3DPPE: 3D Point Positional Encoding for Multi-Camera 3D Object Detection Transformers

Changyong Shu[1*]   Jiajun Deng[2*]   Fisher Yu[3]   Yifan Liu[4†]

[1]Houmo AI, [2]University of Sydney, [3]ETH Zürich, [4]University of Adelaide,

`changyong.shu89@gmail.com`, `jiajun.deng@sydney.edu.au`
`fisheryu@ethz.ch`, `yifan.liu04@adelaide.edu.au`

Abstract

Transformer-based methods have swept the benchmarks on 2D and 3D detection on images. Because tokenization before the attention mechanism drops the spatial information, positional encoding becomes critical for those methods. Recent works found that encodings based on samples of the 3D viewing rays can significantly improve the quality of multi-camera 3D object detection. We hypothesize that 3D point locations can provide more information than rays. Therefore, we introduce 3D point positional encoding, 3DPPE, to the 3D detection Transformer decoder. Although 3D measurements are not available at the inference time of monocular 3D object detection, 3DPPE uses predicted depth to approximate the real point positions. Our hybrid-depth module combines direct and categorical depth to estimate the refined depth of each pixel. Despite the approximation, 3DPPE achieves 46.0 mAP and 51.4 NDS on the competitive nuScenes dataset, significantly outperforming encodings based on ray samples. We will make the codes available for further investigation.

1. Introduction

3D object detection is a vital component of autonomous driving perception systems. Particularly, image-based 3D object detection has received increasing attention from both academia and industry due to its lower cost compared to LiDAR-dependent solutions. Despite the fact that autonomous driving vehicles are equipped with multiple cameras, early attempts at image-based 3D object detection, as seen in previous works [19, 21], focus on monocular detection and combine the detection results from multiple cameras. This kind of solution is unable to make use of correspondence in the overlapping area of adjacent cameras, and the paradigm to individually detect objects in each view
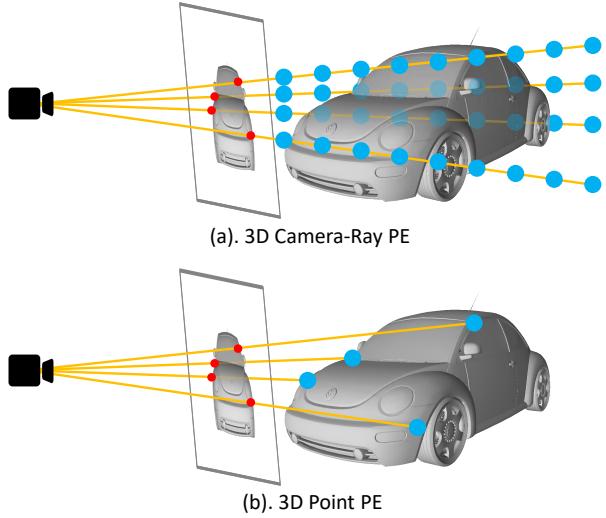
(a). 3D Camera-Ray PE

(b). 3D Point PE

Figure 1 – An illustration of (a) 3D camera-ray positional encoding (PE) and (b) our proposed 3D point PE. The 3D camera-ray PE represents camera-ray information by determining the positions of a set number of discrete points along the direction from the camera optical center to the image plane pixel. This encoding approach is coarse-grained. On the other hand, the 3D point PE provides more precise position information by encoding the location of a single point with an estimated depth. In the figure, four pixels are randomly selected to demonstrate the methods.

involves a large computational overhead. Alternatively, a group of recent studies [7, 8, 29, 31] follow the paradigm of Lift-Splat-Shoot (LSS) [6] to first transform multi-camera images to unified bird-eye-view (BEV) representation in parallel and then perform object detection on the BEV representation. However, such ill-posed view transformation inevitably causes error accumulation, which further affects the accuracy of 3D object detection.

At the same time, the transformer-based (DETR-like) [1] scheme has also been explored in this field. Typically, the methods following this scheme [3, 27, 28, 30, 34] utilizes

a set of learnable 3D object query to iteratively interact with multi-view 2D features, and further perform 3D object detection without explicit view transformation. Within the transformer-based methods, there are two general ways to enable the interaction of 3D queries and 2D image features, *i.e.*, projection-based and position-encoding-based. The former one projects 3D queries into the 2D image plane [30, 34] for feature sampling, which requires extra deployment efforts. Moreover, such a sampling procedure only extracts local features, failing to make use of global coherence for improving 3D object detection. The other way, as first introduced in PETR [28], integrates the 3D information into 2D image features by positional encoding. With 3D positional encoding (PE), 2D image features can be directly exploited by 3D queries, without extra projection efforts.

Enhancement of the 3D PE is anticipated to result in more precise 3D object detection. Despite effectiveness, the mechanism and design options of 3D PE in previous methods have not been fully explored. The typical 3D PE is the 3D camera-ray PE, as shown in Figure. 1 (a). It encodes the ray direction starting from the camera's optical center to the pixel on the image plane. However, the ray direction only provides coarse localization information for the 2D image feature without the depth prior. Moreover, as the object query is embedded from the randomly initialized 3D reference point, the inconsistent embedding space for the reference point and camera-ray PE further hampers the effectiveness of the attention mechanism in the transformer decoder. Thus, reformulating a new 3D positional encoding with depth prior to localize the 2D feature and unify representation for both image feature and object query is still a legacy issue.

In this work, we explore an alternative 3D PE paradigm to ameliorate the aforementioned problem. Formally, we introduce 3D point positional encoding (3DPPE) to improve transformer-based multi-camera 3D object detection. As illustrated in Figure. 1 (b), 3DPPE improves the camera-ray 3D PE by involving depth prior. Moreover, we find that 3D point PE not merely avoids the defects above, but also can provide better representative similarity (shown in Figure. 6). Specifically, in 3DPPE, we first devise a hybrid-depth module that combines direct and categorical ones to estimate the refined depth of each pixel. Then, we transform the pixels to 3D points via the camera parameters and predicted depth. The resulting 3D points are sequentially sent to a position encoder for 3D point PE. Particularly, we exploit a shared position encoder for the transformed 3D points and reference points to develop a unified embedding space.

We conduct extensive experiments to demonstrate the advantages of our proposed 3DPPE on challenging NuScene benchmarks. With the proposed 3D point positional encoding, our proposed 3DPPE can improve the camera-ray-based encoding by 1.9% mAP and 1.0% NDS.

2. Related Work

Transformer-based object detection. Object detection has been an active research topic in computer vision for several decades. Traditional object detection approaches, such as sliding window-based methods (one-stage) and region-based (two-stage) methods, have achieved significant progress in recent years. However, these methods generally rely on hand-designed components, such as non-maximum-suppression (NMS) or anchor generation. DETR [1] is a pioneering work that introduces the transformer-based framework to solve object detection as a set prediction problem, eliminating the need for heuristic target assignment and extra post-processing like non-maximum suppression (NMS). Deformable DETR [35] improves DETR by introducing deformable attention and multi-level image features to ameliorate the slow convergence problem and to improve the poor detection performance for small objects. Two-stage schemes [20, 26, 35] use the top-k scoring region proposals to initialize the object queries for convergence acceleration. [13, 24, 32] use anchor points or anchor boxes to generate object queries, which provide explicit positional priors. SMCA [5] and Conditional DETR [18] enhance the cross-attention mechanisms by leveraging the spatial information in the decoder embedding. DN-DETR [11] and its variant DINO [33] incorporate denoising techniques to ameliorate the instability problem of bipartite graph matching.

Multi-camera 3D object detection. Previous works on multi-camera object detection have typically used monocular detection to process each view separately, followed by post-processing to merge the results into a unified coordinate system. However, this approach is limited in its ability to utilize information from multiple views simultaneously and can lead to missed detections, particularly for truncated objects. A promising paradigm has emerged recently that converts multi-camera features from perspective view to bird's-eye view (BEV) and performs object detection under BEV. Two representative approaches within this paradigm are LSS-based and transformer-based.

