OPEN: Object-wise Position Embedding for Multi-view 3D Object Detection

Jinghua Hou[1], Tong Wang[2], Xiaoqing Ye[2], Zhe Liu[1], Shi Gong[2], Xiao Tan[2], Errui Ding[2], Jingdong Wang[2], and Xiang Bai[1†]

[1] Huazhong University of Science and Technology
{jhhou,zheliu1994,xbai}@hust.edu.cn
[2] Baidu Inc., China

Abstract. Accurate depth information is crucial for enhancing the performance of multi-view 3D object detection. Despite the success of some existing multi-view 3D detectors utilizing pixel-wise depth supervision, they overlook two significant phenomena: 1) the depth supervision obtained from LiDAR points is usually distributed on the surface of the object, which is not so friendly to existing DETR-based 3D detectors due to the lack of the depth of 3D object center; 2) for distant objects, fine-grained depth estimation of the whole object is more challenging. Therefore, we argue that the object-wise depth (or 3D center of the object) is essential for accurate detection. In this paper, we propose a new multi-view 3D object detector named OPEN, whose main idea is to effectively inject object-wise depth information into the network through our proposed object-wise position embedding. Specifically, we first employ an object-wise depth encoder, which takes the pixel-wise depth map as a prior, to accurately estimate the object-wise depth. Then, we utilize the proposed object-wise position embedding to encode the object-wise depth information into the transformer decoder, thereby producing 3D object-aware features for final detection. Extensive experiments verify the effectiveness of our proposed method. Furthermore, OPEN achieves a new state-of-the-art performance with 64.4% NDS and 56.7% mAP on the nuScenes test benchmark. The code is available at `https://github.com/AlmoonYsl/OPEN`.

Keywords: Multi-view 3D object detection · Depth prediction · Position embedding

1 Introduction

3D object detection involves the localization and recognition of 3D objects in the real world, which is a fundamental task in 3D perception and is widely applied in autonomous driving and robotics. In recent years, 3D object detection [9, 14–16, 21, 33, 35] based on camera images has attracted increasing attention due to its lower price than LiDAR sensors. Besides, previous studies

† Corresponding author.

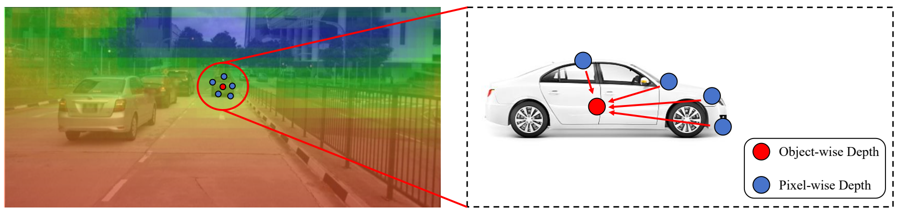

Fig. 1: The illustration of object-wise depth prediction. The blue points represent the pixel-wise depth, which is usually distributed on the surface of objects and supervised by projected LiDAR points. The red points represent the object-wise depth, which is the 3D center of the object and is supervised by the accurate center of projected 3D ground truth bounding boxes annotated by humans.

proposed many architectures for multi-view 3D object detection and achieved promising performance compared with LiDAR-based 3D object detectors.

Depth is important information for accurate 3D object detection based on camera images. Recent studies [13, 15, 33] have explored how to utilize depth information to improve detection performance. In more detail, they usually adopt pixel-wise depth supervision from the projected LiDAR points on the multi-view images to make the network aware of the depth. However, we find that the pixel-wise depth supervision obtained from the projected LiDAR points is mainly distributed on the surface of objects, as shown in Figure 1, which is not so friendly to existing DETR-based 3D object detectors. Besides, for some distant objects, the fine-grained depth estimation of the whole object is more challenging (*e.g.*, ambiguous for distant objects) than only predicting the depth of 3D centers.

In contrast, we observe that the object-wise depth representation can alleviate the aforementioned problems, as shown in Figure 1. On the one hand, it is more suitable for DETR-based 3D object detectors since object queries are usually defined as the center of objects. On the other hand, the object-wise depth is more easily estimated accurately, especially for distant objects. However, there is a new question: *how to effectively introduce the depth information into multi-view 3D object detectors?* Position embedding is an effective operation to encode the geometric information into multi-view image features, which have been mentioned in existing multi-view 3D object detectors [21, 22, 33, 35]. Among them, there are two representative position embedding, including ray-aware manner [35] and point-aware manner [33]. For ray-aware position embedding, it roughly encodes the depth candidates generated in the camera frustum, leading to an uncertain depth. For point-aware position embedding, it only encodes the predicted pixel-wise depth, which ignores the object-wise depth, leading to sub-optimal performance.

Towards this goal, in this paper, we propose a new multi-view 3D detector named OPEN, which mainly introduces the object-wise depth representation to multi-view 3D object detection by our proposed object-wise position embedding. Specifically, OPEN consists of three components: pixel-wise depth encoder (PDE), object-wise depth encoder (ODE), and object-wise position embedding (OPE). PDE first predicts the pixel-wise depth map by aggregating multi-view image features to possess the depth-aware ability of the whole scene. This provides the prior information for the following object-wise depth estimation. Then, the ODE effectively combines the pixel-wise depth representation and temporal information so as to reason the object-wise depth accurately.

Next, to take full advantage of object-wise depth to enhance detection performance, we encode the object-wise depth into the transformer decoder through our object-wise position embedding, leading to 3D object-aware features. Besides, we design a depth-aware focal loss (DFL) to encourage the network to pay more attention to the 3D object center.

Overall, our contributions are summarized as follows:

– We propose a new multi-view 3D object detector named OPEN, which utilizes the 3D object-wise depth representation to achieve better detection performance.
– We introduce the object-wise position embedding to effectively inject object-wise depth information into the transformer decoder, leading to 3D object-aware features.
– The proposed OPEN outperforms previous state-of-the-art methods on the nuScenes [1] dataset.

