International Journal of Computer Vision (2023) 131:531–551
https://doi.org/10.1007/s11263-022-01710-9

PV-RCNN++: Point-Voxel Feature Set Abstraction With Local Vector Representation for 3D Object Detection

Shaoshuai Shi[1,2] · Li Jiang[1,2] · Jiajun Deng[3] · Zhe Wang[4] · Chaoxu Guo[4] · Jianping Shi[4] · Xiaogang Wang[1] · Hongsheng Li[1]

Received: 3 April 2022 / Accepted: 31 October 2022 / Published online: 24 November 2022

Abstract
3D object detection is receiving increasing attention from both industry and academia thanks to its wide applications in various fields. In this paper, we propose Point-Voxel Region-based Convolution Neural Networks (PV-RCNNs) for 3D object detection on point clouds. First, we propose a novel 3D detector, PV-RCNN, which boosts the 3D detection performance by deeply integrating the feature learning of both point-based set abstraction and voxel-based sparse convolution through two novel steps, *i.e.*, the voxel-to-keypoint scene encoding and the keypoint-to-grid RoI feature abstraction. Second, we propose an advanced framework, PV-RCNN++, for more efficient and accurate 3D object detection. It consists of two major improvements: sectorized proposal-centric sampling for efficiently producing more representative keypoints, and VectorPool aggregation for better aggregating local point features with much less resource consumption. With these two strategies, our PV-RCNN++ is about 3× faster than PV-RCNN, while also achieving better performance. The experiments demonstrate that our proposed PV-RCNN++ framework achieves state-of-the-art 3D detection performance on the large-scale and highly-competitive Waymo Open Dataset with 10 FPS inference speed on the detection range of $150m \times 150m$.

Keywords 3D object Detection · Point clouds · LiDAR · Autonomous driving · Sparse convolution

1 Introduction

3D object detection on point clouds aims to localize and recognize 3D objects from a set of 3D points, which is a fundamental task of 3D scene understanding and is widely-adopted in lots of real-world applications like autonomous driving, intelligent traffic system and robotics. Compared to 2D detection methods on images (Girshick, 2015; Ren et al., 2015; Liu et al., 2016; Redmon et al., 2016; Lin et al., 2017, 2018), the sparsity and irregularity of point clouds make it challenging to directly apply 2D detection techniques to 3D detection on point clouds.

To tackle these challenges, most of existing 3D detection methods (Chen et al., 2017; Zhou and Tuzel, 2018; Yang et al., 2018b; Lang et al., 2019; Yan et al., 2018) transform the points into regular voxels that can be processed with conventional 2D/3D convolutional neural networks and well-studied 2D detection heads. But the voxelization operation inevitably brings quantization errors, thus degrading their localization

Communicated by Jean-Sébastien Franco.

✉ Shaoshuai Shi
sshi@mpi-inf.mpg.de

Li Jiang
lijiang@mpi-inf.mpg.de

Jiajun Deng
jiajun.deng@sydney.edu.au

Zhe Wang
wangzhe@sensetime.com

Chaoxu Guo
cxguo@sensetime.com

Jianping Shi
shijianping@sensetime.com

Xiaogang Wang
xgwang@ee.cuhk.com.hk

Hongsheng Li
hsli@ee.cuhk.com.hk

[1] The Chinese University of Hong Kong, Hong Kong SAR, China

[2] Max Planck Institute for Informatics, Saarbrücken, Germany

[3] The University of Sydney, Sydney, Australia

[4] SenseTime Research, Shanghai, China

accuracy. In contrast, the point-based methods (Qi et al., 2018; Shi et al., 2019; Wang and Jia, 2019) naturally preserve accurate point locations in feature extraction but are generally computationally-intensive on handling large-scale points. There are also some existing approaches (Chen et al., 2019b; Li et al., 2021) that simply combine these two strategies by adopting the voxel-based methods for feature extraction and 3D proposal generation in the first stage, but introducing the raw point representation in a second stage to compensate the quantization errors for fine-grained proposal refinement. However, this simple stacked combination ignores deep fusion of their basic operators (*e.g.*, sparse convolution (Graham et al., 2018) and set abstraction (Qi et al., 2017b)) and can not fully explore the feature intertwining of both strategies to take the best of both worlds.

Therefore, we propose a unified framework, namely, Point-Voxel Region-based Convolutional Neural Networks (PV-RCNNs), to take the best of both voxel and point representations by deeply integrating the feature learning strategies from both of them. The principle lies in the fact that the voxel-based strategy can more efficiently encode multi-scale features and generate high-quality 3D proposals from large-scale point clouds, while the point-based strategy can preserve accurate location information with flexible receptive fields for fine-grained proposal refinement. We demonstrate that our proposed point-voxel intertwining framework can effectively improve the 3D detection performance by deeply fusing the feature learning of both point and voxel representations.

Firstly, we introduce our initial 3D detection framework, PV-RCNN, which is a two-stage 3D detector on point clouds. It consists of two novel steps for point-voxel feature aggregation. The first step is voxel-to-keypoint scene encoding, where a 3D voxel CNN with sparse convolutions is adopted for feature learning and proposal generation. The multi-scale voxel features are then summarized into a small set of keypoints by point-based set abstraction, where the keypoints with accurate point locations are sampled by farthest point sampling from the raw points. The second step is keypoint-to-grid RoI feature abstraction, where we propose RoI-grid pooling module to aggregate the above keypoint features back to regular RoI grids of each proposal. It encodes multi-scale contextual information to form regular grid features for proposal refinement. These two steps establish feature intertwining between point-based set abstraction and voxel-based sparse convolution, which have been experimentally evidenced to improve the model representative ability as well as the detection performance.

Secondly, we propose an advanced two-stage detection network, PV-RCNN++, on top of PV-RCNN, for achieving more accurate, efficient and practical 3D object detection. The improvements of PV-RCNN++ lie in two aspects. The first aspect is a novel sectorized proposal-centric keypoint sampling strategy, where we concentrate the limited number of keypoints in and around the 3D proposals to encode more effective scene features. Meanwhile, by considering radial distribution of LiDAR points, we propose to conduct point sampling parallelly in different sectors, which accelerates keypoint sampling process, while also ensuring uniform distribution of keypoints. Our proposed keypoint sampling strategy is much faster and more effective than vanilla farthest point sampling that has a quadratic complexity. The efficiency of the whole framework is thus greatly improved, which is particularly important for large-scale 3D scenes with millions of points. The second aspect is a novel local feature aggregation module, VectorPool aggregation, for more effective and efficient local feature encoding on point clouds. We argue that the relative point locations in a local region are robust, effective and discriminative features for describing local geometry. We propose to split 3D local space into regular and compact sub-voxels, the features of which are sequentially concatenated to form a hyper feature vector. The sub-voxel features in different locations are encoded with separate kernel weights to generate position-sensitive local features. In this way, different local location information is encoded with different feature channels in the hyper feature vector. Compared with set abstraction, our VectorPool aggregation can efficiently handle a very large number of centric points due to the compact local feature representation. Equipped with VectorPool aggregation in both voxel-based backbone and RoI-grid pooling module, our PV-RCNN++ is more memory-friendly and faster than previous counterparts with comparable or even better performance, which helps in establishing a practical 3D detector for resource-limited devices.

In a nutshell, our contributions are three-fold: 1) Our PV-RCNN adopts two novel strategies, voxel-to-keypoint scene encoding and keypoint-to-grid RoI feature abstraction, to deeply integrate the advantages of both point-based and voxel-based feature learning strategies. 2) Our PV-RCNN++ takes a step in more practical 3D detection system with better performance, less resource consumption and faster running speed. This is enabled by our proposed sectorized proposal-centric keypoint sampling to obtain more representative keypoints with faster speed, and is also powered by our novel VectorPool aggregation for achieving local aggregation on a large number of central points with less resource consumption and more effective representation. (3) Our proposed 3D detectors surpass all published methods with remarkable margins on the challenging large-scale Waymo Open Dataset. In particular, our PV-RCNN++ achieves state-of-the-art results with 10 FPS inference speed for $150m \times 150m$ detection range. The source code is available at https://github.com/open-mmlab/OpenPCDet.

2 Related Work

3D Object Detection with 2D images Image-based 3D detection aims to estimate 3D bounding boxes from a monocular image or stereo images. Mono3D (Chen et al., 2016) generates 3D proposals with ground-plane assumption, which are scored by exploiting semantic knowledge from images. The following works (Mousavian et al., 2017; Li et al., 2019a) incorporate the relations between 2D and 3D boxes as geometric constraint. M3D-RPN (Brazil and Liu, 2019) introduces a 3D region proposal network with depth-aware convolutions. (Chabot et al., 2017; Murthy et al., 2017; Manhardt et al., 2019) predict 3D boxes based on a wire-frame template obtained from CAD models. RTM3D (Li et al., 2020) performs coarse keypoints detection to localize 3d objects. On the stereo side, Stereo R-CNN (Li et al., 2019b; Qian et al., 2020) capitalizes on a stereo RPN to associate proposals from left and right images. DSGN (Chen et al., 2020) introduces differentiable 3D volume to learn depth information and semantic cues in an end-to-end optimized pipelines. -LiDARs (Wang et al., 2019a; Qian et al., 2020; You et al., 2020) propose to covert the image pixels to artificial point clouds, where the LiDAR-based detectors can operate on them for 3D box estimation. These image-based 3D detection methods suffer from inaccurate depth estimation and can only generate coarse 3D bounding boxes.

Recently, in addition to image-based 3D detection from monocular image or stereo images, a comprehensive scene understanding with surrounding cameras has drawn a lot of attention, where the well-known bird's-eye-view (BEV) representation is generally adopted for better feature fusion from multiple surrounding images. LSS (Philion and Fidler, 2020) and CaDDN (Reading et al., 2021) predicts depth distribution to "lift" the 2D image features to a BEV feature map for 3D detection. Their follow-up works (Huang et al., 2021; Huang and Huang, 2022; Li et al., 2022a; Xie et al., 2022) learn a depth-based implicit projection to project image features to BEV space. Some other works also explore transformer structure to project image features from perspective view to BEV space via cross attention, such as DETR3D (Wang et al., 2022), PETR (Liu et al., 2022a, b), BEVFormer (Li et al., 2022b), PolarFormer (Jiang et al., 2022), etc. Although these works greatly improve the performance of image-based 3D detection by projecting multi-view images to a unified BEV space, the inaccurate depth estimation is still the main challenge for image-based 3D detection.

Representation Learning on Point Clouds Recently representation learning on point clouds has drawn lots of attention for improving the performance of 3D classification and segmentation (Qi et al., 2017a, b; Wang et al., 2019b; Huang et al., 2018; Zhao et al., 2019; Li et al., 2018; Su et al., 2018; Wu et al., 2019; Jaritz et al., 2019; Jiang et al., 2019; Thomas et al., 2019; Choy et al., 2019; Liu et al., 2020). In terms of 3D detection, previous methods generally project the points to regular 2D pixels (Chen et al., 2017; Yang et al., 2018b) or 3D voxels (Zhou and Tuzel, 2018; Chen et al., 2019b) for processing them with 2D/3D CNN. Sparse convolution (Graham et al., 2018) is adopted in (Yan et al., 2018; Shi et al., 2020b) to effectively learn sparse voxel features from point clouds. Qi et al. (Qi et al., 2017a, b) proposes PointNet to directly learn point features from raw points, where set abstraction enables flexible receptive fields by setting different search radii. (Liu et al., 2019) combines both voxel CNN and point multi-layer percetron network for efficient point feature learning. In comparison, our PV-RCNNs take advantages from both voxel-based (*i.e.*, 3D sparse convolution) and point-based (*i.e.*, set abstraction) strategies to enable both high-quality 3D proposal generation with dense BEV detection heads and flexible receptive fields in 3D space for improving 3D detection performance.

3D Object Detection with Point Clouds Most of existing 3D detection approaches can be roughly classified into three categories in terms of different strategies to learn point cloud features, *i.e.*, the voxel-based methods, the point-based methods as well as the methods combining both points and voxels.

The voxel-based methods project point clouds to regular grids to tackle the irregular data format problem. MV3D (Chen et al., 2017) projects points to 2D bird view grids and places lots of predefined 3D anchors for generating 3D boxes, and the following works (Ku et al., 2018; Liang et al., 2018, 2019; Vora et al., 2020; Yoo et al., 2020; Huang et al., 2020) develop better strategies for multi-sensor fusion. (Yang et al., 2018b, a; Lang et al., 2019) introduce more efficient frameworks with bird-eye view representation while (Ye et al., 2020) proposes to fuse grid features of multiple scales. MVF (Zhou et al., 2020) integrates 2D features from bird-eye view and perspective view before projecting points into pillar representations (Lang et al., 2019). Some other works (Song and Xiao, 2016; Zhou and Tuzel, 2018) divide the points into 3D voxels to be processed by 3D CNN. 3D sparse convolution (Graham et al., 2018) is introduced by (Yan et al., 2018) for efficient 3D voxel processing. (Kuang et al., 2020) utilizes multiple detection heads for detecting 3D objects with different scales. In addition, (Wang et al., 2020; Chen et al., 2019a) predicts bounding box parameters following anchor-free paradigm. These grid-based methods are generally efficient for accurate 3D proposal generation but the receptive fields are constraint by the kernel size of 2D/3D convolutions.

