FUTR3D: A Unified Sensor Fusion Framework for 3D Detection

Xuanyao Chen[1,2,*] Tianyuan Zhang[1, 3, *] Yue Wang[5] Yilun Wang[6] Hang Zhao[1,4]
[1]Shanghai Qi Zhi Institute [2]Fudan University [3]CMU [4]Tsinghua University
[5]MIT [6]Li Auto

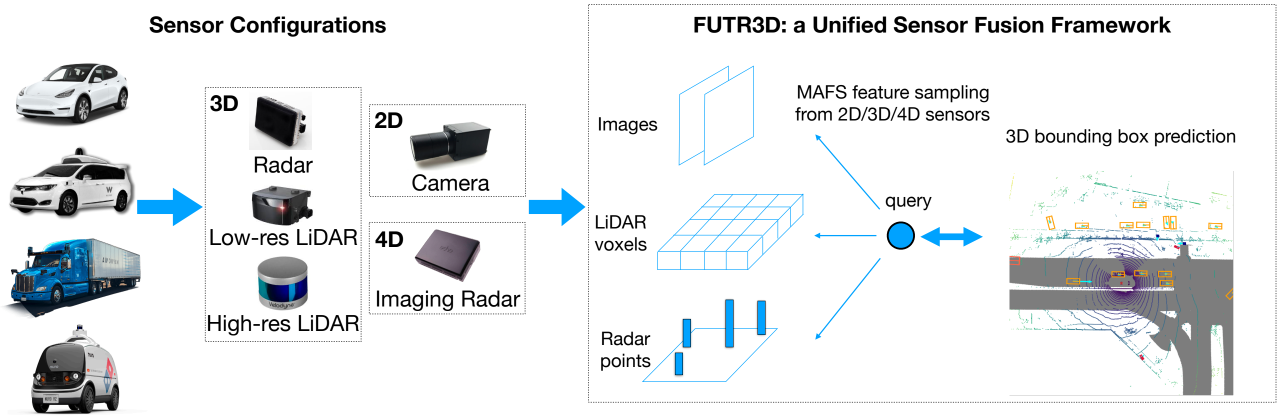


Figure 1. Different self-driving cars have different sensor combinations and setups. FUTR3D is a unified end-to-end sensor fusion framework for 3D detection, which can be used in any sensor configuration, including 2D cameras, 3D LiDARs, 3D radars and 4D imaging radars.

Abstract

Sensor fusion is an essential topic in many perception systems, such as autonomous driving and robotics. Existing multi-modal 3D detection models usually involve customized designs depending on the sensor combinations or setups. In this work, we propose the first unified end-to-end sensor fusion framework for 3D detection, named FUTR3D, which can be used in (almost) any sensor configuration. FUTR3D employs a query-based Modality-Agnostic Feature Sampler (MAFS), together with a transformer decoder with a set-to-set loss for 3D detection, thus avoiding using late fusion heuristics and post-processing tricks. We validate the effectiveness of our framework on various combinations of cameras, low-resolution LiDARs, high-resolution LiDARs, and Radars. On NuScenes dataset, FUTR3D achieves better performance over specifically designed methods across different sensor combinations. Moreover, FUTR3D achieves great flexibility with different sensor configurations and enables low-cost autonomous driving. For example, only using a 4-beam LiDAR with cameras, FUTR3D (58.0 mAP) surpasses state-of-the-art 3D detection model [41] (56.6 mAP) using a 32-beam LiDAR. Our code is available on the project page.

1. Introduction

Sensor fusion is the process of integrating sensory data from disparate information sources. It is an integral part of autonomous perception systems, such as autonomous driving, internet of things, and robotics. With the complementary information captured by different sensors, fusion helps to reduce the uncertainty of state estimation and make more comprehensive and accurate predictions. For instance, on a self-driving car, LiDARs can effectively detect and localize obstacles, while cameras are more capable of recognizing the types of obstacles.

However, multi-sensory systems often come with diverse sensor setups. As shown in Fig.1, each self-driving car system has a proprietary sensor configuration and placement design. For example, a robo-taxi [1, 5, 26] normally has a 360-degree LiDAR and surround-view cameras on the top, together with perimeter LiDARs or Radars around the vehicle; a robo-truck usually has two or more LiDARs on the trailer head, together with long focal length cameras

[*] Equal contribution, work done at Shanghai Qi Zhi Institute.

for long-range perception; a passenger car relies on cameras and Radars around the vehicle to perform driver assistance. Customizing specialized algorithms for different sensor configurations requires huge engineering efforts. Therefore, designing a unified and effective sensor fusion framework is of great value.

Previous research works have proposed several sophisticated designs for LiDAR and camera fusion. Proposal-based methods either propose bounding boxes from the LiDAR point clouds and then extract corresponding features from camera images [6, 10], or propose frustums from the images and further refine bounding boxes according to point clouds in the frustums [21]. Feature projection-based methods [15, 37] associate modalities by either projecting point features onto the image feature maps, or painting the point clouds with colors [28]. Conducting camera-radar fusion usually involves more complicated feature alignment techniques due to the sparsity of Radar signals [3, 9, 18, 19].

Our work introduces the first end-to-end 3D detection framework, named FUTR3D (**Fu**sion **Tr**ansformer for **3D Detection**), that can work with any sensor combinations and setups, *e.g.* camera-LiDAR fusion, camera-radar fusion, camera-LiDAR-Radar fusion. FUTR3D first encodes features for each modality individually, and then employs a query-based Modality-Agnostic Feature Sampler (MAFS) that works in a unified domain and extract features from different modalities. Finally, a transformer decoder operates on a set of 3D queries and performs set predictions of objects. The design of MAFS and transformer decoder makes the model end-to-end and inherently modality agnostic.