2 Related Work

2D Object Detection with DETR. DETR [2] is the pioneering work that achieves end-to-end object detection by transformer [34]. DETR takes learnable object queries as the query and image features as the key and value for the transformer. To achieve end-to-end detection, DETR utilizes Hungarian Matching for ground-truth label assignment. Many works [4,12,20,24,37,43,47] follow the successful architecture of DETR and further improve the performance. For example, Deformable DETR [47] proposed deformable attention, which greatly alleviates the problem of slow convergence of DETR. DINO [43] utilized denoising training to reduce the learning difficulty of Hungarian Matching.

LiDAR-based 3D Object Detection. According to the point cloud processing method, LiDAR-based 3D object detectors can be divided into point-based and voxel-based. Point-based 3D object detectors [5, 6, 28, 32, 41, 42, 44] usually randomly sample point clouds and directly consume sampled point clouds for extracting 3D features by a PointNet-like [29] backbone. For voxel-based 3D object detectors [3, 11, 30, 31, 39, 45], these methods first quantify point clouds into regular grids and then utilize a 3D sparse convolution backbone to extract 3D features.

Multi-view 3D Object Detection. Multi-view 3D object detectors consume multi-view images from surrounding camera sensors and detect 3D objects in the 3D space. According to the view transformation paradigm, multi-view 3D object detectors can be divided into LSS-based and transformer-based. For LSS-based multi-view 3D object detectors [9, 14, 15], these methods utilize Lift-Splat-Shot (LSS) operation [27] to convert multi-view image features to bird's-eye view (BEV) feature and BEV-based 3D detection head to detect objects on the BEV feature. For transformer-based methods [16, 21, 35, 38], these methods utilize the transformer to convert multi-view image features to 3D features. For example, BEVFormer [16] utilizes Deformable attention [47] to aggregate multi-view image features to BEV feature. PETR [21] utilizes 3D position embedding to project multi-view image features to 3D space. Depth is important information to achieve accurate 3D object detection. Therefore, many works explore the utilization of depth to improve detection performance. BEVDepth [15] and BEVStereo [14] utilize projected LiDAR points to supervise depth prediction, which is used to enhance the LSS and improve performance. DFA3D [13] proposes the 3D deformable attention based on the predicted depth map to achieve better view transformation. 3DPPE [33] proposes point-aware position embedding by 3D points generated by the predicted depth map and pixel coordinates to 3D space.

However, previous methods ignored the problems of the predicted depth map supervised by projected LiDAR points. In this paper, we propose a new multi-view 3D object detector named OPEN, which utilizes the object-wise position embedding to effectively inject object-wise depth information into the network through our proposed object-wise position embedding and enhance the interaction with object queries in the transformer decoder. Benefiting the object-wise depth information, our OPEN achieves the state-of-the-art performance for 3D object detection on the nuScenes [1] dataset.

3 Method

3.1 Overall Architecture

Many existing works [13, 15, 33] also utilize the projected LiDAR points to supervise the depth prediction. However, the depth supervision obtained from the LiDAR points is usually distributed on the surface of the object, which is not so friendly to existing DETR-based 3D object detectors, and the fine-grained depth estimation of the whole object is more challenging, especially for some distant objects. Therefore, we propose a new multi-view 3D object detector named OPEN, whose overall architecture is presented in Figure 2. Specifically, given N multi-view images, OPEN first utilizes a 2D backbone to extract N multi-view features. Then, for N views, OPEN follow DepthNet [33] to predict the pixel-wise depth map supervised by projected LiDAR points. Next, the object-wise depth encoder (ODE) combines the pixel-wise depth representation and temporal information to reason the object-wise depth accurately. Finally, OPEN encodes the object-wise depth into the transformer decoder through our object-wise position embedding (OPE) to predict final detection results.

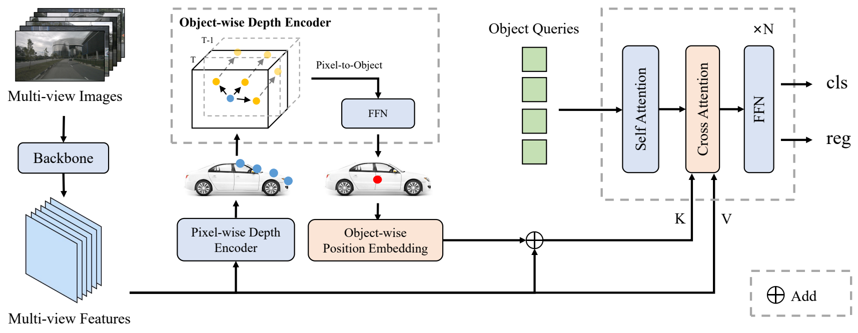

Fig. 2: The overall architecture of the proposed OPEN, which consists of the pixel-wise depth encoder (PDE), the object-wise depth encoder (ODE), and object-wise position embedding (OPE). Specifically, the PDE first utilizes a DepthNet to predict the pixel-wise depth map supervised by projected LiDAR points. Then, the ODE predicts the object-wise depth, supervised by the center of projected 3D bounding boxes, based on the predicted pixel-wise depth map. Finally, OPEN utilizes the object-wise position embedding based on predicted object-wise depth and corresponding 2D object centers to convert the multi-view image features to object-wise 3D features for interaction with object queries and generate final detection results.

3.2 Pixel-wise Depth Encoder

In order to obtain a more accurate object-wise depth prediction, OPEN first utilizes PDE to predict the pixel-wise depth map, which serves as the depth prior of subsequent object-wise depth prediction and is supervised by projected LiDAR points. Given multi-view features $\mathbf{F} = \{\mathbf{F}_i \in \mathbb{R}^{C\times H\times W}, i = 1, 2, 3, ..., N\}$ generated from N multi-view images, where C, H, and W are the channel dimension, height, and width of image features. PDE first encodes camera intrinsic $\mathbf{K} \in \mathbb{R}^{4\times 4}$ by an MLP for modulating $\mathbf{F}$. Then, it utilizes a DepthNet [33], which consists of several residual blocks and deformable convolutions, to predict the pixel-wise depth map. Besides, we follow the fusion depth [33], which combines regression depth and probabilistic depth to generate the final pixel-wise depth map $\mathbf{D}_i \in \mathbb{R}^{H\times W\times 1}$ for the following object-wise depth encoder.

