Inc.
yzhu25@ucmerced.edu

Li Zhang
Fudan University
lizhangfd@fudan.edu.cn

Yanwei Fu†
Fudan University
yanweifu@fudan.edu.cn

Xiangyang Xue
Fudan University
xyxue@fudan.edu.cn

Abstract

Conventional 3D object detection approaches concentrate on bounding boxes representation learning with several parameters, i.e., localization, dimension, and orientation. Despite its popularity and universality, such a straightforward paradigm is sensitive to slight numerical deviations, especially in localization. By exploiting the property that point clouds are naturally captured on the surface of objects along with accurate location and intensity information, we introduce a new perspective that views bounding box regression as an implicit function. This leads to our proposed framework, termed Implicit Detection or ImpDet, which leverages implicit field learning for 3D object detection. Our ImpDet assigns specific values to points in different local 3D spaces, thereby high-quality boundaries can be generated by classifying points inside or outside the boundary. To solve the problem of sparsity on the object surface, we further present a simple yet efficient virtual sampling strategy to not only fill the empty region, but also learn rich semantic features to help refine the boundaries. Extensive experimental results on KITTI and Waymo benchmarks demonstrate the effectiveness and robustness of unifying implicit fields into object detection.

1. Introduction

3D object detection has attracted substantial attention in both academia and industry due to its wide applications in autonomous driving [9, 35, 1], virtual reality [29, 24] and robotics [2]. Although point clouds generated from 3D LiDAR sensors capture precise distance measurements and geometric information of surrounding environments, the irregular, sparse and orderless properties make it hard to be encoded and non-trivial to directly apply 2D detection methods [37].

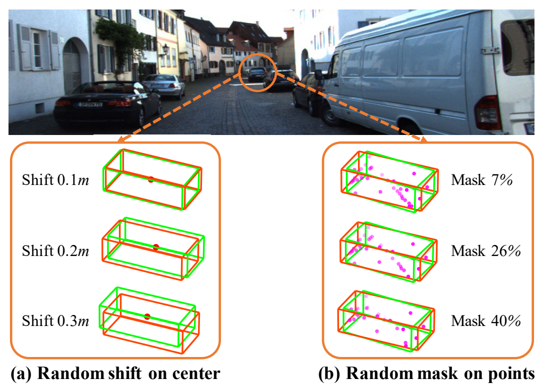

Figure 1. Illustration of different 3D bounding box representations under numerical deviations. Ground truth and deviated boxes are drawn in red and green respectively. **(a) Parameters:** random shift ground-truth centers in range $\pm(0.1, 0.2, 0.3)\ m$ along x/y/z axis. **(b) Implicit fields:** random mask 7/26/40% predicted inside points. We show boxes represented with implicit fields are more robust than conventional parameters when facing some outliers.

Generally, object bounding boxes in 3D scenes are represented with several parameters, such as center localization, box dimension, and orientation. Previous literatures [32, 43, 46, 18, 48, 17] are mostly built upon this representation and utilize convolutional neural networks (CNN) to regress these values. Nevertheless, when there are fewer points on objects caused by object occlusion or other factors for sparsity, directly learning these parameters would be fragile. Even worse, several studies [22, 40] have demonstrated that even minor numerical deviations of these parameters may cause significant performance drop, as shown in Fig. 1 (a). Consequently, this motivates us to consider an open question: *Can we have the more robust 3D bounding box representations for learning?*

Interestingly, recent learning based 3D object modeling

---

*Work done outside Amazon

†Corresponding author. Dr. Fu is also with Fudan ISTBI—ZJNU Algorithm Centre for Brain-inspired Intelligence, Zhejiang Normal University, Jinhua, China

works [4, 25] employ as the nature recipe the implicit fields, which nevertheless has less touched in 3D object detection. Thus to nicely answer the above question, this paper particularly highlights the potential of exploiting implicit fields for 3D object detection. More precisely, implicit field assigns a value (*e.g.*, 0 or 1) to each point in the 3D space; then the object's mesh can be represented by all points assigned to a specific value. Inspired by this, we advocate an implicit way to build bounding boxes for object detection, since point clouds are naturally captured on the surface of objects, with accurate location and intensity information. More precisely, we first classify/assign points into two categories, *i.e.*, inside or outside the box. Then, we can fit a bounding box directly according to these points. As illustrated in Fig. 1 (b), compared with the conventional box representation, such an implicit way can benefit from the best of both worlds: (1) providing high-quality boxes without any pre-defined anchor and being more robust even to some outliers; (2) naturally leveraging implicit fields for multi-task learning, improving features with point-based representation; (3) effectively enhancing the features of inside points and suppressing the outside points according to the implicit assignments.

This paper, for the first time, systematically explores the implicit field learning for 3D object detection, and proposes the ImpDet. As shown in Fig. 2, our ImpDet mainly consists of three key components: (1) candidate shifting, (2) implicit boundary generation and (3) occupant aggregation. Specifically, the *candidate shifting* first shifts and samples points closest to the ground-truth centers as candidates and divides local 3D space surrounding the candidate, in order to relieve the computational pressure caused by implicit functions. Different from previous 3D object detectors explicitly regressing box parameters based on candidates, *implicit boundary generation* adopts the implicit function to fit a high-quality boundary in a local space by assigning implicit values to classify inside and outside points. Furthermore, we come up with a refinement strategy, termed *occupant aggregation*, to refine the boundaries by aggregating features of inside points. Finally, we output the parameter-based representation for detection evaluation.

In summary, our primary contributions are listed as: (1) We for the first time show a perspective of incorporating implicit fields into 3D object detection and propose a framework named ImpDet. Different from previous detectors explicitly regressing box parameters, our ImpDet uses the implicit function to assign values to each point and then fit high-quality boundaries without any pre-defined anchor. (2) We propose a simple yet effective virtual sampling strategy to assist the implicit boundary generation since points in objects may be incompleted due to occlusion or sparsity. With multi-task learning, it can not only fill the empty region, but also learn rich semantic information as auxiliary features. (3) Extensive experiments are conducted on KITTI and Waymo benchmarks to demonstrate the effectiveness and robustness of our ImpDet.

