PETR: Position Embedding Transformation for Multi-View 3D Object Detection

Yingfei Liu* Tiancai Wang* Xiangyu Zhang Jian Sun

MEGVII Technology
{liuyingfei,wangtiancai,zhangxiangyu,sunjian}@megvii.com

Abstract. In this paper, we develop position embedding transformation (PETR) for multi-view 3D object detection. PETR encodes the position information of 3D coordinates into image features, producing the 3D position-aware features. Object query can perceive the 3D position-aware features and perform end-to-end object detection. PETR achieves state-of-the-art performance (**50.4%** NDS and **44.1%** mAP) on standard nuScenes dataset and ranks 1*st* place on the benchmark. It can serve as a simple yet strong baseline for future research. Code is available at https://github.com/megvii-research/PETR.

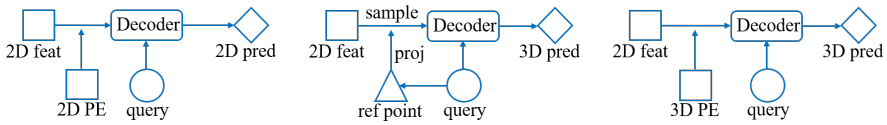
Keywords: Position embedding, transformer, 3D object detection

1 Introduction

3D object detection from multi-view images is appealing due to its low cost in autonomous driving system. Previous works [6,33,49,34,48] mainly solved this problem from the perspective of monocular object detection. Recently, DETR [4] has gained remarkable attention due to its contribution on end-to-end object detection. In DETR [4], each object query represents an object and interacts with the 2D features in transformer decoder to produce the predictions (see Fig. 1(a)). Simply extended from DETR [4] framework, DETR3D [51] provides an intuitive solution for end-to-end 3D object detection. The 3D reference point, predicted by object query, is projected back into the image spaces by the camera parameters and used to sample the 2D features from all camera views (see Fig. 1(b)). The decoder will take the sampled features and the queries as input and update the representations of object queries.

However, such 2D-to-3D transformation in DETR3D [51] may introduce several problems. First, the predicted coordinates of reference point may not that accurate, making the sampled features out of the object region. Second, only the image feature at the projected point will be collected, which fails to perform the representation learning from global view. Also, the complex feature sampling procedure will hinder the detector from practical application. Thus, building an end-to-end 3D object detection framework without the online 2D-to-3D transformation and feature sampling is still a remaining problem.

* Equal contribution.



(a) DETR (b) DETR3D (c) PETR

Fig. 1: Comparison of DETR, DETR3D, and our proposed PETR. (a) In DETR, the object queries interact with 2D features to perform 2D detection. (b) DETR3D repeatedly projects the generated 3D reference points into image plane and samples the 2D features to interact with object queries in decoder. (c) PETR generates the 3D position-aware features by encoding the 3D position embedding (3D PE) into 2D image features. The object queries directly interact with 3D position-aware features and output 3D detection results.

In this paper, we aim to develop a simple and elegant framework based on DETR [4] for 3D object detection. We wonder if it is possible that we transform the 2D features from multi-view into 3D-aware features. In this way, the object query can be directly updated under the 3D environment. Our work is inspired by these advances in implicit neural representation [17,8,32]. In MetaSR [17] and LIFF [8], the high-resolution (HR) RGB values are generated from low-resolution (LR) input by encoding HR coordinates information into the LR features. In this paper, we try to transform the 2D features from multi-view images into the 3D representation by encoding 3D position embedding (see Fig. 1(c)).

To achieve this goal, the camera frustum space, shared by different views, is first discretized into meshgrid coordinates. The coordinates are then transformed by different camera parameters to obtain the coordinates of 3D world space. Then 2D image features extracted from backbone and 3D coordinates are input to a simple 3D position encoder to produce the 3D position-aware features. The 3D position-aware features will interact with the object queries in transformer decoder and the updated object queries are further used to predict the object class and the 3D bounding boxes.

The proposed PETR architecture brings many advantages compared to the DETR3D [51]. It keeps the end-to-end spirit of original DETR [4] while avoiding the complex 2D-to-3D projection and feature sampling. During inference time, the 3D position coordinates can be generated in an offline manner and served as an extra input position embedding. It is relatively easier for practical application.

To summarize, our contributions are:

- We propose a simple and elegant framework, termed PETR, for multi-view 3D object detection. The multi-view features are transformed into 3D domain by encoding the 3D coordinates. Object queries can be updated by interacting with the 3D position-aware features and generate 3D predictions.
- A new 3D position-aware representation is introduced for multi-view 3D object detection. A simple implicit function is introduced to encode the 3D position information into 2D multi-view features.

– Experiments show that PETR achieves state-of-the-art performance (**50.4%** NDS and **44.1%** mAP) on standard nuScenes dataset and ranks 1*st* place on 3D object detection leaderboard.

2 Related Work

2.1 Transformer-based Object Detection

Transformer [47] is an attention block that widely applied to model the long-range dependency. In transformer, the features are usually added with position embedding, which provides the position information of the image [13,53,27], sequence [15,47,11,10,54], and video [1,24,52]. Transformer-XL [10] uses the relative position embedding to encode the relative distance of the pairwise tokens. ViT [13] adds the learned position embedding to the patch representations that encode distance of different patches. MViT [24] decomposes the distance computation of the relative position embedding and model the space-time structure.

Recently, DETR [4] involves the transformer into 2D object detection task for end-to-end detection. In DETR [4], each object is represented as an object query which interacts with 2D images features through transformer decoder. However, DETR [4] suffers from the slow convergence. [44] attributes the slow convergence to the cross attention mechanism and designs a encoder-only DETR. Furthermore, many works accelerate the convergence by adding position priors. SMAC [14] predicts 2D Gaussian-like weight map as spatial prior for each query. Deformable DETR [58] associates the object queries with 2D reference points and proposes deformable cross-attention to perform sparse interaction. [50,30,26] generate the object queries from anchor points or anchors that use position prior for fast convergence. Extended from DETR [58], SOLQ [12] uses object queries to perform classification, box regression and instance segmentation simultaneously.