3.3 Object-wise Depth Encoder

After obtaining the pixel-wise depth map, ODE performs an attention-based temporal depth aggregation to predict object-wise depth. The object-wise depth is supervised by the projected 3D ground truth bounding boxes. Given the predicted pixel-wise depth $\mathbf{D}_{(m,n)}$, where (m, n) represents the pixel in the m^{th} row and n^{th} column on the i^{th} image. As shown in Figure 3, ODE converts image pixel coordinates $(u, v)^{\mathrm{T}}$ to the camera coordinate based on the predicted

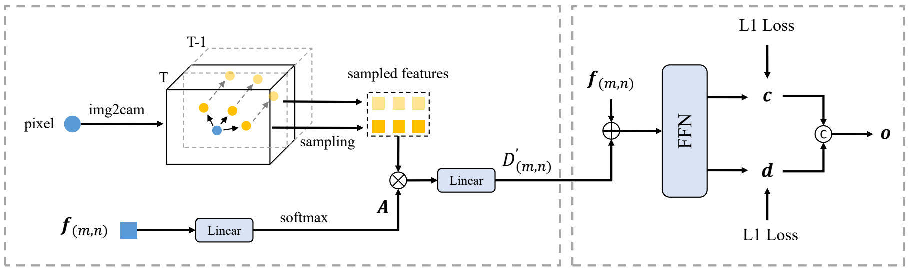

Fig. 3: The overall architecture of the ODE. ODE first converts image pixels from the pixel coordinate to the camera coordinate and aggregates current and historical features to generate depth embedding for object-wise depth prediction by streaming temporal fusion strategy. Finally, ODE utilizes an FFN to predict the object-wise depth d and corresponding object center c based on the depth embedding.

pixel-wise depth $\mathbf{D}_{(m,n)}$. The transformation is formulated as:

$$\mathbf{p}^{'}_{(m,n)} = (u \times \mathbf{D}_{(m,n)}, v \times \mathbf{D}_{(m,n)}, \mathbf{D}_{(m,n)}, 1)^{\mathrm{T}}, \tag{1}$$

$$\mathbf{p}_{(m,n)} = \mathbf{K}^{-1}\mathbf{p}^{'}_{(m,n)}, \tag{2}$$

where $\mathbf{p}_{(m,n)}$ is the projected point on the camera coordinate, which will be used as the reference point. After generating a series of reference points $\mathbf{P} = \{\mathbf{p}_{(m,n)}, m = 1, 2, 3, ..., H \text{ and } n = 1, 2, 3, ..., W\}$. ODE first generates pixel-wise depth embedding and then combines this embedding and image features to predict the object-wise depth. Specifically, given the reference point $\mathbf{p}_{(m,n)}$ and the corresponding image feature $\mathbf{f}_{(m,n)}$, ODE utilizes a linear function to predict k 3D offsets, which are added to $\mathbf{p}_{(m,n)}$ to obtain 3D sampling points. Then, we project 3D sampling points to the pixel coordinate of the current and previous frames. These projected points $\mathbf{p}^{*}_{(m,n)}$ are used to sample corresponding features. Meanwhile, the attention weight $\mathbf{A}$ is predicted by feeding $\mathbf{f}_{(m,n)}$ into a linear and softmax function. Finally, the sampled features $\mathbf{f}_s$ is multiplied with $\mathbf{A}$ for generating pixel-wise depth embedding $\mathbf{E}_{(m,n)}$. We can formulate the above process as follows:

$$\mathbf{E}_{(m,n)} = \phi(\sum_{j=1}^{k} \mathbf{A}_j \cdot \text{Concat}(\mathbf{F}_i(\mathbf{p}^{*}_{(m,n)}), \mathbf{F}^{'}_i(\mathbf{p}^{*}_{(m,n)}))), \tag{3}$$

where ϕ is a linear function. Concat denotes the concatenation operation. $\mathbf{F}_i(*)$ and $\mathbf{F}^{'}_i(*)$ denote sampling the corresponding features of the coordinates $*$ on the current image features $\mathbf{F}$ and previous image features $\mathbf{F}^{'}$ by a bilinear interpolation operation, respectively.

Finally, the pixel-wise depth embedding $\mathbf{E}_{(m,n)}$ and image features $\mathbf{F}_i$ are fed into a FFN to predict the object-wise depth $\mathbf{d} \in \mathbb{R}^{(H \times W) \times 1}$ and corresponding object center $\mathbf{c} \in \mathbb{R}^{(H \times W) \times 2}$ on the image.

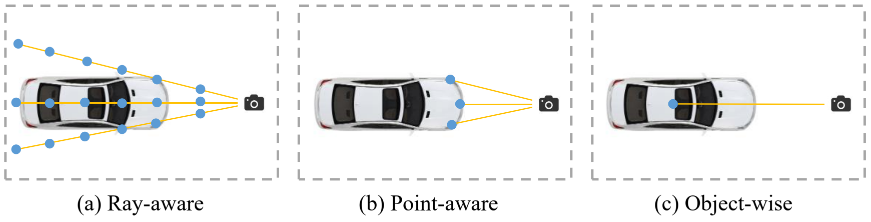

Fig. 4: Comparison of the ray-aware position embedding (a), point-aware position embedding (b), and the object-wise position embedding (c). Compared with other methods, OPE utilizes the 3D object center to generate the position embedding, which can achieve better 3D representation.

3.4 Object-wise Position Embedding

To utilize the generated object-wise depth, we present OPE to encode the object-wise depth into the transformer decoder to enhance 3D-aware features and improve the performance of multi-view 3D object detection. Next, we will discuss the differences between the different position embedding methods and introduce the details of OPE.

