FrustumFormer: Adaptive Instance-aware Resampling for Multi-view 3D Detection

Yuqi Wang[1,2] Yuntao Chen[3] Zhaoxiang Zhang[1,2,3]

[1] CRIPAC, Institute of Automation, Chinese Academy of Sciences (CASIA)
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences (UCAS)
[3] Centre for Artificial Intelligence and Robotics, HKISI_CAS
{wangyuqi2020,zhaoxiang.zhang}@ia.ac.cn chenyuntao08@gmail.com

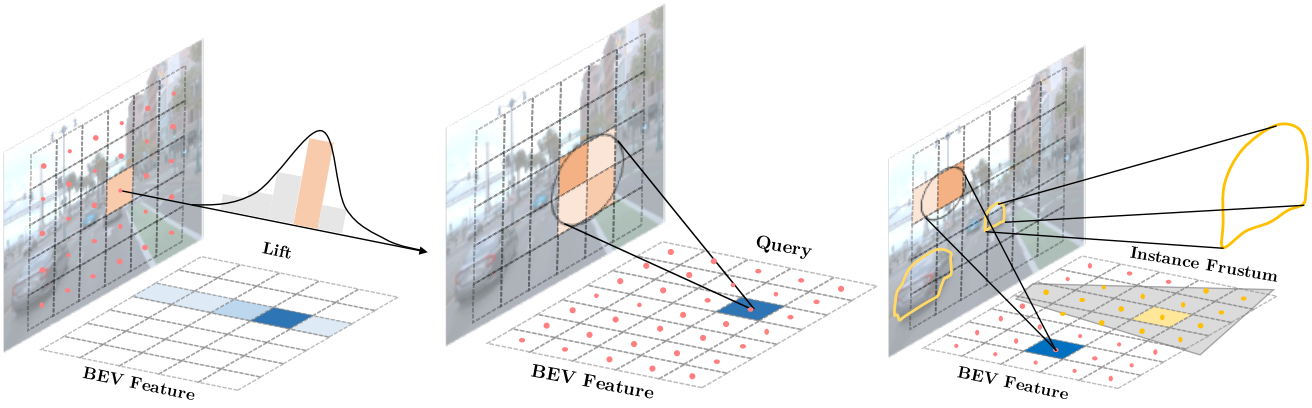
Abstract

The transformation of features from 2D perspective space to 3D space is essential to multi-view 3D object detection. Recent approaches mainly focus on the design of view transformation, either pixel-wisely lifting perspective view features into 3D space with estimated depth or grid-wisely constructing BEV features via 3D projection, treating all pixels or grids equally. However, choosing what to transform is also important but has rarely been discussed before. The pixels of a moving car are more informative than the pixels of the sky. To fully utilize the information contained in images, the view transformation should be able to adapt to different image regions according to their contents. In this paper, we propose a novel framework named ***FrustumFormer****, which pays more attention to the features in instance regions via adaptive instance-aware resampling. Specifically, the model obtains instance frustums on the bird's eye view by leveraging image view object proposals. An adaptive occupancy mask within the instance frustum is learned to refine the instance location. Moreover, the temporal frustum intersection could further reduce the localization uncertainty of objects. Comprehensive experiments on the nuScenes dataset demonstrate the effectiveness of FrustumFormer, and we achieve a new state-of-the-art performance on the benchmark. Codes and models will be made available at* https://github.com/Robertwyq/Frustum.

1. Introduction

Perception in 3D space has gained increasing attention in both academia and industry. Despite the success of LiDAR-based methods [14, 33, 41, 44], camera-based 3D object detection [19, 35, 36, 43] has earned a wide audience, due to the low cost for deployment and advantages for long-range detection. Recently, multi-view 3D detection in Bird's-Eye-View (BEV) has made fast progresses. Due to the unified representation in 3D space, multi-view features and temporal information can be fused conveniently, which leads to significant performance improvement over monocular methods [5, 28, 35, 39].

Transforming perspective view features into the bird's-eye view is the key to the success of modern BEV 3D detectors [12, 18, 19, 22]. As shown in Fig. 1, we categorize the existing methods into lifting-based ones like LSS [30] and BEVDet [12] and query-based ones like BEVFormer [19] and Ego3RT [25]. However, these methods mainly focus on the design of view transformation strategies while overlooking the significance of choosing the right features to transform during view transformation. Regions containing objects like vehicles and pedestrians are apparently more informative than the empty background like sky and ground. But all previous methods treat them with equal importance. We suggest that the view transformation should be adaptive with respect to the image content. Therefore, we propose *Adaptive Instance-aware Resampling (AIR)*, an instance-aware view transformation, as shown in Fig. 1c. The core idea of AIR is to reduce instance localization uncertainty by focusing on a selective part of BEV queries. Localizing instance regions is difficult directly on the BEV plane but relatively easy in the image view. Therefore, the *instance frustum*, lifting from instance proposals in image views, gives geometrical hints of the possible locations of objects in the 3D space. Though the instance frustum has provided initial prior locations, it is still a large uncertain area. We propose an *occupancy mask predictor* and a *temporal frustum fusion module* to further reduce the localization uncertainty. Our model learns an occupancy mask for frustum queries on the BEV plane, predicting the possibility that a region might contain objects. We also fuse instance frustums across different time steps, where the intersection area poses geomet-



(a) Grid Sampling in Image. (b) Grid Sampling in BEV. (c) Instance-aware Sampling in Frustum.

Figure 1. **Comparison of different sampling strategies for the feature transformation from image view to bird's eye view.** (a) represents the sampling in image view and lift features [12] to BEV plane with pixel-wise depth estimation. (b) shows the grid sampling in BEV and queries back [19] to obtain image features via cross-attention. (c) illustrates our proposed instance-aware sampling strategy in the frustum, which adapts to the view content by focusing more attention on instance regions. This approach is designed to enhance the learning of instance-aware BEV features.

ric constraints for actual locations of objects.