2.2 Vision-based 3D Object Detection

Vision-based 3D object detection is to detect 3D bounding boxes from camera images. Many previous works [6,33,20,21,41,19,2,49,48] perform 3D object detection in the image view. M3D-RPN [2] introduces the depth-aware convolution, which learns position-aware features for 3D object detection. FCOS3D [49] transforms the 3D ground-truths to image view and extends FCOS [46] to predict 3D cuboid parameters. PGD [48] follows the FCOS3D [49] and uses a probabilistic representation to capture the uncertainty of depth. It greatly alleviates the depth estimation problem while introducing more computation budget and larger inference latency. DD3D [34] shows that depth pre-training on large-scale depth dataset can significantly improve the performance of 3D object detection.

Recently, several works attempt to conduct the 3D object detection in 3D world space. OFT [39] and CaDDN [38] map the monocular image features into the bird's eye view (BEV) and detect 3D objects in BEV space. ImVoxelNet [40] builds a 3D volume in 3D world space and samples multi-view features

to obtain the voxel representation. Then 3D convolutions and domain-specific heads are used to detect objects in both indoor and outdoor scenes. Similar to CaDDN [38], BEVDet [18] employs the Lift-Splat-Shoot [37] to transform 2D multi-view features into BEV representation. With the BEV representation, a CenterPoint [55] head is used to detect 3D objects in an intuitive way. Following DETR [4], DETR3D [51] represents 3D objects as object queries. The 3D reference points, generated from object queries, are repeatedly projected back to all camera views and sample the 2D features.

BEV-based methods tend to introduce the Z-axis error, resulting in poor performance for other 3D-aware tasks (e.g., 3D lane detection). DETR-based methods can enjoy more benefits from end-to-end modeling with more training augmentations. Our method is DETR-based that detects 3D objects in a simple and effective manner. We encode the 3D position information into 2D features, producing the 3D position-aware features. The object queries can directly interact with such 3D position-aware representation without projection error.

2.3 Implicit Neural Representation

Implicit neural representation (INR) usually maps the coordinates to visual signal by a multi-layer perceptron (MLP). It is a high efficiency way for modeling 3D objects [35,9,31], 3D scenes [32,43,5,36] and 2D images [17,8,45,42]. NeRF [32] employs a fully-connected network to denote a specific scene. To synthesize a novel view, the 5D coordinates along camera rays are input to the network as queries and output the volume density and view-dependent emitted radiance. In MetaSR [17] and LIFF [8], the HR coordinates are encoded into the LR features and HR images of arbitrary size can be generated. Our method can be regarded as an extension of INR in 3D object detection. The 2D images are encoded with 3D coordinates to obtain 3D position-aware features. The anchor points in 3D space are transformed to object queries by a MLP and further interact with 3D position-aware features to predict the corresponding 3D objects.

3 Method

3.1 Overall Architecture

Fig. 2 shows the overall architecture of the proposed PETR. Given the images $I = \{I_i \in R^{3\times H_I \times W_I}, i = 1, 2, \ldots, N\}$ from N views, the images are input to the backbone network (e.g. ResNet-50 [16]) to extract the 2D multi-view features $F^{2d} = \{F_i^{2d} \in R^{C\times H_F \times W_F}, i = 1, 2, \ldots, N\}$. In 3D coordinates generator, the camera frustum space is first discretized into a 3D meshgrid. Then the coordinates of meshgrid are transformed by camera parameters and generate the coordinates in 3D world space. The 3D coordinates together with the 2D multi-view features are input to the 3D position encoder, producing the 3D position-aware features $F^{3d} = \{F_i^{3d} \in R^{C\times H_F \times W_F}, i = 1, 2, \ldots, N\}$. The 3D features are further input to the transformer decoder and interact with the object queries, generated from query generator. The updated object queries are used to predict the object class as well as the 3D bounding boxes.

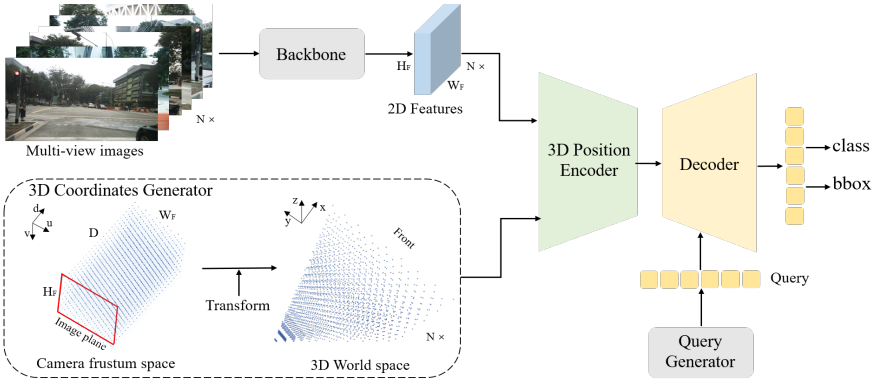


Fig. 2: The architecture of the proposed PETR paradigm. The multi-view images are input to the backbone network (e.g. ResNet) to extract the multi-view 2D image features. In 3D coordinates generator, the camera frustum space shared by all views is discretized into a 3D meshgrid. The meshgrid coordinates are transformed by different camera parameters, resulting in the coordinates in 3D world space. Then 2D image features and 3D coordinates are injected to proposed 3D position encoder to generate the 3D position-aware features. Object queries, generated from query generator, are updated through the interaction with 3D position-aware features in transformer decoder. The updated queries are further used to predict the 3D bounding boxes and the object classes.