Discussion of Different Position Embedding. As shown in Figure 4, we compare the ray-aware position embedding of StreamPETR [35], the point-aware position embedding of 3DPPE [33], and our object-wise position embedding (OPE). The ray-aware position embedding (a) first generates a 3D mesh grid in the camera frustum space based on depth candidates and then converts these points to the LiDAR coordinate as the position embedding. The point-aware position embedding (b) first utilizes a DepthNet [33] to predict the pixel-wise depth map and then converts these 3D points generated by the predicted depth map to the LiDAR coordinate as the position embedding. Although these methods achieve satisfactory performance, there are still ignored problems. For ray-aware position embedding, the uncertainty depth estimation without depth supervision makes it difficult to generate accurate 3D-aware features. For point-aware position embedding, although it adopts projected LiDAR points to supervise the pixel-wise depth prediction and encodes 3D points for position embedding to improve performance, it ignores the importance of object-wise depth for DETR-based 3D object detectors, leading to sub-optimal performance. Therefore, we propose object-wise position embedding (OPE) to alleviate the above problems. Compared with other methods, the OPE can generate more accurate 3D-aware features based on object-wise depth representation, resulting in better detection performance.

Details of OPE. Given the j^{th} object-wise depth d_j and corresponding object center (x, y) predicted by ODE on the i^{th} image, OPE generates object position $\mathbf{o}_j = (x, y, d_j)$. After generating $\mathbf{o}_j$, OPE converts $\mathbf{o}_j$ in the pixel coordinate to the 3D object center $\mathbf{O}_j$ in the LiDAR coordinate:

$$\mathbf{o}_j^{'} = (x \times d_j, y \times d_j, d_j, 1)^{\mathrm{T}}, \tag{4}$$

$$\mathbf{O}_j = \mathbf{R}^{-1}\mathbf{K}^{-1}\mathbf{o}_j^{'}, \tag{5}$$

where $\mathbf{K}$ and $\mathbf{R}$ denote the camera intrinsic and extrinsic. Then, OPE normalizes the 3D object center by the perception range and utilizes a 3D position embedding operation to encode the 3D object center in a cosine manner. Finally, OPE adopts a multi-layer perceptron (MLP) to generate the object-wise position embedding:

$$\mathbf{OPE}_j = \mathrm{MLP}((\mathrm{PE}_{3D}(\mathrm{Norm}(\mathbf{O}_j)))), \tag{6}$$

where $\mathbf{OPE}_j$, Norm, and PE_{3D} denote the j^{th} object-wise position embedding, normalization, and 3D position embedding operation respectively. Finally, we add $\mathbf{OPE}_j$ to the j^{th} image feature for feature interaction with learnable object queries in the transformer decoder.

3.5 Depth-aware Focal Loss

DFL aims to further encourage OPEN to pay more attention to the object center. Different from the traditional focal loss [17], DFL takes the depth score s as the classification label to generate a soft label for focal loss. Specifically, given the predicted 3D object center $\hat{\mathbf{C}}$. the corresponding ground truth $\mathbf{C}$, the binary target class label $\mathbf{t}$, and the predicted classification probability $\hat{\mathbf{p}}$, then the DFL is formulated as:

$$\mathcal{L}_{DFL} = -\boldsymbol{\alpha}^{'} \cdot |\mathbf{t} \cdot \mathbf{s} - \hat{\mathbf{p}}|^{\gamma} \cdot \log(|1 - \mathbf{t} - \hat{\mathbf{p}}|), \tag{7}$$

where $\mathbf{s} = e^{-\mathrm{L2}(\hat{\mathbf{C}} - \mathbf{C})}$, $\boldsymbol{\alpha}^{'} = \alpha \cdot \mathbf{t} \cdot \mathbf{s} + (1 - \alpha) \cdot (1 - \mathbf{t} \cdot \mathbf{s})$. L2 is the euclidean distance. α and γ are hyper-parameters.
Total Loss. Given the depth-aware focal loss $\mathcal{L}_{DFL}$, 3D bounding box regression loss $\mathcal{L}_{reg}$, the pixel-wise depth prediction loss $\mathcal{L}_{PDE}$, and the object-wise depth prediction loss $\mathcal{L}_{ODE}$, we adopt the Hungarian Matching to achieve label assignment and the total loss can be calculated as:

$$\mathcal{L} = \lambda_1\mathcal{L}_{PDE} + \lambda_2\mathcal{L}_{ODE} + \lambda_3\mathcal{L}_{DFL} + \lambda_4\mathcal{L}_{reg}, \tag{8}$$

where λ_1, λ_2, λ_3, and λ_4 are hyper-parameter to balance different losses.

4 Experiments

4.1 Datasets and Metrics

We validate our method on the nuScenes [1] benchmark. nuScenes is the widely used autonomous driving dataset for multi-view 3D object detection. The nuScenes dataset consists of 1000 scenes, which are divided into three parts: 750 for training, 150 for validation, and 150 for testing, where each scene is roughly 20s long and annotated at 2Hz. The nuScenes dataset provides point clouds collected by 32-beam LiDAR and 6 multi-view images collected by 6 surrounding cameras. Besides, the evaluation of the nuScnes dataset adopts the Mean Average Precision (mAP) and nuScenes detection score (NDS) to evaluate the performance of 3D detectors for 10 foreground classes. Consistent with previous methods, we report NDS and mAP, along with mean Average Translation Error (mATE), mean Average Scale Error (mASE), mean Average Orientation Error(mAOE), mean Average Velocity Error(mAVE), and mean Average Attribute Error(mAAE).

4.2 Implementation Details

We follow StreamPETR [35] to conduct experiments with ResNet50 [7], ResNet101, and V2-99 [36] backbones on the nuScenes [1] dataset without any test-time augmentation or future information. OPEN is trained with the nuImages [1] pre-trained model for the nuScenes `val` set and with the DD3D [25] pre-trained model for the nuScenes `test` set. For the pixel-wise depth encoder, we take the $8\times$ downsampled depth map generated by projected LiDAR points as the supervision. For the object-wise depth encoder, we set the number of prediction offsets k to 13. For the depth-aware focal loss, we set α to 0.25 and γ to 2.0. For the balancing factors of different losses, we set λ_1, λ_2, λ_3, and λ_4 to 1.0, 5.0, 2.0 and 0.25.

For training OPEN, we use the same data augmentation methods as PETR [21] and AdamW [23] optimizer with a batch size of 16 on 8 NVIDIA Tesla V100 GPUs without CBGS [46] strategy. For comparison with other methods on the nuScenes `val` set, we use the streaming video training method [35] to train OPEN for 90 epochs with a starting learning rate of $4e^{-4}$ and cosine annealing policy. It is worth noting that compared with the sliding window training method, the streaming video training method needs more epochs for convergence but still saves much training time. For comparison with other methods on the nuScenes `test` set, we train 60 epochs to prevent over-fitting.