2. Related Work

3D Mesh Representation. There are two commonly used implicit functions, signed distance functions (SDF) [28, 15, 41] and occupancy functions [25, 4, 11, 10]. For SDF, values inside the shape are negative, and then increase to zero as points approach the boundary, and become positive when points are outside the shape. Occupancy functions classify points into two categories, 0 for being inside and 1 for being outside. Previous studies [25, 28, 15, 5, 14] have been proposed to extract features for each point and multi-layer perceptrons are adopted to predict values. Then, methods like Marching Cubes [21] can be used to extract a surface based on both functions. Given the simplicity of binarized representation, we adopt the occupancy functions as an implicit way to build bounding boxes for 3D object detection. Compared to the conventional box representation, our method provides high-quality boxes without any pre-defined anchor and is more robust even with some outliers.

3D Object Detection. Although image-based object detection has achieved remarkable progress, it is far from meeting the requirements for real-world applications, such as autonomous driving. Therefore, researches on 3D data are gradually emerging and flourishing. Most existing 3D object detection methods can be classified in two directions, *i.e.*, point-based and voxel-based. Point-based methods [30, 33, 44, 45] take raw point clouds as input and extract local features with set abstraction. However, the sampling and grouping operations in set abstraction make it time-consuming. For voxel-based approaches [32, 7, 6, 48, 12], they divide point clouds into regular grids so that 3D CNNs can be applied for feature extraction. In this work, we adopt the voxel-based CNN as the backbone in consideration of its efficiency.

3D Object Detection with Segmentation Branch. As another important branch for 3D scene understanding, instance segmentation is gradually applied to assist 3D object detection on account of no cost for annotation. [12, 49] adds another segmentation branch as an auxiliary network to guide the features to be aware of object structures. [50, 32, 42, 33] propose to utilize segmentation results to reweight features or vote the predicted boxes for refinement. [39, 42, 38, 3] obtain segmentation labels/features from 2D space to enhance the point representations in the 3D space. Methods on this line mostly use simple fully-connected layers to build the extra segmentation branch, except that [49] introduces the concept of implicit function. Different from existing works, we propose a novel unified 3D object detection framework, which for the first time directly benefits from the implicit field learning to achieve more precise 3D object detection. Such a framework attempts to assign a special value for each point via implicit functions. Then the network is able to make full use of the assignment results to provide high-quality boundaries and leverage more discriminative inside features (natural by-product) for refinement.

3. Methodology

Figure 2 illustrate the framework of our proposed ImpDet. After obtaining point- and voxel-wise features from the *backbone network* (in Sec. 3.1), the *candidate shifting* module first shifts and samples points as candidate centers in order to partition local 3D space surrounding the candidates (in Sec. 3.2). Next, a high-quality boundary box can be fitted in the local space by the proposed *implicit boundary generation* module (in Sec. 3.3). Finally, we perform the *occupant aggregation* module to refine the boundaries by aggregating the feature of interior points (in Sec. 3.4).

3.1. Backbone Network

We adopt the voxel-based CNN as the backbone due to its efficiency. In order to prevent the loss of geometry information, which is crucial for implicit boundary generation, we simultaneously extract point- and voxel-wise features in one backbone [26, 52]. As the yellow block shown in Fig. 2, we first feed raw point clouds $\mathcal{P} = \{x_i, y_i, z_i, r_i\}_{i=1}^{N}$ into a multi-layer perceptron (MLP) for initial point-wise features $f^{(p_0)}$, where (x_i, y_i, z_i) and r_i mean the coordinates and intensity of point p_i, N is the total number of points. Then, we utilize stacked voxel feature encoding (VFE) layers [52] to obtain initial voxel-wise features $f^{(v_0)}$, where each voxel maintains a feature vector for points fall in it. For point-wise features, $f^{(p_0)}$ is subsequently combined with $f^{(v_0)}$ and fed into another MLP layer to calculate the final features $f^{(point)}$. For voxel-wise features, $f^{(v_0)}$ is followed by several 3D sparse convolution blocks to gradually produce multi-scale features $f^{(v_i)}|_{i=1}^{5}$. Similar to [6], we compress the voxel-wise tensor $f^{(v_5)}$ by concatenating features along z-axis, and further apply a feature pyramid network (FPN) [20]. By fusing output features, we get 2D birds-eye-view (BEV) map features $f^{(bev)} \in \mathbb{R}^{L\times W\times C}$, where L and W represent the length and width of BEV map respectively.

3.2. Candidate Shifting

To reduce the computational costs for the following stages, we first shift points on BEV maps toward the centers of their corresponding ground-truth boxes and then sample those closest to the centers. By doing so, we can apply the implicit boundary generation only to a small number of local 3D space surrounding the shifted points, rather than the entire space.

Concretely, we use a MLP layer to generate the central offset $p^{(ofs)} \in \mathbb{R}^{LW\times 3}$ as well as the feature offset $f^{(ofs)} \in \mathbb{R}^{LW\times C}$ of each pixel on BEV maps. By adding offsets, the candidate centers can be generated as,

$$\begin{gathered} p^{(ctr)} = p^{(ofs)} + p^{(bev)},\ f^{(ctr)} = f^{(ofs)} + f^{(bev)} \\ \left[p^{(ofs)};\ f^{(ofs)}\right] = \mathcal{M}\left(f^{(bev)}\right) \end{gathered} \tag{1}$$

where $p^{(bev)} \in \mathbb{R}^{LW\times 3}$ indicates the coordinates of points on BEV maps, the height is set to 0 by default; $\mathcal{M}$ denotes a MLP layer; $[*;*]$ means the concatenation operation. To measure the quality of the shifted centers for sampling, we choose 3D centerness [37, 44] as metric indicator, which can be written as,

$$s^{(ctrns)} = \sqrt[3]{\frac{\min(x_f, x_b)}{\max(x_f, x_b)} \times \frac{\min(y_l, y_r)}{\max(y_l, y_r)} \times \frac{\min(z_t, z_b)}{\max(z_t, z_b)}} \tag{2}$$

where $(x_f, x_b, y_l, y_r, z_t, z_b)$ denotes the distance from candidate centers to front, back, left, right, top and bottom surfaces of the corresponding boxes they fall in. $s^{(ctrns)}$ is close to 1 when the shifted candidate centers are more accurate, and set as 0 for those outside the bounding boxes. Since $s^{(ctrns)}$ is not accessible during testing, we train a MLP layer attached a sigmoid function to predict its value using candidate center features $f^{(ctr)}$ as input. The predicted centerness is used as confidence score to sample high-quality centers with non-maximum suppression (NMS) by treating each center as $1 \times 1 \times 1$ cube.