The point-based methods directly detect 3D objects from raw points. F-PointNet (Qi et al., 2018) applies PointNet (Qi et al., 2017a, b) for 3D detection from the cropped points based on 2D image boxes. PointRCNN (Shi et al., 2019) generates 3D proposals directly from raw points by only taking 3D points. (Qi et al., 2019) proposes hough voting strategy for feature grouping. 3DSSD (Yang et al., 2020)

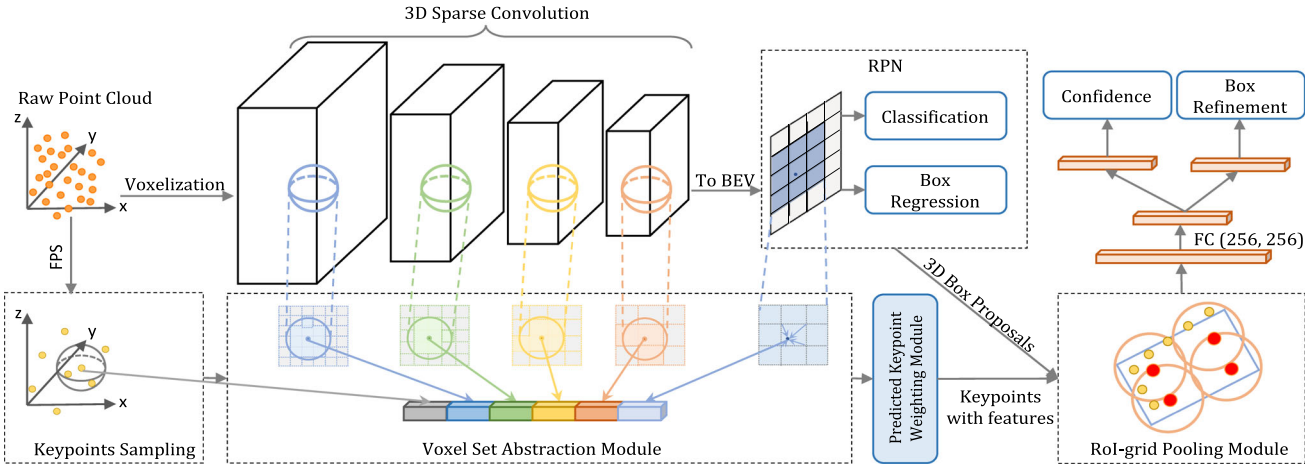

Fig. 1 The overall architecture of our proposed PV-RCNN. The raw point clouds are first voxelized to feed into the 3D sparse convolution based encoder to learn multi-scale semantic features and generate 3D object proposals. Then the learned voxel-wise feature volumes at multiple neural layers are summarized into a small set of key points via the novel voxel set abstraction module. Finally the keypoint features are aggregated to the RoI-grid points to learn proposal specific features for fine-grained proposal refinement and confidence prediction

introduces hybrid feature-distance based farthest point sampling on raw points. These point-based methods are mostly based on PointNet series, especially set abstraction (Qi et al., 2017b), which enables flexible receptive fields for point cloud feature learning. However, it is challenging to extend these point-based methods to large-scale point clouds since they generally consume much more memory/computation resources than the above voxel-based methods.

There are also some works that utilize both the point-based and voxel-based representations. STD (Yang et al., 2019) transforms point-wise features to dense voxels for refining the proposals. Fast Point R-CNN (Chen et al., 2019b) fuses the deep voxel features with raw points for 3D detection. Part-A2-Net (Shi et al., 2020b) aggregates the point-wise part locations by the voxel-based RoI-aware pooling to improve 3D detection performance. However, these methods generally simply transform features between two representations and do not fuse the deeper features from the specific basic operations of these two representations. In contrast, our PV-RCNN frameworks explore on how to deeply aggregate features by learning with both point-based (*i.e.*, set abstraction) and voxel-based (*i.e.*, sparse convolution) feature learning modules to boost the 3D detection performance.

3 PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection

Most of state-of-the-art 3D detectors (Shi et al., 2020b; Yin et al., 2021; Sheng et al., 2021) adopt 3D sparse convolution for learning representative features from irregular points thanks to its efficiency and effectiveness on handling large-scale point clouds. However, 3D sparse convolution network suffers from losing accurate point information due to the indispensable voxelization process. In contrast, the point-based approaches (Qi et al., 2017a, b) naturally preserve accurate point locations and can capture rich context information with flexible receptive fields, where the accurate point locations are essential for estimating accurate 3D bounding boxes.

In this section, we briefly review our initial 3D detection framework, PV-RCNN (Shi et al., 2020a), for 3D object detection from point clouds. It deeply integrates the voxel-based sparse convolution and point-based set abstraction operations to take the best of both worlds.

As shown in Fig. 1, PV-RCNN is a two-stage 3D detection framework that adopts a 3D voxel CNN with sparse convolution as the backbone for efficient feature encoding and proposal generation (Sec. 3.1), and then we generate the proposal-aligned features for predicting accurate 3D bounding boxes by intertwining point-voxel features through two novel steps, which are voxel-to-keypoint scene encoding (Sec. 3.2) and keypoint-to-grid RoI feature abstraction (Sec. 3.3).

3.1 Voxel Feature Encoding and Proposal Generation

In order to handle 3D object detection on the large-scale point clouds, we adopt the 3D voxel CNN with sparse convolution (Graham et al., 2018) as the backbone network to generate initial 3D proposals.

The input points $\mathcal{P}$ are first divided into small voxels with spatial resolution of $L \times W \times H$, where non-empty voxel features are directly calculated by averaging the coordinates

of inside points. The network utilizes a series of 3D sparse convolution to gradually convert the points into feature volumes with $1\times$, $2\times$, $4\times$, $8\times$ downsampled sizes. We follow (Yan et al., 2018) to stack the 3D feature volumes along Z axis to obtain the $\frac{L}{8} \times \frac{W}{8}$ bird-view feature maps, which can be naturally combined with the 2D detection heads (Liu et al., 2016; Yin et al., 2021) for high quality 3D proposal generation.

It is worth noting that the sparse feature volumes at each level can be viewed as a set of sparse voxel-wise feature vectors, and these multi-scale semantic features are considered as the input of our following voxel-to-keypoint scene encoding step.

3.2 Voxel-to-Keypoint Scene Encoding

Given the multi-scale scene features, we propose to summarize these features into a small number of keypoints, which serve as the courier to propagate features from the above 3D voxel CNN to the refinement network.

Keypoint Sampling We simply adopt farthest point sampling algorithm as in (Qi et al., 2017b) to sample a small number of keypoints $\mathcal{K} = \{p_i \mid p_i \in \mathbb{R}^3\}_{i=1}^{n}$ from the raw points $\mathcal{P}$, where n is a hyper-parameter (*e.g.*, n=4,096 for Waymo Open Dataset (Sun et al., 2020)). It encourages that the keypoints are uniformly distributed around non-empty voxels and can be representative to the overall scene.

Voxel Set Abstraction Module To aggregate the multi-scale semantic features from 3D feature volumes to the keypoints, we propose *Voxel Set Abstraction* (VSA) module. The set abstraction (Qi et al., 2017b) is adopted for aggregating voxel-wise feature volumes. The key difference is that the surrounding local points are now regular voxel-wise semantic features from 3D voxel CNN, instead of the neighboring raw points with features learned by PointNet (Qi et al., 2017a).

Specifically, we denote the number of non-empty voxels in the k-th level of 3D voxel CNN as N_k, and the voxel-wise features and 3D coordinates are denoted as $\mathcal{F}^{(l_k)} = \left\{[f_i^{(l_k)}, v_i^{(l_k)}] \mid f_i^{(l_k)} \in \mathbb{R}^C, v_i^{(l_k)} \in \mathbb{R}^3\right\}_{i=1}^{N_k}$, where C indicates the number of feature dimensions.

For each keypoint $p_i \in \mathcal{K}$, to retrieve the set of neighboring voxel-wise feature vectors, we first identify its neighboring non-empty voxels at the k-th level within a radius r_k as

$$S_i^{(l_k)} = \left\{ \left[f_j^{(l_k)}, v_j^{(l_k)} - p_i \right] \,\middle|\, \left\| v_j^{(l_k)} - p_i \right\| < r_k \right\}, \tag{1}$$

where $[f_j^{(l_k)}, v_j^{(l_k)}] \in \mathcal{F}^{(l_k)}$, and the local relative position $v_j^{(l_k)} - p_i$ is concatenated to indicate the relative location of $f_j^{(l_k)}$ in this local area. The features within neighboring set $S_i^{(l_k)}$ are then aggregated by a PointNet-block (Qi et al., 2017a) to generate keypoint feature as

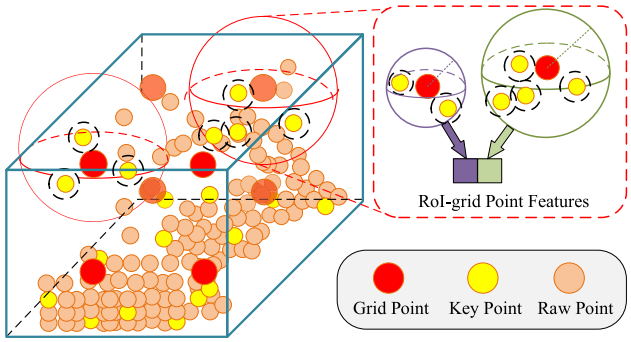

Fig. 2 Illustration of RoI-grid pooling module. Rich context information of each 3D RoI is aggregated by set abstraction operation with multiple receptive fields

$$f_i^{(\mathrm{pv}_k)} = \max\left\{\mathrm{SharedMLP}\left(S_i^{(l_k)}\right)\right\}, \tag{2}$$

where SharedMLP$(\cdot)$ denotes a shared multi-layer perceptron (MLP) network to encode voxel-wise features and relative locations, and max$\{\cdot\}$ conducts permutation invariant feature aggregation to map diverse number of neighboring voxel features to a single keypoint feature $f_i^{(\mathrm{pv}_k)}$. Here multiple radii are utilized to capture richer contextual information.

The above voxel feature aggregation is performed at the outputs of different levels of 3D voxel CNN, and the aggregated features from different scales are concatenated to obtain the multi-scale semantic feature for keypoint p_i as

$$f_i^{(p)} = \mathrm{Concat}\left(\left\{f_i^{(\mathrm{pv}_k)}\right\}_{k=1}^{4}, f_i^{(\mathrm{raw})}, f_i^{(\mathrm{bev})}\right), \tag{3}$$

where $i \in \{1, \ldots, n\}$, and $k \in \{1, \ldots, 4\}$ indicates that the keypoint features are aggregated from four-level voxel-wise features of 3D voxel CNN. Note that the keypoint features are further enriched with two extra information sources, where the raw point features $f_i^{(\mathrm{raw})}$ are aggregated as in Eq. (2) to partially make up the quantization loss of point voxelization, while 2D bird-view features $f_i^{(\mathrm{bev})}$ are obtained by bilinear interpolation on the $8\times$ downsampled 2D feature maps to achieve larger receptive fields along the height axis.

Predicted Keypoint Weighting Intuitively, the keypoints belonging to the foreground objects should contribute more to the proposal refinement, while the keypoints from the background regions should contribute less. Hence, we propose a *Predicted Keypoint Weighting* (PKW) module to re-weight the keypoint features with extra supervisions from point segmentation as

$$f_i^{(p)} = \mathrm{MLP}(f_i^{(p)}) \cdot f_i^{(p)}, \tag{4}$$

where MLP$(\cdot)$ is a three-layer MLP network with a sigmoid function to predict foreground confidence. It is trained with focal loss (Lin et al., 2018) by the default parameters and the segmentation labels can be directly generated from the 3D box annotations as in (Shi et al., 2019). Note that this PKW module is optional for our framework as it only leads small gains (see Table. 7).

The keypoint features $\mathcal{F} = \left\{ f_i^{(p)} \right\}_{i=1}^{n}$ not only incorporate multi-scale semantic features from the 3D voxel backbone network, but also naturally preserve accurate location information through its 3D keypoint coordinates $\mathcal{K} = \{p_i\}_{i=1}^{n}$, which provides strong capacity of preserving 3D structural information of the entire scene for the following fine-grained proposal refinement.

3.3 Keypoint-to-Grid RoI Feature Abstraction

Given the aggregated keypoint features and their 3d coordinates, in this step, we propose keypoint-to-grid RoI feature abstraction to generate accurate proposal-aligned features for fine-grained proposal refinement.