4.3 Main Results

We compare the proposed OPEN with previous state-of-the-art multi-view 3D object detectors on the nuScenes `val` and `test` sets. As shown in Table 1, for nuScenes `val` set, OPEN achieves 56.4% NDS and 46.5% mAP performance with an image size of 256×704 and ResNet50 backbone pre-trained on nuImages, which outperforms the state-of-the-art method (SparseBEV) by 0.6% NDS and 0.7% mAP, and outperforms our baseline (StreamPETR) by 1.4% NDS and 1.5% mAP. When adopting an image size of 512×1408 and ResNet101 backbone pre-trained on nuImage, OPEN achieves 60.6% NDS and 51.6% mAP performance, which yields a new state-of-the-art result on the nuScenes `val` set. OPEN exceeds the state-of-the-art method (Far3D) by 1.2% NDS and 0.6% mAP. Finally, our

Table 1: Comparison of other methods on the nuScenes `val` set. † The backbone benefits from perspective pretraining.

Method	Backbone	Input Size	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
BevDet4D [8]	ResNet50	256×704	45.7	32.2	0.703	0.278	0.495	0.354	0.206
PETRv2 [22]	ResNet50	256×704	45.6	34.9	0.700	0.275	0.580	0.437	0.187
BEVDepth [15]	ResNet50	256×704	47.5	35.1	0.639	0.267	0.479	0.428	0.198
BEVStereo [14]	ResNet50	256×704	50.0	37.2	0.598	0.270	0.438	0.367	0.190
BEVFormerv2† [40]	ResNet50	-	52.9	42.3	0.618	0.273	0.413	0.333	0.188
SOLOFusion [26]	ResNet50	256×704	53.4	42.7	0.567	0.274	0.511	0.252	0.181
Sparse4Dv2 [18]	ResNet50	256×704	53.8	43.9	0.598	0.270	0.475	0.282	0.179
StreamPETR† [35]	ResNet50	256×704	55.0	45.0	0.613	0.267	0.413	0.265	0.196
SparseBEV† [19]	ResNet50	256×704	55.8	44.8	0.581	0.271	0.373	0.247	0.190
OPEN†	ResNet50	256×704	**56.4**	**46.5**	0.573	0.275	0.413	0.235	0.193
3DPPE [33]	ResNet101	512×1408	45.8	39.1	0.674	0.282	0.395	0.830	0.191
BEVDepth [15]	ResNet101	512×1408	53.5	41.2	0.565	0.266	0.358	0.331	0.190
SOLOFusion [26]	ResNet101	512×1408	58.2	48.3	0.503	0.264	0.381	0.246	0.207
SparseBEV† [19]	ResNet101	512×1408	59.2	50.1	0.562	0.265	0.321	0.243	0.195
StreamPETR† [35]	ResNet101	512×1408	59.2	50.4	0.569	0.262	0.315	0.257	0.199
Sparse4Dv2† [18]	ResNet101	512×1408	59.4	50.5	0.548	0.268	0.348	0.239	0.184
Far3D† [10]	ResNet101	512×1408	59.4	51.0	0.551	0.258	0.372	0.238	0.195
OPEN†	ResNet101	512×1408	**60.6**	**51.6**	0.528	0.266	0.312	0.222	0.190

Table 2: Comparison of other methods on nuScenes `test` set. These results are reported without test-time augmentation, model ensembling, and any future information.

Method	Backbone	Input Size	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
BEVDepth [15]	V2-99	640×1600	60.0	50.3	0.445	0.245	0.378	0.320	0.126
BEVStereo [14]	V2-99	640×1600	61.0	52.5	0.431	0.246	0.358	0.357	0.138
CAPE-T [38]	V2-99	640×1600	61.0	52.5	0.503	0.242	0.361	0.306	0.114
FB-BEV [19]	V2-99	640×1600	62.4	53.7	0.439	0.250	0.358	0.270	0.128
HoP [48]	V2-99	640×1600	61.2	52.8	0.491	0.242	0.332	0.343	0.109
StreamPETR [35]	V2-99	640×1600	63.6	55.0	0.479	0.239	0.317	0.241	0.119
SparseBEV [19]	V2-99	640×1600	63.6	55.6	0.485	0.244	0.332	0.246	0.117
Sparse4Dv2 [18]	V2-99	640×1600	63.8	55.6	0.462	0.238	0.328	0.264	0.115
OPEN	V2-99	640×1600	**64.4**	**56.7**	0.456	0.244	0.325	0.240	0.129

OPEN has an obvious performance improvement over our baseline with NDS of 1.4% and mAP of 1.2%.

Furthermore, we provide the detection results on nuScenes `test` set, shown in Table 2. We observe that OPEN achieves 64.4% NDS and 56.7% mAP under the image size of 640×1600 based on V2-99 [36] backbone pre-trained on DD3D, which yields a new state-of-the-art result on the nuScenes `test` set.These experiments clearly demonstrate the effectiveness of our method.

4.4 Ablation Study

In this section, we conduct ablation studies to investigate the effectiveness of OPEN on the nuScenes `val` set. If not specified, we adopt the V2-99 backbone

Table 3: Ablation studies for each component in OPEN on the nuScenes `val` set. The PDE, ODE, OPE, and DFL represent the pixel-wise depth encoder, object-wise depth encoder, object-wise position embedding, and depth-aware focal loss, respectively.

#	PDE	ODE	OPE	DFL	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
I					59.4	50.3	0.575	0.258	0.300	0.243	0.196
II	✓				59.4	50.5	0.564	0.257	0.320	0.252	0.190
III	✓	✓			59.7	50.6	0.568	0.257	0.305	0.245	**0.187**
IV	✓	✓	✓		60.8	**52.4**	0.553	0.258	0.291	0.242	0.197
V	✓	✓	✓	✓	**61.3**	52.1	**0.525**	**0.256**	**0.281**	**0.216**	0.199

Table 4: Ablation studies of the object-wise depth encoder on the nuScenes `val` set. The temporal denotes the utilization of temporal information in ODE.

temporal	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
	61.0	51.7	0.530	**0.256**	**0.280**	0.218	0.201
✓	**61.3**	**52.1**	**0.525**	**0.256**	0.281	**0.216**	**0.199**

pre-trained on DD3D with an image size of 320×800. Here, we train OPEN for 24 epochs in ablation studies.