3.3. Implicit Boundary Generation

After sampling candidate centers, we perform implicit functions on points in a local 3D space around each center to generate boundaries.

Virtual Sampling Strategy. Given a candidate center $p_k^{(ctr)}$, we get its surrounding local space by drawing a ball with radius r, and randomly select m points from the space. The set of sampled points are defined as $\mathcal{B}_k^p = \mathcal{Q}\left(p_k^{(ctr)}\right) = \left\{p_i \in \mathcal{P} \mid \|p_k^{(ctr)} - p_i\|_2 < r\right\}$, where $card\left(\mathcal{B}_k^p\right) = m$. We assign r a relatively large value to ensure the ball covers as many points as possible. For sampled points in $\mathcal{B}_k^p$, we also gather their features from $f^{(point)}$ and denote them as $\mathcal{B}_k^{f_p}$.

However, along with distance increase, point clouds become sparser and fewer points fall on the object's surface. For distant objects, the point coordinate information may be insufficient to predict boxes. To this end, we present a *virtual sampling strategy* as shown in Fig. 3(a). Concretely, a set of virtual points $\mathcal{V}_k$ are uniformly placed around the candidate center $p_k^{(ctr)}$ with the grid size of $S \times S \times S$ and the interval of (x_s, y_s, z_s). On account of less computation cost, we also randomly sample m virtual points

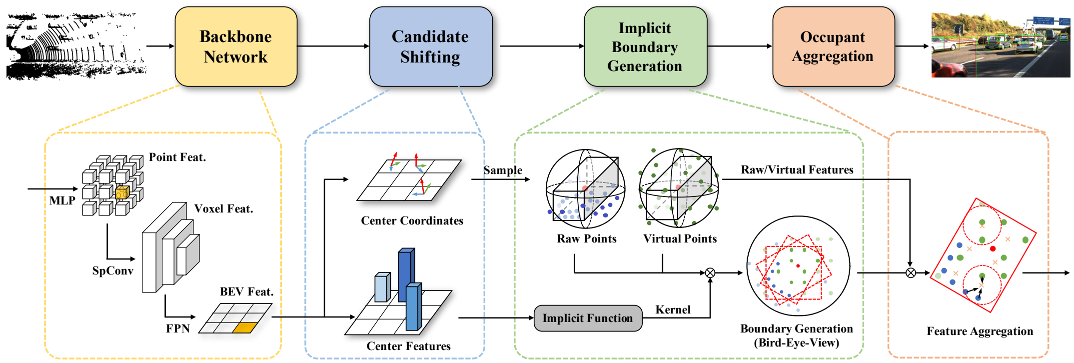

Figure 2. An overview of our proposed ImpDet. The *candidate shifting* module shifts and samples points as candidate centers, and divides local 3D space surrounding the candidates. Next, an implicit function can perform in the local space to fit a high-quality boundary box by assigning implicit values to classify inside and outside points. A virtual sampling strategy is further introduced to not only fill the empty region in the local space but also learn rich semantic features. Finally, we adopt the *occupant aggregation* to refine the boundaries.

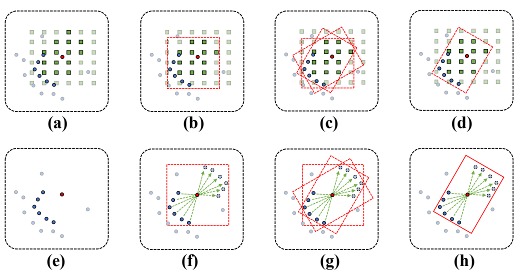

Figure 3. The illustration of implicit boundary generation. Note that (a)-(d) represents the *sampling* strategy, and (e)-(h) means the *centrosymmetry* strategy. The red point denotes a candidate center, blue and green points are sampled raw points and virtual points. Particularly, darker color represents the inside points filtered by a threshold t. Red boxes are generated boundaries with different orientations. We omit virtual points in (e)-(h) for better view.

from $\mathcal{V}_k$. To get features of sampled virtual points, we apply K-Nearest Neighbor to interpolate virtual point features from voxel-wise features $f^{(v_4)}$ because of its larger receptive field and smaller feature dimension. A MLP layer is further employed to encode the interpolated features as well as their coordinates. Similarly, we denote the set of sampled virtual points and their features as $\mathcal{B}_k^v$ and $\mathcal{B}_k^{f_v}$. Experiments in Tab. 8 show that this simple yet effective strategy plays a key role in boundary generation, because it can not only fill the empty region, but also learn rich semantic information.

Implicit Function. Intuitively, whether a sampled point belongs to a box (*i.e.*, inside the box) depends on its corresponding candidate center. The closer the euclidean or feature distances of two points are, the higher probability that they belong to the same box (object). Such a conditional relation inspires us to introduce an implicit function, which produces kernels conditioned on the candidate centers. The kernels further convolve with sampled points, so that the implicit values can be adjusted dynamically. More precisely, the generated kernels are reshaped as parameters of two 1×1 convolution layers with the channel of 16, the relative distance between the candidate center and sampled points are also involved. Take the sampled virtual points $\mathcal{B}_k^v$ as an example, the formulations are defined as,

$$\begin{aligned} \theta_k =& \mathcal{M}\left(\left[f_k^{(ctr)};\, p_k^{(ctr)}\right]\right) \\ \mathcal{H}_k^v =& \, sigmoid\left(\mathcal{O}\left(\left[\mathcal{B}_k^{f_v};\, \mathcal{B}_k^v - p_k^{(ctr)}\right],\, \theta_k\right)\right) \end{aligned} \tag{3}$$

where $\mathcal{H}_k^v \in \mathbb{R}^{1\times m}$ is the assigned implicit values; $\mathcal{O}\left(*;\theta\right)$ means the convolution operation with kernel θ. All implicit values $\mathcal{H}_k$ of candidate center $p_k^{(ctr)}$ is achieved by integrating outputs both from raw points $\mathcal{B}_k^p$ and virtual points $\mathcal{B}_k^v$.