The LSS-based methods, such BEVDet [8] and BEVDet4D [7], are effective approaches for converting multi-camera features into a dense bird's-eye view (BEV) representation using LSS [6]. Specifically, these methods predict the categorical depth distribution of each pixel in the image feature map to generate the dense BEV representation, which can provide comprehensive information for 3D object detection. Following methods, such as BEVDepth [29] and STS [31] explicitly introduce a sub-network for depth estimation to refine the depth prediction.

Among the transformer-based methods, BEVFormer [34] constructs a dense BEV representation using a set of
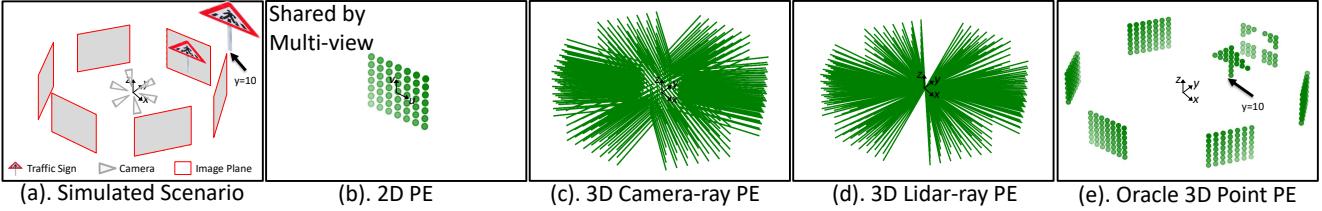
Figure 2 – Illustration of different positional encoding in the surround-view system. (a) presents the simulated scenario with 6 cameras, only one traffic sign appears in front view (the distance is set as 10m), rather than other cameras. Comparing all the PE from (b) to (e), only the oracle 3D point PE can encode the precise 3D point location of the object. Best viewed in color.

grid-shaped BEV queries to aggregate spatial and temporal features. DETR3D [30] samples 2D image features by projecting 3D reference points generated by object queries onto all views. The PETR series [27, 28] proposes the 3D position-encoding (PE) to transform the image features into 3D position-aware features, which can be directly interacted with object queries in 3D space. Following PETR series, Focal-PETR [23] utilizes instance-guided supervision and spatial alignment module to adaptively focus object queries on discriminative foreground regions; MV2D [25] generates a dynamic object query from 2D detector result, and one object query aggregates the feature from its corresponding 2D bounding box region.

In this paper, we follow PETR to perform 3D object detection with the transformer-based paradigm. However, in contrast to the previous approaches [23,25] that leverage 2D prior to improving 3D object detection, we devote our main efforts on investigate the 3D positional encoding, which has been rarely studied in the literature.

3. Preliminary of Positional Encoding

3.1. Ray-based Positional Encoding

The PETR series methods, *i.e.*, PETR [28] and PETRv2 [27], introduce a technique for multi-camera 3D object detection by encoding 3D coordinate information into multi-camera image features. This approach allows for the production of 3D position-aware features, which can improve the accuracy of object detection in 3D space. Specifically, PETR and PETRv2 obtain the 3D coordinate information from a set of points along the camera ray, namely camera-ray PE. Given the depth range $R_D = [D_{\min}, D_{\max}]$, camera-ray PE first divides the depth into N_D bins via linear-increasing discretization (LID) [28]. The center of each bin is exploited to represent the corresponding bin, and thus the 3D position information of a pixel is represented as N_D points along the camera-ray direction. After that, by utilizing the extrinsic and intrinsic parameters of the camera, points corresponding to different camera views are transformed into a unified coordinate system. For each pixel, the camera-ray points are concatenated together and

fed into an embedding layer for positional encoding. We perform further analysis on the ray-based positional encoding in PETR and PETRv2 in the supplementary material.

3.2. 3D Point Positional Encoding

For optimal accuracy in positional encoding, it is important to have access to the true 3D position of a point on a 2D plane, as demonstrated in Figure. 2-(e). In contrast, camera-ray PE encodes the direction from the camera's optical center to the pixel on the image plane, while LiDAR-ray PE depicts the orientation from the LiDAR center to the 3D point. While both ray PEs encode direction, they cannot accurately determine the 3D location without precise depth information.

To confirm that accurate 3D point positioning can enhance detection performance, we project 2D features into 3D space using the ground truth depth of a 2D image. Specifically, we project 3D point clouds onto surround-view images to generate a sparse depth map, and then use depth completion [10] to obtain the ground truth (GT) dense depth. We compare the performance of different positional encoding settings in Figure. 2 and list the results in Table. 1. All experiments are compared following the same training scheme. Compared to 3D camera-ray PE, 2D PE scores worst due to its complete inability to multi-camera distinction. 3D LiDAR-ray PE can achieve on-par or inferior performance depending on the fixed depth d. As for the 3D point PE with ground truth depth named Oracle 3D Point, a significant improvement is achieved with 6.7% NDS, 10.9% mAP, and 18.7% mATE respectively, which verify that *the 3D PE encoded from precise 3D point location is the key to improve multi-camera 3D object detection*.

As a camera-only system cannot gather ground truth depth information, we introduce a lightweight depth estimation module to substitute for the inaccessible GT depth. A more precise depth estimation results in improved 3D object detection performance. This study illustrates the potential of encoding 2D image features in 3D space with the help of estimated depth information.
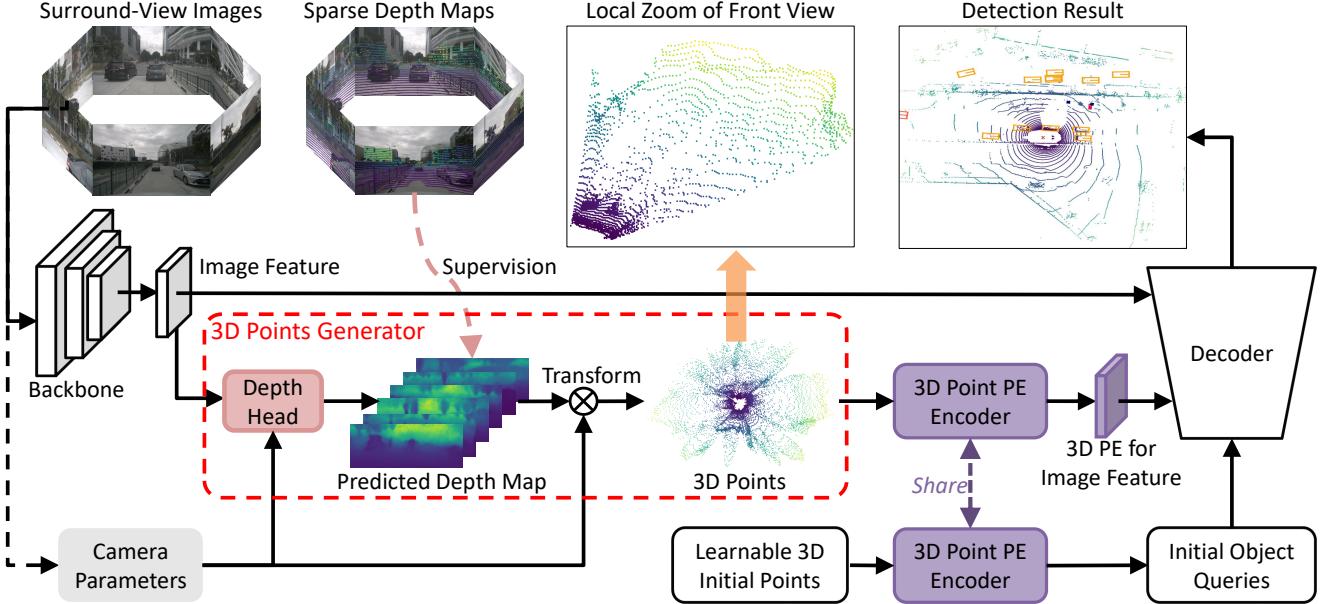
Figure 3 – The overall architecture of our proposed pipeline for method. Best viewed in color and zoom. The plot on the up shows intuitive illustration of surround-view images, sparse point projection, local zoom of front view and detection result. In local zoom of front view, the two dense point blocks located in left bottom corner indicate the two car object in the front view. The plot on the bottom shows the detail component of our method.