RoI-grid Pooling via Set Abstraction We propose the *RoI-grid pooling* module to aggregate the keypoint features to the RoI-grid points by adopting multi-scale local feature grouping. For each given 3D proposal, we uniformly sample $6 \times 6 \times 6$ grid points according to the 3D proposal box, which are then flattened and denoted as $\mathcal{G} = \{g_i\}_{i=1}^{6\times6\times6=216}$. To aggregate the features of keypoints to the RoI grid points, we firstly identify the neighboring keypoints of a grid point g_i as

$$\Psi = \left\{ \left[f_j^{(p)}, p_j - g_i \right] \middle| \left\| p_j - g_i \right\| < r^{(g)} \right\}, \tag{5}$$

where $p_j \in \mathcal{K}$ and $f_j^{(p)} \in \mathcal{F}$. We concatenate $p_j - g_i$ to indicate the local relative location within the ball of radius $r^{(g)}$. Then we adopt the similar process with Eq. (2) to summarize the neighboring keypoint feature set Ψ to obtain the features of grid point g_i as

$$f_i^{(g)} = \max \left\{ \text{SharedMLP}\left(\Psi\right) \right\}. \tag{6}$$

Note that we set multiple radii $r^{(g)}$ and aggregate keypoint features with different receptive fields, which are concatenated together for capturing richer multi-scale contextual information. Next, all RoI-grid features $\{f_i^{(g)}\}_{i=1}^{216}$ of the same RoI can be vectorized and transformed by a two-layer MLP with 256 feature dimensions to represent the overall features of this proposal box.

Our proposed RoI-grid pooling operation can aggregate much richer contextual information than the previous RoI-pooling/RoI-align operation (Shi et al., 2019; Yang et al., 2019; Shi et al., 2020b). It is because a single keypoint can contribute to multiple RoI-grid points due to the overlapped neighboring balls of RoI-grid points, and their receptive fields are even beyond the RoI boundaries by capturing the contextual keypoint features outside the 3D RoI. In contrast, the previous state-of-the-art methods either simply average all point-wise features within the proposal as the RoI feature (Shi et al., 2019), or pool many uninformative zeros as the RoI features because of the very sparse point-wise features (Shi et al., 2020b; Yang et al., 2019).

Proposal Refinement Given the above RoI-aligned features, the refinement network learns to predict the size and location (*i.e.* center, size and orientation) residuals relative to the 3D proposal box. Two sibling sub-networks are employed for confidence prediction and proposal refinement. Each sub-network consists of a two-layer MLP and a linear prediction layer. We follow (Shi et al., 2020b) to conduct the IoU-based confidence prediction. The binary cross-entropy loss is adopted to optimize the IoU branch while the box residuals are optimized with smooth-L1 loss.

4 PV-RCNN++: Faster and Better 3D Detection with PV-RCNN Framework

Although our proposed PV-RCNN 3D detection framework achieves state-of-the-art performance (Shi et al., 2020a), it suffers from the efficiency problem when handling large-scale point clouds. To make PV-RCNN framework more practical for real-world applications, we propose an advanced 3D detection framework, *i.e.,* PV-RCNN++, for more accurate and efficient 3D object detection with less resource consumption.

As shown in Fig. 3, we present two novel modules to improve both the accuracy and efficiency of PV-RCNN framework. One is sectorized proposal-centric strategy for much faster and better keypoint sampling, and the other one is VectorPool aggregation module for more effective and efficient local feature aggregation from large-scale point clouds. These two modules are adopted to replace their counterparts in PV-RCNN, which are introduced in Sec. 4.1 and Sec. 4.2, respectively.

4.1 Sectorized Proposal-Centric Sampling for Efficient and Representative Keypoint Sampling

The keypoint sampling is critical for PV-RCNN framework as keypoints bridge the point-voxel representations and heavily influence the performance of proposal refinement. However, previous keypoint sampling algorithm (see Sec. 3.2) has two main drawbacks. (i) Farthest point sampling is time-consuming due to its quadratic complexity, which hinders the training and inference speed of PV-RCNN, espe-

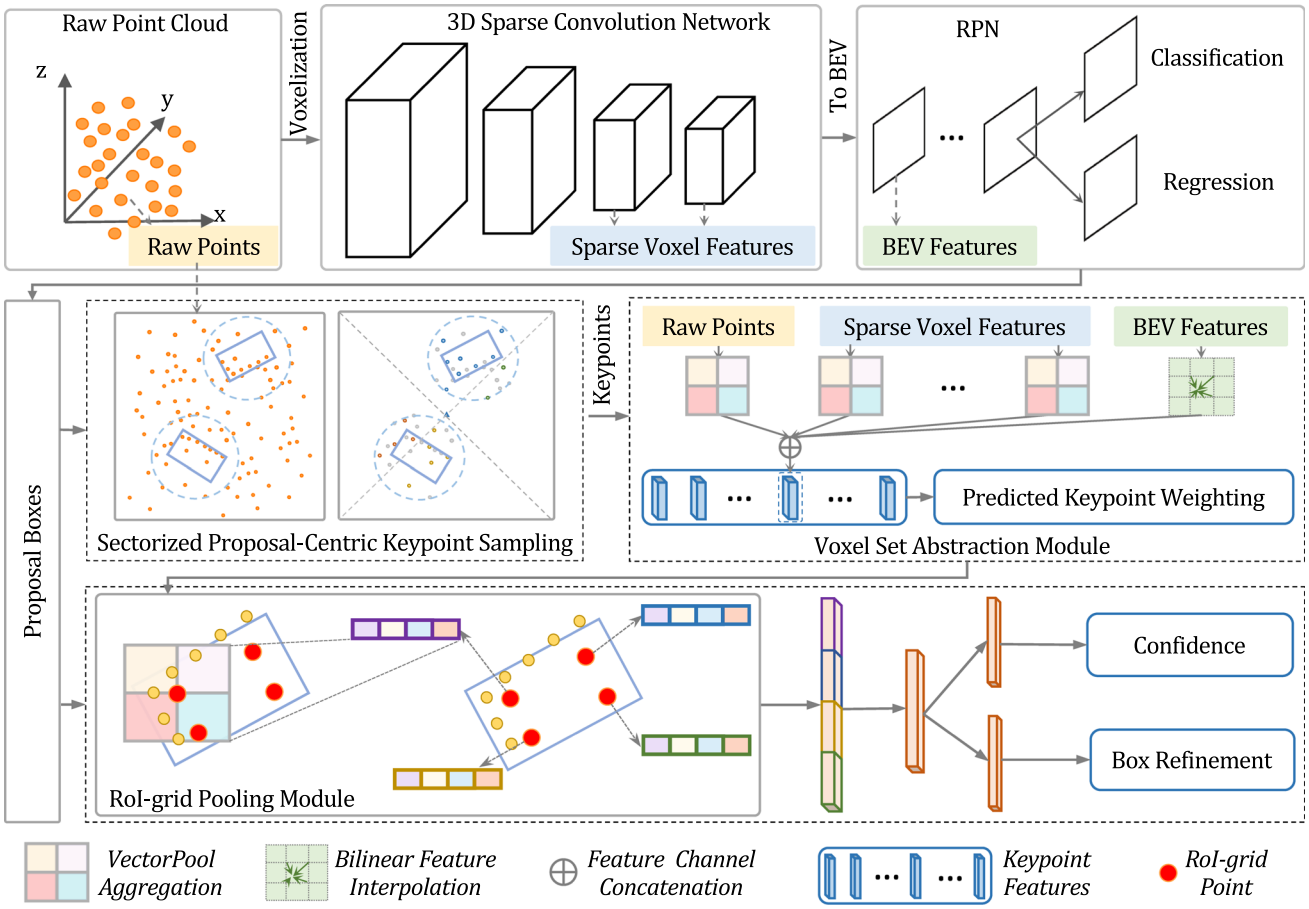

Fig. 3 The overall architecture of our proposed PV-RCNN++ framework. We propose sectorized proposal-centric keypoint sampling module to concentrate keypoints to the neighborhoods of 3D proposals while it can also accelerate the process with sectorized farthest point sampling. Moreover, our proposed VectorPool module is utilized in both voxel set abstraction module and RoI-grid pooling module to improve the local feature aggregation and save memory/computation resources

cially for keypoint sampling on large-scale point clouds. (ii) It would generate a large number of background keypoints that are generally useless to proposal refinement, since only the keypoints around proposals can be retrieved by RoI-grid pooling module.

To mitigate these drawbacks, we propose the *Sectorized Proposal-Centric* (SPC) keypoint sampling to uniformly sample keypoints from more concentrated neighboring regions of proposals, while also being much faster than the vanilla farthest point sampling algorithm. It mainly consists of two novel steps, which are the proposal-centric filtering and the sectorized sampling, which are illustrated in the following paragraphs.

Proposal-Centric Filtering To better concentrate the keypoints on the more important areas and also reduce the complexity of the next sampling process, we first adopt the proposal-centric filtering step.

Specifically, we denote a number of N_p 3D proposals as $\mathcal{D} = \{[c_i, d_i] \mid c_i \in \mathbb{R}^3, d_i \in \mathbb{R}^3\}_{i=1}^{N_p}$, where c_i and d_i are the center and size of each proposal box, respectively. We restrict the keypoint candidates $\mathcal{P}'$ to the neighboring point sets of all proposals as

$$\mathcal{P}' = \left\{ p_i \,\middle|\, \|p_i - c_j\| < \frac{1}{2} \cdot \max\left(d_j\right) + r^{(s)} \right\}, \tag{7}$$

where $[c_j, d_j] \in \mathcal{D}$, $p_i \in \mathcal{P}$ indicates the raw point, and $\max(\cdot)$ obtains the maximum length of 3D box size. $r^{(s)}$ is a hyperparameter indicating the maximum extended radius of the proposals. Through this proposal-centric filtering process, the number of candidate keypoints for sampling is greatly reduced from $|\mathcal{P}|$ to $|\mathcal{P}'|$. For instance, for the Waymo Open Dataset (Sun et al., 2020), generally $\mathcal{P}$ is about $180k$ and $\mathcal{P}'$ can be smaller than $90k$ in most cases (the exact point number depends on the number of proposal boxes in each scene).

Hence, this step not only reduces the time complexity of the follow-up keypoint sampling, but also concentrates the

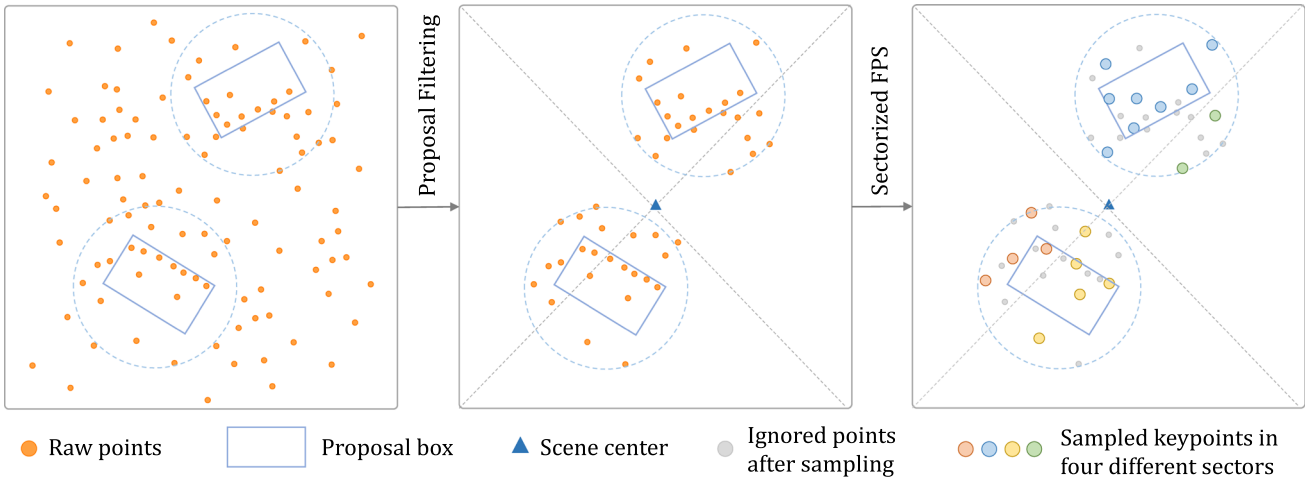

Fig. 4 Illustration of Sectorized Proposal-Centric (SPC) keypoint sampling. It contains two steps, where the first proposal filtering step concentrates the limited number of keypoints to the neighborhoods of proposals, and the following sectorized-FPS step divides the whole scene into several sectors for accelerating the keypoint sampling process while also keeping the keypoints uniformly distributed

limited number of keypoints to better encode the neighboring regions of the proposals.

Sectorized Keypoint Sampling To further parallelize the keypoint sampling process for acceleration, as shown in Fig. 4, we propose the sectorized keypoint sampling strategy, which takes advantage of radial distribution of the LiDAR points to better parallelize and accelerate the keypoint sampling process.

Specifically, we divide proposal-centric point set $\mathcal{P}'$ into s sectors centered at the scene center, and the point set of k-th sector can be represented as

$$S'_k = \left\{ p_i \,\middle|\, \left\lfloor \left(\arctan\left(p_i^y, p_i^x\right) + \pi\right) \cdot \frac{s}{2\pi} \right\rfloor = k - 1 \right\}, \quad (8)$$

where $k \in \{1, \ldots, s\}$, $p_i = (p_i^x, p_i^y, p_i^z) \in \mathcal{P}'$, and $\arctan(p_i^y, p_i^x) \in (-\pi, \pi]$ indicates the angle between the positive X axis and the ray ended with (p_i^x, p_i^y) in terms of the bird's eye view.