3.2 3D Coordinates Generator

To build the relation between the 2D images and 3D space, we project the points in camera frustum space to 3D space since the points between these two spaces are one-to-one assignment. Similar to DGSN [7], we first discretize the camera frustum space to generate a meshgrid of size (W_F, H_F, D). Each point in the meshgrid can be represented as $p_j^m = (u_j \times d_j, v_j \times d_j, d_j, 1)^T$, where (u_j, v_j) is a pixel coordinate in the image, d_j is the depth value along the axis orthogonal to the image plane. Since the meshgrid is shared by different views, the corresponding 3D coordinate $p_{i,j}^{3d} = (x_{i,j}, y_{i,j}, z_{i,j}, 1)^T$ in 3D world space can be calculated by reversing 3D projection:

$$p_{i,j}^{3d} = K_i^{-1} p_j^m \tag{1}$$

where $K_i \in R^{4\times4}$ is the transformation matrix of i-th view that establish the transformation from 3D world space to camera frustum space. As illustrated in Fig. 2, the 3D coordinates of all views cover the panorama of the scene after the transformation. We further normalize the 3D coordinates as in Eq. 2.

$$\begin{cases} x_{i,j} = & (x_{i,j} - x_{min})/(x_{max} - x_{min}) \\ y_{i,j} = & (y_{i,j} - y_{min})/(y_{max} - y_{min}) \\ z_{i,j} = & (z_{i,j} - z_{min})/(z_{max} - z_{min}) \end{cases} \tag{2}$$

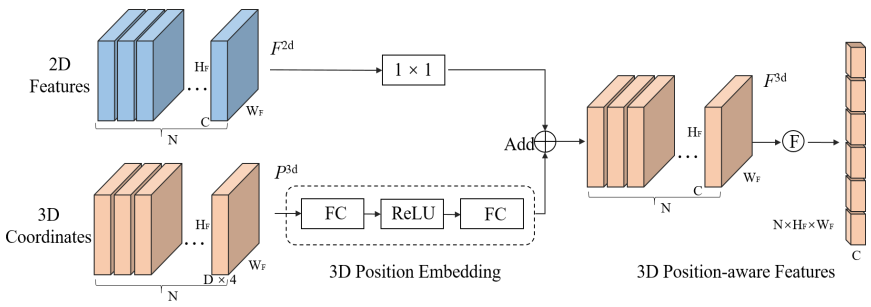


Fig. 3: Illustration of the proposed 3D Position Encoder. The multi-view 2D image features are input to a 1×1 convolution layer for dimension reduction. The 3D coordinates produced by the 3D coordinates generator are transformed into 3D position embedding by a multi-layer perception. The 3D position embeddings are added with the 2D image features of the same view, producing the 3D position-aware features. Finally, the 3D position-aware features are flattened and serve as the input of the transformer decoder. Ⓕ is the flatten operation.

where $[x_{min}, y_{min}, z_{min}, x_{max}, y_{max}, z_{max}]$ is the region of interest (RoI) of 3D world space. The normalized coordinates of $H_F \times W_F \times D$ points are finally transposed as $P^{3d} = \{P_i^{3d} \in R^{(D\times 4)\times H_F \times W_F}, i = 1, 2, \ldots, N\}$.

3.3 3D Position Encoder

The purpose of the 3D position encoder is to obtain 3D features $F^{3d} = \{F_i^{3d} \in R^{C\times H_F \times W_F}, i = 1, 2, \ldots, N\}$ by associating 2D image features $F^{2d} = \{F_i^{2d} \in R^{C\times H_F \times W_F}, i = 1, 2, \ldots, N\}$ with 3D position information. Analogously to Meta SR [17], the 3D position encoder can be formulated as:

$$F_i^{3d} = \psi(F_i^{2d}, P_i^{3d}), \quad i = 1, 2, \ldots, N \tag{3}$$

where $\psi(.)$ is the position encoding function that is illustrated in Fig. 3. Next, we describe the detailed implementation of $\psi(.)$. Given the 2D features F^{2d} and 3D coordinates P^{3d}, the P^{3d} is first feed into a multi-layer perception (MLP) network and transformed to the 3D position embedding (PE). Then, the 2D features F^{2d} is transformed by a 1×1 convolution layer and added with the 3D PE to formulate the 3D position-aware features. Finally, we flatten the 3D position-aware features as the key component of transformer decoder.

Analysis on 3D PE: To demonstrate the effect of 3D PE, we randomly select the PE at three points in the front view and compute the PE similarity between these three points and all multi-view PEs. As shown in Fig. 4, the regions close to these points tend to have the higher similarity. For example, when we select the left point in the front view, the right region of front-left view will have relatively higher response. It indicates that 3D PE implicitly establishes the position correlation of different views in 3D space.

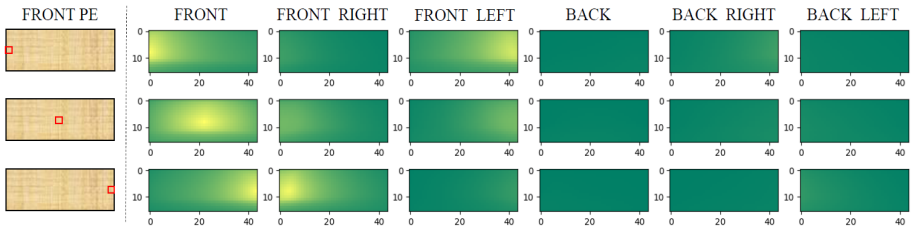


Fig. 4: 3D position embedding similarity. The red points are selected positions in the front view. We calculated the similarity between the position embedding of these selected positions and all image views. It shows that the regions close to these selective points tend to have higher similarity.