We propose a novel framework called *FrustumFormer* based on the insights mentioned previously, which effectively enhances the learning of instance-aware BEV features via Adaptive Instance-aware Resampling. FrustumFormer utilizes the instance frustum to establish the connection between perspective and bird's eye view regions, which contains two key designs: (1) A frustum encoder that enhances instance-aware features via adaptive instance-aware resampling. (2) A temporal frustum fusion module that aggregates historical instance frustum features for accurate localization and velocity prediction. In conclusion, the contributions of this work are as follows:

- We propose *FrustumFormer*, a novel framework that exploits the geometric constraints behind perspective view and birds' eye view by instance frustum.
- We propose that choosing what to transform is also important during view transformation. The view transformation should adapt to the view content. Instance regions should gain more attention rather than be treated equally. Therefore, we design *Adaptive Instance-aware Resampling (AIR)* to focus more on the instance regions, leveraging sparse instance queries to enhance the learning of instance-aware BEV features.
- We evaluate the proposed *FrustumFormer* on the nuScenes dataset. We achieve improved performance compared to prior arts. *FrustumFormer* achieves 58.9 NDS and 51.6 mAP on nuScenes test set without bells and whistles.

2. Related Work

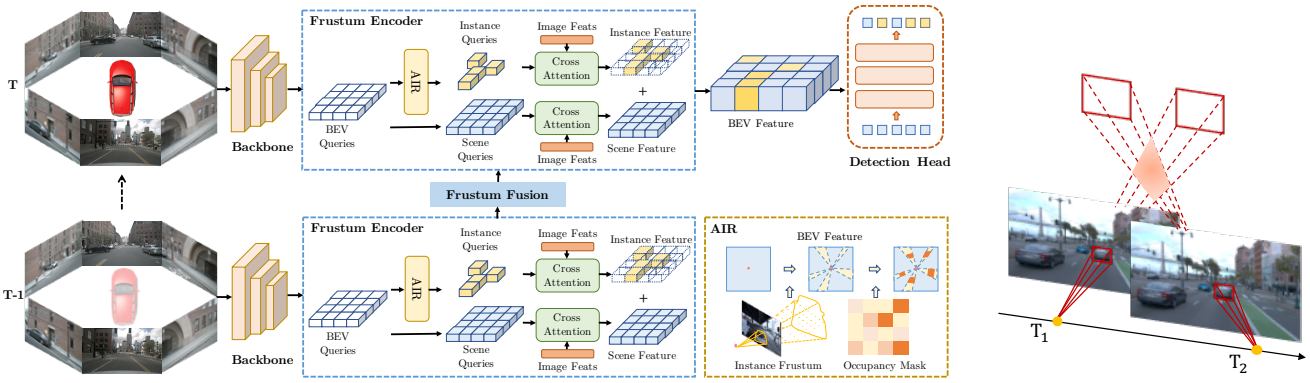
2.1. Frustum-based 3D Object Detection

Frustum indicates the possible locations of 3D objects in a 3D space by projecting 2D interested regions. The frustum is commonly used to aid fusion [8, 27, 31, 37, 42] in 3D object detection when RGB images and LiDAR data are available. Frustum PointNets [31] takes advantage of mature 2D object detectors and performs 3D object instance segmentation within the trimmed 3D frustums. Frustum Fusion [26] leverages the intersection volume of the two frustums induced by the 2D detection on stereo images. To deal with LiDAR sparsity, Faraway-Frustum [42] proposes a novel fusion strategy for detecting faraway objects. In this paper, we introduce the idea of frustum into camera-only 3D detection for enhancing instance-aware BEV features.

2.2. Multi-view 3D Object Detection

Multi-view 3D object detection aims to predict the 3D bounding boxes and categories of the objects with multi-view images as input. Current methods can be divided into two schemes: *lifting 2D to 3D* and *Querying 2D from 3D*.

Lifting 2D to 3D. Following the spirit of LSS [30], BEVDet [12] lifts multi-view 2D image features into a depth-aware frustum and splats into a unified bird's-eye-view (BEV) representation and applies to the detection task. BEVDepth [18] utilizes LiDAR points as supervision to learn reliable depth estimation. BEVDet4D [11] incorporates the temporal information and extends the BEVDet to the spatial-temporal 4D working space. Recently, STS [38], BEVStereo [16] and SOLOFusion [29] further attempt to improve the depth learning by combining temporal geometric constraints. Overall, these works demonstrate the im-



(a) Overall architecture of FrustumFormer.

(b) Temporal Frustum Fusion.

Figure 2. **Illustration of our proposed FrustumFormer.** (a) shows the overall pipeline. The image backbone first extracts the multi-view image features. These features are transformed into a unified BEV feature by the frustum encoder, which integrates temporal information from frustum fusion. Then the detection head decodes the BEV feature to the final outputs. Adaptive Instance-aware Resampling (AIR) is utilized to adaptively adjust the sampling area and select instance queries according to the view content. It consists of instance frustum query generation and frustum occupancy mask prediction. (b) illustrates the hints for object locations during temporal frustum fusion.

portance of incorporating depth and temporal information in improving detection performance.

Querying 2D from 3D. Following DETR [2], DETR3D [36] predicts learnable queries in 3D space and projects back to query the corresponding 2D image features. PETR [22, 23] proposes to query directly with 3D position-aware features, which are generated by encoding the 3D position embedding into 2D image features. Ego3RT [25] introduces the polarized grid of dense imaginary eyes and sends rays backward to 2D visual representation. BEVFormer [19] learns spatiotemporal BEV features via deformable attention, which explicitly constructs the BEV grid samples in 3D space and queries back to aggregate multi-view image features. PolarFormer [13] further generates polar queries in the Polar coordinate and encodes BEV features in Polar space.

3. Method

In multi-view 3D object detection task, N monocular views of images $I = \{I_i \in \mathbb{R}^{3\times H\times W}\}_{i=1}^{N}$, together with camera intrinsics $K = \{K_i \in \mathbb{R}^{3\times 3}\}_{i=1}^{N}$ and camera extrinsics $T = \{T_i \in \mathbb{R}^{4\times 4}\}_{i=1}^{N}$ are given. The objective of the model is to output the 3D attributes (locations, size, and velocity) and the corresponding category of objects contained in multi-view images.