The contributions of our work are the following:

- To the best of our knowledge, FUTR3D is the first unified sensor fusion framework that can work with any sensor configuration in an end-to-end manner.
- We design a Modality-Agnostic Feature Sampler, called MAFS. It samples and aggregates features from cameras, high-resolution LiDARs, low-resolution LiDARs and Radars. MAFS enables our method to operate on any sensors and their combinations in a modality agnostic way. This module is potentially applicable to any multi-modal use cases.
- FUTR3D outperforms specifically designed fusion methods across different sensor combinations. For example, FUTR3D outperforms PointPainting [28] without bells and whistles even though PointPainting is designed to work on high-resolution LiDARs and images.
- FUTR3D achieves excellent flexibility with different sensor configurations and enables low-cost perception systems for autonomous driving. On the nuScenes [1] dataset, FUTR3D achieves 58.0 mAP with a 4-beam LiDAR and camera images, which surpasses the state-of-the-art 3D detection model with a 32-beam LiDAR.
- We will release code to promote future research.

2. Related Work

2.1. LiDAR-based 3D Detection

The mainstreams of LiDAR-based detectors in autonomous driving quantify the 3D space into voxels or pillars, then use convolutional backbones to extract a stack of Bird's-eye view feature maps [11, 40, 41, 45]. Detectors within this framework draw lots of experiences from 2D detector designs. Besides voxel representations, point-based [22, 23], and range view [4, 7, 12, 27] are also explored. PointNet architecture has been used in VoxelNet [45], Lidar-RCNN [14] to extract feature for a small region of irregular points. Several works [7, 12] demonstrate the computational efficiency of Range view. MVF [44] and Pillar-OD [31] introduce multi-view projection to learn view-complementary features. Object DGCNN [33] models object relations using DGCNN [34] and presents the first set prediction based 3D object detection pipeline.

2.2. Camera-based 3D Detection

Directly migrated from 2D object detection, Monodis [24] learns a single-stage 3D object detector on monocular images. FCOS3D [29] considers 3D object detection on multi-view images. It predicts 3D bounding box per image and aggregates predictions in a post-processing step. Pseudo Lidar [30] lifts images into the 3D space and employs a point cloud based pipeline to perform 3D detection. DETR3D [32] designs a set-based 3D object detection model which operates on multi-view images. DETR3D uses camera transformation to link 2D feature extraction to 3D predictions. Also, it does not require post-processing, thanks to the set prediction module. Our method is closely related to DETR3D in the sense that we use a similar object detection head and feature sampling modules. In contrast to DETR3D, our feature sampling module is *modality agnostic* which makes it work for sensor fusion.

2.3. Multi-modal 3D Detection

Apart from classical heuristic late fusion techniques, we can roughly divide learning-based multi-modal fusion methods into two types: proposal-based methods and feature projection-based methods.

Proposal-based methods have gained a lot of popularity in the past few years. The idea behind such methods is to propose objects from one sensor modality, and then refine it on the other(s). MV3D [6] first generates 3D object proposals in bird's eye view using LiDAR features, and then projects them to camera view and LiDAR front view to fuse LiDAR camera features. Frustum-PointNet [21] and Frustum-ConvNet [36] use 2D object detectors to generate 2D proposals in the camera view, then lift 2D proposals to 3D frustums, and finally use perform 3D box estimation

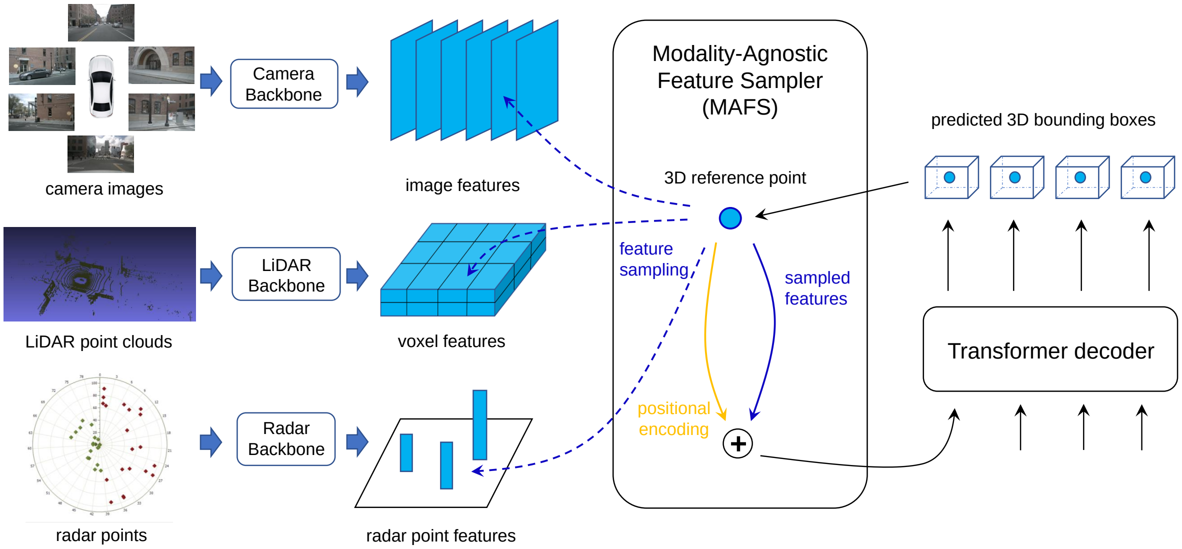


Figure 2. Overview of FUTR3D. Each sensor modality is encoded individually in its own coordinate. Then a query-based Modality-Agnostic Feature Sampler (MAFS) extracts features from all available modalities according to the 3D reference point of each query. Finally a transformer decoder predicts 3D bounding boxes from queries. The predicted boxes can be iteratively fed back into MAFS and transformer decoder to refine the predictions.

within the frustums. AVOD [10] places dense anchor boxes in bird's eye view and then projects these anchors to camera images and LiDAR voxels for feature fusion and region proposal.