3.4 Query Generator and Decoder

Query Generator: Original DETR [4] directly uses a set of learnable parameters as the initial object queries. Following Deformable-DETR [58], DETR3D [51] predicts the reference points based on the initialized object queries. To ease the convergence difficulty in 3D scene, similar to Anchor-DETR [50], we first initialize a set of learnable anchor points in 3D world space with uniform distribution from 0 to 1. Then the coordinates of 3D anchor points are input to a small MLP network with two linear layers and generate the initial object queries Q_0. In our practice, employing anchor points in 3D space can guarantee the convergence of PETR while adopting the setting in DETR or generating the anchor points in BEV space fail to achieve satisfying detection performance. For more details, please kindly refer to our experimental section.
Decoder: For the decoder network, we follow the standard transformer decoder in DETR [4], which includes L decoder layers. Here, we formulate the interaction process in decoder layer as:

$$Q_l = \Omega_l(F^{3d}, Q_{l-1}), \quad l = 1, \ldots, L \tag{4}$$

where Ω_l is the l-th layer of the decoder. $Q_l \in R^{M \times C}$ is the updated object queries of l-th layer. M and C are the number of queries and channels, respectively. In each decoder layer, object queries interact with 3D position-aware features through the multi-head attention and feed-forward network. After iterative interaction, the updated object queries have the high-level representations and can be used to predict corresponding objects.

3.5 Head and Loss

The detection head mainly includes two branches for classification and regression. The updated object queries from the decoder are input to the detection head and predict the probability of object classes as well as the 3D bounding boxes. Note that the regression branch predicts the relative offsets with respect to the coordinates of anchor points. For fair comparison with DETR3D, we also

adopt the focal loss [25] for classification and $L1$ loss for 3D bounding box regression. Let $y = (c, b)$ and $\hat{y} = (\hat{c}, \hat{b})$ denote the set of ground truths and predictions, respectively. The Hungarian algorithm [22] is used for label assignment between ground-truths and predictions. Suppose that σ is the optimal assignment function, then the loss for 3D object detection can be summarized as:

$$L(y, \hat{y}) = \lambda_{cls} * L_{cls}(c, \sigma(\hat{c})) + L_{reg}(b, \sigma(\hat{b})) \tag{5}$$

Here L_{cls} denotes the focal loss for classification, L_{reg} is $L1$ loss for regression. λ_{cls} is a hyper-parameter to balance different losses.

4 Experiments

4.1 Datasets and Metrics

We validate our method on nuScenes benchmark [3]. NuScenes is a large-scale multimodal dataset that is composed of data collected from 6 cameras, 1 lidar and 5 radars. The dataset has 1000 scenes and is officially divided into 700/150/150 scenes for training/validation/testing, respectively. Each scene has 20s video frames and is fully annotated with 3D bounding boxes every 0.5s. Consistent with official evaluation metrics, we report nuScenes Detection Score (NDS) and mean Average Precision (mAP), along with mean Average Translation Error (mATE), mean Average Scale Error (mASE), mean Average Orientation Error(mAOE), mean Average Velocity Error(mAVE), mean Average Attribute Error(mAAE).

4.2 Implementation Details

To extract the 2D features, ResNet [16], Swin-Transformer [27] or VoVNetV2 [23] are employed as the backbone network. The C5 feature (output of 5th stage) is upsampled and fused with C4 feature (output of 4th stage) to produce the P4 feature. The P4 feature with 1/16 input resolution is used as the 2D feature. For 3D coordinates generation, we sample 64 points along the depth axis following the linear-increasing discretization (LID) in CaDDN [38]. We set the region to $[-61.2m, 61.2m]$ for the X and Y axis, and $[-10m, 10m]$ for Z axis. The 3D coordinates in 3D world space are normalized to [0, 1]. Following DETR3D [51], we set $\lambda_{cls} = 2.0$ to balance classification and regression.

PETR is trained using AdamW [29] optimizer with weight decay of 0.01. The learning rate is initialized with 2.0×10^{-4} and decayed with cosine annealing policy [28]. Multi-scale training strategy is adopted, where the shorter side is randomly chosen within [640, 900] and the longer side is less or equal to 1600. Following CenterPoint [55], the ground-truths of instances are randomly rotated with a range of $[-22.5°, 22.5°]$ in 3D space. All experiments are trained for 24 epochs (2x schedule) on 8 Tesla V100 GPUs with a batch size of 8. No test time augmentation methods are used during inference.

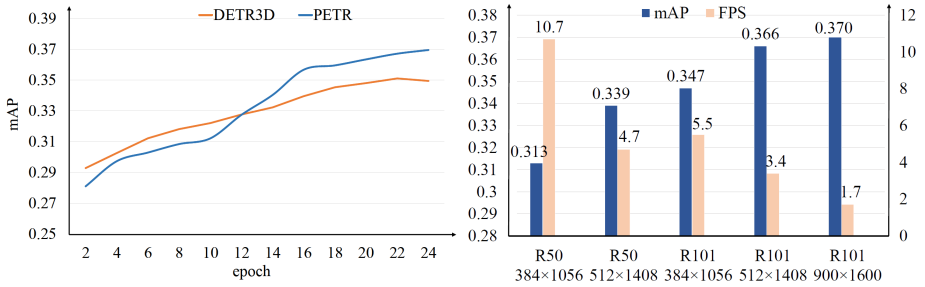
Table 1: Comparison of recent works on the nuScenes val set. The results of FCOS3D and PGD are fine-tuned and tested with test time augmentation. The DETR3D, BEVDet and PETR are trained with CBGS [57]. † is initialized from an FCOS3D backbone.

Methods	Backbone	Size	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
CenterNet	DLA	-	0.328	0.306	0.716	0.264	0.609	1.426	0.658
FCOS3D	Res-101	1600×900	0.415	0.343	0.725	0.263	0.422	1.292	0.153
PGD	Res-101	1600×900	0.428	0.369	0.683	0.260	0.439	1.268	0.185
BEVDet	Res-50	1056×384	0.381	0.304	0.719	0.272	0.555	0.903	0.257
BEVDet	Res-101	1056×384	0.389	0.317	0.704	0.273	0.531	0.940	0.250
BEVDet	Swin-T	1408×512	0.417	0.349	**0.637**	**0.269**	0.490	0.914	0.268
PETR	Res-50	1056×384	0.381	0.313	0.768	0.278	0.564	0.923	0.225
PETR	Res-50	1408×512	0.403	0.339	0.748	0.273	0.539	0.907	0.203
PETR	Res-101	1056×384	0.399	0.333	0.735	0.275	0.559	0.899	0.205
PETR	Res-101	1408×512	0.421	0.357	0.710	0.270	**0.490**	0.885	0.224
PETR	Swin-T	1408×512	**0.431**	**0.361**	0.732	0.273	0.497	**0.808**	**0.185**
DETR3D†	Res-101	1600×900	0.434	0.349	0.716	0.268	**0.379**	0.842	0.200
PETR†	Res-101	1056×384	0.423	0.347	0.736	0.269	0.448	0.844	0.202
PETR†	Res-101	1408×512	0.441	0.366	0.717	0.267	0.412	**0.834**	**0.190**
PETR†	Res-101	1600×900	**0.442**	**0.370**	**0.711**	**0.267**	0.383	0.865	0.201