As shown in Fig. 2a, FrustumFormer mainly focuses on the feature transformation process and is composed of four components: an image backbone, a frustum encoder, a frustum fusion module, and a detection head. The image backbone first extracts multi-scale image features from multi-view images. Aiding by the frustum fusion module, the image features transform into a unified BEV feature via the frustum encoder. Finally, a query-based detection head decodes the BEV feature to the outputs of the detection task.

3.1. Frustum Encoder

Frustum encoder transforms the multi-scale multi-view image features $\mathbf{F}$ to a unified BEV feature $\mathbf{B}$. Instead of treating all regions equally during the feature view transformation, our frustum encoder adaptively transforms the image features according to the view content. As shown in Fig. 2a, we use two types of BEV queries to construct the final BEV features, the scene queries $\mathbf{Q}_s$ and the instance queries $\mathbf{Q}_i$. Scene queries(blue grids) are dense and generated from regular grids, while instance queries(yellow grids) are sparse and generated from irregular instance frustum. The learning process of the scene query is similar to BEVFormer [19]. However, the instance queries are learned inside instance regions, and the learned instance feature are further combined with the scene feature to form the final instance-aware BEV features. Specifically, the selection of instance regions is made via *adaptive instance-aware resampling*, which consists of *(1) instance frustum query generation* and *(2) frustum occupancy mask prediction*. Finally, the instance feature is learned by *(3) instance frustum cross-attention* computed in selected instance regions. We will introduce these three parts in the following.

Instance Frustum Query Generation. This section introduces the query generation for a single instance frustum $\mathbf{Q}_f$, which is a subset of instance queries $\mathbf{Q}_i$. The core insight is to leverage the instance mask from perspective views and select the corresponding region on the BEV plane. Following the query-based [13, 19] view transformation, we define a group of grid-shape learnable parameters $\mathbf{Q}_i \in \mathbb{R}^{H\times W\times C}$ as the instance queries. H, W are the spatial shape of BEV queries, and C is the channel dimension. We first generate sampling points $\{\mathbf{p}_i^k = (x_i, y_i, z_k), i \in H \times W, k \in K\}$ corresponding to a single BEV query $\mathbf{Q}^{p_i}$ at grid region center $\mathbf{p}_i = (x_i, y_i)$, and then project these

points to different image views. K is the number of sampling points in the vertical direction of pillars. The projection between sampling points $\mathbf{p}_i^k$ and its corresponding 2D reference point (u_{ij}^k, v_{ij}^k) on the j-th image view is formulated as:

$$\pi_j(\mathbf{p}_i^k) = (u_{ij}^k, v_{ij}^k) \tag{1}$$

$$d_{ij}^k \cdot [u_{ij}^k, v_{ij}^k, 1]^T = \mathbf{T}_j \cdot [x_i^k, y_i^k, z_i^k, 1]^T \tag{2}$$

where $\pi_j(\mathbf{p}_i^k)$ denotes the projection of the k-th sampling point at location $\mathbf{p}_i$ on the j-th camera view. $\mathbf{T}_j \in \mathbb{R}^{3\times4}$ is the projection matrix of the j-th camera. x_i^k, y_i^k, z_i^k represents the 3D location of the sampling point in the vehicle frame. u_{ij}^k, v_{ij}^k denotes corresponding 2D reference point on j-th view projected from 3D sampling point $\mathbf{p}_i^k$. d_{ij}^k is the depth in the camera frame.

Predicting the instance frustum region directly in bird's eye view is challenging, but detecting the objects in perspective view [9, 32] is relatively mature. Inspired by this, we take advantage of object masks on the image plane and leverage its geometric clues for the BEV plane. The instance frustum queries $\mathbf{Q}_f$ of a specific 2D instance could be defined as all instance BEV queries $\mathbf{Q}_i$ with image plane projection points inside the object mask S in Eq. (3):

$$\mathbf{Q}_f = \{\mathbf{Q}^{p_i} \in \mathbf{Q}_i | \quad \exists \quad \pi(\mathbf{p}_i^k) \in S\} \tag{3}$$

$\mathbf{Q}^{p_i} \in \mathbb{R}^{1\times C}$ is the query located at $\mathbf{p}_i = (x_i, y_i)$. $\mathbf{p}_i^k$ represents the k-th sampling points in the pillars at $\mathbf{p}_i$. $\pi(\mathbf{p}_i^k)$ denotes the projection points of $\mathbf{p}_i^k$ to the image plane.

Frustum Occupancy Mask Prediction. Although the instance frustum provides potential locations for objects, its corresponding area on the BEV plane is often large due to depth uncertainty. To reduce this localization uncertainty, we propose to predict an occupancy mask for all frustums. Specifically, given the union of all instance frustum queries $\cup\mathbf{Q}_f$, we design a *OccMask* module to predict a binary occupancy mask $\mathbf{O}_{bev} \in \mathbb{R}^{H\times W\times1}$ on the BEV plane for all instance frustum queries in a single shot. The occupancy mask reflects the probability of a grid-wise region containing the objects, and is computed by Eq. (4):

$$\mathbf{O}_{bev} = \mathit{OccMask}(\cup\mathbf{Q}_f) \tag{4}$$

OccMask is a learned module composed of 2D convolutions. The supervision comes from the projection of the ground truth 3D bounding boxes on the BEV plane. We use the focal loss [21] for learning occupancy mask in Eq. (5):

$$\mathcal{L}_m = \mathit{Focal\ Loss}(\mathbf{O}_{bev}, \mathbf{\Omega}) \tag{5}$$

where $\mathbf{\Omega}$ represents the projection mask of the 3D bounding boxes onto the BEV plane. We select the minimum projecting bounding box on the BEV plane as the supervisory signal, which is made up of the outermost corners of the objects, considering their rotation. To further refine our predictions, the supervision signal for $\mathbf{O}_{bev}$ is added to each layer of the frustum encoder for iterative refinement, alongside BEV instance feature learning. In each layer, we predict the current occupancy mask using the last output instance frustum query from the previous layer. This approach enables the sampling areas to adapt to the previous layer output.