Effectiveness of Each Component. As shown in Table 3, we adopt the StreamPETR [35] as our baseline (I) and conduct ablation studies on each component in OPEN. Compared with the baseline, the PDE (II) only brings 0.2% mAP performance improvements, indicating that pixel-wise depth supervision can not significantly boost detection performance. Compared with (II), the ODE (III) further brings 0.3% NDS and 0.1% mAP performance improvements. Compared with (III), when adopting the OPE (IV) in the transformer decoder, the performance is further improved by 1.1% NDS and 1.8% mAP, demonstrating the effectiveness of encoding object-wise depth information into the network by our proposed OPE. Compared with (IV), the DFL (V) can further bring 0.5% NDS performance improvements by paying more attention to the 3D object center information. Finally, combining all components, OPEN achieves significant performance improvements (1.9% NDS and 1.8% mAP) over the baseline. These experiments demonstrate the effectiveness of the proposed component.

Ablations of the ODE. The object-wise depth encoder combines the temporal information to predict the object-wise depth accurately based on the pixel-wise depth map. Here, we explore the effect of the temporal information. As shown in Table 4, when ODE disables the temporal information, we find there is a 0.3% NDS and 0.4% mAP performance drop. We think that the temporal information is helpful to improve the accuracy of the object-wise depth prediction. This experiment effectively illustrates the necessity of our ODE by utilizing temporal information.

Effectiveness of the OPE. To further verify the effectiveness of OPE, we compare different position embedding methods on the nuScenes `val` set. For

Table 5: Comparison of other position embedding methods on the nuScenes `val` set. The Ray-aware PE, Point-aware PE, and OPE denote the ray-aware position embedding, point-aware position embedding, and object-wise position embedding, respectively.

Method	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
Ray-aware PE [35]	59.4	50.3	0.575	**0.258**	0.300	0.243	**0.196**
Point-aware PE [33]	60.0 (+0.6)	51.6 (+1.3)	0.568	**0.258**	0.306	0.245	0.203
OPE	**60.8** (**+1.4**)	**52.4** (**+2.1**)	**0.553**	**0.258**	**0.291**	**0.242**	0.197

Table 6: Comparison of other position embedding methods on the nuScenes `val` set in different distances. $\text{NDS}_{>0}$, $\text{NDS}_{>20}$, and $\text{NDS}_{>40}$ represent different evaluation metrics under distance thresholds of 0, 20, and 40 meters, respectively.

Method	$\text{NDS}_{>0}$↑	$\text{NDS}_{>20}$↑	$\text{NDS}_{>40}$↑
Ray-aware PE [35]	59.4	49.5	36.8
Point-aware PE [33]	60.0 (+0.6)	50.4 (+0.9)	37.9 (+1.1)
OPE	**60.8** (**+1.4**)	**51.0** (**+1.5**)	**39.1** (**+2.3**)

StreamPETR [35], it generates a 3D mesh grid in the camera frustum space based on depth candidates and converts these points to the LiDAR coordinate as the ray-aware position embedding. For 3DPPE [33], it generates 3D points based on the predicted pixel-wise depth map and converts these points to the LiDAR coordinate as the point-aware position embedding. For OPE, it generates object centers based on predicted object-wise depth and converts them to the LiDAR coordinate as the object-wise position embedding. As shown in Table 5, when adopting the ray-aware position embedding (I), the performance is 59.4% NDS and 50.3% mAP. The point-aware position embedding (II) can bring 0.6% NDS and 1.3% mAP compared with the ray-aware position embedding. After utilizing the proposed object-wise position embedding (III), there are 1.4% NDS and 2.1% mAP performance improvements compared with the ray-aware position embedding. Moreover, our OPE outperforms the point-aware position embedding with 0.8% NDS and 0.8% mAP, which illustrates the effectiveness and superiority of OPE for enhancing detection performance.

Effectiveness of the OPE for distant objects. To verify the effectiveness of the OPE for distant objects, we compare different position embedding methods based on distance on the nuScenes `val` set. Specifically, we set the minimum distance thresholds between the ground truth and ego as 0, 20, and 40 meters, respectively. As shown in Table 6, when the thresholds are set to 0, 20, and 40 meters, OPE can bring 1.4% NDS, 1.5% NDS, and 2.3% NDS performance improvement, respectively. These experiments indicate that as the distance increases, the performance gain brought by OPE gradually becomes larger compared with the ray-aware position embedding of StreamPETR. Besides, when the thresholds are set to 40 meters, OPE outperforms the point-aware position

Table 7: Ablation studies for the necessity of pixel-wise depth encoder on the nuScenes `val` set. The PDE represents the pixel-wise depth encoder.

PDE	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
	59.9	49.8	0.538	0.258	0.282	0.221	**0.199**
✓	**61.3**	**52.1**	**0.525**	**0.256**	**0.281**	**0.216**	**0.199**