4.3 State-of-the-art Comparison

As shown in Tab. 1, we first compare the performance with state-of-the-art methods on nuScenes val set. It shows that PETR achieves the best performance on both NDS and mAP metrics. CenterNet [56], FCOS3D [49] and PGD [48] are typical monocular 3D object detection methods. When compare with FCOS3D [49] and PGD [48], PETR with ResNet-101 [16] surpasses them on NDS by 2.7% and 1.4%, respectively. However, PGD [48] achieves relatively lower mATE because of the explicit depth supervision. Besides, we also compare PETR with multi-view 3D object detection methods DETR3D [51] and BEVDet [18], which detect 3D objects in a unified view. Since the DETR3D [51] and BEVDet [18] follow different settings on the image size and backbone initialization, we individually compare the PETR with them for fair comparison. Our method outperforms them 0.8% and 1.4% in NDS, respectively.

Tab. 2 shows the performance comparison on nuScenes test set. Our method also achieves the best performance on both NDS and mAP. For fair comparison with BEVDet [18], PETR with Swin-S backbone is also trained with an image size of 2112×768. It shows that PETR surpasses BEVDet [18] by 3.6% in mAP and 1.8% in NDS, respectively. It is worth noting that PETR with Swin-B achieves a comparable performance compared to existing methods using external data. When using the external data, PETR with VOVNetV2 [23] backbone achieves 50.4% NDS and 44.1% mAP. As far as we know, PETR is the first vision-based method that surpasses 50.0% NDS.



(a) (b)

Fig. 5: Convergence and speed analysis on PETR. (a) The convergence comparison between PETR and DETR3D [51]. PETR converges slower at initial stage and requires a relatively longer training schedule for fully convergence. (b) The performance and speed analysis with different backbones and input sizes.

Table 2: Comparison of recent works on the nuScenes test set. $*$ are trained with external data. ‡ is test time augmentation.

Methods	Backbone	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
FCOS3D‡	Res-101	0.428	0.358	0.690	0.249	0.452	1.434	**0.124**
PGD‡	Res-101	0.448	0.386	0.626	0.245	0.451	1.509	0.127
DD3D$*$‡	V2-99	0.477	0.418	0.572	0.249	**0.368**	1.014	**0.124**
DETR3D$*$	V2-99	0.479	0.412	0.641	0.255	0.394	0.845	0.133
BEVDet	Swin-S	0.463	0.398	0.556	**0.239**	0.414	1.010	0.153
BEVDet$*$	V2-99	0.488	0.424	**0.524**	0.242	0.373	0.950	0.148
PETR	Res-101	0.455	0.391	0.647	0.251	0.433	0.933	0.143
PETR	Swin-T	0.450	0.411	0.664	0.256	0.522	0.971	0.137
PETR	Swin-S	0.481	0.434	0.641	0.248	0.437	0.894	0.143
PETR	Swin-B	0.483	**0.445**	0.627	0.249	0.449	0.927	0.141
PETR$*$	V2-99	**0.504**	0.441	0.593	0.249	0.383	**0.808**	0.132

We also perform the analysis on the convergence and detection speed of PETR. We first compare the convergence of DETR3D [51] and PETR (see Fig. 5(a)). PETR converges relatively slower than DETR3D [51] within the first 12 epochs and finally achieves much better detection performance. It indicates that PETR requires a relatively longer training schedule for fully convergence. We guess the reason is that PETR learns the 3D correlation through global attention while DETR3D [51] perceives the 3D scene within local regions. Fig. 5(b) further reports the detection performance and speed of PETR with different input sizes. The FPS is measured on a single Tesla V100 GPU. For the same image size (e.g., 1056×384), our PETR infers with 10.7 FPS compared to the BEVDet [18] with 4.2 FPS. Note that the speed of BEVDet [18] is measured on NVIDIA 3090 GPU, which is stronger than Tesla V100 GPU.

Table 3: The impact of 3D Position Embedding. 2D PE is the common position embedding used in DETR. MV is multi-view position embedding to distinguish different views. 3D PE is the 3D position embedding proposed in our methods.

PE	2D	MV	3D	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
1	✓			0.208	0.069	1.165	0.290	0.773	0.936	0.259
2	✓	✓		0.224	0.089	1.165	0.287	0.738	**0.929**	0.251
3			✓	0.356	0.305	**0.835**	**0.238**	0.639	0.971	**0.237**
4	✓		✓	0.351	0.305	0.838	0.283	**0.633**	1.048	0.256
5	✓	✓	✓	**0.359**	**0.309**	0.844	0.278	0.653	0.945	0.241

Table 4: Analysis of different methods to discrete the camera frustum space and different region of interest (ROI) ranges to normalized the 3D coordinates. UD is the Uniform discretization while LID is the linear-increasing discretization.