Instance Frustum Cross-Attention. Instance Frustum Cross-Attention (IFCA) is designed for the feature interaction between instance queries $\mathbf{Q}_i$ and image view features $\mathbf{F}$. The instance queries $\mathbf{Q}_i$ is selected by Eq. (6):

$$\mathbf{Q}_i = \{\mathbf{Q}^{p_i} \in \cup\mathbf{Q}_f | \mathbf{O}_{bev}(i) = 1\} \tag{6}$$

Instance queries are selected from the instance frustum queries $\mathbf{Q}_f$. $\mathbf{O}_{bev}(i)$ denotes the occupancy value at position $\mathbf{p}_i$ on the BEV plane. For each query $\mathbf{Q}^{p_i}$ in $\mathbf{Q}_f$, if $\mathbf{O}_{bev}(i)$ predicts the occupancy value is 1, then the query $\mathbf{Q}^{p_i}$ is marked as instance query. The process of instance frustum cross-attention (IFCA) can be formulated as:

$$\mathit{IFCA}(\mathbf{Q}_i^{p_i}, \mathbf{F}_j) = \frac{1}{|v|}\sum_{j\in v}\sum_{m=1}^{M}\mathit{DA}(\mathbf{Q}_i^{p_i}, \pi_j(\mathbf{p}_i^m), \mathbf{F}_j) \tag{7}$$

where $\mathbf{Q}_i^{p_i}$ is an instance query at location $\mathbf{p}_i$, $\pi_j(\mathbf{p}_i^m)$ is the projection to get the m-th 2D reference point on the j-th camera view. M is the total number of sampling points for an instance query. $\mathbf{F}_j$ is the image features of the j-th camera view. *DA* represents deformable attention. v is the set of image views for which the 2D reference point can fall on. $|v|$ represents the number of views.

3.2. Frustum Fusion Module

Temporal information is essential for camera-based 3D object detection, especially in inferring the motion state of objects and recognizing objects under heavy occlusions. Beyond learning occupancy mask on the BEV plane, another solution for eliminating the location uncertainty in the instance frustum is to fuse the temporal information.

Temporal Frustum Intersection. As shown in Fig. 2b, the intersection area of the instance frustum at different timestamps leaves hints for the accurate location of 3D objects. Inspired by this, we constrain the query interaction within instance frustum regions, implicitly learning features from interaction areas. Given instance frustum queries $\mathbf{Q}_f$ at current timestamp t and history instance frustum queries $\mathbf{H}_f$ preserved at timestamp t'. For a query $\cup\mathbf{Q}_f^{p_i}$ at position $\mathbf{p}_i$, we use the information from ego-motion $(\Delta x, \Delta y, \Delta\theta)$ to compute the corresponding position $\mathbf{p'}_i$ at timestamp t'. The cross-attention for query $\mathbf{Q}_f^{p_i}$ only compute the history queries around position $\mathbf{p'}_i$ of $\mathbf{H}_f$. Following [19], we adopt a sequential RNN-like [6] way of fusing the historical instance frustum queries. This approach enables the aggregation of long-range hints for the intersection area.

Temporal Frustum Cross-Attention. Temporal Frustum Cross-Attention (TFCA) aggregates the information of history instance frustum queries $\mathbf{H}_f$ into the current instance frustum queries $\mathbf{Q}_f$. Since the objects might be movable in the scene, causing the misalignment if only computing the query at $\mathbf{p'}_i$. Deformable attention [46] is utilized to reduce the influence of object movement. The process of temporal frustum cross-attention (TFCA) can be formulated as follows:

$$TFCA(\mathbf{Q}_f^{p_i}, \mathbf{H}_f) = \sum_{m=1}^{M} DA(\mathbf{Q}_f^{p_i}, \mathbf{p'}_i^m, \mathbf{H}_f) \quad (8)$$

where $\mathbf{Q}_f^{p_i}$ denotes the instance frustum query located at $\mathbf{p}_i = (x_i, y_i)$. $\mathbf{H}_f$ represents the history instance frustum query. $\mathbf{p'}_i$ is the aligned position by ego-motion. For each query at location $\mathbf{p'}_i$, we sample M points $\mathbf{p'}_i^m$ to query the history instance frustum feature. *DA* represents deformable attention.

4. Experiment

4.1. Datasets

nuScenes dataset [1]. The nuScenes dataset provides 1000 sequences of different scenes collected in Boston and Singapore. These sequences are officially split into 700/150/150 ones for training, validation, and testing. Each sequence is roughly about 20s duration, and the key samples are annotated at 2Hz, contributing to a total of 1.4M objects bounding boxes. Each sample consists of RGB images from 6 cameras covering the 360-degree horizontal FOV: front, front left, front right, back, back left, and back right. The image resolution is 1600×900 pixels in all views. 10 classes are annotated for the object-detecting task: car, truck, bus, trailer, construction vehicle, pedestrian, motorcycle, bicycle, barrier, and traffic cone.

Evaluation metrics. For the official evaluation protocol in the nuScenes dataset, the metrics include mean Average Precision (mAP) and a set of True Positive (TP) metrics, which contains the average translation error (ATE), average scale error (ASE), average orientation error (AOE), average velocity error (AVE), and average attribute error (AAE). Finally, the nuScenes detection score (NDS) is defined to consider the above metrics.