Through this process, we divide the task of sampling n keypoints into s subtasks of sampling local keypoints, where k-th sector samples $\left\lfloor \frac{|S'_k|}{|\mathcal{P}'|} \times n \right\rfloor$ keypoints from the point set S'_k. These subtasks are eligible to be executed in parallel on GPUs, while the scale of keypoint sampling (*i.e.*, time complexity) is further reduced from $|\mathcal{P}'|$ to $\max_{k\in\{1,\ldots,s\}} |S'_k|$. Note that we adopt farthest point sampling in each subtask since both the qualitative and quantitative experiments in Sec.?? demonstrate that farthest point sampling can generate more uniformly distributed keypoints to better cover the whole regions, which is critical for the final detection performance.

It is worth noting that our sector-based group partition can roughly produce similar number of points in each group by considering radial distribution of the points generated by LiDAR sensors, which is essential to speed up the keypoint sampling since the overall running time depends on the group with the most points.

Therefore, our proposed keypoint sampling algorithm greatly reduces the scale of keypoint sampling from $|\mathcal{P}|$ to the much smaller $\max_{k\in\{1,\ldots,s\}} |S'_k|$, which not only effectively accelerates the keypoint sampling process, but also increases the capability of keypoint feature representation by concentrating the keypoints to the more important neighboring regions of 3D proposals.

Although the proposed sectorized keypoint sampling is tailored for LiDAR sensors, the main idea behind it, that is, conducting FPS in spatial groups to speed up the operation, is also effective with other types of sensors. It should be noted that the point group generation should be based on spatially partitioning to keep the overall uniform distribution. As shown in Table 8, randomly dividing the points into groups, while ensuring a balance in the number of points between groups, harms the model performance.

4.2 Local Vector Representation for Structure-Preserved Local Feature Learning

The local feature aggregation of point clouds plays an important role in PV-RCNN framework as it is the fundamental operation to deeply integrate the point-voxel features in both voxel set abstraction and RoI-grid pooling modules. However, we observe that set abstraction (see Eqs. (2) and (6)) in PV-RCNN framework can be extremely time- and resource-consuming on large-scale point clouds, since it applies several shared-parameter MLP layers on the point-wise features of each local point separately. Moreover, the

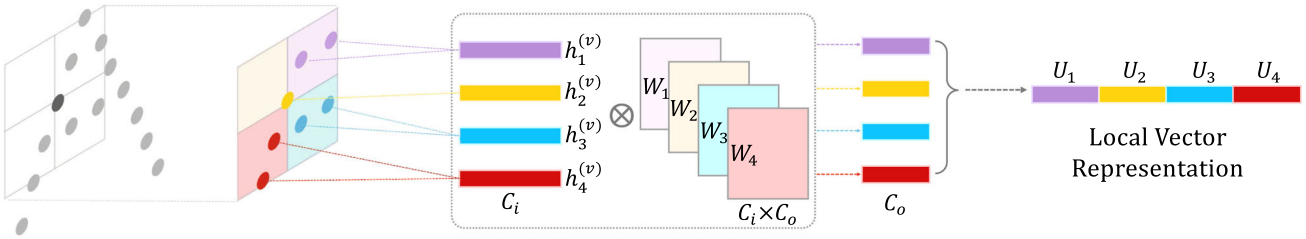

Fig. 5 Illustration of VectorPool aggregation for local feature aggregation from point clouds. The local 3D space around a center point is divided into dense sub-voxels, where the inside point-wise features are generated by interpolating from three nearest neighbors. The features of each volume are encoded with position-specific kernels to generate position-sensitive features, which are sequentially concatenated to generate the local vector representation to explicitly encode the spatial structure information. Note that the notations in the figure follows the same denition as in Sec. 4.2, except that we simplify the channel definition of the kernels as: $C_i = 9 + C_{\text{in}}$, $C_o = C_{\text{mid}}$

max-pooling operation in set abstraction abandons the spatial distribution information of local points and harms the representation capability of locally aggregated features from point clouds.

Therefore, in this section, we propose VectorPool aggregation module for local feature aggregation on the large-scale point clouds, which can better preserve spatial point distribution of local neighborhoods and also costs less memory/computation resources than the commonly-used set abstraction. Our PV-RCNN++ framework adopts it as a basic module to enable more effective and efficient 3D object detection.

Problem Statement The VectorPool aggregation module aims to generate the informative local features for N target center points (denoted as $\mathcal{Q} = \{q_k \mid q_k \in \mathbb{R}^3\}_{k=1}^{N}$) by learning from M given support points and their features (denoted as $\mathcal{I} = \{[h_i, a_i] \mid h_i \in \mathbb{R}^{C_{\text{in}}}, a_i \in \mathbb{R}^3\}_{i=1}^{M}$), where C_{in} is the input feature channels and we are going to extract N local point-wise features with C_{out} channels for each point in $\mathcal{Q}$.

VectorPool Aggregation on Point Clouds In our proposed VectorPool aggregation module, we propose to generate position-sensitive local features by encoding different spatial regions with separate kernel weights and separate feature channels, which are then concatenated as a single vector representation to explicitly represent the spatial structures of local point features.

Specifically, given a target center point q_k, we first identify the support points that are within its cubic neighboring region, which can be represented as

$$\mathcal{Y}_k = \left\{ [h_j, a_j] \,\middle|\, \max(a_j - q_k) < 2 \times \delta \right\}, \tag{9}$$

where $[h_j, a_j] \in \mathcal{I}$, δ is the half length of this cubic space, and $\max(a_j - q_k) \in \mathbb{R}$ obtains the maximum axis-aligned value of this 3D distance. Note that we double the half length (*e.g.*, $2 \times \delta$) of the original cubic space to contain more neighboring points for local feature aggregation of this target point.

To generate position-sensitive features for this local cubic neighborhood centered at q_k, we split its neighboring cubic space into $n_x \times n_y \times n_z$ small local sub-voxels. Inspired by (Qi et al., 2017b), we utilize the inverse distance weighted strategy to interpolate the features of the t^{th} sub-voxel by considering its three nearest neighbors from $\mathcal{Y}_k$, where $t \in \{1, \ldots, n_x \times n_y \times n_z\}$ indicating the index of each sub-voxel and we denote its corresponding sub-voxel center as $v_t \in \mathbb{R}^3$. Then we can generate the features of the t^{th} sub-voxel as

$$h_t^{(v)} = \frac{\sum_{i \in \mathcal{G}_t} (w_i \cdot h_i)}{\sum_{i \in \mathcal{G}_t} w_i}, \quad w_i = (||a_i - v_t||)^{-1}, \tag{10}$$

where $[h_i, a_i] \in \mathcal{Y}_k$, $\mathcal{G}_t$ is the index set indicating the three nearest neighbors (*i.e.*, $(|\mathcal{G}_t| = 3)$) of v_t in neighboring set $\mathcal{Y}_k$. The results $h_t^{(v)}$ encode the local features of the specific t^{th} local sub-voxel in this local cubic.

There are also two other alternative strategies to aggregate the features of local sub-voxels by simply averaging the features within each sub-voxel or by randomly choosing one point within each sub-voxel. Both of them generate lots of empty features in the empty sub-voxels, which may degrade the performance. In contrast, our interpolation based strategy can generate more effective features even on empty local voxels.

Those features in different local sub-voxels may represent very different local features. Hence, instead of encoding the local features with a shared-parameter MLP as in (Qi et al., 2017b), we propose to encode different local sub-voxels with separate local kernel weights for capturing position-sensitive features as

$$U_t = \text{Concat}\left(\{a_i - v_t\}_{i \in \mathcal{G}_t}, \; h_t^{(v)} \right) \times W_t, \tag{11}$$

where $\{a_i - v_t\}_{i \in \mathcal{G}_t} \in \mathbb{R}^{(3 \times 3 = 9)}$ indicates the relative positions of its three nearest neighbors, Concat$(\cdot)$ is the concatenation operation to fuse the relative position and features. $W_t \in \mathbb{R}^{(9 + C_{\text{in}}) \times C_{\text{mid}}}$ is the learnable kernel weights for encod-

ing the specific features of t^{th} local sub-voxel with feature channel C_{mid}, and different positions have different learnable kernel weights for encoding position-sensitive local features.

Finally, we directly sort the local sub-voxel features U_t according to their spatial order along each of 3D axis, and their features are sequentially concatenated to generate the final local vector representation as

$$\mathcal{U} = \text{MLP}\left(\text{Concat}\left(U_1, U_2, \ldots, U_{n_x \times n_y \times n_z}\right)\right), \tag{12}$$

where $\mathcal{U} \in \mathbb{R}^{C_{\text{out}}}$. The inner sequential concatenation encodes the structure-preserved local features by simply assigning the features of different locations to their corresponding feature channels, which naturally preserves the spatial structures of local features in the neighboring space centered at q_k, This local vector representation would be finally processed with several MLPs to encode the local features to C_{out} feature channels for the follow-up processing.

It is worth noting that our VectorPool aggregation module can also be combined with channel reduction technique as in (Sun et al., 2018) to further reduce the computation/memory resources by summarizing the input feature channels before conducting VectorPool aggregation, and we provide the detailed ablation experiments in Sec. 5.3 and Table 10.

Compared with set abstraction, our VectorPool aggregation can greatly reduce the needed computations and memory resources by adopting channel summation and utilizing the proposed local vector representation before MLPs. Moreover, instead of conducting max-pooling on local point-wise features as in set abstraction, our proposed local vector representation can encode the position-sensitive features with different feature channels, to provide more effective representation for local feature learning.

VectorPool Aggregation on PV-RCNN++ Our proposed VectorPool aggregation is integrated in PV-RCNN++ detection framework, to replace set abstraction in both voxel set abstraction module and RoI-grid pooling module. Thanks to our VectorPool aggregation operation, the experiments demonstrate that our PV-RCNN++ not only consumes much less memory and computation resources than PV-RCNN framework, but also achieves better 3D detection performance.

5 Experiments

In this section, we first introduce our experimental setup and implementation details in Sec. 5.1. Then we present the main results of our PV-RCNN/PV-RCNN++ frameworks and compare with state-of-the-art methods in Sec. 5.2. Finally, we conduct extensive ablation experiments and analysis to investigate the individual components of our proposed frameworks in Sec. 5.3.

5.1 Experimental Setup

Datasets and Evaluation Metrics[1]. We evaluate our methods on the Waymo Open Dataset (Sun et al., 2020), which is currently the largest dataset with LiDAR point clouds for 3D object detection of autonomous driving scenarios. There are totally 798 training sequences with around 160k LiDAR samples, 202 validation sequences with 40k LiDAR samples and 150 testing sequences with 30k LiDAR samples.

The evaluation metrics are calculated by the official evaluation tools, where the mean average precision (mAP) and the mAP weighted by heading (mAPH) are used for evaluation. The 3D IoU threshold is set as 0.7 for vehicle detection and 0.5 for pedestrian/cyclist detection. The comparison is conducted in two difficulty levels, where the LEVEL 1 denotes the ground-truth objects with at least 5 inside points while the LEVEL 2 denotes the ground-truth objects with at least 1 inside points. As utilized by the official Waymo evaluation server, the mAPH of LEVEL 2 difficulty is the most important evaluate metric for all experiments.

Network Architecture For the PV-RCNN framework, the 3D voxel CNN has four levels (see Fig. 1) with feature dimensions 16, 32, 64, 64, respectively. Their two neighboring radii r_k of each level in the voxel set abstraction module are set as (0.4m, 0.8m), (0.8m, 1.2m), (1.2m, 2.4m), (2.4m, 4.8m), and the neighborhood radii of set abstraction for raw points are (0.4m, 0.8m). For the proposed RoI-grid pooling operation, we uniformly sample $6 \times 6 \times 6$ grid points in each 3D proposal and the two neighboring radii $\tilde{r}$ of each grid point are (0.8m, 1.6m).

For the PV-RCNN++ framework, we set the maximum extended radius $r^{(s)} = 1.6m$ for proposal-centric filtering, and each scene is split into 6 sectors for parallel keypoint sampling. Two VectorPool aggregation operations are adopted to the $4\times$ and $8\times$ feature volumes of voxel set abstraction module with the half length $\delta = (1.2m, 2.4m)$ and $\delta = (2.4m, 4.8m)$ respectively, and both of them have local voxels $n_x = n_y = n_z = 3$. The VectorPool aggregation on raw points is set with $n_x = n_y = n_z = 2$. For RoI-grid pooling, we adopt the same number of RoI-grid points ($6 \times 6 \times 6$) as PV-RCNN, and the utilized VectorPool aggregation has local voxels $n_x = n_y = n_z = 3$, and half length $\delta = (0.8m, 1.6m)$.

Training and Inference Details Both two frameworks are trained from scratch in an end-to-end manner with ADAM optimizer, learning rate 0.01 and cosine annealing learning

[1] The datasets generated during and/or analysed during the current study are available on the official websites of Waymo Open Dataset (Sun et al., 2020) and KITTI dataset (Geiger et al., 2012).