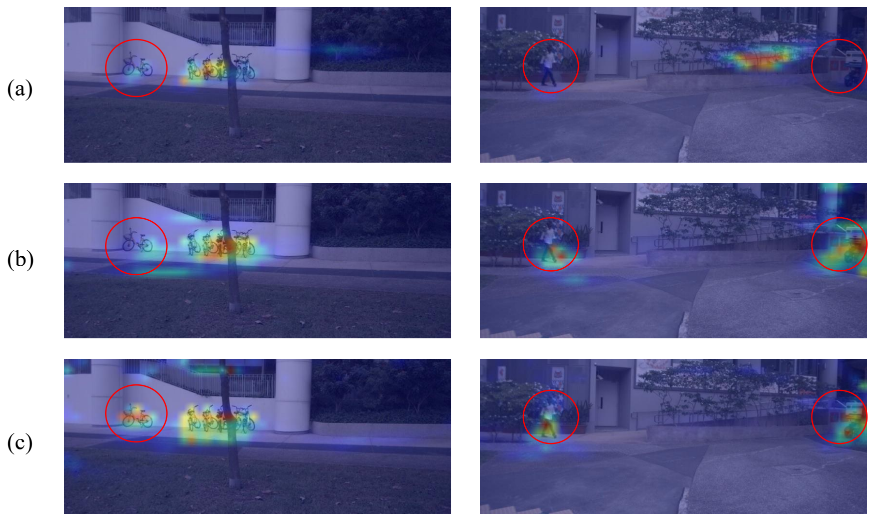

Fig. 5: Comparison of attention weight maps for the ray-aware position embedding of StreamPETR (a), the point-aware position embedding of 3DPPE (b), and the object-wise position embedding (c) on the nuScenes `val` set. After utilizing OPE, our OPEN can generate better attention weight maps for some hard-detected objects, which are highlighted by red circles.

embedding of 3DPPE by 1.2% NDS performance improvement. Therefore, these experiments demonstrate the importance of OPE for distant objects.

Ablations of the PDE. We conduct ablation studies to demonstrate the necessity of pixel-wise depth encoder (PDE) for accurate object-wise depth prediction in OPEN. As shown in Table 7, when PDE is not used, we find there is a 1.4% NDS and 2.3% mAP performance drop. We think taking pixel-wise depth as the prior is necessary to obtain accurate object-wise depth. This experiment illustrates the necessity of PDE.

4.5 Visualization

Attention Map Comparison. To explore the influence of different position embedding methods for the transformer decoder, we visualize the attention weight map of the last transformer decoder layer for the ray-aware position em-

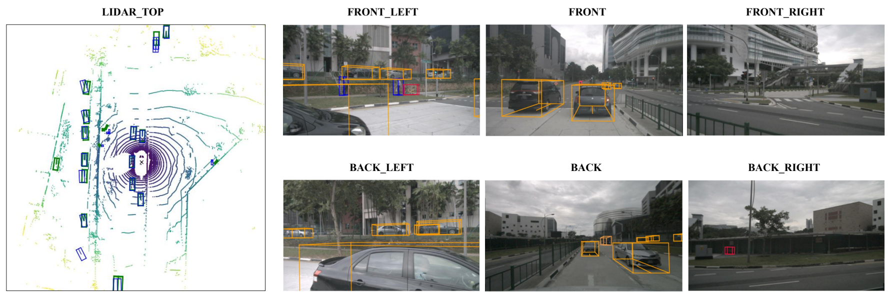

Fig. 6: Qualitative detection results on multi-view images and the BEV space on the nuScenes `val` set. The 3D predicted bounding boxes are drawn with different colors for different classes on multi-view images. Blue and green boxes are the prediction and ground truth boxes on the BEV space.

bedding of StreamPETR [35], the point-aware position embedding of 3DPPE [33], and our OPE. As shown in Figure 5, compared with other methods, our OPE has higher attention weights for some hard-detected objects, which are highlighted by red circles. Furthermore, the attention weight map obtained by OPE is more focused than point-aware position embedding. These visualization results indicate the effectiveness and superiority of OPE compared with other position embedding methods.

Qualitative Results. As shown in Figure 6, we show the qualitative detection results of OPEN on multi-view images and BEV space. We can observe that our OPEN can detect some distant objects successfully. More qualitative results are provided in the supplemental material.

5 Conclusion

In this paper, we have proposed a new multi-view 3D object detector named OPEN, which mainly introduces the object-wise depth information to multi-view 3D object detection for 3D object-aware feature representation by our proposed object-wise position embedding. Specifically, OPEN first employs an object-wise depth encoder, which combines temporal information and takes the pixel-wise depth map as a prior, to accurately estimate the object-wise depth. Then, OPEN utilizes the proposed object-wise position embedding to encode the object-wise depth information into the transformer decoder, thereby producing 3D object-aware features for final detection. Besides, we design the depth-aware focal loss to encourage OPEN to pay more attention to the 3D object center information. Extensive experiments demonstrate the effectiveness of each component of our OPEN. Furthermore, OPEN has achieved state-of-the-art detection performance on the nuScenes dataset. Finally, we hope OPEN could further promote the research of depth in multi-view 3D object detection.

Acknowledgements

This work was supported by the National Science Fund for Distinguished Young Scholars of China (Grant No.62225603) and the Hubei Key R&D Program (Grant No.2022BAA078).

References

1. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: CVPR. pp. 11621–11631 (2020)
2. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV. pp. 213–229. Springer (2020)
3. Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., Li, H.: Voxel r-cnn: Towards high performance voxel-based 3d object detection. In: AAAI. vol. 35, pp. 1201–1209 (2021)
4. Gao, P., Zheng, M., Wang, X., Dai, J., Li, H.: Fast convergence of detr with spatially modulated co-attention. In: ICCV. pp. 3621–3630 (2021)
5. He, C., Li, R., Li, S., Zhang, L.: Voxel set transformer: A set-to-set approach to 3d object detection from point clouds. In: CVPR. pp. 8417–8427 (2022)
6. He, C., Zeng, H., Huang, J., Hua, X.S., Zhang, L.: Structure aware single-stage 3d object detection from point cloud. In: CVPR. pp. 11873–11882 (2020)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
8. Huang, J., Huang, G.: Bevdet4d: Exploit temporal cues in multi-camera 3d object detection. arXiv preprint arXiv:2203.17054 (2022)
9. Huang, J., Huang, G., Zhu, Z., Ye, Y., Du, D.: Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. arXiv preprint arXiv:2112.11790 (2021)
10. Jiang, X., Li, S., Liu, Y., Wang, S., Jia, F., Wang, T., Han, L., Zhang, X.: Far3d: Expanding the horizon for surround-view 3d object detection. In: AAAI (2024)
11. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: CVPR (2019)
12. Li, F., Zhang, H., Liu, S., Guo, J., Ni, L.M., Zhang, L.: Dn-detr: Accelerate detr training by introducing query denoising. In: CVPR (2022)
13. Li, H., Zhang, H., Zeng, Z., Liu, S., Li, F., Ren, T., Zhang, L.: Dfa3d: 3d deformable attention for 2d-to-3d feature lifting. In: ICCV. pp. 6684–6693 (2023)
14. Li, Y., Bao, H., Ge, Z., Yang, J., Sun, J., Li, Z.: Bevstereo: Enhancing depth estimation in multi-view 3d object detection with temporal stereo. In: AAAI. vol. 37, pp. 1486–1494 (2023)
15. Li, Y., Ge, Z., Yu, G., Yang, J., Wang, Z., Shi, Y., Sun, J., Li, Z.: Bevdepth: Acquisition of reliable depth for multi-view 3d object detection. In: AAAI. vol. 37, pp. 1477–1485 (2023)
16. Li, Z., Wang, W., Li, H., Xie, E., Sima, C., Lu, T., Qiao, Y., Dai, J.: Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In: ECCV. pp. 1–18. Springer (2022)
17. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV. pp. 2980–2988 (2017)

