Sparse4D: Multi-view 3D Object Detection with Sparse Spatial-Temporal Fusion

Xuewu Lin, Tianwei Lin, Zixiang Pei, Lichao Huang, Zhizhong Su
Horizon Robotics
xuewu.lin@horizon.ai

Abstract

*Bird-eye-view (BEV) based methods have made great progress recently in multi-view 3D detection task. Comparing with BEV based methods, sparse based methods lag behind in performance, but still have lots of non-negligible merits. To push sparse 3D detection further, in this work, we introduce a novel method, named Sparse4D, which does the iterative refinement of anchor boxes via sparsely sampling and fusing spatial-temporal features. (1) **Sparse 4D Sampling**: for each 3D anchor, we assign multiple 4D keypoints, which are then projected to multi-view/scale/timestamp image features to sample corresponding features; (2) **Hierarchy Feature Fusion**: we hierarchically fuse sampled features of different view/scale, different timestamp and different keypoints to generate high-quality instance feature. In this way, Sparse4D can efficiently and effectively achieve 3D detection without relying on dense view transformation nor global attention, and is more friendly to edge devices deployment. Furthermore, we introduce an instance-level depth reweight module to alleviate the ill-posed issue in 3D-to-2D projection. In experiment, our method outperforms all sparse based methods and most BEV based methods on detection task in the nuScenes dataset. Code is available at https://github.com/linxuewu/Sparse4D.*

1. Introduction

Multi-view visual 3D perception plays a critical role in autonomous driving systems, especially for low-cost deployment. Compared with Lidar modality, cameras can provide valuable visual cues for long-range distance detection and vision-only element identification. However, without explicit depth cues, 3D perception from 2D images is an ill-posed issue, leading to a long-standing challenge of how to properly fuse multi-camera images to address 3D perception tasks such as 3D detection. There are two mainstream categories of recent methods: the BEV-based methods and the sparse-based methods.

BEV-based methods [12, 17–19, 29, 49] address 3D detection via converting multi-view image features into an unified BEV space, and achieve excellent performance promotion. However, besides the advantages of the BEV fashion, there still exist some unavoidable disadvantages as follows: (1) The image-to-BEV perspective transformation requires dense feature sampling or rearrangement, which is complex and computationally expensive for low-cost edge devices deployment; (2) The maximum perception range is limited by the size of BEV feature map, making it difficult to trade off among perception range, efficiency and precision; (3) The height dimension is compressed in BEV feature with losing of texture cues. Thus BEV features are incompetent for some perception tasks such as sign detection.

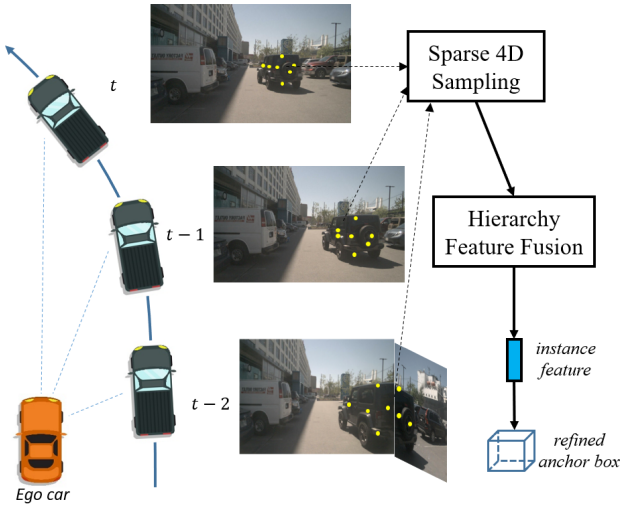


Figure 1. Overview of the Sparse4D. For each candidate anchor instance, we sparsely sampling multi-timestamp/view/scale features of multiple keypoints, then hierarchically fuse these feature as instance feature for precise anchor refinement.

Different from BEV based methods, the sparse based algorithms [5, 35, 41] do not require a dense perspective transformation module, but directly sample sparse feature for 3D anchors refinement, thus can alleviate above issues. Among them, the most representative sparse 3D detection method is DETR3D [41]. However, its model capacity is limited since DETR3D only sample feature of a single 3D reference

point for each anchor query. Recently, SRCN3D [35] utilizes RoI-Align [9] to sample multi-view feature, but is not efficient enough and cannot precisely align feature points from different views. Meanwhile, existing sparse 3D detection methods have not taken advantage of rich temporal context, and have a significant performance gap compared with state-of-the-art BEV based methods.