Table 1 – Performance comparison of different PE settings. Comparing all PEs listed in row 1, only the oracle 3D pint PE can encode precise 3D point location of objects.

PE	NDS↑	mAP↑	mATE↓
Camera-ray	0.337	0.274	0.852
2D	0.193	0.055	1.209
LiDAR-ray	0.338	0.275	0.849
Oracle 3D Point	**0.404**	**0.383**	**0.665**

4. Method

In this section, we present how to utilize the proposed unified depth-guided 3D point PE to transform the 2D features from multi-view images into the 3D space to perform multi-camera 3D object detection. We start by giving the architecture overview (Section. 4.1), then depict 3D point generator (Section. 4.2) and 3D point encoder (Section. 4.3), ultimately elaborate 3D point-aware feature (Section. 4.4) and decoder modification (Section. 4.5) respectively.

4.1. Framework Overview

In this work, we present 3D point positional encoding (3DPPE) for transformer-based multi-camera 3D object detection. As shown in Figure. 3, We first send N surround-view images $\boldsymbol{I} = \{I_i \in \mathbb{R}^{3 \times H_{I_i} \times W_{I_i}}, i = 1, 2, \ldots, N\}$ to backone (e.g. ResNet [9], Swintransformer [14]) for im-

age features $\boldsymbol{F} = \{F_i \in \mathbb{R}^{C \times H_{F_i} \times W_{F_i}}, i = 1, 2, \ldots, N\}$, where H_{I_i} and W_{I_i} is the i-th image shape, H_{F_i} and W_{F_i} is the i-th feature shape, C is the channel number of i-th feature. Then the image feature $\boldsymbol{F}$ undergo the depth head in 3D point generator for dense depth maps $\boldsymbol{D} = \{D_i \in \mathbb{R}^{1 \times H_{F_i} \times W_{F_i}}, i = 1, 2, \ldots, N\}$, and $\boldsymbol{D}$ is further transferred to the 3D points $P^{\mathrm{3D}} = \{P_i^{\mathrm{3D}} \in \mathbb{R}^{3 \times H_{F_i} \times W_{F_i}}, i = 1, 2, \ldots, N\}$ via camera parameter. The shared 3D point PE generator imports the 3D points $\boldsymbol{P}^{3D}$ above to produce the 3D point PE following $\mathrm{PE} = \{\mathrm{PE}_i \in \mathbb{R}^{C \times H_{F_i} \times W_{F_i}}, i = 1, 2, \ldots, N\}$ for 2D image feature. The 3D point PE generator also takes in the learnable 3D anchor points for 3D object queries $Q = \{Q_i \in \mathbb{R}^{C \times 1}, i = 1, 2, \ldots, K\}$, where PE and Q are unified 3D presentation with fine-grained point-aware position in 3D space. Finally, the 3D queries can directly interact with the image features supplemented by the 3D point PE in decoder to perform 3D object detection.

4.2. 3D Point Generator

We introduce a depth estimation module to provide dense depth map, and then transfer it to 3D point via camera back-projection.

Hybrid-Depth Module. Inspired by BEVdepth [29] and PGD [22], we design a hybrid-depth module that fuses the directly regressed depth $D^{\mathrm{R}} \in \mathbb{R}^{H_F \times W_F}$ and categorical depth $D^{\mathrm{P}} \in \mathbb{R}^{H_F \times W_F}$ with a learnable weight α. The architecture of our proposed hybrid-depth module is illustrated in
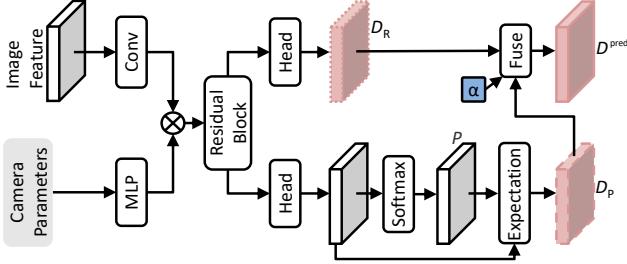
Figure 4 – Framework for depth head. The depth estimation D^{pred} is the fusion result of regressed depth D^R and probabilistic depth D^P, α is the fusion weight, P is the probabilistic over depth bins.

Figure. 4. For a given depth range $[d_{\min}, d_{\max}]$ with identical interval d_Δ, we could get $N_D = \frac{d_{\max} - d_{\min}}{d_\Delta}$ discrete depth bins: $\mathbf{D} = \{d_1, d_2, \ldots, d_{N_D}\}$. Instead of directly regressing the relative depth, the probabilistic P over these depth bins could be generated for each pixel, $P \in \mathbb{R}^{N_D \times H_F \times W_F}$. Thus, the pixel depth can be formulated as:

$$D^P = \sum_{i=1}^{N_D} P_{u,v,i} \times d_i. \tag{1}$$

The ultimate depth estimation D^{pred} comes from the fusion result of D^R and D^P above:

$$D^{pred} = \alpha D^R + (1 - \alpha) D^P, \tag{2}$$

where α is the learnable fusion weight for proportion balance. To achieve reliable depth prediction, predicted depth are supervised by projected depth D^{gt} from point cloud, and smooth L1 loss [21, 22] and distribution focal loss [12] are utilized:

$$\begin{aligned} L_{\text{depth}} = &\lambda_{\text{sm}} L_{\text{smooth-L1}}(D^{pred}, D^{gt}) \\ &+ \lambda_{\text{dfl}} L_{\text{dfl}}(D^{pred}, D^{gt}, \mathbf{D}), \end{aligned} \tag{3}$$

where λ_{sm} and λ_{dfl} is the hyper-parameters. The L_{dfl} aims to enlarge the probabilities of nearest two bins d^i and d^{i+1} around the ground truth D^{gt} ($d_i < D^{gt} < d_{i+1}$) for efficient learning:

$$\begin{aligned} L_{\text{dfl}}(D^{pred}, D^{gt}, \mathbf{D}) = &- \frac{d_{i+1} - D^{gt}}{d_\Delta} log(P_i) \\ &- \frac{D^{gt} - d_i}{d_\Delta} log(P_{i+1}). \end{aligned} \tag{4}$$

Coordinate Transformation from 2D to 3D. We transfer the 2D pixels in surround view to 3D point P^{3D} in the LiDAR coordinate system via the camera parameters. This process can be formulated as follow:

$$\begin{bmatrix} P_i^{3D}[0, u, v] \\ P_i^{3D}[1, u, v] \\ P_i^{3D}[2, u, v] \end{bmatrix} = R_i K_i^{-1} D_i^{pred}[u, v] \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} + T_i. \tag{5}$$

where $D_i^{pred}[u, v]^*$ is the predicted depth of 2D pixel (u, v), $P_i^{3D}[0, u, v]$, $P_i^{3D}[1, u, v]$ and $P_i^{3D}[2, u, v]$ are the x-axis, y-axis and z-axis coordinate of the correspond 3D point for 2D pixel (u, v) in i-th camera. $K_i \in \mathbb{R}^{3 \times 3}$ is i-th camera intrinsic matrix, $R_i \in \mathbb{R}^{3 \times 3}$ and $T_i \in \mathbb{R}^{3 \times 1}$ are the rotation and translation matrix from the camera coordinate system of i-th view to LiDAR coordinate system.