18. Lin, X., Lin, T., Pei, Z., Huang, L., Su, Z.: Sparse4d v2: Recurrent temporal fusion with sparse model. arXiv preprint arXiv:2305.14018 (2023)
19. Liu, H., Teng, Y., Lu, T., Wang, H., Wang, L.: Sparsebev: High-performance sparse 3d object detection from multi-camera videos. In: ICCV. pp. 18580–18590 (2023)
20. Liu, S., Li, F., Zhang, H., Yang, X., Qi, X., Su, H., Zhu, J., Zhang, L.: Dab-detr: Dynamic anchor boxes are better queries for detr. In: ICLR (2022)
21. Liu, Y., Wang, T., Zhang, X., Sun, J.: Petr: Position embedding transformation for multi-view 3d object detection. In: ECCV. pp. 531–548. Springer (2022)
22. Liu, Y., Yan, J., Jia, F., Li, S., Gao, A., Wang, T., Zhang, X.: Petrv2: A unified framework for 3d perception from multi-camera images. In: ICCV. pp. 3262–3272 (2023)
23. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: arXiv preprint arXiv:1711.05101 (2017)
24. Meng, D., Chen, X., Fan, Z., Zeng, G., Li, H., Yuan, Y., Sun, L., Wang, J.: Conditional detr for fast training convergence. In: ICCV (2021)
25. Park, D., Ambrus, R., Guizilini, V., Li, J., Gaidon, A.: Is pseudo-lidar needed for monocular 3d object detection? In: ICCV. pp. 3142–3152 (2021)
26. Park, J., Xu, C., Yang, S., Keutzer, K., Kitani, K., Tomizuka, M., Zhan, W.: Time will tell: New outlooks and a baseline for temporal multi-view 3d object detection. In: ICLR (2022)
27. Philion, J., Fidler, S.: Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In: ECCV. pp. 194–210. Springer (2020)
28. Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep hough voting for 3d object detection in point clouds. ICCV (2019)
29. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: CVPR. pp. 652–660 (2017)
30. Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., Li, H.: Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In: CVPR (2020)
31. Shi, S., Jiang, L., Deng, J., Wang, Z., Guo, C., Shi, J., Wang, X., Li, H.: Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection. IJCV (2021)
32. Shi, S., Wang, X., Li, H.: Pointrcnn: 3d object proposal generation and detection from point cloud. In: CVPR (2019)
33. Shu, C., Deng, J., Yu, F., Liu, Y.: 3dppe: 3d point positional encoding for transformer-based multi-camera 3d object detection. In: ICCV. pp. 3580–3589 (2023)
34. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: NeurIPS. vol. 30 (2017)
35. Wang, S., Liu, Y., Wang, T., Li, Y., Zhang, X.: Exploring object-centric temporal modeling for efficient multi-view 3d object detection. In: ICCV (2023)
36. Wang, T., Zhu, X., Pang, J., Lin, D.: Fcos3d: Fully convolutional one-stage monocular 3d object detection. In: ICCV. pp. 913–922 (2021)
37. Wang, Y., Zhang, X., Yang, T., Sun, J.: Anchor detr: Query design for transformer-based detector. In: AAAI (2022)
38. Xiong, K., Gong, S., Ye, X., Tan, X., Wan, J., Ding, E., Wang, J., Bai, X.: Cape: Camera view position embedding for multi-view 3d object detection. In: CVPR. pp. 21570–21579 (2023)
39. Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors **18**(10), 3337 (2018)

40. Yang, C., Chen, Y., Tian, H., Tao, C., Zhu, X., Zhang, Z., Huang, G., Li, H., Qiao, Y., Lu, L., et al.: Bevformer v2: Adapting modern image backbones to bird's-eye-view recognition via perspective supervision. In: CVPR. pp. 17830–17839 (2023)
41. Yang, Z., Sun, Y., Liu, S., Jia, J.: 3dssd: Point-based 3d single stage object detector. In: CVPR. pp. 11040–11048 (2020)
42. Yang, Z., Sun, Y., Liu, S., Shen, X., Jia, J.: Std: Sparse-to-dense 3d object detector for point cloud. In: CVPR. pp. 1951–1960 (2019)
43. Zhang, H., Li, F., Liu, S., Zhang, L., Su, H., Zhu, J., Ni, L.M., Shum, H.Y.: Dino: Detr with improved denoising anchor boxes for end-to-end object detection. ICLR (2023)
44. Zhang, Y., Hu, Q., Xu, G., Ma, Y., Wan, J., Guo, Y.: Not all points are equal: Learning highly efficient point-based detectors for 3d lidar point clouds. In: CVPR. pp. 18953–18962 (2022)
45. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: CVPR (2018)
46. Zhu, B., Jiang, Z., Zhou, X., Li, Z., Yu, G.: Class-balanced grouping and sampling for point cloud 3d object detection. arXiv preprint arXiv:1908.09492 (2019)
47. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. In: ICLR (2021)
48. Zong, Z., Jiang, D., Song, G., Xue, Z., Su, J., Li, H., Liu, Y.: Temporal enhanced training of multi-view 3d object detector via historical object prediction. In: ICCV (2023)