Table 1 Performance comparison on the validation set of Waymo Open Dataset. ∗: re-implemented by (Zhou et al., 2020). †: re-implemented by ourselves. ‡: performance reported in the official open-source codebase of (Yin et al., 2021). "2f", "3f", "16f": the performance is achieved by using multiple point cloud frames

Method	Veh. (LEVEL 1)		Veh. (LEVEL 2)		Ped. (LEVEL 1)		Ped. (LEVEL 2)		Cyc. (LEVEL 1)		Cyc. (LEVEL 2)	
	mAP	mAPH	mAP	mAPH	mAP	mAPH	mAP	mAPH	mAP	mAPH	mAP	mAPH
†SECOND (Yan et al., 2018)	72.27	71.69	63.85	63.33	68.70	58.18	60.72	51.31	60.62	59.28	58.34	57.05
StarNet (Ngiam et al., 2019)	53.70	–	–	–	66.80	–	–	–	–	–	–	–
*PointPillar (Lang et al., 2019)	56.62	–	–	–	59.25	–	–	–	–	–	–	–
MVF (Zhou et al., 2020)	62.93	–	–	–	65.33	–	–	–	–	–	–	–
Pillar-based (Wang et al., 2020)	69.80	–	–	–	72.51	–	–	–	–	–	–	–
†Part-A2-Net (Shi et al., 2020b)	77.05	76.51	68.47	67.97	75.24	66.87	66.18	58.62	68.60	67.36	66.13	64.93
LiDAR R-CNN (Li et al., 2021)	76.0	75.5	68.3	67.9	71.2	58.7	63.1	51.7	68.6	66.9	66.1	64.4
CenterPoint (Yin et al., 2021)	76.7	76.2	68.8	68.3	79.0	72.9	71.0	65.3	–	–	–	–
‡CenterPoint (Yin et al., 2021)	–	–	–	67.9	–	–	–	65.6	–	–	–	68.6
RSN (Sun et al., 2021)	75.1	74.6	66.0	65.5	77.8	72.7	68.3	63.7	–	–	–	–
VoTr-TSD (Mao et al., 2021)	74.95	74.25	65.91	65.29	–	–	–	–	–	–	–	–
CT3D (Sheng et al., 2021)	76.30	–	69.05	–	–	–	–	–	–	–	–	–
PV-RCNN (Ours)	78.00	77.50	69.43	68.98	79.21	73.03	70.42	64.72	71.46	70.27	68.95	67.79
PV-RCNN++ (Ours)	**79.25**	**78.78**	**70.61**	**70.18**	**81.83**	**76.28**	**73.17**	**68.00**	**73.72**	**72.66**	**71.21**	**70.19**
3D-MAN+16f (Yang et al., 2021)	74.53	74.03	67.61	67.14	–	–	–	–	–	–	–	–
RSN+3f (Sun et al., 2021)	78.4	78.1	69.5	69.1	79.4	76.2	69.9	67.0	–	–	–	–
PV-RCNN++2f (Ours)	**80.17**	**79.70**	**72.14**	**71.70**	**83.48**	**80.42**	**75.54**	**72.61**	**74.63**	**73.75**	**72.35**	**71.50**

Table 2 Performance comparison on the test set of Waymo Open Dataset by submitting to the official test evaluation server. ∗: re-implemented by (Zhou et al., 2020). ‡: performance reported in the official open-source codebase of (Yin et al., 2021). "2f", "3f": the performance is achieved by using multiple point cloud frames

Method	Veh. (LEVEL 1)		Veh. (LEVEL 2)		Ped. (LEVEL 1)		Ped. (LEVEL 2)		Cyc. (LEVEL 1)		Cyc. (LEVEL 2)	
	mAP	mAPH	mAP	mAPH	mAP	mAPH	mAP	mAPH	mAP	mAPH	mAP	mAPH
StarNet (Ngiam et al., 2019)	61.5	61.0	54.9	54.5	67.8	59.9	61.1	54.0	–	–	–	–
*PointPillar (Lang et al., 2019)	63.3	62.8	55.6	55.1	62.1	50.2	55.9	45.1	–	–	–	–
CenterPoint (Yin et al., 2021)	80.2	79.7	72.2	71.8	78.3	72.1	72.2	66.4	–	–	–	–
‡CenterPoint (Yin et al., 2021)	–	–	–	71.9	–	–	–	67.0	–	–	–	**68.2**
PV-RCNN (Ours)	80.60	80.15	72.81	72.39	78.16	72.01	71.81	66.05	71.80	70.42	69.13	67.80
PV-RCNN++ (Ours)	**81.62**	**81.20**	**73.86**	**73.47**	**80.41**	**74.99**	**74.12**	**69.00**	**71.93**	**70.76**	**69.28**	**68.15**
3D-MAN+16f (Yang et al., 2021)	78.71	78.28	–	–	69.97	65.98	–	–	–	–	–	–
CenterPoint+2f (Yin et al., 2021)	81.05	80.59	73.42	72.99	80.47	77.28	74.56	71.52	**74.60**	**73.68**	**72.17**	**71.28**
RSN+3f (Sun et al., 2021)	80.7	80.3	71.9	71.6	78.9	75.6	70.7	67.8	–	–	–	–
PV-RCNN++2f (Ours)	**83.74**	**83.32**	**76.31**	**75.92**	**82.60**	**79.38**	**76.63**	**73.55**	74.44	73.43	72.06	71.09

rate decay. To train the proposal refinement stage, we randomly sample 128 proposals with 1:1 ratio for positive and negative proposals, where a proposal is considered as a positive sample if it has at least 0.55 3D IoU with the ground-truth boxes, otherwise it is treated as a negative sample. Both two frameworks are trained with three losses with equal loss weights (*i.e.*, region proposal loss, keypoint segmentation loss and proposal refinement loss), where the region proposal loss is same as (Yin et al., 2021) and the proposal refinement loss is same as (Shi et al., 2020b).

During training, we adopt the widely used data augmentation strategies for 3D detection, including random scene flipping, global scaling with a scaling factor sampled from $[0.95, 1.05]$, global rotation around Z axis with an angle sampled from $[-\frac{\pi}{4}, \frac{\pi}{4}]$, and the ground-truth sampling augmentation (Yan et al., 2018) to randomly "paste" some new objects from other scenes to current training scene for simulating objects in various environments. The detection range is set as $[-75.2, 75.2]m$ for X and Y axes, and $[-2, 4]m$ for the Z axis, while the voxel size is set as $(0.1m, 0.1m, 0.15m)$. More training details can be found in our open source codebase https://github.com/open-mmlab/OpenPCDet.

For the inference speed, our PV-RCNN++ framework can achieve state-of-the-art performance with 10 FPS for $150m \times 150m$ detection range on Waymo Open Dataset (three times faster than PV-RCNN), where a single TITAN RTX GPU card is utilized for profiling.

5.2 Main Results

In this section, we demonstrate the main results of our proposed PV-RCNN/PV-RCNN++ frameworks, and make the comparison with state-of-the-art methods on the large-scale Waymo Open Dataset (Sun et al., 2020). By default, we adopt the center-based RPN head as in (Yin et al., 2021) to generate 3D proposals in the first stage, and we train a single model in each setting for detecting the objects of all three categories.

Comparison with State-of-the-Art Methods As shown in Table 1, for the 3D object detection setting of taking a single frame point cloud as input, our PV-RCNN++ (*i.e.*, "PV-RCNN++") outperforms previous state-of-the-art works (Yin et al., 2021; Shi et al., 2020b) on all three categories with remarkable performance gains (+1.88% for vehicle, +2.40% for pedestrian and +1.59% for cyclist in terms of mAPH of LEVEL 2 difficulty). Moreover, following (Sun et al., 2021), by simply concatenating an extra neighboring past frame as input, our PV-RCNN framework can also be evaluated on the multi-frame setting. Table 1 (*i.e.*, "PV-RCNN++2f") demonstrates that the performance of our PV-RCNN++ framework can be further boosted by using 2 frames, which outperforms previous multi-frame method (Sun et al., 2021) with remarkable margins (+2.60% for vehicle, +5.61% for pedestrian in terms of mAPH of LEVEL 2).

Table 3 Performance comparison of PV-RCNN and PV-RCNN++ on the validation set of Waymo Open Dataset. We adopt two settings for both two frameworks by equipping with different RPN heads for proposal generation, which are the anchor-based RPN head as in (Shi et al., 2020b) and the center-based RPN head as in (Yin et al., 2021). Note that PV-RCNN++ adopts the same backbone network (without residual connection) with PV-RCNN for fair comparison

Difficulty	Method	Veh. mAPH	Ped. mAPH	Cyc. mAPH
LEVEL 1	PV-RCNN (anchor)	76.89	65.65	66.35
	PV-RCNN++ (anchor)	**78.64**	**69.26**	**70.86**
	PV-RCNN (center)	77.50	73.03	70.27
	PV-RCNN++ (center)	**78.63**	**74.62**	**72.38**
LEVEL 2	PV-RCNN (anchor)	68.41	57.61	63.98
	PV-RCNN++ (anchor)	**69.95**	**60.94**	**68.22**
	PV-RCNN (center)	68.98	64.72	67.79
	PV-RCNN++ (center)	**69.91**	**66.30**	**69.62**

Table 4 Performance comparison of PV-RCNN and PV-RCNN++ on the test set of KITTI dataset. The results are evaluated by the most important moderate difficulty level of KITTI evaluation metric by submitting to the official KITTI evaluation server

Method	Car	Pedestrian	Cyclist	Average
PV-RCNN	81.43	43.29	63.71	62.81
PV-RCNN++	**81.88**	**47.19**	**67.33**	**65.47**

Meanwhile, we also evaluate our frameworks on the test set by submitting to the official test server of Waymo Open Dataset (Sun et al., 2020). As shown in Table 2, without bells and whistles, in both single-frame and multi-frame settings, our PV-RCNN++ framework consistently outperforms previous state-of-the-art (Yin et al., 2021) significantly in both vehicle and pedestrian categories, where for single-frame setting we achieve a performance gain of +1.57% for vehicle and +2.00% for pedestrian in terms of mAPH of LEVEL 2 difficulty, and for multi-frame setting we achieve a performance gain of +2.93% for vehicle detection and +2.03% for pedestrian detection. We also achieve comparable performance for the cyclist category on both the single-frame and multi-frame settings. Note that we do not use any test-time augmentation or model ensemble tricks in the evaluation process. The significant improvements on the large-scale Waymo Open dataset manifest the effectiveness of our proposed framework.

Comparison of PV-RCNN and PV-RCNN++ Table 3 demonstrates that no matter which type of RPN head is adopted, our PV-RCNN++ framework consistently outperforms previous PV-RCNN framework on all three categories of all difficulty levels. Specifically, for the anchor-based setting, PV-RCNN++ surpasses PV-RCNN with a performance gain of +1.54% for vehicle, +3.33% for pedestrian and 4.24% for cyclist in terms of mAPH of LEVEL 2 difficulty. By taking the center-based head, PV-RCNN++ also outperforms PV-RCNN with a +0.93% mAPH gain for vehicle, a +1.58% mAPH gain for pedestrian and a +1.83% mAPH gain for cyclist in terms of LEVEL 2 difficulty.

The stable and consistent improvements prove the effectiveness of our proposed sectorized proposal-centric sampling algorithm and VectorPool aggregation module. More importantly, our PV-RCNN++ consumes much less calculations and GPU memory than PV-RCNN framework, while also increasing the processing speed from 3.3 FPS to 10 FPS for the 3D detection of $150m \times 150m$ such a large area, which further validates the efficiency and the effectiveness of our PV-RCNN++.

As shown in Table 4, we also provide the performance comparison of PV-RCNN and PV-RCNN++ on the KITTI dataset (Geiger et al., 2012). Compared with Waymo Open Dataset, KITTI dataset adopts different kinds of LiDAR sensor and the scene in KITTI dataset is about four times smaller than the scene in the Waymo Open Dataset. Table 3 shows that PV-RCNN++ outperforms previous PV-RCNN on all three categories of KITTI dataset with remarkable average performance margin, demonstrating its effectiveness on handling different kinds of scenes and different LiDAR sensors.

5.3 Ablation Study

In this section, we investigate the individual components of our PV-RCNN++ framework with extensive ablation experiments. We conduct all experiments on the large-scale Waymo Open Dataset (Sun et al., 2020). For efficiently conducting the ablation experiments, we generate a small representative training set by uniformly sampling 20% frames (about $32k$ frames) from the training set[2], and all results are evaluated on the full validation set (about $40k$ frames) with the official evaluation tool. All models are trained with 30 epochs and batch size 16 on 8 GPUs.

We conduct all ablation experiments with the center-based RPN head (Yin et al., 2021) on three categories (vehicle, pedestrian and cyclist) of Waymo Open Dataset (Sun et al., 2020), and the mAPH of LEVEL 2 difficulty is adopted as the evaluation metric.

Effects of Voxel-to-Keypoint Scene Encoding In Sec. 3.2, we propose the voxel-to-keypoint scene encoding strategy to encode the global scene features to a small set of keypoints, which serves as a bridge between the backbone network and the proposal refinement network. As shown in the 2^{nd} and 4^{th} rows of Table 5, our proposed voxel-to-keypoint scene encoding strategy achieves better performance than the UNet-based decoder while summarizing the scene features to much less point-wise features than the UNet-based decoder.