4.2. Experimental Settings

Implementation Details. Following previous methods [19, 35, 36], we utilize two types of backbone: ResNet101-DCN [7, 10] that initialized from FCOS3D [35], and VoVnet-99 [15] that initialized from DD3D [28]. We utilize the output multi-scale features from FPN [20] with sizes of 1/8, 1/16, 1/32, and 1/64 and a feature dimension of 256. The frustum encoder includes 6 layers and is implemented based on BEVFormer [19]. The default size of BEV queries is 200x200, and the perception ranges are [-51.2m, 51.2m] for the X and Y axis and [-3m, 5m] for the Z axis. We sample K=8 points for each pillar-like region of the BEV query. We adopt learnable position embedding for BEV queries. There are two types of queries in frustum encoder: sparse instance queries and dense scene queries. we followed BEVformer [19] to extract the scene feature by scene queries. The instance feature is extracted by instance queries. For the 2D instance proposals, we utilized the Mask R-CNN [9] pre-trained on the nuImages [1]. We use the output bounding boxes to generate object mask regions, and the score threshold is set to 0.5. The loss weight for $\mathcal{L}_m$ is set to 5. The frustum fusion module uses a temporal window size of 8 and randomly samples 4 key-frames during training. We adopted a query-based detection head to decode the BEV features, the same in [19]. The number of the object query is set to 900 and has 3 groups of queries [4] during training.

Training. We train the model on 8 NVIDIA A100 GPUs with batch size 1 per GPU. We train our model with AdamW [24] optimizer for 24 epochs, an initial learning rate of 2×10^{-4} with a cosine annealing schedule. The input of the images is cropped to 1600 × 640. We adopt data augmentations like image scaling, flipping, color distortion, and GridMask [3]. For the ablation study, we train the model with a total batch size of 8 for 24 epochs without data augmentation. We use the ResNet-50 [10] as the backbone. The image resolution is resized at a scale of 0.8, which is 1280 × 512.

Inference. During inference, the previous BEV features are saved and used for the next, corresponding to the infinite temporal window of a sequence. This online inference strategy is time-efficient. Since we adopted three groups of queries during training, only one group is utilized at inference time. We do not adopt model-agnostic tricks such as model ensemble and test-time augmentation when evaluating our model on both *val* and *test* sets.

4.3. 3D Object Detection Results

We compare our method with the state of the art on both *val* and *test* sets of nuScenes.

nuScenes test set. Table 1 presents the results of our model on the nuScenes *test* set, where we achieved a remarkable performance of **51.6 mAP** and **58.9 NDS** without utilizing any extra depth supervision from LiDAR. Under the setting without utilizing LiDAR as supervision, our method outperforms the previous state of the art. We evaluate our model in two types of backbone mentioned in the implementation details. With R101-DCN [7] as the backbone, we could achieve **47.8 mAP** and **56.1 NDS**, a significant improvement (+2.1 mAP and +1.8 NDS) over previous methods. For the final performance, we train FrustumFormer on the *trainval* split for 24 epochs without CBGS [45], using

Methods	Backbone	CBGS	LiDAR	mAP↑	NDS↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
FCOS3D‡ [35]	R101†			0.358	0.428	0.690	0.249	0.452	1.434	**0.124**
PGD [34]	R101†			0.386	0.448	0.626	**0.245**	0.451	1.509	0.127
BEVFormer [19]	R101†			0.445	0.535	0.631	0.257	0.405	0.435	0.143
PolarFormer [13]	R101†			0.457	0.543	0.612	0.257	**0.392**	0.467	0.129
FrustumFormer	R101†			**0.478**	**0.561**	**0.575**	0.257	0.402	**0.411**	0.132
DD3D [28]‡	V2-99*			0.418	0.477	0.572	0.249	0.368	1.014	0.124
DETR3D‡ [36]	V2-99*	✓		0.412	0.479	0.641	0.255	0.394	0.845	0.133
Ego3RT [25]	V2-99*			0.425	0.473	0.549	0.264	0.433	1.014	0.145
M2BEV [40]	X-101			0.429	0.474	0.583	0.254	0.376	1.053	0.190
BEVDet4D‡ [11]	Swin-B	✓		0.451	0.569	**0.511**	**0.241**	0.386	**0.301**	0.121
UVTR [17]	V2-99*			0.472	0.551	0.577	0.253	0.391	0.508	0.123
BEVFormer [19]	V2-99*			0.481	0.569	0.582	0.256	0.375	0.378	0.126
PolarFormer [13]	V2-99*			0.493	0.572	0.556	0.256	0.364	0.440	0.127
PETRv2 [23]	V2-99*			0.490	0.582	0.561	0.243	**0.361**	0.343	**0.120**
BEVDepth‡ [18]	V2-99*	✓	✓	0.503	0.600	0.445	0.245	0.378	0.320	0.126
BEVStereo [16]	V2-99*	✓	✓	0.525	0.610	0.431	0.246	0.358	0.357	0.138
FrustumFormer	V2-99*			**0.516**	**0.589**	0.555	0.249	0.372	0.389	0.126

Table 1. **Comparison to state-of-art on the nuScenes *test* set.** * notes that VoVNet-99(V2-99) [15] was pre-trained on the depth estimation task with extra data [28]. †Initialized from FCOS3D [35] backbone. ‡ means utilizing test-time augmentation during inference. The commonly used scheme for training is 24 epochs, and *CBGS* [45] would increase the training epochs by nearly $4.5\times$. *LiDAR* means training depth branch utilizing extra modality supervision from LiDAR.

Methods	Backbone	CBGS	LiDAR	mAP↑	NDS↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
FCOS3D [35]	R101†			0.295	0.372	0.806	0.268	0.511	1.315	0.170
DETR3D [36]	R101†	✓		0.349	0.434	0.716	0.268	0.379	0.842	0.200
PGD [34]	R101†			0.358	0.425	0.667	**0.264**	0.435	1.276	**0.177**
PETR [22]	R101†	✓		0.370	0.442	0.711	0.267	0.383	0.865	0.201
UVTR [17]	R101†			0.379	0.483	0.731	0.267	0.350	0.510	0.200
BEVFormer [19]	R101†			0.416	0.517	0.673	0.274	0.372	0.394	0.198
PolarFormer [13]	R101†			0.432	0.528	0.648	0.270	**0.348**	0.409	0.201
BEVDepth [18]	R101	✓	✓	0.412	0.535	0.565	0.266	0.358	0.331	0.190
STS [38]	R101	✓	✓	0.431	0.542	0.525	0.262	0.380	0.369	0.204
FrustumFormer	R101†			**0.457**	**0.546**	**0.624**	0.265	0.362	**0.380**	0.191

Table 2. **Comparison to state-of-art on the nuScenes *val* set.** †Initialized from FCOS3D [35] backbone. Our model is trained for 24 epochs without *CBGS* [45]. *LiDAR* means training depth branch utilizing extra modality supervision from LiDAR.