Setting the region of 3D perception space $[x_{\max}, x_{\min}, y_{\max}, y_{\min}, z_{\max}, z_{\min}]$, the normalization is further conducted on each 3D point:

$$\begin{cases} P_i^{3D}[0, u, v] = (P_i^{3D}[0, u, v] - x_{\min})/(x_{\max} - x_{\min}) \\ P_i^{3D}[1, u, v] = (P_i^{3D}[1, u, v] - y_{\min})/(y_{\max} - y_{\min}) \\ P_i^{3D}[2, u, v] = (P_i^{3D}[2, u, v] - z_{\min})/(z_{\max} - z_{\min}). \end{cases} \tag{6}$$

4.3. 3D point Encoder

The 3D point P^{3D} is embedded in 3D point PE PE via the 3D point encoder:

$$\begin{aligned} PE_i[:, u, v] = \text{MLP}(\text{Cat}(&\text{Sine}(P_i^{3D}[0, u, v]), \\ &\text{Sine}(P_i^{3D}[1, u, v]), \\ &\text{Sine}(P_i^{3D}[2, u, v]))), \end{aligned} \tag{7}$$

where the sine/cosine positional encoding function Sine [1] maps a 1-dimensional coordinate value to a $\frac{C}{2}$-dimensional vector, the sequential Cat operator concatenate $\text{Sine}(P_i^{3D}[0, u, v])$, $\text{Sine}(P_i^{3D}[1, u, v])$ and $\text{Sine}(P_i^{3D}[2, u, v])$ to generate a $\frac{3C}{2}$-dimensional vector, then the MLP consisted of two linear layers and a RELU activation reduces the vector dimension from $\frac{3C}{2}$ to C.

4.4. 3D Point-Aware Features

Given the resulting 3D point PE for image features above, we add it element-wisely with the image feature F for generating the 3D point-aware features F^{3D}. For a better understanding of our proposed 3D point-aware feature, Figure. 5 illustrates the difference among ours 3D point-aware feature and 3D position-aware feature in PETR series: (1) the channel dimensionality of point cloud in petr series is $N_D \times 3$, where N_D denotes the depth bin number along camera-ray, and the positional encoding is generated in a ray-aware paradigm. Whereas our method performs the positional encoding in point-aware manner with the channel dimensionality of point cloud reduce to 1×3 (e.g., the definite depth leading to better locating capability in 3D space). (2) Our scheme is compact for explicit motivation, as the function-ambiguity multi-layer modulation for feature-guided 3D PE in PETRv2 is not used.

4.5. Modification in Decoder

As depicted in Figure. 3, the learnable 3D anchor points go through the shared embedding generator used for 2D im-

---

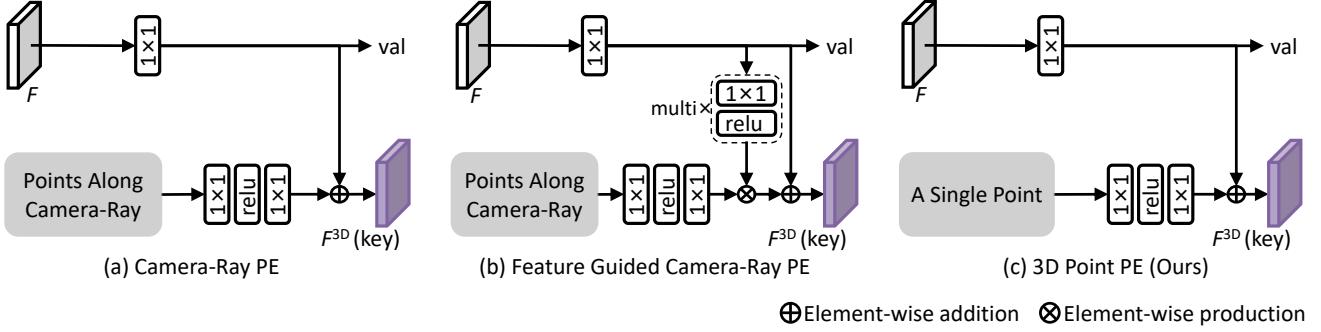*The $[\ldots]$ in the whole paper denotes the tensor slice in pytorch.

Figure 5 – Comparison between (a) camera-ray PE, (b) feature guided camera-ray PE, and (c) our proposed 3D point PE.

Table 2 – **Performance comparison of 3D object detection on nuScenes val set.** † indicates using the pre-trained FCOS3D backbone for model initialization. "S" denotes model with a single time stamp input. ∗ is trained with CBGS.

Methods	Backbone	Resolution	mAP↑	NDS↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
BEVDet [8]	Res-50	704×256	0.298	0.379	0.725	0.279	0.559	0.860	0.245
BEVDepth-S [29]	Res-50	704×256	0.315	0.367	0.702	0.271	0.621	1.042	0.315
PETR∗ [28]	Res-50	1408×512	0.339	0.403	0.748	0.273	0.539	0.907	0.203
3DPPE∗	Res-50	1408×512	**0.370**	**0.433**	0.689	0.279	0.524	0.828	0.202
FCOS3D [21]	Res-101	1600×900	0.295	0.372	0.806	0.268	0.511	1.315	0.170
PGD [22]	Res-101	1600×900	0.335	0.409	0.732	0.263	0.423	1.285	0.172
DETR3D∗† [30]	Res-101	1600×900	0.349	0.434	0.716	0.268	0.379	0.842	0.200
BEVFormer-S∗† [34]	Res-101	1600×900	0.375	0.448	0.725	0.272	0.391	0.802	0.200
Ego3RT∗† [17]	Res-101	1600×900	0.375	0.450	0.657	0.268	0.391	0.850	0.206
SpatialDETR∗† [4]	Res-101	1600×900	0.351	0.425	0.772	0.274	0.395	0.847	0.217
PETR∗† [28]	Res-101	1408×512	0.366	0.441	0.717	0.267	0.412	0.834	0.190
3DPPE∗†	Res-101	1408×512	**0.391**	**0.458**	0.674	0.282	0.395	0.830	0.191

age features above to produce the 3D point PE E^Q for random initialized object queries Q, thus the E^F and E^Q are essentially encoded in the sympatric representation, which further enhances the object queries with precise positioning for indexing the useful 3D point-aware feature and performing the accurate 3D object.

5. Experiment

In this section, we first present the main results of our 3DPPE and compare with other state-of-the-art methods on nuScenes dataset in Section. 5.1. Then, in Section. 5.2, we conduct extensive ablative experiments to investigate the effectiveness of each component in our proposed method. After that, in Section. 5.3, we show the qualitative comparison between our method and the previous ray-based positional encoding. Finally, we discuss further potential improvements of 3DPPE in Section. 5.4. We detail the benchmark and metrics, as well as experimental details, in the supplementary material.

5.1. Comparison with State-of-the-art Methods

We compare the proposed method with other state-of-the-art multi-camera 3D object detectors on the validation and test sets of nuScenes dataset. All of the reported methods follow the single frame paradigm, and the P4 feature [28] is leveraged by default. Note that test time augmentation is not used in our method.

Table. 2 shows the comparison between state-of-the-art methods in nuScenes val set. Both the results with ResNet-50 and ResNet-101 are evaluated. Specifically, with ResNet-50, our 3DPPE achieves 0.370 mAP and 0.433 NDS, outperforming PETR by 3.1% and 3.0%. When using a stronger ResNet-101 backbone, the performance of 3DPPE boosts to 0.391 mAP and 0.458 NDS, performing better than other competitors. This comparison shows the superiority of our point PE against the camera-ray PE.

We also present the results evaluated by the test server in Table. 3. In this experiment, we follow the common practice to exploit the DD3D pre-trained VoVNet-99 models. Both the train and val sets are involved in the training phase. Remarkably, 3DPPE achieves 46.0% mAP and 51.4% NDS, exceeds PETR by absolute 1.9% mAP and 1.0% NDS.

Table 3 – **Performance comparison of 3D object detection performance on nuScenes test set.** "S" denotes model with a single time stamp input.