Feature projection-based methods usually perform feature projection and fusion before the detection head. By finding the point-pixel correspondence, [37] enables middle-level feature fusion, and makes a singe-stage detector. Cont-Fuse [15] further uses a KNN to better find matching points for image pixels, and fuses features at multiple levels. The top-performing camera-LiDAR based 3D detection algorithm PointPaint [28] projects LiDAR points onto the prediction map of a pre-trained image semantic segmentation network, and fuse the semantic prediction label with the intensity measurement of each point. MVP [42] generate dense virtual points with semantic label to make more use of image information. Although feature projection-based methods have recently achieved impressive performance, their designs usually require a lot of heuristics and understanding of the sensor modalities.

There are also a handful of works on fusing camera images and radar signals that share similar spirits with camera-LiDAR fusion. [3, 9, 19] encode both camera images and radar signals in the perspective view and fuse them by simple feature map concatenation. CenterFusion [18] proposes 3D bounding boxes from images, and further refines them by fusing their features with radar signals in the bird's eye view representations.

3. Approach

FUTR3D can be conceptually divided into four parts. First, the data from different sensor modalities can be encoded by their modality-specific feature encoders (§3.1). Then, a query-based Modality Agnostic Feature Sampler (MAFS) is used to sample and aggregate features from all modalities, according to initial positions of the query (§3.2); this is the major novelty of this work. Next, a shared transformer decoder head is used to refine the bounding box predictions based on the fused features using an iterative refine module (§3.3). Finally, our loss is based on set-to-set matching between predictions and ground-truths (§3.4). FUTR3D is designed to be a unified framework for *multi-modal* sensor fusion, which makes single-modal methods like DETR3D [32] and Object DGCNN [33] special cases of our method. For ease of presentation, we use the same notation as FUTR3D. An overview of FUTR3D is shown in Figure 2.

3.1. Modality-specific Feature Encoding

FUTR3D learns features from each modality independently. Since our framework does not make assumptions about the modalities used or their model architectures, our model works with *any* choices of feature encoders. This work focuses on three types of data: LiDAR point clouds, radar point clouds, and multi-view camera images.

For LiDAR point clouds, we use a VoxelNet [40, 45] to encode LiDAR point clouds. After 3D backbone and FPN [16], we obtain multi-scale Bird's-eye view (BEV) feature maps $\{\mathcal{F}^j_{\text{lid}} \in \mathbb{R}^{C \times H_j \times W_j}\}_{j=1}^m$, where C is the output channel and $H_i \times W_i$ is the size of the i-th BEV feature map.

We pillarize Radar points $\{r_j\}_{j=1}^{N} \in \mathbb{R}^{C_{ri}}$ into 0.8m pillars. We adopt a MLP Φ_{rad} to obtain per-pillar features $\mathcal{F}_{\text{rad}}^{j} = \Phi_{\text{rad}}(r_j) \in \mathbb{R}^{C_{ro}}$, where C_{ro} denoted the number of encoded radar features. We obtain the Radar BEV feature map $\mathcal{F}_{\text{rad}} \in \mathbb{R}^{C_{ro} \times H \times W}$.

On a typical self-driving car, we have N surrounding cameras. Following prior works [28, 29, 32, 41], we use ResNet [8] or VoVNet [20] and FPN [16] for image feature extraction, it outputs multi-scale features for each image, denoted as $\mathcal{F}_{\text{cam}}^{k} = \{\mathcal{F}_{\text{cam}}^{kj} \in \mathbb{R}^{C \times H_j \times W_j}\}_{j=1}^{m}$ for the k-th image.

3.2. Modality-Agnostic Feature Sampler

The most critical part of FUTR3D is the feature sampling process, termed Modality-Agnostic Feature Sampler (MAFS). The input of our detection head is a set of object queries $Q = \{q_i\}_{i=1}^{N_q} \subset \mathbb{R}^{C}$, and features from all sensors. MAFS updates each query by sampling features from each sensor feature and fusing them.

Initial 3D reference points. Following Anchor-DETR [35], we first randomly initialize the queries with N_q reference points $\{c_i\}_{i=1}^{N_q}$, where $c_i \in [0,1]^3$ presents relative coordinates in 3D space. Then, this 3D reference point serves as an anchor to gather features from multiple sources. The initial reference point does not depend on features from any sensors, and it will update dynamically after fusing features from all modalities.

LiDAR point feature sampling The point cloud features after the 3D backbone and FPN [16] are denoted as $\{\mathcal{F}_{lid}^{j}\}_{j=1}^{m}$. Following Deformable Attention [46], we sample K points from each scale feature map. We use $\mathcal{P}(c_i)$ to denote the projection of the 3D reference point in the BEV. We sample LiDAR features from all multi-scale BEV feature maps and sum them:

$$\mathcal{SF}_{\text{lid}}^{i} = \sum_{j=1}^{m} \sum_{k=1}^{K} \mathcal{F}_{\text{lid}}^{j}(\mathcal{P}(c_i) + \Delta_{\text{lid}}^{ijk}) \cdot \sigma_{\text{lid}}^{ijk} \tag{1}$$

where $\mathcal{SF}_{\text{lid}}^{i}$ means the sampled LiDAR point features for i-th reference point, $\Delta_{\text{lid}}^{ijk}$ and $\sigma_{\text{lid}}^{ijk}$ is the predicted sampling offsets and attention weights, $\mathcal{F}_{\text{lid}}^{j}(\mathcal{P}(c_i) + \Delta_{ijk})$ represents the bilinear sampling from the BEV feature map.

Radar point feature sampling Similar to LiDAR feature sampling, we use Deformable Attention to sample Radar features as:

$$\mathcal{SF}_{\text{rad}}^{i} = \sum_{k=1}^{K} \mathcal{F}_{\text{rad}}(\mathcal{P}(c_i) + \Delta_{\text{rad}}^{ik}) \cdot \sigma_{\text{rad}}^{ik} \tag{2}$$

where Δ_{rad}^{ik} and σ_{rad}^{ik} is the predicted sampling offsets and attention weights.