VoVNet (V2-99) as the backbone architecture with a pre-trained checkpoint from DD3D [28].

nuScenes validation set. Table 2 shows that our method achieves leading performance on the nuScenes *val* set. We achieved **45.7 mAP** and **54.6 NDS** without bells and whistles. Unlike the evaluation on *test* set, all the methods are compared with a fair backbone here. Since BEVDepth [18] and STS [38] utilized extra modality supervision in training, our NDS metric only improved slightly compared to them, but our mAP improved significantly. The translation error would be reduced with LiDAR supervision for the depth estimation, but this required extra modality data from LiDAR. Besides, our model is trained for 24 epochs, while they actually trained 90 epochs if using CBGS [45].

4.4. Ablation Study

We conduct several ablation experiments on the nuScenes *val* set to validate the design of FrustumFormer. For all ablation experiments, we used ResNet-50 as the backbone and resized the image resolution to 0.8 scales.

Ablation of Components in FrustumFormer. Table 3 ablates the components designed in FrustumFormer. (a) is the baseline setting of our method. (b) is the baseline with the instance frustum queries, which resamples the points in the whole instance frustum region. (c) is the baseline with the occupancy mask prediction. (d) is the baseline with adaptive instance-aware resampling, which consists of instance frustum query and occupancy mask prediction. Utilizing

adaptive instance-aware resampling to improve the learning of instance-aware BEV feature can significantly enhance both mAP and NDS metrics. (e) is based on (d) and further adds the history frustum information to incorporate temporal clues. Above all, our FrustumFormer could improve 4.2 mAP and 9.7 NDS compared to the baseline.

	IF	OM	FF	mAP↑	NDS↑	mATE↓
(a)				0.318	0.366	0.771
(b)	✓			0.326	0.373	0.765
(c)		✓		0.328	0.381	0.759
(d)	✓	✓		0.337	0.383	0.749
(e)	✓	✓	✓	**0.360**	**0.463**	**0.719**

Table 3. **Ablation of components in FrustumFormer.** IF denotes instance frustum, OM denotes occupancy mask, and FF means temporal frustum fusion. Adaptive instance-aware resampling is the combination of IF and OM, shown in (d).

Ablation of Instance-aware Sampling. Table 4 proves the effectiveness of instance-aware sampling. (a) represents the baseline setting, which treats all regions equally and samples 1x points for a grid cell region. (b) increases the sampling points to 2x for all regions. (c) selectively resamples the points inside instance regions. Compared with (b) and (c), we found that instance-aware sampling is more effective since simply increasing the sampling points for all regions has no gain.

	Total	Scene	Instance	mAP↑	NDS↑
(a)	1×	1×	-	0.318	0.366
(b)	2×	2×	-	0.318	0.362
(c)	2×	1×	1×	**0.326**	**0.373**

Table 4. **Ablation of instance-aware sampling.** In our method, 1× means sampling 8 points for a cell region on the BEV plane.

Ablation of Occupancy Mask Learning. Table 5 compares different supervision for learning occupancy masks on the BEV plane. (a) is the baseline without explicit supervision. (b) adds the supervision on the BEV plane, in which the instance area can be obtained by projecting annotated bounding boxes. To ease the learning, we slightly enlarge the bounding boxes by 1.0 meters in this setting. (c) uses the strict projection area (without enlargement) from the bounding boxes. (d) increases the loss weight for $\mathcal{L}_m$ from 5.0 to 10.0. (e) decreases the loss weight for $\mathcal{L}_m$ from 5.0 to 1.0. We choose the (c) as our default setting.

Ablation for Temporal Frustum Fusion. In Table 6, we demonstrate the benefits of incorporating frustum information in temporal fusion, as well as examine the impact of two important parameters: window size W and keyframes K. We start with the baseline (a) that uses a window size of 4 and 2 keyframes. Subsequently, we introduce the temporal frustum information in (b), leading to a notable improvement in performance. Then, we experiment with a larger temporal window by setting W to 8 and K to 4 in (c), which yields the best performance. As we extend the temporal window further in (d), the mAP continues to improve, but the NDS metric is affected by the mAVE score, causing degradation. Considering the NDS and mAP comprehensively, we finally utilize setting (c) as the default.

	Supervision	α	mAP↑	NDS↑	mATE↓
(a)	w/o	0.0	0.318	0.366	0.771
(b)	w/ BEV box*	5.0	0.324	0.374	0.756
(c)	w/ BEV box	5.0	**0.328**	**0.381**	0.759
(d)	w/ BEV box	10.0	0.322	0.381	**0.749**
(e)	w/ BEV box	1.0	0.326	0.379	0.751

Table 5. **Ablation of occupancy mask learning.** BEV box means utilizing the ground truth bounding boxes' projection on the BEV plane as supervision. * denotes enlarged bounding box. α is the loss weight for learning occupancy mask.

	W	K	Frustum	mAP↑	NDS↑	mAVE↓
(a)	4	2		0.353	0.454	0.497
(b)	4	2	✓	0.355	0.457	0.479
(c)	8	4	✓	0.360	**0.463**	**0.463**
(d)	16	4	✓	**0.364**	0.457	0.568

Table 6. **Ablation for temporal frustum fusion.** W means the history window size. K determines the key frames sampled in temporal window during model training.

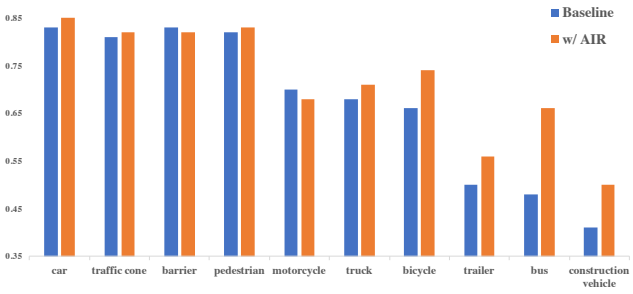


Figure 3. **Improvement of Recall Under Low Visibility.** We compute the recall under the visibility at 0-40% for all categories on the nuScenes *val* set. The recall for bus, bicycle, trailer, and construction vehicle categories improved significantly.