Boundary Generation. By setting a threshold $t = 0.5$, we can easily distinguish the inside and outside points with $\mathcal{H}$. The key challenge now is *how to fit a boundary according to the classified points*. Generally, a regular boundary box in 3D space should include two factors: size and orientation.

For the size, we apply a strategy named *'sampling'* to directly fit a minimum bounding box by using inside points, because (1) point clouds are mostly on the surface of objects; and (2) virtual points can significantly complement point clouds, reducing the sparsity in objects caused by distance or occlusion. Particularly, the center of the boundary can be easily computed, which may be different from the candidate center, as illustrated in Fig. 3(a)-(b). As a contrast, we also introduce an algorithm termed *'centrosymmetry'* to first project the symmetric point of each inside point according to the candidate center[1], and then draw a minimum bounding box with both original and projected points, as shown in Fig. 3(e)-(f). Obviously, this strategy uses the parameter of center and the quality of the boundary depends on the accuracy of the candidate center. Experiments in Tab. 8 clearly suggests that the boundary boxes generated by our proposed implicit fields are more robust.

[1]An object or its surface points are not centrosymmetric but the bounding box is.

For the orientation of objects in 3D object detection, it naturally ranges from 0 to 2π and is usually not parallel to x-y axes. Therefore, it is necessary to fit inside points better by rotating boundary boxes. Concretely, we first narrow down the search space from $[0, 2\pi)$ to $\left[0, \frac{\pi}{2}\right)$ (*i.e.*, convert to the first quadrant) and then divide it into h different angles, thereby producing h different minimum bounding boxes with different angles. As a result, we accumulate the point-to-surface distance for each box and select the minimum one as the final boundary, shown in Fig. 3(c)-(d) and (g)-(h). We assign the rotation of the minimum one $r_a \in \left[0, \frac{\pi}{2}\right)$ as the boundary's orientation. Furthermore, denote the boundary size as (l_a, w_a, h_a), we empirically correct the orientation and expand the range to $[0, \pi)$ by,

$$r_a = \begin{cases} r_a, \; if \; l_a \geq w_a \\ r_a + \dfrac{\pi}{2}, \; otherwise \end{cases} \tag{4}$$

3.4. Occupant Aggregation

As shown in Tab. 6, boundary boxes predicted by our implicit boundary generation stage achieve the competitive recall performance. However, for 3D object detection, it still lacks the classification score and the accurate orientation (which should range from $[0, 2\pi)$). To this end, we reuse the implicit values $\mathcal{H}$ to refine the boundary boxes by aggregating features of inside points and suppressing the effect from outside points. Concretely, we uniformly sample $6 \times 6 \times 6$ grid points within each boundary box. Then, a set abstraction layer is applied to aggregate features of inside points as well as the voxel-wise features $f^{(v_3)}$ and $f^{(v_4)}$ at the location of each grid point. Finally, we concatenate all grid points' features and feed them into a detection head. The head is built with three branches for classification confidence, direction prediction and box refinement respectively. Particularly, each branch has four MLP layers with a channel of 256 and shares the first two layers.

3.5. Loss Function

The overall loss functions are composed of six terms, *i.e.*, the candidate shifting loss, the centerness confidence loss, the implicit function loss, the classification loss, the box refinement loss and the direction prediction loss,

$$\begin{aligned} \mathcal{L} = \lambda_1 \mathcal{L}_{ofs} + & \lambda_2 \mathcal{L}_{ctrns} + \lambda_3 \mathcal{L}_{imp} \\ & + \lambda_4 \mathcal{L}_{cls} + \lambda_5 \mathcal{L}_{box} + \lambda_6 \mathcal{L}_{dir} \end{aligned} \tag{5}$$

where λ_i is the coefficient to balance each term. Similar to [32, 6], we empirically set $\lambda_1 = \lambda_2 = \lambda_4 = 1.0$, $\lambda_3 = \lambda_5 = 2.0$ and $\lambda_6 = 0.2$.

Here, we only describe the first three objectives proposed by us, and leave other common losses to the supplemental materials. Denote the symbols with hat '$\wedge$' as ground truth, each formulation can be defined as,

$$\mathcal{L}_{ofs} = \frac{1}{|\mathfrak{N}_{pixel}|} \sum_{i \in \mathfrak{N}_{pixel}} \mathcal{L}_{smooth_{L1}} \left(p_i^{(ofs)}, \, \widehat{p_i^{(ofs)}} \right) \tag{6}$$

$$\mathcal{L}_{ctrns} = \frac{1}{|\mathfrak{N}_{pixel}|} \sum_{i=1}^{HW} \mathcal{L}_{focal} \left(s_i^{(ctrns)}, \, \widehat{s_i^{(ctrns)}} \right) \tag{7}$$

$$\mathcal{L}_{imp} = \frac{1}{|\mathfrak{N}_{center}|} \sum_{i \in \mathfrak{N}_{center}} \mathcal{L}_{BCE} \left(\mathcal{H}_i, \, \widehat{\mathcal{H}_i} \right) \tag{8}$$

where $\mathfrak{N}_{pixel}$ and $\mathfrak{N}_{center}$ indicate the set of indices of positive pixels/candidate centers if they are inside objects' bounding boxes; '$|\cdot|$' means the cardinality.

4. Experiments

4.1. Dataset and Protocols

To verify the efficacy of our proposed model, we evaluate it on two popular public benchmarks, KITTI 3D detection benchmark [8] and Waymo Open Dataset [35] (WOD).

KITTI Setup. The KITTI dataset contains $7,481$ training frames and $7,518$ testing frames in autonomous driving scenes. Following the standard setting, the training data are divided into a *train* set with $3,712$ samples and a *val* set with $3,769$ samples. We report the mean average precision of 3D object detection ($\mathrm{AP_{3D}}$) and birds-eye-view ($\mathrm{AP_{BEV}}$) on both the *val* set and online *test* server. For fair comparison, the 40 recall positions based metric $\mathrm{AP}|_{\mathrm{R40}}$ is reported on *test* server while $\mathrm{AP}|_{\mathrm{R11}}$ with 11 recall positions is reported on *val* set. On the KITTI benchmark, according to the object size, occlusion ratio, and truncation level, the task can be categorized into 'Easy', 'Mod.' and 'Hard', we report the results in all three tasks, and ranks all methods based on the $\mathrm{AP_{3D}}$ of 'Mod.' setting as in KITTI benchmark. In particular, we focus on the 'Car' category as many recent works [6, 50, 26] and adopt IoU = 0.7 for evaluation. When performing experimental studies on the *val* set, we use the train data for training. For the *test* server, we randomly select 80% samples for training and use the remaining 20% data for validation.