[2] Reference:https://github.com/open-mmlab/OpenPCDet.

Table 5 Effects of voxel set abstraction (VSA) and RoI-grid pooling modules, where the UNet-decoder and RoI-aware pooling are the same with (Shi et al., 2020b). All experiments are based on PV-RCNN++ framework with a center-based RPN head

Point Feature Extraction	RoI Pooling Module	Veh.	Ped.	Cyc.	Average
UNet-decoder	RoI-aware Pooling	66.42	63.41	67.48	65.77
UNet-decoder	RoI-grid Pooling	67.37	63.77	67.08	66.05
VSA	RoI-aware Pooling	66.15	60.70	66.07	64.31
VSA	RoI-grid Pooling	68.62	63.74	68.26	66.87

Table 6 Effects of different feature components for voxel set abstraction. "Frame Rate" indicates frames per seconds in terms of testing speed. All experiments are conducted on PV-RCNN++ framework with a center-based RPN head. Note that the default setting of PV-RCNN++ does not use the voxel features $f_i^{(\text{pv}_{1,2})}$ by considering its negligible gain and higher latency

VSA Input Feature				Frame Rate	Veh.	Ped.	Cyc.	Average
$f_i^{(\text{pv}_{1,2})}$	$f_i^{(\text{pv}_{3,4})}$	$f_i^{(\text{bev})}$	$f_i^{(\text{raw})}$					
		✓		16.3	67.01	61.23	66.54	64.93
			✓	13.2	68.02	62.73	66.80	65.85
	✓			12.3	68.13	63.24	67.37	66.25
	✓	✓		11.9	68.15	63.35	67.73	66.41
	✓		✓	10.3	68.46	63.59	67.58	66.54
	✓	✓	✓	10.0	68.62	63.74	68.26	66.87
✓	✓	✓	✓	*7.6*	*68.55*	*64.48*	*67.94*	*66.99*

For instance, our voxel set abstraction module encodes the whole scene to around $4k$ keypoints for feeding into the RoI-grid pooling module, while the UNet-based decoder network needs to summarize the scene features to around $80k$ point-wise features in most cases, which validates the effectiveness of our proposed voxel-to-keypoint scene encoding strategy. We consider that it might benefit from the fact that the keypoint features are aggregated from multi-scale feature volumes and raw point clouds with large receptive fields, while also keeping the accurate point locations. Besides that, we should also note that the feature dimension of UNet-based decoder is generally smaller than the feature dimensions of our keypoints since the UNet-based decoder is limited to its large memory consumption on large-scale point clouds, which may degrade its performance.

We also notice that our voxel set abstraction module achieves worse performance (the 1^{st} and 3^{rd} rows of Table 5) than the UNet-decoder when it is combined with RoI-aware pooling (Shi et al., 2020b). This is to be expected since RoI-aware pooling module will generate lots of empty voxels in each proposal by taking only $4k$ keypoints, which may degrade the performance. In contrast, our voxel set abstraction module can be ideally combined with our RoI-grid pooling module and they can benefit each other by taking a small number of keypoints as the intermediate connection.

Effects of Different Features for Voxel Set Abstraction The voxel set abstraction module incorporates multiple feature components (see Sec. 3.2), and their effects are explored in Table 6. We can summarize the observations as follows: (i) The performance drops a lot if we only aggregate features from high level bird-view semantic features ($f_i^{(\text{bev})}$) or accurate point locations ($f_i^{(\text{raw})}$), since neither 2D-semantic-only nor point-only are enough for the proposal refinement. (ii) As shown in 6^{th} row of Table 6, $f_i^{(\text{pv}_3)}$ and $f_i^{(\text{pv}_4)}$ contain both 3D structure information and high level semantic features, which can improve the performance a lot by combining with the bird-view semantic features $f_i^{(\text{bev})}$ and the raw point locations $f_i^{(\text{raw})}$. (iii) The shallow semantic features $f_i^{(\text{pv}_1)}$ and $f_i^{(\text{pv}_2)}$ can slightly improve the performance but also greatly increase the training cost. Hence, the proposed PV-RCNN++ framework does not use such shallow semantic features.

Table 7 Effects of Predicted Keypoint Weighting module. All experiments are conducted on our PV-RCNN++ framework with a center-based RPN head

Use PKW	Vehicle	Pedestrian	Cyclist	Average
×	68.48	63.90	67.62	66.66
✓	68.62	63.74	68.26	66.87

Effects of Predicted Keypoint Weighting The predicted keypoint weighting is proposed in Sec. 3.2 to re-weight the point-wise features of keypoints with extra keypoint segmentation supervision. As shown in Table 7, the experiments show that the performance slightly drops after removing this module, which demonstrates that the predicted keypoint weighting enables better multi-scale feature aggregation by focusing more on the foreground keypoints, since they are more important for the succeeding proposal refinement network. Although this module only leads small additional cost to our frameworks, we should also notice that it is optional for our frameworks by considering its limited gains.

Effects of RoI-grid Pooling Module RoI-grid pooling module is proposed in Sec. 3.3 for aggregating RoI features from the very sparse keypoints. Here we investigate the effects of RoI-grid pooling module by replacing it with the RoI-aware pooling (Shi et al., 2020b) and keeping other modules consistent. As shown in the 3^{rd} and 4^{th} rows Table 5, the experiments show that the performance drops significantly when replacing RoI-grid pooling. It validates that our proposed RoI-grid pooling module can aggregate much richer contextual information to generate more discriminative RoI features.

Compared with the previous RoI-aware pooling module (Shi et al., 2020b), our proposed RoI-grid pooling module can generate denser grid-wise feature representation by supporting different overlapped ball areas among different grid points, while RoI-aware pooling module may generate lots of zeros due to the sparse inside points of RoIs. That means our proposed RoI-grid pooling module is especially effective for aggregating local features from the very sparse point-wise features, such as in our PV-RCNN framework to aggregate features from a very small number of keypoints.

Effects of Proposal-Centric Filtering In the 1^{st} and 2^{nd} rows of Table 8, we investigate the effectiveness of our proposal-centric keypoint filtering (see Sec. 4.1), where we find that compared with the strong baseline PV-RCNN++ framework equipped with vanilla farthest point sampling, our proposal-centric keypoint filtering further improves the average detection performance by 1.12 mAPH in LEVEL 2 difficulty (65.87% vs. 66.99%). It validates our argument that our proposed proposal-centric keypoint sampling strategy can generate more representative keypoints by concentrating the small number of keypoints to the more informative neighboring regions of proposals. Moreover, improved by our proposal-centric keypoint filtering, our keypoint sampling algorithm is about five times (133ms vs. 27ms) faster than the vanilla farthest point sampling algorithm by reducing the number of candidate keypoints.

Effects of Sectorized Keypoint Sampling To investigate the effects of sectorized farthest point sampling (Sec. 4.1), we compare it with four alternative strategies for accelerating the keypoint sampling process: (i) Random Sampling: the keypoints are randomly chosen from raw points. (ii) Voxelized-FPS-Voxel: the raw points are firstly voxelized to reduce the number of points (*i.e.,* voxels), then farthest point sampling is applied to sample keypoints from voxels by taking the voxel centers. (iii) Voxelized-FPS-Point: unlike Voxelized-FPS-Voxel, here a raw point is randomly selected within the selected voxels as the keypoints. (iv) RandomParallel-FPS: the raw points are randomly split into several groups, then farthest point sampling is utilized to these groups in parallel for faster keypoint sampling.

As shown in Table 8, compared with the vanilla farthest point sampling (2^{nd} row) algorithm, the detection perfor-

Table 8 Effects of different keypoint sampling algorithms. The running time is the average running time of keypoint sampling process on the validation set of the Waymo Open Dataset. The coverage rate is calculated by averaging the coverage rate of each scene on the validation set of the Waymo Open Dataset. "FPS" indicates the farthest point sampling and "PC-Filter" indicates our proposal-centric filtering strategy. All experiments are conducted by adopting different keypoint sampling algorithms to our PV-RCNN++ framework with a center-based RPN head

Keypoint Sampling Algorithm	Running Time	Performance of L2 mAPH				Coverage Rate (CR) under Different Radii					Mean CR
		Veh.	Ped.	Cyc.	Average	0.1m	0.2m	0.3m	0.4m	0.5m	CR
FPS	133ms	67.70	63.18	66.74	65.87	–	–	–	–	–	–
PC-Filter+FPS	27ms	68.78	64.09	68.11	66.99	43.22	82.78	97.97	99.95	99.99	84.78
PC-Filter+Random Sampling	<1ms	65.10	60.59	65.57	63.75	51.44	77.06	87.32	92.16	94.74	80.54
PC-Filter+Voxelized-FPS-Voxel	17ms	68.01	63.21	67.36	66.19	6.52	52.00	89.93	98.89	99.92	69.45
PC-Filter+Voxelized-FPS-Point	17ms	68.37	63.45	67.30	66.37	34.14	75.84	96.28	99.77	99.99	81.20
PC-Filter+RandomParallel-FPS	2ms	68.05	63.20	66.79	66.01	38.12	64.47	81.72	91.82	96.90	74.61
PC-Filter+Sectorized-FPS	9ms	68.62	63.74	68.26	66.87	47.63	82.38	95.20	98.87	99.72	84.76

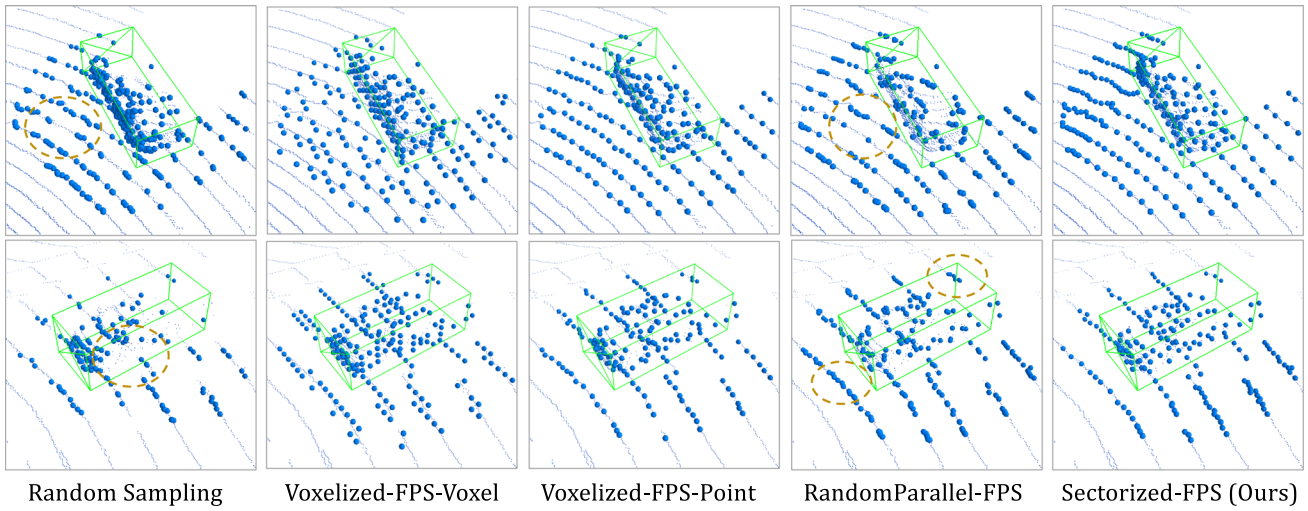

Fig. 6 Illustration of the keypoint distributions from different keypoint sampling strategies. Some dashed circles are utilized to highlight the missing parts and the clustered keypoints after using these keypoint sampling strategies. We find that our Sectorized-FPS generates better uniformly distributed keypoints that cover more input points to better encode the scene features for proposal refinement, while other strategies may miss some important regions or generate some clustered keypoints

mances of all four alternative strategies drop a lot. In contrast, the performance of our proposed sectorized farthest point sampling algorithm is on par with the vanilla farthest point sampling (66.99% vs. 66.87%) while being three times (27ms vs. 9ms) faster than the vanilla farthest point sampling algorithm.

Analysis of the Coverage Rate of Keypoints We argue that the uniformly distributed keypoints are important for the proposal refinement, where a better keypoint distribution should cover more input points. Hence, to evaluate the quality of different keypoint sampling strategies, we propose an evaluation metric, *coverage rate*, which is defined as the ratio of input points that are within the coverage region of any keypoints. Specifically, for a set of input points $\mathcal{P} = \{p_i\}_{i=1}^{m}$ and a set of sampled keypoints $\mathcal{K} = \{p'_j\}_{j=1}^{n}$, the coverage rate $\mathbf{C}$ can be formulated as:

$$\mathbf{C} = \frac{1}{m} \cdot \sum_{i=1}^{m} \min(1.0, \quad \sum_{j=1}^{n} \mathbb{1}(||p_i - p'_j|| < R_c)), \qquad (13)$$

where R_c is a scalar that denotes the coverage radius of each keypoint, and $\mathbb{1}(\cdot) \in \{0, 1\}$ is the indicator function to indicate whether p_i is covered by p'_j.