Image feature sampling We project the reference point $\underline{\text{c}}_i$ to image of k-th camera by utilizing camera's intrinsic and extrisic parameters and denote the coordinates of projected reference point as $\mathcal{T}_k(c_i)$ We use the projected image coordinates $\mathcal{T}_k(c_i)$ to sample point features from feature maps of all cameras, and perform weighted sum:

$$\mathcal{SF}_{\text{cam}}^{i} = \sum_{k=1}^{N} \sum_{j=1}^{m} \mathcal{F}_{\text{cam}}^{kj}(\mathcal{T}_k(c_i)) \cdot \sigma_{\text{cam}}^{ikj} \tag{3}$$

where $\mathcal{F}_{\text{cam}}^{kj}(\mathcal{T}_k(c_i))$ denotes the blinear sampling using the image coordinates. Then scalar weight $\sigma_{\text{cam}}^{ijk}$ is also decoded from object query q_i using a linear layer and normilzing by sigmoid.

Modality-agnostic feature fusion After sampling point features from all modalities, we fuse features and update queries. First, we concatenate sampled features from all modalities and encode them using a MLP network Φ_{fus} given by:

$$\mathcal{SF}_{\text{fus}}^{i} = \Phi_{\text{fus}}(\mathcal{SF}_{\text{lid}}^{i} \oplus \mathcal{SF}_{\text{cam}}^{i} \oplus \mathcal{SF}_{\text{rad}}^{i}), \tag{4}$$

where $\mathcal{SF}_{\text{fus}}^{i}$ is the fused per query features. Then, we add the positional encoding $\text{PE}(c_i)$ of reference points to the fused features to make them location aware. Finally, we update the queries accordingly by $q_i = q_i + \Delta q_i$, where

$$\Delta q_i = \mathcal{SF}_{\text{fus}}^{i} + \text{PE}(c_i). \tag{5}$$

Then object queries are updated using self-attention modules and FFN. Our method works with any sensor combinations thanks to this modality-agnostic feature fusion module. We use the same object detection head throughout.

3.3. Iterative 3D Box Refinement

Each block $\ell \in \{1, 2, \ldots, L\}$ of transformer decoder in our detection head produces a set of updated object queries $Q^{\ell} = \{q_i^{\ell}\}_{i=1}^{M} \subset \mathbb{R}^{C}$. We predict a sequence of iteratively refined boxes given the queries. Specifically, for each object query q_i^{ℓ}, we use a shared MLP Φ_{reg} to predict offset to box center coordinate $\Delta x_i^{\ell} \in \mathbb{R}^3$, box size $(w_i^{\ell}, h_i^{\ell}, l_i^{\ell})$, orientation $(\sin\theta_i^{\ell}, \cos\theta_i^{\ell})$, velocity ($v_i^{\ell} \in \mathbb{R}^2$) and another Φ_{cls} for its categorical label $\hat{y}_i^{\ell}$.

Following [32, 33, 46], we adopt an iterative refinement approach. We use the predictions of box center coordinates in the last layer as the 3D reference points for each query, except for the first layer which directly decode and input-agnostic reference points from the object queries. The next layer's reference points are given by:

$$c_i^{\ell+1} = c_i^{\ell} + \Delta x_i^{\ell} \tag{6}$$

Table 1. **Comparison with leading methods on nuScenes *test* set.** FUTR3D either surpasses or achieves comparable performance with state-of-the-art methods in single-modality settings, including those that use LiDAR and cameras separately, as well as in LiDAR-camera fusion settings. 'L' and 'C' represent LiDAR and cameras respectively. Our methods use VoVNet for camera backbones and VoxelNet with 0.075 meter size for LiDAR backbone.

	Modality	NDS ↑	mAP ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓
FCOS3D [29]	C	40.2	32.6	74.3	25.9	44.1	134.1	16.3
DD3D [20]	C	47.7	41.8	57.2	24.9	36.8	101.4	12.4
FUTR3D	C	47.9	41.2	64.1	25.5	39.4	84.5	13.3
UVTR [13]	L	69.7	63.9	30.2	24.6	35.0	20.7	12.3
TransFusion [39]	L	70.2	65.5	25.6	24.0	35.1	27.8	12.9
FUTR3D	L	69.9	65.3	28.1	24.7	36.8	25.3	12.4
MVP [42]	L+C	70.5	66.4	26.3	23.8	32.1	31.3	13.4
FusionPainting [38]	L+C	71.6	68.1	25.6	23.6	34.6	27.4	13.2
UVTR [13]	L+C	71.1	67.1	30.6	24.5	35.1	22.5	12.4
TransFusion [39]	L+C	71.7	68.9	25.9	24.3	35.9	28.8	12.7
FUTR3D	L+C	72.1	69.4	28.4	24.1	31.0	30.0	12.0

Table 2. **Camera with Low-Resolution LiDAR results on nuScenes *val* set.** FUTR3D significantly outperforms CenterPoint with low-resolution LiDAR, and PointPainting with camera+low-resolution LiDAR. Under camera + 4-beam LiDAR setting, FUTR3D achieves 58.0 mAP, which surpasses state-of-the-art LiDAR detector CenterPoint with 32-beam LiDAR (56.6 mAP).

(a) **Camera+4 beam LiDAR**

	Modality	NDS ↑	mAP ↑
CenterPoint [41]	L	53.6	38.5
FUTR3D	L	56.4	44.3
PointPainting [28]	L+C	59.4	50.0
FUTR3D	L+C	64.2	58.0

(b) **Camera+1 beam LiDAR**

	Modality	NDS ↑	mAP ↑
CenterPoint [41]	L	36.9	14.5
FUTR3D	L	39.2	16.9
PointPainting [28]	L+C	41.0	22.0
FUTR3D	L+C	51.9	43.4

3.4. Loss

Following [2, 32, 33, 46], we compute a set-to-set loss between predictions and ground-truths using one-to-one matching. We adopt the focal loss for classification and L1 regression loss for 3D bounding box as DETR3D [32].

As noted by Co-DETR [47], sparse supervision, such as one-to-one set loss, can impede learning effectiveness. To learn more discriminative LiDAR feature, we incorporate an auxiliary one-stage detector head, i.e. head used in CenterPoint [41], after the LiDAR encoder. This one-stage head is jointly trained with original transformer decoder. Notably, the auxiliary head is only used during LiDAR training and is not used during inference.