Depth Range	$(x_{min}, y_{min}, z_{min}, x_{max}, y_{max}, z_{max})$	UD	LID	NDS↑	mAP↑	mATE↓
(1,51.2)	(-51.2, -51.2, -10.0, 51.2, 51.2, 10.0)	✓		0.352	0.303	0.862
(1,51.2)	(-51.2, -51.2, -5, 51.2, 51.2, 3)	✓		0.352	0.305	0.854
(1,61.2)	(-61.2, -61.2, -10.0, 61.2, 61.2, 10.0)	✓		**0.358**	**0.308**	**0.850**
(1,61.2)	(-61.2, -61.2, -5, 61.2, 61.2, 3)	✓		0.342	0.297	0.860
(1,51.2)	(-51.2, -51.2, -10.0, 51.2, 51.2, 10.0)		✓	0.350	**0.310**	0.843
(1,51.2)	(-51.2, -51.2, -5, 51.2, 51.2, 3)		✓	0.355	0.306	**0.838**
(1,61.2)	(-61.2, -61.2, -10.0, 61.2, 61.2, 10.0)		✓	**0.359**	0.309	0.839
(1,61.2)	(-61.2, -61.2, -5, 61.2, 61.2, 3)		✓	0.346	0.304	0.842

4.4 Ablation Study

In this section, we perform the ablation study on some important components of PETR. All the experiments are conducted using single-level C5 feature of ResNet-50 backbone without the CBGS [57].

Impact of 3D Position Embedding. We evaluate the impact of different position embedding (PE) (see Tab. 3). When only the standard 2D PE in DETR is used, the model can only converge to 6.9% mAP. Then we add the multi-view prior (convert the view numbers into PE) to distinguish different views and it brings a slight improvement. When only using the 3D PE generated by 3D coordinates, PETR can directly achieve 30.5% mAP. It indicates that 3D PE provides a strong position prior to perceive the 3D scene. In addition, the performance can be improved when we combine the 3D PE with both 2D PE and multi-view prior. It should be noted that the main improvements are from the 3D PE and the 2D PE/multi-view prior can be selectively used in practice.

3D Coordinate Generator. In 3D coordinates generator, the perspective view in camera frustum space is discretized into 3D meshgrid. The transformed coordinates in 3D world space are further normalized with a region of interest (RoI).

Table 5: The ablation studies of different components in the proposed PETR.

PE Networks	NDS↑	mAP↑	mATE↓
None	0.311	0.256	1.00
1×1 ReLU 1×1	**0.359**	**0.309**	**0.839**
3×3 ReLU 3×3	0.017	0.000	1.054

(a) The network to generate the 3D PE. "None" means that the normalized 3D coordinates are directly used as 3D PE.

Fusion Ways	NDS↑	mAP↑	mATE↓
Add	**0.359**	**0.309**	0.839
Concat	0.358	0.309	**0.832**
Multiply	0.357	0.303	0.848

(b) Different ways to fuse the 2D multi-view features with 3D PE in the 3D position encoder.

Anc-Points	NDS↑	mAP↑	mATE↓
None	-	-	-
Fix-BEV	0.337	0.295	0.852
Fix-3D	0.343	0.303	0.864
Learned-3D	**0.359**	**0.309**	**0.839**

(c) "None" means no anchor points following DETR. "Fix-BEV" and "Fix-3D" mean the grid anchor points in BEV space and 3D space respectively.

Points-Num	NDS↑	mAP↑	mATE↓
600	0.339	0.300	0.847
900	0.351	0.303	0.860
1200	0.354	0.303	0.854
1500	**0.359**	**0.309**	**0.839**

(d) Results with different numbers of anchor points. We explored anchor point numbers ranging from 600 to 1500. More points perform better.

Here, we explore the effectiveness of different discretization methods and RoI range (see Tab. 4). The Uniform discretization (UD) shows similar performance compared to the linear-increasing discretization (LID). We also tried several common ROI regions and the RoI range of $(-61.2m, -61.2m, -10.0m, 61.2m, 61.2m, 10.0m)$ achieves better performance than others.

3D Position Encoder. The 3D position encoder is used to encode the 3D position into the 2D features. Here we first explore the effect of the multi-layer perception (MLP) that converts the 3D coordinates into 3D position embedding. It can be seen in Tab. 5(a) that the network with a simple MLP can improve the performance by 4.8% and 5.3% on NDS and mAP compared to the baseline without MLP (aligning the channel number of 2D features to $D \times 4$). When using two 3×3 convolution layers, the model will not converge as the 3×3 convolution destroys the correspondence between 2D feature and 3D position. Furthermore, we compare different ways to fuse the 2D image features with 3D PE in Tab. 5(b). The concatenation operation achieves similar performance compared to addition while surpassing the multiply fusion.

Query Generator. Tab. 5(c) shows the effect of different anchor points to generate queries. Here, we compare four types of anchor points: "None", "Fix-BEV", "Fix-3D" and "Learned-3D". Original DETR ("None") directly employs a set of learnable parameters as object queries without anchor points. The global feature of object query fail to make the model converge. "Fix-BEV" is the fixed anchor points are generated with the number of 39×39 in BEV space. "Fix-3D"

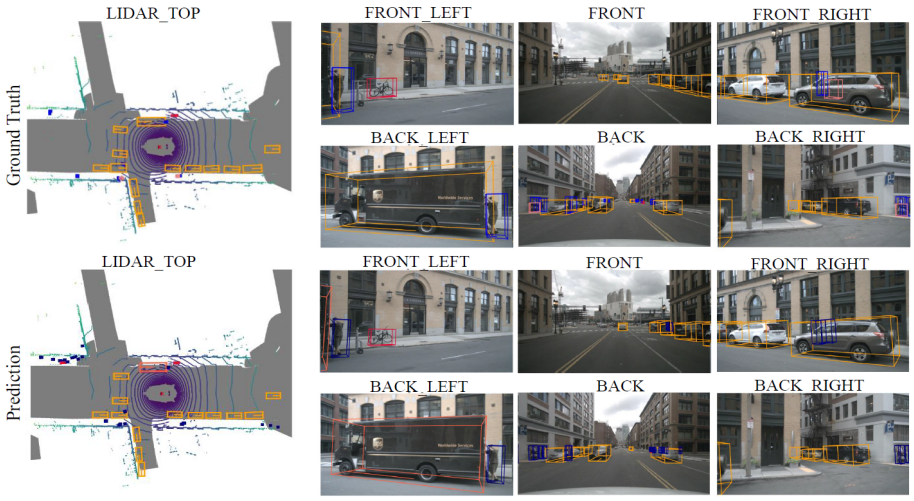


Fig. 6: Qualitative analysis of detection results in BEV and image views. The score threshold is 0.25, while the backbone is ResNet-101. The 3D bounding boxes are drawn with different colors to distinguish different classes.