As shown in Table 8, we evaluate the coverage rate of different keypoint sampling algorithms in terms of multiple coverage radii. Our sectorized farthest point sampling achieves similar average coverage rate (84.76%) with the vanilla farthest point sampling (84.78%), which is much better than other sampling algorithms. The higher average coverage rate demonstrates that our proposed sectorized farthest point sampling can sample more uniformly distributed keypoints to better cover the input points, which is consistent with the qualitative results of different sampling strategies as in Fig. 6.

In short, our sectorized farthest point sampling can generate uniformly distributed keypoints to better cover the input points, by splitting raw points into different groups based on the fact of radial distribution of LiDAR points. Although there may still exist a very small number of clustered keypoints in the margins of different sectors, the experiments show that they have negligible effect on the performance. We consider the reason may be that the clustered keypoints are mostly in the regions around the scene centers, where the objects are generally easier to detect since the raw points around scene centers are much denser than distant regions.

Effects of VectorPool Aggregation In Sec. 4.2, to tackle the resource-consuming problem of set abstraction, we propose VectorPool aggregation module to effectively and efficiently summarize the structure-preserved local features from point clouds. As shown in Table 9, by adopting VectorPool aggregation in both voxel set abstraction module and RoI-grid pooling module, PV-RCNN++ framework consumes much less computations (*i.e.*, a reduction of 4.679 GFLOPS) and GPU memory (from 10.62GB to 7.58GB) than original PV-RCNN framework, while the performance is also consistently increased from 65.92% to 66.87% in terms of average mAPH (LEVEL 2) of three categories. Note that the batch size is only set as 2 in all of our settings and the reduction of memory consumption / calculations can be more significant with larger batch size.

The significant reduction of resource consumption demonstrates the effectiveness of our VectorPool aggregation

Table 9 Effects of different components in our proposed PV-RCNN++ frameworks. All models are trained with 20% frames from the training set and are evaluated on the full validation set of the Waymo Open Dataset, and the evaluation metric is the mAPH in terms of LEVEL_1 (L1) and LEVEL_2 (L2) difficulties as used in (Sun et al., 2020). "FPS" denotes farthest point sampling, "SPC-FPS" denotes our proposed sectorized proposal-centric keypoint sampling strategy, "VSA" denotes the voxel set abstraction module, "SA" denotes the set abstraction operation and "VP" denotes our proposed VectorPool aggregation. All models adopt the center-based RPN head for proposal generation

Setting	Keypoint Sampling	Point Feature Extraction	RoI Feature Aggregation	Vehicle		Pedestrian		Cyclist		Average		GFLOPS	Memory (MB)
				L1	L2	L1	L2	L1	L2	L1	L2		
PV-RCNN	FPS	VSA (SA)	RoI-grid pool (SA)	75.43	67.54	69.40	61.62	68.98	66.57	71.27	65.24	-0	10617
	SPC-FPS	VSA (SA)	RoI-grid pool (SA)	76.44	68.03	70.52	62.43	69.84	67.29	72.27	65.92	-0	10617
	SPC-FPS	VSA (VP)	RoI-grid pool (SA)	77.41	68.73	70.98	63.30	69.63	67.19	72.67	66.41	-1.712	9453
	SPC-FPS	VSA (SA)	RoI-grid pool (VP)	77.12	68.43	70.82	63.15	70.11	67.66	72.68	66.41	-2.967	8565
PV-RCNN++	SPC-FPS	VSA (VP)	RoI-grid pool (VP)	77.32	68.62	71.36	63.74	70.71	68.26	73.13	66.87	-4.679	7583

for feature learning from large-scale point clouds, which makes our PV-RCNN++ framework a more practical 3D detector to be used on resource-limited devices. Moreover, PV-RCNN++ framework also benefits from the structure-preserved spatial features from our VectorPool aggregation, which is critical for the following fine-grained proposal refinement.

We further analyze the effects of VectorPool aggregation by removing channel reduction (Sun et al., 2018) in our VectorPool aggregation. As shown in Table 10, our VectorPool aggregation is effective in reducing memory consumption no matter whether channel reduction is incorporated (by comparing the 1^{st} / 3^{rd} rows or the 2^{nd} / 4^{th} rows), since the model activations in our VectorPool aggregation modules consume much less memory than those in set abstraction, by adopting a single local vector representation before multi-layer perceptron networks. Meanwhile, Table 10 also demonstrates that our VectorPool aggregation can achieve better performance than set abstraction (Qi et al., 2017b) in both two cases (with or without channel reduction). Meanwhile, we also notice that VectorPool aggregation slightly improves the number of parameters compared with previous set abstraction module (*e.g.*, from 13.05M to 14.11M for the setting with channel reduction), which is generally negligible given the fact that VectorPool aggregation consumes smaller GPU memory.

Effects of Different Feature Aggregation Strategies for Local Sub-Voxels As mentioned in Sec. 4.2, in addition to our adopted interpolation-based method, there are two alternative strategies (average pooling and random selection) for aggregating features of local sub-voxels. Table 11 demonstrates that our interpolation based feature aggregation achieves much better performance than the other two strategies, especially for the small objects like pedestrian and cyclist. We consider that our strategy can generate more effective features by interpolating from three nearest neighbors (even beyond the sub-voxel), while both of the other two methods might generate lots of zero features on the empty sub-voxels, which may degrade the final performance.

Effects of Separate Local Kernel Weights in VectorPool Aggregation We adopt separate local kernel weights (see Eq. (11)) on different local sub-voxels to generate position-sensitive features. The 1^{st} and 2^{nd} rows of Table 12 show that the performance drops a bit if we remove the separate local kernel weights and adopt shared kernel weights for relative position encoding. It validates that the separate local kernel weights are better than previous shared-parameter MLP for local feature encoding, and it is important in our proposed VectorPool aggregation module.

Effects of Dense Voxel Numbers in VectorPool Aggregation Table 12 investigates the number of dense voxels $n_x \times n_y \times n_z$ in VectorPool aggregation for voxel set abstraction module and RoI-grid pooling module, where we can see that VectorPool aggregation with $3 \times 3 \times 3$ and

Table 10 Effects of VectorPool aggregation with and without channel reduction (Sun et al., 2018). "SA" denotes set abstraction, "VP" denotes VectorPool aggregation module and "CR" denotes channel reduction. "#Param." indicates the number of parameters of the model. All experiments are based on our PV-RCNN++ framework with a center-based RPN head for proposal generation, and only the local feature extraction modules are changed during the ablation experiments

Stra- tegy	CR	Veh.	Ped.	Cyc.	Average	GFLOPS	Mem. (MB)	#Param.
SA	×	68.03	62.43	67.29	65.92	-0	10617	13.07M
SA	✓	68.43	62.06	66.96	65.81	-3.467	9795	13.05M
VP	×	68.82	64.06	67.96	66.95	-1.988	8549	14.32M
VP	✓	68.62	63.74	68.26	66.87	-4.679	7583	14.11M

Table 11 Effects of the feature aggregation strategies to generate the local sub-voxel features of VectorPool aggregation. All experiments are based on our PV-RCNN++ framework with a center-based RPN head for proposal generation

Aggregation of Sub-Voxels	Vehicle	Pedestrian	Cyclist	Average
Average Pooling	68.35	62.33	67.50	66.06
Random Selection	68.36	62.82	67.68	66.29
Interpolation	68.62	63.74	68.26	66.87

Table 12 Effects of separate local kernel weights and the number of dense voxels in our proposed VectorPool aggregation module. All experiments are based on our PV-RCNN++ framework with a center-based head for proposal generation

Kernel Weights	Number of Dense Voxels	Vehicle	Pedestrian	Cyclist	Average
Share	$3 \times 3 \times 3$	68.17	63.28	67.36	66.27
Separate	$3 \times 3 \times 3$	68.62	63.74	68.26	66.87
Separate	$2 \times 2 \times 2$	68.21	62.88	67.44	66.18
Separate	$3 \times 3 \times 3$	68.62	63.74	68.26	66.87
Separate	$4 \times 4 \times 4$	68.74	63.99	67.98	66.90

Table 13 Effects of the number of keypoints for encoding the global scene. All experiments are based on our PV-RCNN++ framework with a center-based head for proposal generation

Number of Keypoints	Vehicle	Pedestrian	Cyclist	Average
8192	68.85	64.11	67.88	66.95
4096	68.62	63.74	68.26	66.87
2048	67.99	62.14	67.41	65.85
1024	66.67	59.21	65.07	63.65

Table 14 Effects of the grid size in RoI-grid pooling module. All experiments are based on our PV-RCNN++ framework with a center-based head for proposal generation

RoI-grid Size	Vehicle	Pedestrian	Cyclist	Average
$8 \times 8 \times 8$	68.88	63.74	67.84	66.82
$7 \times 7 \times 7$	68.76	63.81	68.00	66.85
$6 \times 6 \times 6$	68.62	63.74	68.26	66.87
$5 \times 5 \times 5$	68.28	63.54	67.69	66.50
$4 \times 4 \times 4$	68.21	63.58	67.56	66.45
$3 \times 3 \times 3$	67.33	62.93	67.22	65.83

$4 \times 4 \times 4$ achieve similar performance while the performance of $2 \times 2 \times 2$ setting drops a lot. We consider that our interpolation-based VectorPool aggregation can generate effective voxel-wise features even with large dense voxels, hence the setting with $4 \times 4 \times 4$ achieves slightly better performance than the setting with $3 \times 3 \times 3$. However, since the setting with $4 \times 4 \times 4$ greatly improves the calculations and memory consumption, we finally choose the setting of $3 \times 3 \times 3$ dense voxel representation in both voxel set abstraction module (except the raw point layer) and RoI-grid pooling module of our PV-RCNN++ framework.

Effects of the Number of Keypoints In Table 13, we investigate the effects of the number of keypoints for encoding the scene features. Table 13 shows that larger number of keypoints achieves better performance, and similar performance is observed when using more than 4,096 keypoints. Hence, to balance the performance and computation cost, we empirically choose to encode the whole scene to 4,096 keypoints for the Waymo Open dataset. The above experiments show that our method can effectively encode the whole scene to 4,096 keypoints while keeping similar performance with a large number of keypoints, which demonstrates the effectiveness of the keypoint feature encoding strategy of our proposed PV-RCNN/PV-RCNN++ frameworks.

Table 15 Comparison of PV-RCNN/PV-RCNN++ on different sizes of scenes. "FoV" indicates the field of view of each scene, where for each scene in Waymo Open Dataset, we crop a specific angle (*e.g.*, 90°, 180°) of frontal view for training and testing, and 360° FoV indicates the original scene. "#Points" indicates the average number of points in each scene. "Frame Rate" indicates frames per seconds in terms of testing speed

FoV	#Points	Method	Frame Rate	Veh.	Ped.	Cyc.	Average
90°	∼45k	PV-RCNN	10.5	66.50	59.33	65.96	63.93
90°	∼45k	PV-RCNN++	16.0	**67.29**	**61.51**	**66.91**	**65.24**
180°	∼90k	PV-RCNN	6.8	65.34	60.20	61.70	62.41
180°	∼90k	PV-RCNN++	12.3	**66.05**	**62.08**	**63.44**	**63.86**
360°	∼180k	PV-RCNN	3.3	67.54	61.62	66.57	65.24
360°	∼180k	PV-RCNN++	10.0	**68.62**	**63.74**	**68.26**	**66.87**

Effects of the Grid Size in RoI-grid Pooling. Table 14 shows the performance of adopting different RoI-grid sizes for RoI-grid pooling module. We can see that the performance increases along with the RoI-grid sizes from $3 \times 3 \times 3$ to $6 \times 6 \times 6$, and the settings with larger RoI-grid sizes than $6 \times 6 \times 6$ achieve similar performance. Hence we finally adopt RoI-grid size $6 \times 6 \times 6$ for the RoI-grid pooling module. Moreover, from Table 14 and Table 9, we also notice that PV-RCNN++ with a much smaller RoI-grid size $4 \times 4 \times 4$ (66.45% in terms of mAPH@L2) can also outperform PV-RCNN with larger RoI-grid size $6 \times 6 \times 6$ (65.24% in terms of mAPH@L2), which further validates the effectiveness of our proposed sectorized proposal-centric sampling strategy and the VectorPool aggregation module.

Comparison on Different Sizes of Scenes. To investigate the effects of our proposed PV-RCNN++ on handling large-scale scenes, we further conduct ablation experiments to compare the effectiveness and efficiency of PV-RCNN and PV-RCNN++ frameworks on different sizes of scenes. As shown in Table 15, we compare these two frameworks on three sizes of scenes by cropping different angles of frontal view of the scene in Waymo Open Dataset for training and testing. PV-RCNN++ framework consistently outperforms previous PV-RCNN framework on all three sizes of scenes with large performance gains. Table 15 also demonstrates that as the scales of the scenes get larger, PV-RCNN++ becomes much more efficient than PV-RCNN. In particular, when the comparison is conducted on the original scene of Waymo Open Dataset, the running speed of PV-RCNN++ is about three times faster than PV-RCNN, demonstrating the efficiency of PV-RCNN++ on handling large-scale scenes.