4. Experiments

4.1. Implementation Details

Dataset. We use nuScenes [1] dataset for all experiments. This dataset consists of 3 modalities, namely 6 cameras, 5 radars and 1 LiDAR. All are captured with a full 360-degree field of view. There are totally 1000 sequences, where each sequence has roughly 40 annotated keyframes. The keyframes are synchronized across sensors with a sampling rate of 2 FPS.

Cameras. In each frame, nuScenes [1] provides images from six cameras `[front_left, front, front_right, back_left, back, back_right]`; there are overlap regions between cameras and the whole scene is covered. The resolution is 1600×900.

LiDAR. nuScenes provides a 32-beam LiDAR, which spins at 20 FPS. Since only key frames are annotated at 2 FPS, we follow the common practice to transform points from the past 9 frames to the current frame.

Low-resolution LiDAR. Low-resolution LiDARs are often used in many low cost uses cases. We consider these low-resolution LiDARs as complementary setups to high-resolution LiDARs because they are scalable to be deployed on production-ready platforms. We simulate low-resolution LiDAR outputs from the 32-beam LiDAR. We first convert points from Cartesian coordinate system to spherical coordinates: range r, inclination θ, and azimuth ϕ. The range of inclination is $[-30^o, 10^o]$ For 4-beam LiDAR, we select beams whose inclination θ fall within $[-7.1°, -5.8°] \cup [-4.5°, -3.2°] \cup [-1.9°, -0.6°] \cup [0.7°, 2.0°]$. For 1-beam LiDAR, we select the beam with pitch angle in $[-1.9°, -0.6°]$.

Radar. We stack points captured by all five radars into a single point cloud; each point cloud contains 200 to 300 points each frame. We use radar coordinates, velocity measurements, and intensities. We filter radar points using the official tool provided by nuScenes.

Model setting. The feature dimension C is all 256 for LiDAR feature $\mathcal{F}_{\text{lid}}$, image feature $\mathcal{F}_{\text{cam}}$, object queries Q. For Radar points, $C_{ri} = 6$ and $C_{ro} = 64$. There are $N_q = 900$ object queries. In LiDAR and image feature extraction, we use $M = 4$ layers of multi-scale features encoded by FPN. We use $K = 4$ sampling offsets when using Deformable Attention. There are total $L = 6$ blocks in the transformer decoder of the detection head.

Training details. For LiDAR-based detectors, we train them for 20 epochs using AdamW [17] optimizer. We set the learning rate as 1.0×10^{-4} and adopt cyclic learning rate policy [25]. We remove object sampling [40] augmentation in the last 5 epochs. For LiDAR-camera models, we pre-train the LiDAR backbone and the camera backbone respectively, then jointly fine-tune the model for another 6 epochs. For camera-radar models, we pre-train the image model, followed by joint training on cameras and radars. We set classification loss and L1 regression weights as 2.0 and 0.25. The auxiliary head loss weight is set as 0.5 in LiDAR training.

Evaluation metrics. Mean Average Precision (mAP) and nuScenes Detection Score (NDS) are the major metrics for the nuScenes 3D detection benchmark. For mAP, nuScenes considers the distances between the centers of the bounding boxes on bird's eye view. NDS measures the quality of detection results by consolidating several breakdown metrics: Average Translation Error (ATE), Average Scale Error (ASE), Average Orientation Error (AOE), Average Velocity Error (AVE), and Average Attribute Error (AAE). We average the metrics over all classes, following the official evaluation protocol.

4.2. Multi-modal Detection

We demonstrate the effectiveness of our framework under several sensor combinations.

High-resolution LiDAR with Cameras is the most commonly used sensor combination for autonomous driving. We compare our method with state-of-the-art methods in Table 1. FUTR3D achieves 72.1% NDS and 69.4% mAP on nuScenes *test* set, surpassing TransFusion by 0.4% NDS and 0.5% mAP. Moreover, FUTR3D achieves results comparable to the state-of-the-art in LiDAR-only and camera-only settings.

Low-resolution LiDAR with Cameras. We also investigate the use of low-resolution LiDAR with cameras for cost-effective applications. Specifically, we simulate 4-beam and 1-beam LiDAR configurations, as described in §4.1, and compare our method against the specialized method PointPainting [28] in Table 2. We report the results of the single-modality approaches with low-resolution LiDAR in Table 2.

FUTR3D outperforms PointPainting by **8.0** mAP on 4-beam LiDARs plus cameras setting, and **21.4** mAP on 1-beam LiDARs plus cameras setting. PointPainting uses extra image data from nuImages to pre-train its image segmentation models while our method does not include any additional data. Notably, FUTR3D reaches 58.0 mAP with 4-beam LiDAR with cameras, which outperforms (by 1.4 mAP) one of the top-performing LiDAR detectors, CenterPoint [41] with 32-beam LiDAR (56.6 mAP).

FUTR3D achieves high performance with both low-resolution LiDAR and high-resolution LiDAR, which demonstrates that FUTR3D is a general sensor fusion framework.

Cameras and Radars is a cost-effective sensor setup commonly used in driver-assist systems for passenger cars. Radars provide sparse object localization and velocity information. In addition to the mean average precision (mAP) and normalized detection score (NDS), we also report the mean average velocity error (mAVE) metric, which measures the error of box velocity predictions and is significantly reduced by the use of radar. As demonstrated in Table 3, our method outperforms the state-of-the-art CenterFusion [18] by a significant margin. These results highlight the effectiveness of our approach in utilizing the useful information contained in the sparse radar points. Adding Radars improves over our camera-only version by 5.3 mAP and 8.6 NDS score. ResNet-101 Backbone is used for the camera backbone in this group of experiments.