Improvement of Recall Under Low Visibility. The nuScenes dataset provides the visibility labels of objects in four subsets {0-40%, 40-60%, 60-80%, 80-100%}. As shown in Fig. 3, we compare the recall between baseline and baseline with adaptive instance-aware resampling under low visibility (0-40%). We found that the recall for categories of *bicycle*, *trailer*, *bus*, and *construction vehicle* improved a lot under the low visibility. Since nearly 29% of objects belong to the visibility of 0-40%, such improvement is crucial for a better 3D object detector.

4.5. Qualitative Analysis

Visualization of Recall Improvement. As shown in Fig. 4, we present the recall improvement achieved by utilizing the adaptive instance resampling (AIR) approach on the prediction results of the nuScenes *val* dataset. The predicted boxes are highlighted in blue, while the ground truth boxes are displayed in green. The red circle indicates the region where AIR helps to discover objects that were previously missed by enhancing the learning of instance features.

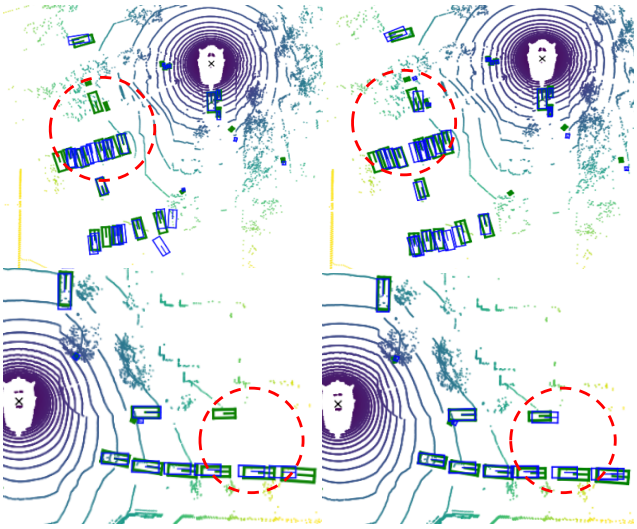


Figure 4. **Visualization of Recall Improvement**. The left side shows the baseline, while the right side shows the baseline with AIR. The prediction boxes are marked in blue, while the ground truth boxes are marked in green. In the red circle region, our method discovers more objects than the baseline.

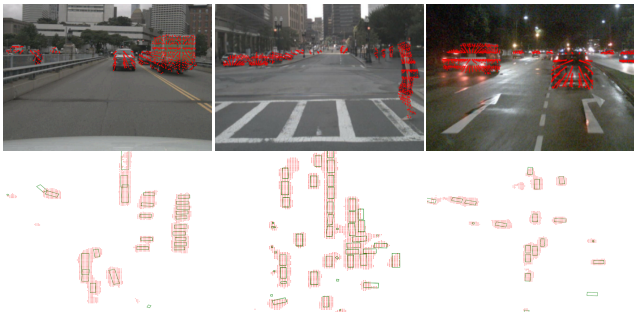


Figure 5. **Visualization of Instance-aware Sampling.** We visualize the instance-aware sampling on the perspective view and bird's eye view. Ground truth bounding boxes are marked in green color.

Visualization of Instance-aware Sampling. As shown in Fig. 5, we illustrate the instance-aware sampling points of our method on both perspective view and bird's eye view. The sampling points are highly related to the instance regions, enhancing the learning of the instance-aware feature.

Visualization of Instance-aware BEV Feature. As shown in Fig. 6, we visualize the BEV feature output by our frustum encoder. The BEV feature learned by our frustum encoder is instance-aware and has strong relations to the real positions. Here we visualize the corresponding ground truth boxes on the right side. Furthermore, we compare the BEV feature between the baseline and the baseline with adaptive instance-aware resampling(AIR). When utilizing AIR, more instance regions would be discovered (corresponding to the recall improvement), and the features are more instance discriminative in the dense areas.

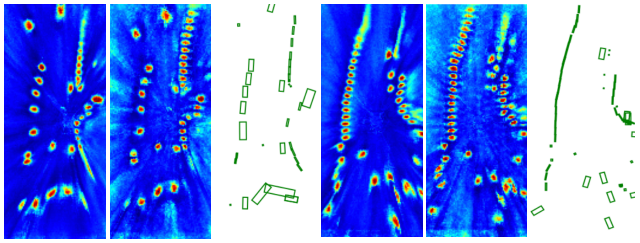


Figure 6. **Visualization of the instance-aware BEV feature.** We compared the feature heatmaps output by the frustum encoder without AIR, with AIR, and ground truth boxes (in green) shown from left to right. The colors in the feature heatmaps correspond to the norm value. The feature learned by frustum encoder is instance-aware and has strong correlations to the actual positions in 3D space. By using AIR, our model is able to discover more instance regions and learn discriminative features in dense areas.

5. Conclusion

In this paper, we propose *FrustumFormer*, a novel framework for multi-view 3D object detection. The core insight of FrustumFormer is to transform adaptively according to the view contents. To achieve this, we designed adaptive instance-aware resampling, which pays more attention to instance regions during feature view transformation. By utilizing this technique in the frustum encoder and temporal frustum fusion module, the model learned to better locate instance regions while learning instance-aware BEV features. Experimental results on the nuScenes dataset demonstrate the effectiveness of our method for multi-view 3D object detection. Our method significantly improved mAP over previous methods by focusing on instance regions. We hope that our framework can serve as a new baseline for future 3D perception research, shining a light on the significance of view content during feature transformation.

Acknowledgments

This work was supported in part by the Major Project for New Generation of AI (No.2018AAA0100400), the National Natural Science Foundation of China (No.61836014, No.U21B2042, No.62072457, No.62006231) and the InnoHK program. We extend our sincere thanks and appreciation to Jiawei He and Lue Fan for their valuable suggestions.