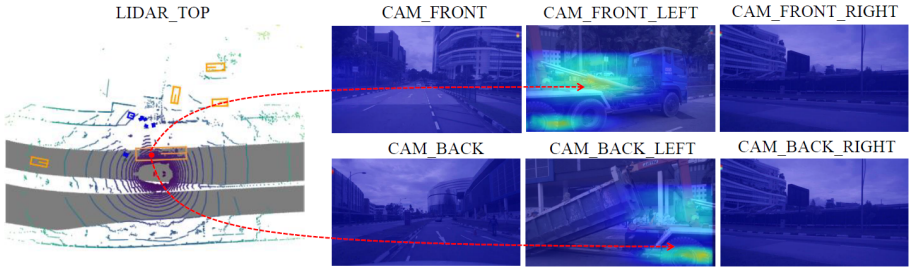


Fig. 7: Visualization of attention maps, generated from an object query (corresponding to the truck) on multi-view images. Both front-left and back-left views have a high response on the attention map.

means the fixed anchor points are with the number of $16 \times 16 \times 6$ in 3D world space. "Learned-3D" are the learnable anchor points defined in 3D space. We find the performance of both "Fix-BEV" and "Fix-3D" are lower than learned anchor points. We also explore the number of anchor points (see Tab. 5(d)), which ranges from 600 to 1500. The model achieve the best performance with 1500 anchor points. Considering of the computation cost is increasing with the number of anchor points, we simply use 1500 anchor points to make a trade-off.

4.5 Visualization

Fig. 6 shows some qualitative detection results. The 3D bounding boxes are projected and drawn in BEV space as well as image view. As shown in the BEV

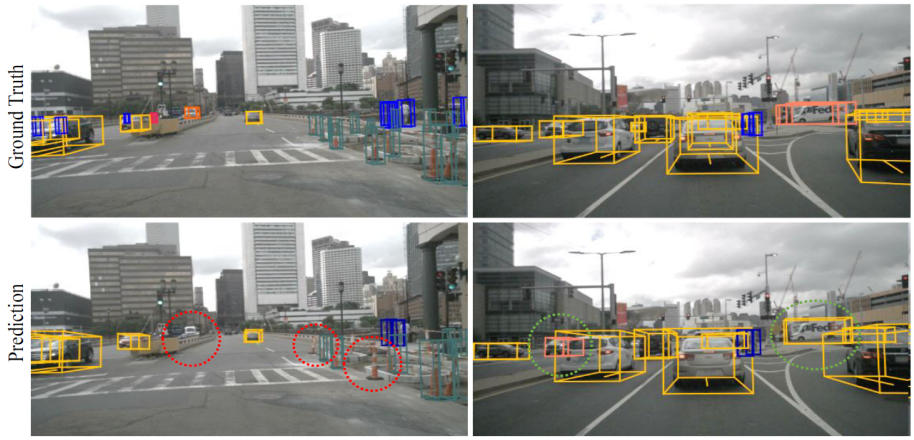


Fig. 8: Failure cases of PETR. We mark the failure cases by red and green circles. The red circles are some small objects that are not detected. The green circles are objects that are wrongly classified.

space, the predicted bounding boxes are close to the ground-truths. This indicates that our method achieves good detection performance. We also visualize the attention maps generated from an object query on multi-view images. As shown in Fig. 7, the object query tends to pay attention to the same object, even in different views. It indicates that 3D position embedding can establish the position correlation between different views. Finally, we provide some failure cases (see Fig. 8). The failure cases are marked by red and green circles. The red circles show some small objects that are not detected. The objects in green circle are wrongly classified. The wrong detection mainly occurs when different vehicles share high similarity on appearance.

5 Conclusions

The paper provides a simple and elegant solution for multi-view 3D object detection. By the 3D coordinates generation and position encoding, 2D features can be transformed into 3D position-aware feature representation. Such 3D representation can be directly incorporated into query-based DETR architecture and achieves end-to-end detection. It achieves state-of-the-art performance and can serve as a strong baseline for future research.

Acknowledgements: This research was supported by National Key R&D Program of China (No. 2017YFA0700800) and Beijing Academy of Artificial Intelligence (BAAI).

References

1. Bertasius, G., Wang, H., Torresani, L.: Is space-time attention all you need for video understanding. arXiv preprint arXiv:2102.05095 **2**(3), 4 (2021) 3
2. Brazil, G., Liu, X.: M3d-rpn: Monocular 3d region proposal network for object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9287–9296 (2019) 3
3. 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: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11621–11631 (2020) 8
4. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European conference on computer vision. pp. 213–229. Springer (2020) 1, 2, 3, 4, 7
5. Chabra, R., Lenssen, J.E., Ilg, E., Schmidt, T., Straub, J., Lovegrove, S., Newcombe, R.: Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In: European Conference on Computer Vision. pp. 608–625. Springer (2020) 4
6. Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., Urtasun, R.: Monocular 3d object detection for autonomous driving. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2147–2156 (2016) 1, 3
7. Chen, Y., Liu, S., Shen, X., Jia, J.: Dsgn: Deep stereo geometry network for 3d object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12536–12545 (2020) 5
8. Chen, Y., Liu, S., Wang, X.: Learning continuous image representation with local implicit image function. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8628–8638 (2021) 2, 4
9. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5939–5948 (2019) 4
10. Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q.V., Salakhutdinov, R.: Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860 (2019) 3
11. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018) 3
12. Dong, B., Zeng, F., Wang, T., Zhang, X., Wei, Y.: Solq: Segmenting objects by learning queries. Advances in Neural Information Processing Systems **34** (2021) 3
13. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020) 3
14. Gao, P., Zheng, M., Wang, X., Dai, J., Li, H.: Fast convergence of detr with spatially modulated co-attention. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3621–3630 (2021) 3
15. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. In: International Conference on Machine Learning. pp. 1243–1252. PMLR (2017) 3
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016) 4, 8, 9