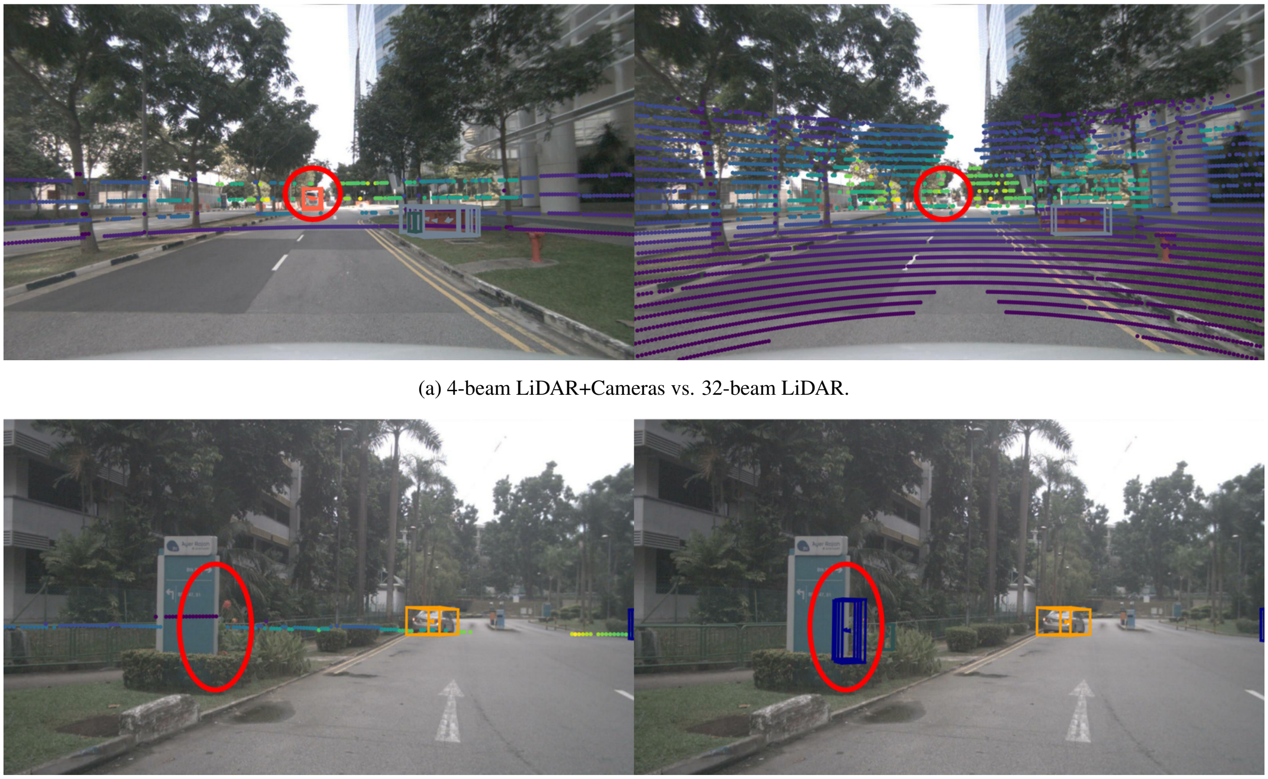
Table 3. **Camera-Radar fusion results** on nuScenes *val* set. FUTR3D outperforms CenterFusion by a large margin. Despite the sparsity of depth estimation and localization information provided by Radar points, they still contribute significantly to improving accuracy and reducing velocity error.

methods	Modality	NDS ↑	mAP ↑	mAVE ↓
CenterNet [43]	C	32.8	30.6	142.6
CenterFusion [18]	C+R	45.3	33.2	54.0
FUTR3D	C	42.5	34.6	84.2
FUTR3D	C+R	**51.1**	**39.9**	**41.3**

4.3. Characteristics of Cameras and LiDARs

As shown in Section 4.2, FUTR3D is a unified detection framework that can work in camera-only, LiDAR-only and camera-LiDAR fusion settings. To the best of our knowledge, it is the first time one can control for the detection method and study the performance gain of sensor fusion comparing to each sensor respectively. To investigate the characteristics of different sensors, we break down the performance of FUTR3D on different camera-LiDAR combinations by object distances, different sizes and object categories. The resultes are based on the FUTR3D with 0.1m voxel for LiDAR and ResNet-101 for cameras.

Object category. We report Average Precision (AP) of



(a) 4-beam LiDAR+Cameras vs. 32-beam LiDAR.

(b) 1-beam LiDAR+Cameras vs. Cameras.

Figure 3. Qualitative results of FUTR3D. We show perspective image view results by projecting LiDAR points onto images. (a) There is a car in the distance marked in red circle which can be detected using 4-beam LiDAR with cameras. (b) The billboard circled in red is detected falsely as pedestrian using vision only. This can be corrected with the help of 1-beam LiDAR.

Table 4. Performance breakdown by object categories. Cameras help LiDAR-based detectors significantly on bicycles, traffic cones and motorcycles. Abbreviations: construction vehicle (CV), pedestrian (Ped), motorcycle (Motor), and traffic cone (TC).

modalities	AP									
	Car	Truck	Bus	Trailer	CV	Ped	Motor	Bicycle	TC	Barrier
Camera only	54.3	29.8	36.2	16.7	7.7	41.6	32.9	29.6	51.1	47.6
1-beam LiDAR	29.6	16.0	30.4	13.8	3.3	32.5	6.4	6.0	9.5	22.0
1-beam LiDAR + Camera	61.1	39.7	49.6	22.6	12.6	55.2	39.7	33.1 ↑ 27.1	52.1 ↑ 42.6	47.2
4-beam LiDAR	70.1	39.4	50.5	30.0	12.2	68.6	44.2	17.3	41.0	47.5
4-beam LiDAR + Camera	78.0	54.4	61.5	32.2	20.4	75.7	61.4	53.9 ↑ 36.6	58.1 ↑ 17.1	53.9
32-beam LiADR	84.3	53.4	65.2	41.6	25.2	81.5	66.4	48.7	64.1	62.6
32-beam LiDAR + Camera	86.3	61.5	71.9	42.1	26.0	82.6	73.6	63.3 ↑ 14.6	70.1 ↑ 6.0	64.4

our LiDAR and cameras methods for every object category in Table 4. Though the overall mAP of the 4-beam LiDAR only FUTR3D is higher than the camera-only FUTR3D (42.1 mAP *v.s.* 34.6 mAP). The camera-only model outperforms the 4-beam LiDAR model on bicycles, traffic cones and barriers, showing that 4-beam LiDAR are not good at detecting small objects. Moreover, when equipping the 4-beam LiDAR with cameras, the performance on bicycles, traffic cones and motorcycles are significantly boosted.