References

[1] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 5

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 3

[3] Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Gridmask data augmentation. *arXiv preprint arXiv:2001.04086*, 2020. 5

[4] Qiang Chen, Xiaokang Chen, Gang Zeng, and Jingdong Wang. Group detr: Fast training convergence with decoupled one-to-many label assignment. *arXiv preprint arXiv:2207.13085*, 2022. 5

[5] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *CVPR*, 2016. 1

[6] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014. 4

[7] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017. 5

[8] Emeç Erçelik, Ekim Yurtsever, and Alois Knoll. Temp-frustum net: 3d object detection with temporal fusion. In *IV*, 2021. 2

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 4, 5

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5

[11] Junjie Huang and Guan Huang. Bevdet4d: Exploit temporal cues in multi-camera 3d object detection. *arXiv preprint arXiv:2203.17054*, 2022. 2, 6

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

[13] Yanqin Jiang, Li Zhang, Zhenwei Miao, Xiatian Zhu, Jin Gao, Weiming Hu, and Yu-Gang Jiang. Polarformer: Multi-camera 3d object detection with polar transformers. In *AAAI*, 2023. 3, 6

[14] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 1

[15] Youngwan Lee, Joong-won Hwang, Sangrok Lee, Yuseok Bae, and Jongyoul Park. An energy and gpu-computation efficient backbone network for real-time object detection. In *CVPRW*, 2019. 5, 6

[16] Yinhao Li, Han Bao, Zheng Ge, Jinrong Yang, Jianjian Sun, and Zeming Li. Bevstereo: Enhancing depth estimation in multi-view 3d object detection with dynamic temporal stereo. In *AAAI*, 2023. 2, 6

[17] Yanwei Li, Yilun Chen, Xiaojuan Qi, Zeming Li, Jian Sun, and Jiaya Jia. Unifying voxel-based representation with transformer for 3d object detection. In *NeurIPS*, 2022. 6

[18] Yinhao Li, Zheng Ge, Guanyi Yu, Jinrong Yang, Zengran Wang, Yukang Shi, Jianjian Sun, and Zeming Li. Bevdepth: Acquisition of reliable depth for multi-view 3d object detection. In *AAAI*, 2023. 1, 2, 6

[19] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *ECCV*, 2022. 1, 2, 3, 4, 5, 6

[20] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 5

[21] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 4

[22] Yingfei Liu, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petr: Position embedding transformation for multi-view 3d object detection. In *ECCV*, 2022. 1, 3, 6

[23] Yingfei Liu, Junjie Yan, Fan Jia, Shuailin Li, Qi Gao, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petrv2: A unified framework for 3d perception from multi-camera images. *arXiv preprint arXiv:2206.01256*, 2022. 3, 6

[24] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. 2018. 5

[25] Jiachen Lu, Zheyuan Zhou, Xiatian Zhu, Hang Xu, and Li Zhang. Learning ego 3d representation as ray tracing. In *ECCV*, 2022. 1, 3, 6

[26] Farzin Negahbani, Onur Berk Töre, Fatma Güney, and Baris Akgun. Frustum fusion: Pseudo-lidar and lidar fusion for 3d detection. *arXiv preprint arXiv:2111.04780*, 2021. 2

[27] Anshul Paigwar, David Sierra-Gonzalez, Özgür Erkent, and Christian Laugier. Frustum-pointpillars: A multi-stage approach for 3d object detection using rgb camera and lidar. In *ICCV*, 2021. 2

[28] Dennis Park, Rares Ambrus, Vitor Guizilini, Jie Li, and Adrien Gaidon. Is pseudo-lidar needed for monocular 3d object detection? In *ICCV*, 2021. 1, 5, 6

[29] Jinhyung Park, Chenfeng Xu, Shijia Yang, Kurt Keutzer, Kris Kitani, Masayoshi Tomizuka, and Wei Zhan. Time will tell: New outlooks and a baseline for temporal multi-view 3d object detection. In *ICLR*, 2023. 2

[30] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *ECCV*, 2020. 1, 2

[31] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *CVPR*, 2018. 2

[32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 2015. 4

[33] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019. 1

[34] Tai Wang, ZHU Xinge, Jiangmiao Pang, and Dahua Lin. Probabilistic and geometric depth: Detecting objects in perspective. In *CoRL*, 2022. 6

[35] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. Fcos3d: Fully convolutional one-stage monocular 3d object detection. In *ICCV*, 2021. 1, 5, 6

[36] Yue Wang, Vitor Campagnolo Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *CoRL*, 2022. 1, 3, 5, 6

[37] Zhixin Wang and Kui Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. In *IROS*, 2019. 2

[38] Zengran Wang, Chen Min, Zheng Ge, Yinhao Li, Zeming Li, Hongyu Yang, and Di Huang. Sts: Surround-view temporal stereo for multi-view 3d detection. *arXiv preprint arXiv:2208.10145*, 2022. 2, 6

[39] Xinshuo Weng and Kris Kitani. Monocular 3d object detection with pseudo-lidar point cloud. In *ICCVW*, 2019. 1

[40] Enze Xie, Zhiding Yu, Daquan Zhou, Jonah Philion, Anima Anandkumar, Sanja Fidler, Ping Luo, and Jose M Alvarez. Mˆ 2bev: Multi-camera joint 3d detection and segmentation with unified birds-eye view representation. *arXiv preprint arXiv:2204.05088*, 2022. 6

[41] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *CVPR*, 2021. 1

[42] Haolin Zhang, Dongfang Yang, Ekim Yurtsever, Keith A Redmill, and Ümit Özgüner. Faraway-frustum: Dealing with lidar sparsity for 3d object detection using fusion. In *ITSC*, 2021. 2

[43] Yunpeng Zhang, Jiwen Lu, and Jie Zhou. Objects are different: Flexible monocular 3d object detection. In *CVPR*, 2021. 1

[44] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018. 1

[45] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv preprint arXiv:1908.09492*, 2019. 5, 6

[46] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2021. 5