Waymo Setup. We also conduct experiments on the recently released large-scale diverse dataset, Waymo Open Dataset [35], to verify the generalization of our method. The dataset collects RGB images and 3D point clouds from five high-resolution cameras and LiDAR sensors, respectively. It provides annotated 798 training sequences, 202 validation sequences from different scenes, and another 150 test sequences without labels. For evaluation, we adopt the officially released evaluation to calculate the average precision (AP) and average precision weighted by heading

(APH). Specifically, two levels are set according to different LiDAR points included by objects. And three distance (0 - 30m, 30 - 50m, 50m - ∞) to sensor are considered under each level.

4.2. Implementation Details

Network Structure. On KITTI dataset, the detection range is limited to $(0, 70.4)\,m$ for the x axis, $(-40, 40)\,m$ for the y axis, and $(-3, 1)\,m$ for the z axis. Before taken as input of our ImpDet, raw point clouds are divided into regular voxels with voxel size of $(0.05, 0.05, 0.1)\,m$. As for Waymo Open Dataset, the range of point clouds is clipped into $(-75.2, 75.2)\,m$ for both the x and y axes, and $(-2, 4)\,m$ for the z axis. The voxel size is $(0.1, 0.1, 0.15)\,m$. For these two datasets, each voxel randomly samples at most 5 points. Following [6], we adopt 2 convolutional layers and 2 deconvolutional layers as FPN structure. The output feature dimension is 128 and 256 for KITTI and Waymo Open Dataset, respectively. Please refer to the supplementary for more implementation details

Hyper Parameters. After the candidate shifting layer, we select top-512 candidate centers for the following stage. The number of sampled points for implicit fields is set to $m = 256$ with the radius $r = 3.2m$. For virtual sampling strategy, we empirically assign $10 \times 10 \times 10$ as grid size, the interval is $(0.6, 0.6, 0.3)\,m$. During implicit boundary generation, we choose the optimal boundary by enumerating $h = 7$ angles from $\left[0,\ \frac{\pi}{2}\right)$. All these settings are applied to both datasets.

Training. Our framework is built on OpenPCDet codebase [36]. We train the whole model with batch size as 3 and learning rate as 0.01 on 8 Tesla V100 GPUs. Adam optimizer is adopted to train our model for totally 80 and 60 epochs on KITTI and Waymo Open Datasets, respectively. Widely-used data augmentation strategies like flipping, rotation, scaling, translation and sampling are also adopted.

Inference. During inference, we first filter the predicted boxes with 0.3 confidence threshold and then perform NMS with 0.1 IoU threshold to remove the redundant predictions. Final 100 boxes are kept for validation or testing.

4.3. Comparison with State-of-the-Arts

KITTI *test* Split. To verify the efficacy of our ImpDet, we evaluate our model on KITTI online *test* server. As shown in Tab. 1, we report the $\mathrm{AP_{3D}}$ results over three settings. From the table, we can observe that: (1) It is obvious that our model can achieve state-of-the-art performance compared with previous methods on the most concerned 'Mod.' setting. This demonstrates the efficacy of our motivation, which leverages the implicit fields to fit high-quality and robust boundaries without any pre-defined anchors for 3D object detection. (2) We group existing methods in tables based on whether containing a segmentation branch. As can be seen, the performance improvement of our ImpDet over the existing 3D object detectors with segmentation branch is significant. Concretely, we achieve $0.71\%/2.35\%$ higher accuracy on 'Mod.' setting than PV-RCNN [32] and SA-SSD [12]. It proves that our implicit field learning has the great potential capacity in 3D object detection task. (3) We observe that our model gets inferior results on easy cases. One possible reason is that there is a trade-off between memory footprint and accuracy during sampling, which is harsh for easy cases (with thousands of foreground points).

Method	Reference	$\mathrm{AP_{3D}}$		
		Mod.	Easy	Hard
VoxelNet [52]	CVPR 2018	64.17	77.82	57.51
PointPillars [16]	CVPR 2019	74.31	82.58	68.99
SECOND [43]	Sensors 2018	75.96	84.65	68.71
HVPR [27]	CVPR 2021	77.92	86.38	73.04
3DSSD [44]	CVPR 2020	79.57	88.36	74.55
CIA-SSD [47]	AAAI 2021	80.28	89.59	72.87
Voxel R-CNN [6]	AAAI 2021	81.62	**90.90**	77.06
VoTr-TSD [23]	ICCV 2021	82.09	89.90	**79.14**
PDV [23]	CVPR 2022	81.86	90.43	77.36
PointRCNN [33]	CVPR 2019	75.64	86.96	70.70
MMLab-PartA^2 [34]	Arxiv 2019	78.49	87.81	73.51
SERCNN [50]	CVPR 2020	78.96	87.74	74.30
STD [45]	ICCV 2019	79.71	87.95	75.09
SA-SSD [12]	CVPR 2020	79.79	88.75	74.16
PV-RCNN [32]	CVPR 2020	81.43	90.25	76.82
VoxSeT [13]	CVPR 2022	82.06	88.53	77.46
ImpDet(Ours)	-	**82.14**	88.39	76.98

Table 1. Comparison with the state-of-the-art competitors on KITTI *test* split. Methods are grouped into two categories: *w/o* (top) or *w/* (bottom) segmentation branch.