Object distance. We split ground truth boxes into three subsets given the distances of box centers to ego vehicle: $[0m, 20m]$, $[20m, 30m]$, $[30m, +\infty]$, with each group taking up 42.93%, 28.27%, 28.8% of all the ground truth boxes. Note that boxes of `traffic_cone` and `barrier` with distances larger than 30 meters will be automatically filtered out following official evaluation protocols of nuScenes 3D detection. Table 5 shows the results. For boxes farther than 30 meters, our camera-only FUTR3D

Table 5. Performance breakdown by object distance. We split boxes given its ego distance and report mAP independently. Results show **cameras help LiDAR-based detectors more on farther objects.**

	mAP		
	$[0m, 20m]$	$[20m, 30m]$	$[30m, +\infty]$
Camera only	49.3	25.4	10.4
4-beam LiDAR	61.1	39.3	16.1
4-beam LiDAR + Camera	69.8 ↑ 8.7	50.5 ↑ 10.2	27.4 ↑ 11.3
32-beam LiDAR	73.8	55.2	29.9
32-beam LiDAR + Camera	76.8 ↑ 3.0	58.8 ↑ 3.6	36.7 ↑ 6.8

Table 6. Performance breakdown by object size. We split the boxes given its longest edge. Results indicate that **cameras improve LiDAR-based detectors more on small objects.**

	mAP	
	$[0m, 4m]$	$[4m, +\infty]$
Camera only	22.3	13.9
1-beam LiDAR	9.2	8.9
1-beam LiDAR + Camera	25.5 ↑ 16.3	17.8 ↑ 8.9
4-beam LiDAR	25.3	19.4
4-beam LiDAR + Camera	33.9 ↑ 8.6	23.4 ↑ 4.0
32-beam LiADR	36.4	25.7
32-beam LiDAR + Camera	39.5 ↑ 3.1	27.4 ↑ 1.7

only achieves 10.4 mAP, when our 4-beam LiDAR model obtains 16.1 mAP. However, fusing these two sensors elevate the performance to the next level (27.4 mAP). Even for the 32-beam LiDAR model, additional camera sensors can improve the performance of our model on farther objects from 29.9 mAP to 36.7 mAP. Adding cameras improves LiDAR perception the most on farther regions.

Object size. We split ground truth boxes into three subsets: $[0m, 4m]$ & $[4m, +\infty]$ based on the longer edge of the 3D box, each group occupying 46.18%, and 53.82% of the gt boxes. Table 6 reports the mAP of our camera LiDAR models on each group. The performance improvements introduced by adding cameras to all LiDAR models are larger on small objects than on large objects. Cameras improve LiDAR-based detectors more on small objects since cameras have much high resolution than even 32-beam LiDARs. However, the performance improvements introduce by adding different LiDARs to camera-only models are roughly the same for small and large objects, meaning depth information are equally useful when localizing both small and large objects.

4.4. Ablation Study

Auxiliary LiDAR Head. We demonstrate the impact of our auxiliary LiDAR head on improving performance in Table 7. Our results clearly show that the auxiliary LiDAR head significantly boosts performance by 3.9 mAP. It is worth noting that the auxiliary head is only utilized during training, and model inference remains unchanged.

Table 7. Ablation on auxiliary LiDAR Head using nuScenes validation set. The auxiliary head is only utilized in training.

Aux. Head	NDS	mAP
w/o	66.1	59.8
w/	69.1	63.7

Camera backbone choices. We show the results of FUTR3D under different camera backbones. We experimented with ResNet-101 and VoVNet for image backbones.

Table 8. Ablation on camera backbones using nuScenes *test* set. Voxelnet with voxel-size of 0.075 meter is used for LiDAR backbone. The first row shows results of LiDAR-only version of FUTR3D.

Cameras	NDS	mAP
—	69.9	65.3
ResNet-101	70.4	67.2
VoVNet	72.1	69.4

4.5. Qualitative Results

In Figure 3, we visualize and compare the results with different settings. In Figure 3a, we show the results of 4-beam LiDAR + cameras (left) and 32-beam LiDAR only (right). Using sparse LiDAR beams and cameras, our method is still able to detect the car in the distance, circled in red, when it is missed using 32-beam LiDAR. Cameras provide denser pixels than LiDAR beams which could be useful for detecting far away objects. In Figure 3b, we show the results of 1-beam LiDAR + cameras (left) and cameras only (right). With the help of only 1-beam LiDAR, camera is able to eliminate the false positive in red circle as LiDAR gives geometry information directly to the model. This validates the effectiveness of FUTR3D framework. Furthermore, this is in line with our assumption that in many cases, low-cost LiDARs and cameras are comparable with expensive LiDARs in object recognition.

5. Discussion and Conclusion

We observe two potential limitations. First, our training pipeline requires a two-stage training: camera encoders and LiDAR encoders are first pre-trained independently on the same detection task, followed by a joint fine-tuning. This suggests a venue for further investigation into the multimodal optimization techniques for 3D object detection.

To conclude, in this work we propose a unified end-to-end sensor fusion framework for 3D object detection. Our insight is that a query-based modality-agnostic feature sampler (MAFS) enables models to work with any sensor combinations and setups. We hope this architecture can serve as a foundation framework for multi-modal fusion and scene understanding.