17. Hu, X., Mu, H., Zhang, X., Wang, Z., Tan, T., Sun, J.: Meta-sr: A magnification-arbitrary network for super-resolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1575–1584 (2019) 2, 4, 6
18. Huang, J., Huang, G., Zhu, Z., Du, D.: Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. arXiv preprint arXiv:2112.11790 (2021) 4, 9, 10
19. Jörgensen, E., Zach, C., Kahl, F.: Monocular 3d object detection and box fitting trained end-to-end using intersection-over-union loss. arXiv preprint arXiv:1906.08070 (2019) 3
20. Kehl, W., Manhardt, F., Tombari, F., Ilic, S., Navab, N.: Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In: Proceedings of the IEEE international conference on computer vision. pp. 1521–1529 (2017) 3
21. Ku, J., Pon, A.D., Waslander, S.L.: Monocular 3d object detection leveraging accurate proposals and shape reconstruction. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11867–11876 (2019) 3
22. Kuhn, H.W.: The hungarian method for the assignment problem. Naval research logistics quarterly **2**(1-2), 83–97 (1955) 8
23. Lee, Y., Park, J.: Centermask: Real-time anchor-free instance segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 13906–13915 (2020) 8, 9
24. Li, Y., Wu, C.Y., Fan, H., Mangalam, K., Xiong, B., Malik, J., Feichtenhofer, C.: Improved multiscale vision transformers for classification and detection. arXiv preprint arXiv:2112.01526 (2021) 3
25. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. pp. 2980–2988 (2017) 8
26. 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. arXiv preprint arXiv:2201.12329 (2022) 3
27. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10012–10022 (2021) 3, 8
28. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016) 8
29. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017) 8
30. Meng, D., Chen, X., Fan, Z., Zeng, G., Li, H., Yuan, Y., Sun, L., Wang, J.: Conditional detr for fast training convergence. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3651–3660 (2021) 3
31. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4460–4470 (2019) 4
32. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: European conference on computer vision. pp. 405–421. Springer (2020) 2, 4
33. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3d bounding box estimation using deep learning and geometry. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 7074–7082 (2017) 1, 3

34. Park, D., Ambrus, R., Guizilini, V., Li, J., Gaidon, A.: Is pseudo-lidar needed for monocular 3d object detection? In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3142–3152 (2021) 1, 3
35. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 165–174 (2019) 4
36. Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., Geiger, A.: Convolutional occupancy networks. In: European Conference on Computer Vision. pp. 523–540. Springer (2020) 4
37. Philion, J., Fidler, S.: Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In: European Conference on Computer Vision. pp. 194–210. Springer (2020) 4
38. Reading, C., Harakeh, A., Chae, J., Waslander, S.L.: Categorical depth distribution network for monocular 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8555–8564 (2021) 3, 4, 8
39. Roddick, T., Kendall, A., Cipolla, R.: Orthographic feature transform for monocular 3d object detection. arXiv preprint arXiv:1811.08188 (2018) 3
40. Rukhovich, D., Vorontsova, A., Konushin, A.: Imvoxelnet: Image to voxels projection for monocular and multi-view general-purpose 3d object detection. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 2397–2406 (2022) 3
41. Simonelli, A., Bulo, S.R., Porzi, L., López-Antequera, M., Kontschieder, P.: Disentangling monocular 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1991–1999 (2019) 3
42. Sitzmann, V., Martel, J., Bergman, A., Lindell, D., Wetzstein, G.: Implicit neural representations with periodic activation functions. Advances in Neural Information Processing Systems **33**, 7462–7473 (2020) 4
43. Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: Continuous 3d-structure-aware neural scene representations. Advances in Neural Information Processing Systems **32** (2019) 4
44. Sun, Z., Cao, S., Yang, Y., Kitani, K.M.: Rethinking transformer-based set prediction for object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3611–3620 (2021) 3
45. Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. Advances in Neural Information Processing Systems **33**, 7537–7547 (2020) 4
46. Tian, Z., Shen, C., Chen, H., He, T.: Fcos: Fully convolutional one-stage object detection. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 9627–9636 (2019) 3
47. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017) 3
48. Wang, T., Xinge, Z., Pang, J., Lin, D.: Probabilistic and geometric depth: Detecting objects in perspective. In: Conference on Robot Learning. pp. 1475–1485. PMLR (2022) 1, 3, 9
49. Wang, T., Zhu, X., Pang, J., Lin, D.: Fcos3d: Fully convolutional one-stage monocular 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 913–922 (2021) 1, 3, 9

50. Wang, Y., Zhang, X., Yang, T., Sun, J.: Anchor detr: Query design for transformer-based detector. arXiv preprint arXiv:2109.07107 (2021) 3, 7
51. Wang, Y., Vitor Campagnolo, G., Zhang, T., Zhao, H., Solomon, J.: Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In: In Conference on Robot Learning. pp. 180–191 (2022) 1, 2, 4, 7, 8, 9, 10
52. Wu, C.Y., Li, Y., Mangalam, K., Fan, H., Xiong, B., Malik, J., Feichtenhofer, C.: Memvit: Memory-augmented multiscale vision transformer for efficient long-term video recognition. arXiv preprint arXiv:2201.08383 (2022) 3
53. Wu, K., Peng, H., Chen, M., Fu, J., Chao, H.: Rethinking and improving relative position encoding for vision transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10033–10041 (2021) 3
54. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R.R., Le, Q.V.: Xlnet: Generalized autoregressive pretraining for language understanding. Advances in neural information processing systems **32** (2019) 3
55. Yin, T., Zhou, X., Krahenbuhl, P.: Center-based 3d object detection and tracking. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11784–11793 (2021) 4, 8
56. Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. arXiv preprint arXiv:1904.07850 (2019) 9
57. 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) 9, 11
58. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020) 3, 7