Method	$\mathrm{AP_{3D}/AP_{BEV}}$		
	Mod.	Easy	Hard
VoxelNet [52]	65.46 / 84.81	81.97 / 89.60	62.85 / 78.57
SECOND [43]	76.48 / 87.07	87.43 / 89.96	69.10 / 79.66
PointPillars [16]	76.99 / 87.06	87.29 / 90.07	70.84 / 83.81
3DSSD [44]	79.45 / -	89.71 / -	78.67 / -
CIA-SSD [47]	79.81 / -	90.04 / -	78.80 / -
RangeIoUDet [19]	81.36 / -	89.32 / -	78.29 / -
HVPR [27]	82.05 / -	**91.14** / -	**79.49** / -
Voxel R-CNN [6]	84.52 / -	89.41 / -	78.93 / -
PI-RCNN [42]	78.53 / -	88.27 / -	77.75 / -
PointRCNN [33]	78.63 / 87.89	88.88 / 90.21	77.38 / 85.51
SERCNN [50]	79.21 / 87.53	89.50 / 90.23	78.16 / 86.45
MMLab-PartA^2 [34]	79.47 / 88.61	89.47 / 90.42	78.54 / 87.31
STD [45]	79.80 / 88.50	89.70 / **90.50**	79.30 / 88.10
SA-SSD [12]	79.99 / -	90.15 / -	78.78 / -
VoTr-TSD [23]	82.14 / -	88.39 / -	76.98 / -
PV-RCNN [32]	83.90 / -	- / -	- / -
ImpDet(Ours)	**85.38 / 89.03**	89.91 / **90.50**	79.25 / **88.24**

Table 2. Performance comparison on KITTI *val* split. Methods are grouped into two categories: *w/o* (top) or *w/* (bottom) segmentation branch.

KITTI *val* Split. We also compare our method with com-

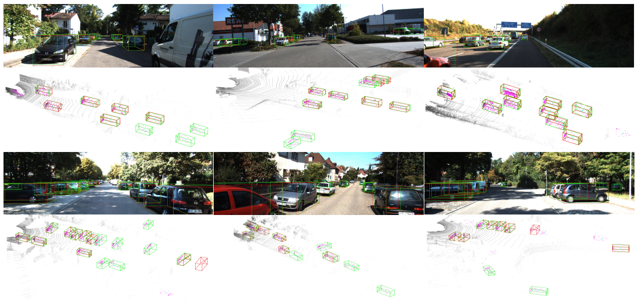

Figure 4. Visualization on KITTI *val* set. The ground truth boxes and our predicted bboxes are drew in red and green. The internal raw points and virtual points predicted by implicit functions are highlighted in purple. Best viewed in color and zoom in.

Method	LEVEL_1(AP/APH)			
	Overall	0 - 30m	30 - 50m	50 - ∞
PointPillars [16]	56.62 / -	81.01 / -	51.75 / -	27.94 / -
MVF [51]	62.93 / -	86.30 / -	60.02 / -	36.02 / -
PV-RCNN [32]	70.30/69.69	91.92/91.34	69.21/68.53	42.17/41.31
PVGNet [26]	74.00 / -	- / -	- / -	- / -
ImpDet(Ours)	**74.38/73.87**	**91.98/91.52**	**72.86/72.29**	**49.13/48.45**

Table 3. Performance comparison on WOD *val* split. We report all distance ranges results on vehicle category.

petitors over the KITTI *val* set. As shown in Tab. 2, our ImpDet can achieve state-of-art performance. Especially, ImpDet outperforms the previous best significantly, *e.g.*, 0.86% over Voxel R-CNN [6] and 1.48% over PV-RCNN [32] on 'Mod.' setting. Similar conclusions are drawn in Tab. 5, which lists the results of other categories, such as pedestrian and cyclist. It suggests that our sampling strategy also works well for small categories. We also show some prediction results in Fig. 4 and we project the 3D bounding boxes detected from LiDAR to RGB images for better visualization. As observed, our ImpDet can produce high-quality 3D bounding boxes via implicit functions in different kinds of scenes. Remarkably, when there are fewer points on objects, our proposed virtual sampling strategy can significantly fill the empty region and thus assist in boundary generation with the assigned implicit values. Our ImpDet may fail on some cases if a candidate center is generated over a large empty area. The sampled virtual points cannot learn enough semantic features from neighboring raw points.

Waymo *val* Split. Table 3 reports the vehicle detection results with 3D AP/APH on validation sequences. Without bells and whistles, our proposed method outperforms all existing state-of-the-art methods on the vehicle category. Improvements on all distance ranges indicate that our methods can robustly represent 3D object bounding boxes containing a various density of points. Especially, a larger gain has been achieved compared with PV-RCNN [32] on distance (50m - ∞), which illustrates that our implicit field learning performs better than directly parameters learning of bounding box with sparse points.

Method	Shift centers / Mask points: c (m) / p (%)				
	0 / 0	0.05 / 7	0.1 / 14	0.2 / 28	0.3 / 42
PV-RCNN	83.2/-	81.6/-	77.3/-	69.5/-	60.0/-
gain (%)	0/-	-1.9/-	-7.1/-	-16.4/-	-27.9/-
Ours	85.4/85.4	85.4/84.3	84.2/83.2	83.0/81.9	79.3/79.6
gain (%)	0/0	**0/-1.3**	**-1.4/-2.6**	**-2.8/-4.1**	**-7.1/-6.8**

Table 4. Comparison of robustness to numerical deviation. For PV-RCNN, random shift centers of proposals in range $\pm c$ m; For ours, random shift candidate centers or mask p % of inside points.

AP_{3D}	PV-RCNN [32]	PointPaint [38]	CT3D [31]	Ours
Mod.	71.95/56.67	71.62/61.67	71.88/55.57	**72.38/64.63**
Easy	88.88/64.26	85.21/69.38	89.01/61.05	**89.25/69.58**
Hard	66.78/51.91	66.98/54.58	67.91/51.10	**69.59/59.14**

Table 5. Performance comparison of Cyclist / Pedestrian categories on KITTI *val* set with R11.

Method	Voxel R-CNN [6]	Ours
Recall (IoU=0.7)	77.10	77.78

Table 6. Comparison of recall using different proposal generation networks. We reproduce performance of competitors.

4.4. Ablation Study

We conduct extensive ablation experiments to explore the effectiveness of different components in our ImpDet and analyze the contributions of implicit fields in 3D object detection. Models are trained on KITTI *train* split and evaluated on the corresponding *val* split. The results of car on the moderate task are reported with R11.

Analysis on Implicit Fields. We first provide the impact analysis of randomly shifting predicted box centers or masking predicted inside points in Tab. 4, to validate the advantages of bringing implicit fields to 3D object detection. Take PV-RCNN [32] as an example, it is sensitive to numerical deviation of centers (drops 28% when shifting centers in range $\pm$0.3m), while our method only drops 7.1% under the same situation. On the other hand, even randomly masking 42% of the internal points, our method shows superior robustness with only 6.8% loss of performance.