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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 1, 2, 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. 5

[3] Simon Chadwick, Will Maddern, and Paul Newman. Distant vehicle detection using radar and vision. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8311–8317. IEEE, 2019. 2, 3

[4] Yuning Chai, Pei Sun, Jiquan Ngiam, Weiyue Wang, Benjamin Caine, Vijay Vasudevan, Xiao Zhang, and Dragomir Anguelov. To the Point: Efficient 3D Object Detection in the Range Image With Graph Convolution Kernels. In *CVPR*, 2021. 2

[5] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019. 1

[6] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-View 3D Object Detection Network for Autonomous Driving. In *CVPR*, pages 1907–1915, 2017. 2

[7] Lue Fan, Xuan Xiong, Feng Wang, Naiyan Wang, and ZhaoXiang Zhang. rangedet: In defense of range view for lidar-based 3d object detection. 2

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, 2016. 4

[9] Vijay John and Seiichi Mita. Rvnet: Deep sensor fusion of monocular camera and radar for image-based obstacle detection in challenging environments. In *Pacific-Rim Symposium on Image and Video Technology*, pages 351–364. Springer, 2019. 2, 3

[10] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3D Proposal Generation and Object Detection from View Aggregation. In *IROS*, pages 1–8. IEEE, 2018. 2, 3

[11] 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*, pages 12697–12705, 2019. 2

[12] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle Detection from 3D Lidar Using Fully Convolutional Network. 2016. 2

[13] Yanwei Li, Yilun Chen, Xiaojuan Qi, Zeming Li, Jian Sun, and Jiaya Jia. Unifying voxel-based representation with transformer for 3d object detection. In *Advances in Neural Information Processing Systems*, 2022. 5

[14] Zhichao Li, Feng Wang, and Naiyan Wang. LiDAR R-CNN: An Efficient and Universal 3D Object Detector. In *CVPR*, 2021. 2

[15] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep Continuous Fusion for Multi-Sensor 3D Object Detection. In *ECCV*, pages 641–656, 2018. 2, 3

[16] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *CVPR*, pages 2117–2125, 2017. 3, 4

[17] Ylya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019. 6

[18] Ramin Nabati and Hairong Qi. Centerfusion: Center-based radar and camera fusion for 3d object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1527–1536, 2021. 2, 3, 6

[19] Felix Nobis, Maximilian Geisslinger, Markus Weber, Johannes Betz, and Markus Lienkamp. A deep learning-based radar and camera sensor fusion architecture for object detection. In *2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pages 1–7. IEEE, 2019. 2, 3

[20] 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. 4, 5

[21] 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*, pages 918–927, 2018. 2

[22] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *CVPR*, pages 652–660, 2017. 2

[23] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *NeurIPS*, pages 5099–5108, 2017. 2

[24] Andrea Simonelli, Samuel Rota Bulo, Lorenzo Porzi, Manuel López-Antequera, and Peter Kontschieder. Disentangling monocular 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1991–1999, 2019. 2

[25] Leslie Smith. Cyclical Learning Rates for Training Neural Networks. In *WACV*, 2017. 6

[26] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *CVPR*, pages 2446–2454, 2020. 1

[27] Pei Sun, Weiyue Wang, Yuning Chai, Gamaleldin Elsayed, Alex Bewley, Xiao Zhang, Cristian Sminchisescu, and Dragomir Anguelov. rsn: Range sparse net for efficient, accurate lidar 3d object detection. 2

[28] Sourabh Vora, Alex H Lang, Bassam Helou, and Oscar Beijbom. PointPainting: Sequential Fusion for 3D Object Detection. In *CVPR*, pages 4604–4612, 2020. 2, 3, 4, 5, 6

[29] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. Fcos3d: Fully convolutional one-stage monocular 3d object detection. *arXiv preprint arXiv:2104.10956*, 2021. 2, 4, 5

[30] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in

3D Object Detection for Autonomous Driving. In *CVPR*, pages 8445–8453, 2019. 2

[31] Yue Wang, Alireza Fathi, Abhijit Kundu, David A. Ross, Caroline Pantofaru, Thomas A. Funkhouser, and Justin M. Solomon. Pillar-based object detection for autonomous driving. In *The European Conference on Computer Vision (ECCV)*, 2020. 2

[32] 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 *5th Annual Conference on Robot Learning*, 2021. 2, 3, 4, 5

[33] Yue Wang and Justin M. Solomon. Object dgcnn: 3d object detection using dynamic graphs. In *2021 Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 2, 3, 4, 5

[34] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 2

[35] Yingming Wang, Xiangyu Zhang, Tong Yang, and Jian Sun. Anchor detr: Query design for transformer-based detector. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2567–2575, 2022. 4

[36] Zhixin Wang and Kui Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1742–1749. IEEE, 2019. 2

[37] Zining Wang, Wei Zhan, and Masayoshi Tomizuka. Fusing bird's eye view lidar point cloud and front view camera image for 3d object detection. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–6. IEEE, 2018. 2, 3

[38] Shaoqing Xu, Dingfu Zhou, Jin Fang, Junbo Yin, Zhou Bin, and Liangjun Zhang. Fusionpainting: Multimodal fusion with adaptive attention for 3d object detection. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 3047–3054. IEEE, 2021. 5

[39] Xinge Zhu Qingqiu Huang Yilun Chen Hongbo Fu Xuyang Bai, Zeyu Hu and Chiew-Lan Tai. TransFusion: Robust Lidar-Camera Fusion for 3d Object Detection with Transformers. *CVPR*, 2022. 5

[40] Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely Embedded Convolutional Detection. *Sensors*, 18(10):3337, 2018. 2, 3, 6

[41] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3D Object Detection and Tracking. *arXiv preprint arXiv:2006.11275*, 2020. 1, 2, 4, 5, 6

[42] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Multimodal virtual point 3d detection. *NeurIPS*, 2021. 3, 5

[43] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as Points. *arXiv preprint arXiv:1904.07850*, 2019. 6

[44] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3d object detection in LiDAR point clouds. In *The Conference on Robot Learning (CoRL)*, 2019. 2

[45] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *CVPR*, pages 4490–4499, 2018. 2, 3

[46] 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. 4, 5

[47] Zhuofan Zong, Guanglu Song, and Yu Liu. Detrs with collaborative hybrid assignments training, 2022. 5