Methods	Backbone	mAP↑	NDS↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
DD3D [19]	VoV-99	0.418	0.477	0.572	0.249	0.368	1.014	0.124
DETR3D [30]	VoV-99	0.412	0.479	0.641	0.255	0.394	0.845	0.133
Ego3RT [17]	VoV-99	0.425	0.473	0.549	0.264	0.433	1.014	0.145
BEVDet [8]	VoV-99	0.424	0.488	0.524	0.242	0.373	0.950	0.148
BEVFormer-S [34]	VoV-99	0.435	0.495	0.589	0.254	0.402	0.842	0.131
SpatialDETR [4]	VoV-99	0.424	0.486	0.613	0.253	0.402	0.857	0.131
PETR [28]	VoV-99	0.441	0.504	0.593	0.249	0.383	0.808	0.132
3DPPE	VoV-99	**0.460**	**0.514**	0.569	0.255	0.394	0.796	0.138

As shown by the results in both tables, the advantage of our 3DPPE in mAP is most pronounced. As the mAP calculation of nuScenes is closely related to the distance to the ground-truth object center, this finding further demonstrates that 3DPPE is capable of more precise positioning.

5.2. Ablation Study

We conduct ablative experiments to study the effect of each component in our method. All of the experiments are performed without CBGS strategy. We use C5 feature out of ResNet-50 as the image feature. The resolution of input images is set to 704×256 by default.

Effectiveness of the Depth Quality. In our hybrid-depth module, smooth L1 loss $L_{\mathrm{smooth-L1}}$ and distribution focal loss L_{dfl} are adopted as training objectives of the regression depth and the classification depth, respectively. As shown in Table.4, without any depth supervision, the baseline model achieves 0.343 NDS and 0.266 mAP. $L_{\mathrm{smooth-L1}}$ improve the model by 2.2% NDS and 2.9% mAP. By involving L_{dfl}, the performance further boosts to 0.368 NDS and 0.299 mAP. The improved depth quality will provide a more accurate localization for the 3D point positional encoding, verifying the potential of the proposed encoding method.

Table 4 – Effect of Losses in our hybrid-depth head. The results reported in this table are evaluated on nuScenes val set. By default, the backbone network is ResNet-50, and the resolution of input images is 704×256.

$L_{\mathrm{smooth-L1}}$	L_{dfl}	NDS↑	mAP↑	mATE↓
		0.343	0.266	0.832
✓		0.365	0.295	0.818
✓	✓	0.368	0.299	0.807

Comparison of 3D Postion-aware Feature. PETR series and our method all tend to transform 2D image feature to 3D position-aware feature, as such 3D representation can be directly integrated into query-based method for 3D object detection. We aim to demonstrate that the depth-guided 3D point PE is most effective way to construct the 3D position-aware feature, and 5 paradigms of positional encoding listed in the second row of Table. 5 are used for comparison. The depth-guided 3D point PE achieves superior performance compared to the PE in PETR series, it outperforming camera-ray in PETR by 3.1% NDS and 2.5% mAP, and surpasses the feature-guided scheme in PETRv2 with 1.6% NDS and 1.6% mAP.

Table 5 – Comparison of different 3D position-aware feature. Camera-ray and feature-guided (extended version of camera-ray) are proposed in PETR and PETRv2 respectively. The last three rows in point-aware scheme are proposed ourselves: topk-aware method selects 5 depth bins with highest probability; depth feature-guided category involves the depth feature in the positional encoding; Depth-guided 3D point approach transforms pixels on image plane to 3D space with predicted depth for precise location.

3D Position-aware Feature		NDS↑	mAP↑	mATE↓
Ray-aware	Camera-Ray [28]	0.337	0.274	0.852
	Feature-guided [27]	0.352	0.283	0.843
Point-aware	Topk-aware	0.327	0.265	0.869
	Depth Feature-guided	0.359	0.291	0.826
	Depth-guided 3D Point	0.368	0.299	0.807

Effect of Shared 3D Point PE Encoder. This study seeks to provide empirical evidence that the incorporation of unified positional encoding, within the sympatric representation, enhances the detection capacity of 3D objects by the query. In order to assess this proposition, we conducted an experiment, details of which are furnished in Table. 6, wherein we manipulated the 3D point PE encoder, by varying between a shared and a separate configuration. Our findings demonstrate that the shared positional encoding methodology outperformed the separate approach by a margin of 0.6% NDS and 0.5% mAP, demonstrating the effectiveness of our proposed method.

Table 6 – Effect of separated and shared embedding generator. Shared embedding generator encourages consistency between PE representations and object queries.

Embedding Generator	NDS↑	mAP↑	mATE↓
Separated	0.362	0.294	0.813
Shared	0.368	0.299	0.807

Figure 6 – Qualitative comparison of 3D point PE and camera-ray PE in terms of the local similarity (best viewed in color). The red box of the first line indicates a selected pixel.

5.3. Qualitative Comparison

We randomly select a pixel from the front view, and compute the similarity of the selected position with all surrounding pixels. We find that the similarity of 3D point PE among positions in the local region is higher than that of camera-ray PE, as shown in Figure. 6, yellow region of the former is more cohesive compared to the latter. This indicates that 3D point PE is capable of more precise positioning.

5.4. Discussion on Further Improvements

Our proposed 3DPPE can serve as a simple yet effective baseline, which can be easily extended to achieve better performance. Here, we discuss the potential improvements by (a) leveraging temporal coherence and (b) reusing ground-truth depth for knowledge distillation.

Leveraging Temporal Coherence. Similar to the extension of PETR (*i.e.*, PETR-v2), our 3DPPE can perform temporal modeling by making use of more frames and taking 3D coordinates calibration. As shown in Table. 7, to validate that our 3DPPE is effective even the backbone is initialized with parameters pretrained on depth estimation tasks, we select VoVNet-99 as the backbone network. The original temporal coherence modeling in PETR-v2 improves the baseline with 7.6% NDS and 3.3% mAP. The gain increases to 8.4% NDS and 3.9% mAP when the temporal coherence modeling is transferred to our 3DPPE. Furthermore, our temporal coherence 3DPPE boosts the temporal coherence PETR-v2 with 2.2% NDS and 2.2% mAP. The observed results indicate the potential of our 3DPPE as an effective extension for temporal coherence.

Table 7 – Results of leveraging temporal coherence by involving multiple frames. Here, we exploit VoVNet-99 as the backbone network and set the input resolution as 800×320, P4 feature is selected as 2d image feature. By default, 2 frames are used if a model is with temporal information.

Method	Temporal	mAP↑	NDS↑
PETR-v2		0.377	0.426
PETR-v2	✓	0.410	0.502
3DPPE		0.393	0.440
3DPPE	✓	0.432	0.524

Reusing GT Depth for Knowledge Distillation. We show that besides directly exploiting the ground-truth depth as the supervision of our depth estimation network, we can also reuse it for knowledge distillation to achieve further model boosting. The experimental results are shown in Table. 8. Specifically, we first train a 3DPPE model with ground-truth depth, as discussed in Section. 3.2. This model is denoted as 3DPPE-oracle. With VoVNet-99 backbone and 800×320 input resolution, 3DPPE-oracle achieves 0.4740 NDS and 0.4493 mAP. Inspired by [2], to obtain the distilled model 3DPPE-distill, we add an auxiliary branch sibling to the original transformer decoder at the training stage. The parameters of the auxiliary branch are shared with the original transformer decoder, but the reference points are initialized from the 3DPPE-oracle and will not be finetuned during the training phase. This auxiliary branch of 3DPPE-distill follows the same target assignment as that of 3DPPE-oracle at each iteration and is supervised by the ground-truth boxes together with the predicted result out of 3DPPE-oracle. As shown in this table, 3DPPE-oracle boosts the original 3DPPE to 0.454 NDS and 0.397 mAP, which further validates the extension potential of our method.

Table 8 – Results of reusing the ground-truth depth for knowledge distillation. Here, we exploit VoVNet-99 as the backbone network and set the input resolution as 800×320, P4 feature is selected as 2d image feature.