6 Conclusion

In this paper, we present two novel frameworks, named PV-RCNN and PV-RCNN++, for accurate 3D object detection from point clouds. Our PV-RCNN framework adopts a novel voxel set abstraction module to deeply integrates both the multi-scale 3D voxel CNN features and the PointNet-based features to a small set of keypoints, and the learned discriminative keypoint features are then aggregated to the RoI-grid points through our proposed RoI-grid pooling module to capture much richer contextual information for proposal refinement. Our PV-RCNN++ further improves PV-RCNN framework by efficiently generating more representative keypoints with our novel sectorized proposal-centric keypoint sampling strategy, and also by equipping with our proposed VectorPool aggregation module to learn structure-preserved local features in both the voxel set abstraction module and RoI-grid pooling module. Thus, our PV-RCNN++ finally achieves better performance with much faster running speed than the original PV-RCNN framework.

Our final PV-RCNN++ framework significantly outperforms previous 3D detection methods and achieve state-of-the-art performance on both the validation set and testing set of the large-scale Waymo Open Dataset, and extensive experiments have been designed and conducted to deeply investigate the individual components of our proposed frameworks.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Brazil, G., Liu, X., (2019) M3d-rpn: Monocular 3d region proposal network for object detection. In: ICCV.

Chabot, F., Chaouch, M., Rabarisoa, J., Teuliere, C., Chateau, T. (2017) Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In: CVPR.

Chen, Q., Sun, L., Wang, Z., Jia, K., Yuille, A. (2019a) Object as hotspots: An anchor-free 3d object detection approach via firing of hotspots.

Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., Urtasun, R. (2016) Monocular 3d object detection for autonomous driving. In: CVPR.

Chen, X., Ma, H., Wan, J., Li, B., Xia, T. (2017) Multi-view 3d object detection network for autonomous driving. In: CVPR.

Chen, Y., Liu, S., Shen, X., Jia, J. (2019b) Fast point r-cnn. In: ICCV.

Chen, Y., Liu, S., Shen, X., Jia, J. (2020) Dsgn: Deep stereo geometry network for 3d object detection. In: CVPR.

Choy, C., Gwak, J., Savarese, S. (2019) 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: CVPR.

Geiger, A., Lenz, P., Urtasun, R. (2012) Are we ready for autonomous driving? the kitti vision benchmark suite. In: CVPR.

Girshick, R. (2015) Fast r-cnn. In: ICCV.

Graham, B., Engelcke, M., van der Maaten, L. (2018) 3d semantic segmentation with submanifold sparse convolutional networks. CVPR.

Huang, J., Huang, G. (2022) Bevdet4d: Exploit temporal cues in multi-camera 3d object detection. arXiv preprint arXiv:2203.17054.

Huang, J., Huang, G., Zhu, Z., Du, D. (2021) Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. arXiv preprint arXiv:2112.11790.

Huang, Q., Wang, W., Neumann, U. (2018) Recurrent slice networks for 3d segmentation of point clouds. In: CVPR.

Huang, T., Liu, Z., Chen, X., Bai, X. (2020) Epnet: Enhancing point features with image semantics for 3d object detection. In: ECCV.

Jaritz, M., Gu, J., Su, H. (2019) Multi-view pointnet for 3d scene understanding. In: ICCV Workshops.

Jiang, L., Zhao, H., Liu, S., Shen, X., Fu, C. W., Jia, J. (2019) Hierarchical point-edge interaction network for point cloud semantic segmentation. In: ICCV.

Jiang, Y., Zhang, L., Miao, Z., Zhu, X., Gao, J., Hu, W., Jiang, Y. G. (2022) Polarformer: Multi-camera 3d object detection with polar transformers. arXiv preprint arXiv:2206.15398.

Ku, J., Mozifian, M., Lee, J., Harakeh, A., Waslander, S. (2018) Joint 3d proposal generation and object detection from view aggregation. IROS.

Kuang, H., Wang, B., An, J., Zhang, M., Zhang, Z. (2020) Voxel-fpn: Multi-scale voxel feature aggregation for 3d object detection from lidar point clouds. Sensors.

Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O. (2019) Pointpillars: Fast encoders for object detection from point clouds. CVPR.

Li, B., Ouyang, W., Sheng, L., Zeng, X., Wang, X. (2019a) Gs3d: An efficient 3d object detection framework for autonomous driving. In: CVPR.

Li, P., Chen, X., Shen, S. (2019b) Stereo r-cnn based 3d object detection for autonomous driving. In: CVPR.

Li, P., Zhao, H., Liu, P., Cao, F. (2020) Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving. In: ECCV.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B. (2018) Pointcnn: Convolution on x-transformed points. In: NeurIPS.

Li, Y., Ge, Z., Yu, G., Yang, J., Wang, Z., Shi, Y., Sun, J., Li, Z. (2022a) Bevdepth: Acquisition of reliable depth for multi-view 3d object detection. arXiv preprint arXiv:2206.10092.

Li, Z., Wang, F., Wang, N. (2021) Lidar r-cnn: An efficient and universal 3d object detector. In: CVPR.

Li, Z., Wang, W., Li, H., Xie, E., Sima, C., Lu, T., Yu, Q., Dai, J. (2022b) Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. arXiv preprint arXiv:2203.17270.

Liang, M., Yang, B., Wang, S., Urtasun, R. (2018) Deep continuous fusion for multi-sensor 3d object detection. In: ECCV.

Liang, M., Yang, B., Chen, Y., Hu, R., Urtasun, R. (2019) Multi-task multi-sensor fusion for 3d object detection. In: CVPR.

Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S. (2017) Feature pyramid networks for object detection. In: CVPR.

Lin, T. Y., Goyal, P., Girshick, R., He, K., Dollár, P. (2018) Focal loss for dense object detection. TPAMI.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., Berg, A. C. (2016) Ssd: Single shot multibox detector. In: ECCV.

Liu, Y., Wang, T., Zhang, X., Sun, J. (2022a) Petr: Position embedding transformation for multi-view 3d object detection. arXiv preprint arXiv:2203.05625.

Liu, Y., Yan, J., Jia, F., Li, S., Gao, Q., Wang, T., Zhang, X., Sun, J. (2022b) Petrv2: A unified framework for 3d perception from multi-camera images. arXiv preprint arXiv:2206.01256.

Liu, Z., Tang, H., Lin, Y., Han, S. (2019) Point-voxel cnn for efficient 3d deep learning. In: NeurIPS.

Liu, Z., Hu, H., Cao, Y., Zhang, Z., Tong, X. (2020) A closer look at local aggregation operators in point cloud analysis. arXiv preprint arXiv:2007.01294.

Manhardt, F., Kehl, W., Gaidon, A. (2019) Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape. In: CVPR.

Mao, J., Xue, Y., Niu, M., Bai, H., Feng, J., Liang, X., Xu, H., Xu, C. (2021) Voxel transformer for 3d object detection. In: ICCV.

Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J. (2017) 3d bounding box estimation using deep learning and geometry. In: CVPR.

Murthy, J. K., Krishna, G. S., Chhaya, F, Krishna, K. M. (2017) Reconstructing vehicles from a single image: Shape priors for road scene understanding. In: ICRA.

Ngiam, J., Caine, B., Han, W., Yang, B., Chai, Y., Sun, P., Zhou, Y., Yi, X., Alsharif, O., Nguyen, P., et al. (2019) Starnet: Targeted computation for object detection in point clouds. arXiv preprint arXiv:1908.11069.

Philion, J., Fidler, S. (2020) Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In: ECCV.

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

Qi, C. R., Yi, L., Su ,H., Guibas, L. J. (2017b) Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: NeurIPS.

Qi, C. R., Liu, W., Wu, C., Su, H., Guibas, L. J. (2018) Frustum pointnets for 3d object detection from rgb-d data. In: CVPR.

Qi, C. R., Litany, O., He, K., Guibas, L. J. (2019) Deep hough voting for 3d object detection in point clouds. In: ICCV.

Qian, R., Garg, D., Wang, Y., You, Y., Belongie, S., Hariharan, B., Campbell, M., Weinberger, K. Q., Chao, W. L. (2020) End-to-end pseudo-lidar for image-based 3d object detection. In: CVPR.

Reading, C., Harakeh, A., Chae, J., Waslander, S. L. (2021) Categorical depth distribution network for monocular 3d object detection. In: CVPR.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016) You only look once: Unified, real-time object detection. In: CVPR.

Ren, S., He, K., Girshick, R., Sun, J. (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. In: NeurIPS.

Sheng, H., Cai, S., Liu, Y., Deng, B., Huang, J., Hua, X. S., Zhao, M. J. (2021) Improving 3d object detection with channel-wise transformer. In: ICCV.

Shi, S., Wang, X., Li, H. (2019) Pointrcnn: 3d object proposal generation and detection from point cloud. In: CVPR.

Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., Li, H. (2020a) Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In: CVPR.
Shi, S., Wang, Z., Shi, J., Wang, X., Li, H. (2020b) From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. TPAMI.
Song, S., Xiao, J. (2016) Deep sliding shapes for amodal 3d object detection in rgb-d images. In: CVPR.
Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M. H., Kautz, J. (2018) Splatnet: Sparse lattice networks for point cloud processing. In: CVPR.
Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., Anguelov, D. (2020) Scalability in perception for autonomous driving: Waymo open dataset. In: CVPR.
Sun, P., Wang, W., Chai, Y., Elsayed, G., Bewley, A., Zhang, X., Sminchisescu, C., Anguelov, D. (2021) Rsn: Range sparse net for efficient, accurate lidar 3d object detection. In: CVPR.
Sun, S., Pang, J., Shi, J., Yi, S., Ouyang, W. (2018) Fishnet: A versatile backbone for image, region, and pixel level prediction. In: NeurIPS.
Thomas, H., Qi, C. R., Deschaud, J. E., Marcotegui, B., Goulette, F., Guibas, L. J. (2019) Kpconv: Flexible and deformable convolution for point clouds. In: ICCV.
Vora, S., Lang, A. H., Helou, B., Beijbom, O. (2020) Pointpainting: Sequential fusion for 3d object detection. In: CVPR.
Wang, Y., Chao, W. L., Garg, D., Hariharan, B., Campbell, M., Weinberger, K. Q. (2019a) Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In: CVPR.
Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., Solomon, J. M. (2019b) Dynamic graph cnn for learning on point clouds. TOG.
Wang Y, Fathi A, Kundu A, Ross DA, Pantofaru C, Funkhouser T, Solomon J (2020) Pillar-based object detection for autonomous driving. In: ECCV
Wang, Y., Guizilini, V. C., Zhang, T., Wang, Y., Zhao, H., Solomon, J. (2022) Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In: CoRL.
Wang, Z., Jia, K. (2019) Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. In: IROS.
Wu, W., Qi, Z., Fuxin, L. (2019) Pointconv: Deep convolutional networks on 3d point clouds. In: CVPR.
Xie, E., Yu, Z., Zhou, D., Philion, J., Anandkumar, A., Fidler, S., Luo, P., Alvarez, J. M. (2022) M2bev: Multi-camera joint 3d detection and segmentation with unified birds-eye view representation. arXiv preprint arXiv:2204.05088.
Yan, Y., Mao, Y., Li, B. (2018) Second: Sparsely embedded convolutional detection. Sensors.
Yang, B., Liang, M., Urtasun, R. (2018a) Hdnet: Exploiting hd maps for 3d object detection. In: CoRL.
Yang, B., Luo, W., Urtasun, R. (2018b) Pixor: Real-time 3d object detection from point clouds. In: CVPR.
Yang. Z., Sun, Y., Liu, S., Shen, X., Jia, J. (2019) STD: sparse-to-dense 3d object detector for point cloud. ICCV.
Yang, Z., Sun, Y., Liu, S., Jia, J. (2020) 3dssd: Point-based 3d single stage object detector. In: CVPR.
Yang, Z., Zhou, Y., Chen, Z., Ngiam, J. (2021) 3d-man: 3d multi-frame attention network for object detection. In: CVPR.
Ye, M., Xu, S., Cao, T. (2020) Hvnet: Hybrid voxel network for lidar based 3d object detection. In: CVPR.
Yin, T., Zhou, X., Krahenbuhl, P. (2021) Center-based 3d object detection and tracking. In: CVPR.
Yoo, J. H., Kim, Y., Kim, J. S., Choi, J. W. (2020) 3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection. In: ECCV.
You, Y., Wang, Y., Chao, W. L., Garg, D., Pleiss, G., Hariharan, B., Campbell, M., Weinberger, K. Q. (2020) Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. In: ICLR.
Zhao, H., Jiang, L., Fu, C. W., Jia, J. (2019) Pointweb: Enhancing local neighborhood features for point cloud processing. In: CVPR.
Zhou, Y., Tuzel, O. (2018) Voxelnet: End-to-end learning for point cloud based 3d object detection. In: CVPR.
Zhou, Y., Sun, P., Zhang, Y., Anguelov, D., Gao, J., Ouyang, T., Guo, J., Ngiam, J., Vasudevan, V. (2020) End-to-end multi-view fusion for 3d object detection in lidar point clouds. In: CoRL.