Second, we conduct several variants to analyze designs of the implicit function, and adpot both detection metric (AP_{3D}) and segmentation metrics (Pixel Accuracy and IoU). For PA and IoU, we report both results on the categories of 0 and 1. Tab. 7 shows that (1) When the relative distance is not involved in the convolution layer (termed '*w/o* dist.'), the performance drops a lot; (2) By directly using the vanilla convolution layers with sampled point features and relative distances as input (termed '*w/o* cond.'), it gets much worse results. Those suggest the superiority of our design in the implicit function, and the better accuracy of implicit values facilitates much higher performance

Method	PA (1/0)	IoU (1/0)	AP_{3D} (Mod./Easy/Hard)
w/o cond.	71.38/95.91	57.46/91.49	76.21/85.98/68.19
w/o dist.	88.36/97.04	75.38/95.13	84.82/89.46/78.79
Ours	89.82/97.20	77.28/95.53	85.38/89.91/79.25

Table 7. Ablation study of the design in implicit function. '*w/o* dist.' denotes the relative distance is not involved. '*w/o* cond.' denotes the vanilla convolution layers with sampled point features and relative distances as inputs.

Method	*centrosymmetry*		*sampling*	
	p	p + v	p	p + v
AP_{3D} (%)	79.43	84.33	70.65	85.38
AP_{BEV} (%)	87.81	88.48	79.27	89.03

Table 8. Result comparisons of different boundary generation strategies with both AP_{3D} and AP_{BEV}. 'p' and 'v' denote raw points and virtual points.

of object detection.

Analysis of Boundary Generation. In Tab. 8, we first compare the performance with two boundary generation strategies, *i.e.*, *sampling* and *centrosymmetry*. 'point+virtual' means we utilize both sampled raw points and virtual points for boundary generation. First of all, we observe that additionally using virtual points can boost the performance by a large margin of 4.9% and 14.73% on both strategies. It clearly demonstrates the effectiveness of our proposed virtual sampling strategy in boundary generation, which can significantly fill empty regions in objects. Second, the *sampling* strategy only with raw points achieves the worst results of $70.65/79.27\%$ on $AP_{3D/BEV}$, we explain that too sparse point clouds may make the implicit fields inapplicable since there is no enough points to fit a boundary. Third, our *sampling* strategy outperforms the *centrosymmetry* by 1.05% and 0.55% on 3D and BEV accuracy. Recall the difference between these two strategies, the *centrosymmetry* strategy additional needs the predicted center to perform the centrosymmetric projection for each point, thereby it strongly shows the robustness of our proposed implicit fields, even with some outliers.

We also discuss the values of h in Tab. 9. As expected, if the division of angles is too large, it cannot fit a boundary well, resulting in a drop of detection performance. On the contrary, the more angles we divide, the less the accuracy gains and the higher the computation costs. We choose the optimal value when the model achieves the best performance, *i.e.*, $h = 7$.

Finally, to validate the quality of the predicted boundary boxes via implicit fields, we compute the recall rate with the ground-truth boxes. To be fair, we show the recall rates for all methods with top-100 proposals on the car category over all difficulty levels. As shown in Tab. 6, only with the supervision of center coordinates, our introduced implicit field achieves a competitive result with 77.78% recall rate, outperforming Voxel R-CNN [6]. It indicates that our implicit learning can robustly fit high-quality bounding boxes.

h	3	5	7	9
AP_{3D} (%)	84.12	85.22	85.38	85.28

Table 9. Result comparisons with different number of angle partition on AP_{3D}. The best performance is achieved when $h = 7$.

$f^{(v_3)}$	$f^{(v_4)}$	$\mathcal{B}^{f_p}$	$\mathcal{B}^{f_v}$	$\mathcal{H}^p\mathcal{B}^{f_p}$	$\mathcal{H}^v\mathcal{B}^{f_v}$	AP_{3D} (%)
✓	✓					84.29
✓	✓	✓				85.05
✓	✓		✓			84.85
✓	✓			✓		85.10
✓	✓				✓	85.16
✓	✓			✓	✓	**85.38**

Table 10. Ablation study of different feature choices in occupant aggregation.

Analysis of Occupant Aggregation. In order to explore the contribution of our occupant aggregation module, we do experiments with different combinations of voxel-wise features ($f^{(v_3)}$ and $f^{(v_4)}$), sampled point features ($\mathcal{B}^{f_p}$), sampled virtual point features ($\mathcal{B}^{f_v}$) and those with implicit values ($\mathcal{H}^p\mathcal{B}^{f_p}$ and $\mathcal{H}^v\mathcal{B}^{f_v}$). As shown in Tab. 10, the comparisons between $\mathcal{B}^{f_p}$ and $\mathcal{H}^p\mathcal{B}^{f_p}$ or $\mathcal{B}^{f_v}$ and $\mathcal{H}^v\mathcal{B}^{f_v}$ consistently proves that the implicit values can effectively enhance the features of inside points, suggesting a solid advantage of incorporating implicit fields into 3D object detection. Interestingly, we observe that the virtual point features contribute more to the performance when $\mathcal{H}^v$ is applied (the second/third row and the fourth/fifth row). One possible explanation is that virtual points contain both rich semantic features and confused geometric features since they are randomly sampled in the 3D space. With the cooperation of implicit values, we can successfully suppress the distracting information. Moreover, the result from the last row demonstrates the complementarity of raw points and virtual points.

5. Conclusion and Discussion

In this paper, we introduce a new perspective to represent 3D bounding boxes with implicit fields. Our proposed framework, dubbed Implicit Detection or ImpDet, leverages the implicit function to generate high-quality boundaries by classifying points into two categories, *i.e.*, inside or outside the boundary. A virtual sampling strategy is consequently designed to fill the empty regions around objects, making the boundary generation more robust. Our approach achieves comparable results to the current state-of-the-art methods both on KITTI and WOD benchmarks.

ImpDet also encounters some challenges, including the trade-off between the computation cost and accuracy when sampling points in the local 3D space, and the results on easy objects. Nevertheless, we believe that this work can be inspiring and helpful for encouraging more researches.

Acknowledgements. This work is supported by China Postdoctoral Science Foundation (2022M710746).

References

[1] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 11618–11628. IEEE, 2020.

[2] Teng Hooi Chan, Henrik Hesse, and Song Guang Ho. Lidar-based 3d slam for indoor mapping. In *2021 7th International Conference on Control, Automation and Robotics (ICCAR)*, pages 285–289. IEEE, 2021.