Method	NDS↑	mAP↑
3DPPE-oracle	0.474	0.449
3DPPE	0.440	0.393
3DPPE-distill	0.454	0.397

6. Conclusion

In this paper, we analyze the formulation of positional encoding that maps 2D image feature into 3D representation. We revisit various positional encoding designs and show that 3D point PE encoded from precise 3D point location is vital to multi-camera 3D object detection. Capitalizing on the hybrid-depth module for precise positioning, our proposed 3DPPE achieves state-of-the-art performance among single-frame methods. Moreover, we also demonstrate extension potential of our method on leveraging temporal coherence and reusing ground-truth depth for knowledge distillation. We hope the proposed depth-guided 3D point PE can serve as a strong baseline for 3D perception.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 1, 2, 5

[2] Xiaokang Chen, Jiahui Chen, Yan Liu, and Gang Zeng. D³etr: Decoder distillation for detection transformer. *arXiv preprint arXiv:2211.09768*, 2022. 8

[3] Xuanyao Chen, Tianyuan Zhang, Yue Wang, Yilun Wang, and Hang Zhao. Futr3d: A unified sensor fusion framework for 3d detection. *arXiv preprint arXiv:2203.10642*, 2022. 1

[4] Simon Doll, Richard Schulz, Lukas Schneider, Viviane Benzin, Markus Enzweiler, and Hendrik PA Lensch. Spatialdetr: Robust scalable transformer-based 3d object detection from multi-view camera images with global cross-sensor attention. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIX*, pages 230–245. Springer, 2022. 6, 7

[5] Peng Gao, Minghang Zheng, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. Fast convergence of detr with spatially modulated co-attention. 2021. 2

[6] Philion Jonah and Fidler Sanja. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *European Conference on Computer Vision*, pages 194–210. Springer, 2020. 1, 2

[7] Huang Junjie and Huang Guan. Bevdet4d: Exploit temporal cues in multi-camera 3d object detection. *arXiv preprint arXiv:2203.17054*, 2022. 1, 2

[8] Huang Junjie, Huang Guan, Zhu Zheng, and Du Dalong. Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. *arXiv preprint arXiv:2112.11790*, 2021. 1, 2, 6, 7

[9] He Kaiming, Zhang Xiangyu, Ren Shaoqing, and Sun Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[10] Jason Ku, Ali Harakeh, and Steven L Waslander. In defense of classical image processing: Fast depth completion on the cpu. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pages 16–22. IEEE, 2018. 3

[11] F. Li, H. Zhang, S. Liu, J. Guo, L. M. Ni, and L. Zhang. Dn-detr: Accelerate detr training by introducing query denoising. 2022. 2

[12] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *arXiv preprint arXiv:2006.04388*, 2020. 5

[13] S. Liu, F. Li, H. Zhang, X. Yang, X. Qi, H. Su, J. Zhu, and L. Zhang. Dab-detr: Dynamic anchor boxes are better queries for detr. 2022. 2

[14] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 4

[15] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 11

[16] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 11

[17] Jiachen Lu, Zheyuan Zhou, Xiatian Zhu, Hang Xu, and Li Zhang. Learning ego 3d representation as ray tracing. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVI*, pages 129–144. Springer, 2022. 6, 7

[18] D. Meng, X. Chen, Z. Fan, G. Zeng, H. Li, Y. Yuan, L. Sun, and J. Wang. Conditional detr for fast training convergence. 2021. 2

[19] Dennis Park, Rares Ambrus, Vitor Guizilini, Jie Li, and Adrien Gaidon. Is pseudo-lidar needed for monocular 3d object detection? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3142–3152, 2021. 1, 7

[20] Zhiqing Sun, Shengcao Cao, Yiming Yang, and Kris M Kitani. Rethinking transformer-based set prediction for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3611–3620, 2021. 2

[21] Wang Tai, Zhu Xinge, Pang Jiangmiao, and Lin Dahua. FCOS3D: Fully convolutional one-stage monocular 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2021. 1, 5, 6

[22] Wang Tai, Zhu Xinge, Pang Jiangmiao, and Lin Dahua. Probabilistic and Geometric Depth: Detecting objects in perspective. In *Conference on Robot Learning (CoRL) 2021*, 2021. 4, 5, 6

[23] Shihao Wang, Xiaohui Jiang, and Ying Li. Focal-petr: Embracing foreground for efficient multi-camera 3d object detection. *arXiv preprint arXiv:2212.05505*, 2022. 3

[24] Y. Wang, X. Zhang, T. Yang, and J. Sun. Anchor detr: Query design for transformer-based object detection. 2021. 2

[25] Zitian Wang, Zehao Huang, Jiahui Fu, Naiyan Wang, and Si Liu. Object as query: Equipping any 2d object detector with 3d detection ability. *arXiv preprint arXiv:2301.02364*, 2023. 3

[26] Zhuyu Yao, Jiangbo Ai, Boxun Li, and Chi Zhang. Efficient detr: Improving end-to-end object detector with dense prior. *arXiv preprint arXiv:2104.01318*, 2021. 2

[27] Liu Yingfei, Yan Junjie, Jia Fan, Li Shuailin, Gao Qi, Wang Tiancai, Zhang Xiangyu, and Sun Jian. Petrv2: A unified framework for 3d perception from multi-camera images. *arXiv preprint arXiv:2206.01256*, 2022. 1, 3, 7

[28] Liu Yingfei, Wang Tiancai, Zhang Xiangyu, and Sun Jian. Petr: Position embedding transformation for multi-view 3d object detection. *arXiv preprint arXiv:2203.05625*, 2022. 1, 2, 3, 6, 7

[29] Li Yinhao, Ge Zheng, Yu Guanyi, Yang Jinrong, Wang Zengran, Shi Yukang, Sun Jianjian, and Li Zeming. Bevdepth: Acquisition of reliable depth for multi-view 3d object detection. *arXiv preprint arXiv:2206.10092*, 2022. 1, 2, 4, 6

[30] Wang Yue, Guizilini Vitor Campagnolo, Zhang Tianyuan, Wang Yilun, Zhao Hang, and Solomon Justin. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *Conference on Robot Learning*, pages 180–191. PMLR, 2022. 1, 2, 3, 6, 7

[31] Wang Zengran, Min Chen, Ge Zheng, Li Yinhao, Li Zeming, Yang Hongyu, and Huang Di. Sts: Surround-view temporal stereo for multi-view 3d detection. *arXiv preprint arXiv:2208.10145*, 2022. 1, 2

[32] G. Zhang, Z. Luo, Y. Yu, K. Cui, and S. Lu. Accelerating detr convergence via semantic-aligned matching. *arXiv e-prints*, 2022. 2

[33] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L. M. Ni, and H. Y. Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv e-prints*, 2022. 2

[34] Li Zhiqi, Wang Wenhai, Li Hongyang, Xie Enze, Sima Chonghao, Lu Tong, Yu Qiao, and Dai Jifeng. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. *arXiv preprint arXiv:2203.17270*, 2022. 1, 2, 6, 7

[35] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 2

Appendices

A. Dataset and Metric

Dataset. we conduct experiments on nuScenes dataset, a comprehensive autonomous driving dataset that encompasses a variety of perception tasks, such as detection, tracking, and LiDAR segmentation. The nuScenes dataset comprises 1,000 distinct driving scenes, divided into three distinct subsets for training (700), validation (150), and testing (150) purposes, respectively. Each of these driving scenes includes 20 seconds of perceptual data that are annotated with a keyframe at a frequency of 2 Hz. The data collection vehicle employed in this study is equipped with one LiDAR, five radars, and six cameras that capture a surround view of the vehicle's environment.

Metrics. We follow the official protocol to report the nuScense Score (NDS), mean Average Precision (mAP), along with five true positive metrics including 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).

B. Experimental Details

For comprehensive comparison, we have conducted experiments with ResNet-50, ResNet-101 and VoVNet-99 as the backbone networks in our experiments. Following the setting of the PETR series, we use P4 feature by default. Specifically, P4 feature is obtained by upsampling the C5 feature (output of the 5th stage) and fused with the C4 feature (output of the 4th stage). The P4 feature with 1/16 input resolution or the C5 feature with 1/8 input resolution is used as the 2D feature. The monocular depth ranges from 0 to 61m. The region of 3D perception space is set to $[-61.2m, 61.2m]$ for X and Y dimension and $[-10m, 10m]$ for Z dimension. The 3D coordinates in point cloud are normalized to [0,1]. As for the hyper-parameters in each loss component, we set $\lambda_{sl1}/\lambda_{DFL}/\lambda_{cls}/\lambda_{reg}$ to be 0.25/0.25/2.0/1.0/ respectively, and λ_{cls} and λ_{reg} is the loss weight for classification and regression follow PETR series. AdamW [16] optimizer with a weight decay of 0.01 is used for training model, and the learning rate is initialized as 2.0e-4 and decayed with cosine annealing scheme [15]. Unless otherwise stated, all experiments with a batch size of 8 are trained for 24 epochs on 4 Tesla V100 GPUs. Test augmentation methods are not used during the inference.

C. Analysis of 3D Positional Encoding

In this section, we first decoupled the positional encoding of PETR into three factors, *i.e.*, depth values number N_D, discretization method M_D and depth range R_D. Then, the influence of each factor is explored through ablation
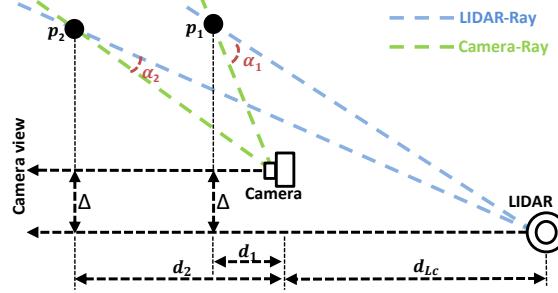

Figure 7 – A mathematic model for the included angle α between camera-ray and LiDAR-ray in a surround-view system. d_{Lc} is the distance along the camera ray from LIDAR to the camera, Δ is the distance perpendicular to camera ray from LIDAR to the camera, Apparently, α_2 will be smaller than α_1 (approaching 0), and meanwhile d_2 becomes larger than d_1.

studies in Sec. C.1. According to the experimental results, we summarize a feasible physical model to explain the meaning of the positional encoding in PETR. In Sec. C.2, a new assumption of using LiDAR-ray as positional encoding is proposed. Extensive experiments provide evidence for the new assumption. All experiments in this section are performed without CBGS and the backbone is set to ResNet-50, C5 feature is selected as 2D image feature, the train image size is set to 704×256 defaultly.

C.1. 3D Camera-Ray PE

PETR series divide the depth range R_D [1m, 61m] into $N^D = 64$ depth bins following linear-increasing discretization (LID). Therefore, one pixel corresponds to 64 separated 3d points lying on the corresponding camera ray. The 3D coordinates of these points are fed together into a 3D positional encoding encoder to generate the PE. In order to clarify what location information is encoded in 3D PE, we further explore the effectiveness of different depth bin numbers N_D, discretization methods M_D and depth range R_D as shown in Table 9. Surprisingly, the results turn out to be almost invariable under different settings, where the fluctuations of NDS, mAP and $mATE$ are smaller than 0.8%, 0.4% and 1.0% respectively. It gives a intuitive hint that the performance remains virtually unchanged through the separated 3D points sliding on the camera-ray. If the sampled points can represent the direction of the camera-ray, it already provides equivalent information to the PETR's 3D encoding. Thus, we propose a 3D camera-ray assumption that we could encode the 2D feature by two points on the camera-ray penetrating this pixel.

C.2. LiDAR-Ray PE Assumption

In this section, we further reduce N_D to 1 with fixed depth d such as 0.2m, 1m, 15m, 30m and 60m respectively. As listed in Table 10, smaller d leads to inferior performance

Table 9 – Quantitative comparison of different depth values number N_D, discretization methods M_D and depth range R_D. SID and UD denote spacing-increasing discretization and uniform discretization respectively. The invariable performances of top 2-4 rows with diverse M_D indicate that M_D is irrelevant. Thus we fix M_D as simplest UD but change R_D as in row 5 and 6, consistent performances demonstrate that R_D is also incoherence. Finally, we fix M_D and R_D but reduce the N_D to 32 and 2 respectively in last 2 rows, the immune performances declare that N_D also does not largely affect the results.

N^D	M_D	R_D	NDS↑	mAP↑	mATE↓
64	LID	[1,61]	0.338	0.275	0.853
64	SID	[1,61]	0.343	0.275	0.856
64	UD	[1,61]	0.337	0.273	0.847
64	UD	[1,31]	0.340	0.274	0.855
64	UD	[31,61]	0.336	0.272	0.849
32	UD	[1,61]	0.342	0.274	0.857
2	-	[1,61]	0.345	0.276	0.845

(row 1 and 2 in Table 10). It indicates that the scheme of current PE is no longer camera-ray, while on-par result (compared to results listed in Table 9) is achieved when d is larger than 15m. The phenomenon above enlightens us that the PE in PETR with $N^\mathrm{D} = 1$ represents a LiDAR-ray. Determination of a ray direction requires two points. As the fixed LiDAR's location provides the start point of ray, thus we can determine LIDAR-ray direction with one point.

As shown in Figure 7 (b), we calculate the discrepancy (Dis) between camera-ray and LiDAR-ray with the cosine of their included angle:

$$
\begin{aligned}
\mathtt{Dis} &= 1 - \cos(\alpha) \\
&= 1 - \cos(\alpha_c - \mathtt{arctan}(\frac{\tan\alpha_c + \frac{\Delta}{d}}{1 + \frac{d_{L_c}}{d}})) \quad (8) \\
&\approx 0.0 \quad \text{when} \quad d \gg d_{L_c} \text{and} \ d \gg \Delta,
\end{aligned}
$$

where α_c is the azimuth angle of camera-ray, α is the included angle between camera-ray and LiDAR-ray, d_{L_c} is the distance between camera and LiDAR along camera view (which ranges from 0.5m to 1.2m in universal sensor configuration), Δ is the distance between camera and LiDAR vertical to camera view (which ranges from 0.0m to 1.0m in universal sensor configuration). When $d \gg d_{L_c}$ and $d \gg \Delta$, the discrepancy between camera-ray and LiDAR-ray is almost eliminated. This further demonstrates that when d is larger, the LiDAR-ray will have a similar direction as the camera-ray, thus, the experiment results will be on par with the original PETR positional encoding even with one point.

Table 10 – Quantitative comparison of different fixed depth d when depth point number N_D is 1.

N^D	M_D	d	NDS↑	mAP↑	mATE↓
1	-	0.2	0.304	0.229	0.948
1	-	1	0.323	0.251	0.907
1	-	15	0.333	0.275	0.842
1	-	30	0.340	0.271	0.844
1	-	60	0.338	0.275	0.849

D. More Ablation Study

Experiment settings in this section are following setting in the above analysis of 3D positional encoding.

D.1. More Studies on PE Similarity

To demonstrate the 3D point PE is capable of more precise locating capability, we randomly select position on background and object (appeared on the cross-view) respectively from the front view, the similarity between the position and all pixels of the surround views is computed. Figure 8 is the complete version of the Figure 6 in main paper. In terem of point selected on cross-view object, as illustrated in Figure 9, the 3D point PE can find the related object from other view without redundant focus on the background. In term of the PE selected at the road, the 3D point PE tend to focus on the closer region round the selected position compared to the 3D camera-ray PE, as vividly shown in Figure 10.