[3] Xia Chen, Jianren Wang, David Held, and Martial Hebert. Panonet3d: Combining semantic and geometric understanding for lidar point cloud detection. In *2020 International Conference on 3D Vision (3DV)*, pages 753–761. IEEE, 2020.

[4] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[5] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6970–6981, 2020.

[6] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1201–1209, 2021.

[7] Liang Du, Xiaoqing Ye, Xiao Tan, Jianfeng Feng, Zhenbo Xu, Errui Ding, and Shilei Wen. 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*, pages 13329–13338, 2020.

[8] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[9] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 3354–3361. IEEE Computer Society, 2012.

[10] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4857–4866, 2020.

[11] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7154–7164, 2019.

[12] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11873–11882, 2020.

[13] Jordan SK Hu, Tianshu Kuai, and Steven L Waslander. Point density-aware voxels for lidar 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8469–8478, 2022.

[14] Moritz Ibing, Isaak Lim, and Leif Kobbelt. 3d shape generation with grid-based implicit functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13559–13568, 2021.

[15] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6001–6010, 2020.

[16] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[17] Jiale Li, Yu Sun, Shujie Luo, Ziqi Zhu, Hang Dai, Andrey S Krylov, Yong Ding, and Ling Shao. P2v-rcnn: Point to voxel feature learning for 3d object detection from point clouds. *IEEE Access*, 9:98249–98260, 2021.

[18] Ziyu Li, Yuncong Yao, Zhibin Quan, Wankou Yang, and Jin Xie. Sienet: Spatial information enhancement network for 3d object detection from point cloud. *arXiv preprint arXiv:2103.15396*, 2021.

[19] Zhidong Liang, Zehan Zhang, Ming Zhang, Xian Zhao, and Shiliang Pu. Rangeioudet: Range image based real-time 3d object detector optimized by intersection over union. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7140–7149, 2021.

[20] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[21] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[22] Xinzhu Ma, Yinmin Zhang, Dan Xu, Dongzhan Zhou, Shuai Yi, Haojie Li, and Wanli Ouyang. Delving into localization errors for monocular 3d object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[23] Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3164–3173, 2021.

[24] Florian Mathis, John H Williamson, Kami Vaniea, and Mohamed Khamis. Fast and secure authentication in virtual reality using coordinated 3d manipulation and pointing. *ACM Transactions on Computer-Human Interaction (ToCHI)*, 28(1):1–44, 2021.

[25] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[26] Zhenwei Miao, Jikai Chen, Hongyu Pan, Ruiwen Zhang, Kaixuan Liu, Peihan Hao, Jun Zhu, Yang Wang, and Xin Zhan. Pvgnet: A bottom-up one-stage 3d object detector with integrated multi-level features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3279–3288, 2021.

[27] Jongyoun Noh, Sanghoon Lee, and Bumsub Ham. Hvpr: Hybrid voxel-point representation for single-stage 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14605–14614, 2021.

[28] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[29] Youngmin Park, Vincent Lepetit, and Woontack Woo. Multiple 3d object tracking for augmented reality. In *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 117–120. IEEE, 2008.

[30] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[31] Hualian Sheng, Sijia Cai, Yuan Liu, Bing Deng, Jianqiang Huang, Xian-Sheng Hua, and Min-Jian Zhao. Improving 3d object detection with channel-wise transformer. *arXiv preprint arXiv:2108.10723*, 2021.

[32] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.

[33] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019.

[34] Shaoshuai Shi, Zhe Wang, Xiaogang Wang, and Hongsheng Li. Part-aˆ 2 net: 3d part-aware and aggregation neural network for object detection from point cloud. *arXiv preprint arXiv:1907.03670*, 2(3), 2019.

[35] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[36] OD Team et al. Openpcdet: An open-source toolbox for 3d object detection from point clouds, 2020.

[37] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9627–9636, 2019.

[38] Sourabh Vora, Alex H Lang, Bassam Helou, and Oscar Beijbom. Pointpainting: Sequential fusion for 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4604–4612, 2020.

[39] Chunwei Wang, Chao Ma, Ming Zhu, and Xiaokang Yang. Pointaugmenting: Cross-modal augmentation for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11794–11803, 2021.

[40] Li Wang, Li Zhang, Yi Zhu, Zhi Zhang, Tong He, Mu Li, and Xiangyang Xue. Progressive coordinate transforms for monocular 3d object detection. *Adv. Neural Inform. Process. Syst.*, 2021.

[41] Francis Williams, Zan Gojcic, Sameh Khamis, Denis Zorin, Joan Bruna, Sanja Fidler, and Or Litany. Neural fields as learnable kernels for 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18500–18510, 2022.

[42] Liang Xie, Chao Xiang, Zhengxu Yu, Guodong Xu, Zheng Yang, Deng Cai, and Xiaofei He. Pi-rcnn: An efficient multi-sensor 3d object detector with point-based attentive cont-conv fusion module. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12460–12467, 2020.

[43] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[44] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 11037–11045. IEEE, 2020.

[45] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1951–1960, 2019.

[46] Jin Hyeok Yoo, Yecheol Kim, Jisong Kim, and Jun Won Choi. 3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736. Springer, 2020.

[47] Wu Zheng, Weiliang Tang, Sijin Chen, Li Jiang, and Chi-Wing Fu. Cia-ssd: Confident iou-aware single-stage object detector from point cloud. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3555–3562, 2021.

[48] Wu Zheng, Weiliang Tang, Li Jiang, and Chi-Wing Fu. Se-ssd: Self-ensembling single-stage object detector from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14494–14503, 2021.

[49] Yuanxin Zhong, Minghan Zhu, and Huei Peng. Vin: Voxel-based implicit network for joint 3d object detection and segmentation for lidars. *arXiv preprint arXiv:2107.02980*, 2021.

[50] Dingfu Zhou, Jin Fang, Xibin Song, Liu Liu, Junbo Yin, Yuchao Dai, Hongdong Li, and Ruigang Yang. Joint 3d instance segmentation and object detection for autonomous

driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1839–1849, 2020.

[51] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3d object detection in lidar point clouds. In *Conference on Robot Learning*, pages 923–932. PMLR, 2020.

[52] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.