Figure 8 – Representative similarity comparison of 3D camera-ray PE in PETR and ours 3D point PE, best viewed in color.
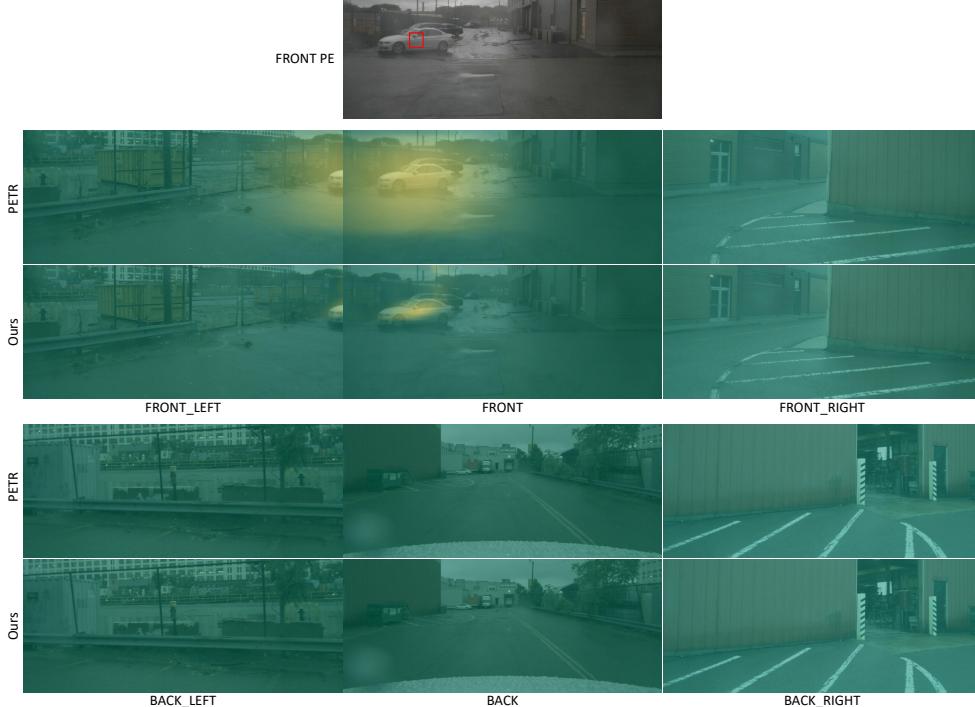

Figure 9 – Representative similarity comparison of 3D camera-ray PE in PETR and ours 3D point PE, best viewed in color. The position is selected on the car object appeared in cross-view.
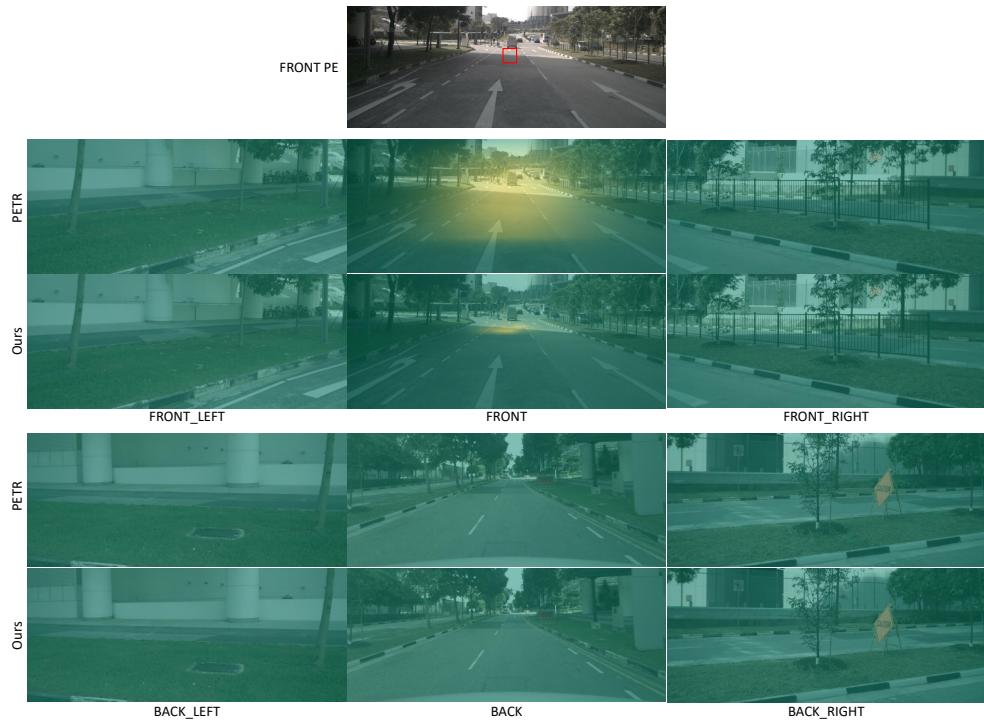
Figure 10 – Representative similarity comparison of 3D camera-ray PE in PETR and ours 3D point PE, best viewed in color. The position is selected on the background.
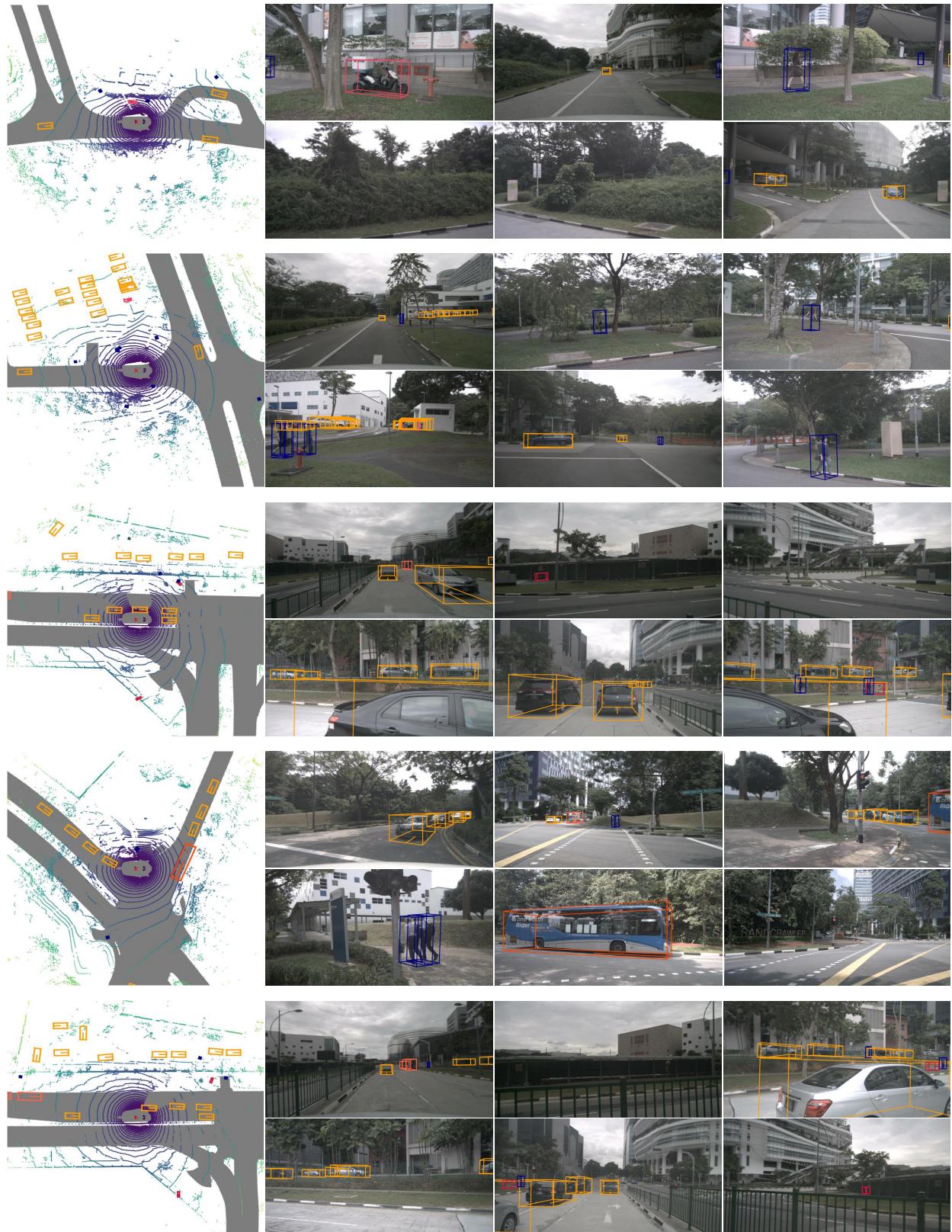
Figure 11 – Qualitative results for our method, best viewed in zoom and color.