In this work, we devote our best effect to expand the limit of sparse based 3D detection. To address existing issues, we introduce a novel framework named Sparse4D, which utilizes multiple keypoints distributed in the region of 3D anchor box to sample feature. Compared with the single point manner [41] and the RoI-Align manner [35], our sampling manner has two main advantages: (1) can efficiently extract rich and complete context inside each anchor box; (2) can be simply extend to temporal dimension as 4D keypoints, then can effectively align temporal information. With 4D keypoints, as illustrated in Fig. 1, Sparse4D first performs multi-timestamp, multi-view and multi-scale for each keypoint. These sampled features then go through a hierarchical fusion module to generate high-quality instance feature for 3D box refinement. Further, to alleviate the ill-posed issue of camera-based 3D detection and improve the perceptual performance, we explicitly add an instance-level depth reweight module, where the instance feature is reweighted by depth confidence sampled from predicted depth distribution. This module is trained in a sparse way without additional Lidar point cloud surpervision.

In summary, our work have four main contributions:

- To the best of our knowledge, our proposed Sparse4D is the first sparse multi-view 3D detection algorithm with temporal context fusion, which can efficiently and effectively align spatial and temporal visual cues to achieve precise 3D detection.
- We propose a deformable 4D aggregation module that can flexibly complete the sampling and fusion of multi-dimensional (point, timestamp, view and scale) features.
- We introduce a depth reweight module to alleviate the ill-posed issue in image-based 3D perception system.
- On the challenging benchmark - nuScenes dataset, Sparse4D outperforms all existing sparse based algorithms and most BEV-based algorithms on 3D detection task, and also performs well on tracking task.

2. Related Work

2.1. Sparse Object Detection

Early object detection methods [7, 22, 33, 37, 38] used dense predictions as output, and then utilized non-maxima suppression (NMS) to process those dense predictions. DETR [3] introduces a new detection paradigm that utilizes set-based loss and transformer to directly predict sparse detection results. DETR performs cross attention between object-query and global image context, leading to heavy computation cost and difficulty in convergence. Due to the use of global cross attention, DETR cannot be regarded as a pure sparse method. Deformable DETR [51] then modifies DETR and proposes a local cross attention based on reference points, which accelerates the model convergence and reduced computational complexity. Sparse R-CNN [36] proposes another sparse detection framework based on the idea of region proposal. The network structure is extremely simple and effective, showing the feasibility and superiority of sparse detection. As the extension of 2D detection, many 3D detection methods have recently paid more attention to these sparse paradigms, such as MoNoDETR [46], DETR3D [41], Sparse R-CNN3D [35], SimMOD [48], etc.

2.2. Monocular 3D Object Detection

The monocular 3D detection algorithm takes a single image as input and outputs the 3D bounding box of the objects. Since the image does not contain depth information, this problem is ill-posed, and is more challenging compared with 2D detection. FCOS3D [39] and SMOKE [25] is extended based on a single-stage 2D detection network, using a fully convolution network to directly regress the depth of each object. [31, 40, 43] convert the 2D image into the 3D pseudo point cloud signal with monocular depth estimation results, and then use the LiDAR-based detection network to complete the 3D detection. OFT [34] and CaDDN [32] transform the dense 2D image feature into BEV space with the help of the view transformation module and then send the BEV feature to the detector to complete 3D object detection. The difference is that OFT uses the 3D to 2D inverse projection relationship to complete the feature space transformation, while CaDDN is based on the 2D to 3D projection, which is more like a pseudo-LiDAR method.

2.3. Multi-view 3D Object Detection

Dense algorithms are the main research direction of multi-view 3D detection, which use dense feature vectors for view transformation, feature fusion or box prediction. Currently, BEV-based methods are the main part of dense algorithms. BEVFormer [18] adopts deformable attention to complete the BEV feature generation and dense spatial-temporal feature fusion. BEVDet [11, 12] uses lift-splat operation [30] to achieve the view transformation. On the basis of BEVDet, BEVDepth [17] adds explicit depth supervision, which significantly improves the accuracy of the detection. BEVStereo [15] and SOLOFusion [29] introduce temporal stereo technology into 3D detection, further improving the depth estimation effect. PETR [23, 24] utilizes 3D position encoding and global cross attention for feature fusion, but the global cross attention is computationally expensive. Like vanilla DETR [3], PETR cannot be regarded as a purely sparse method. DETR3D [41] is a representa-

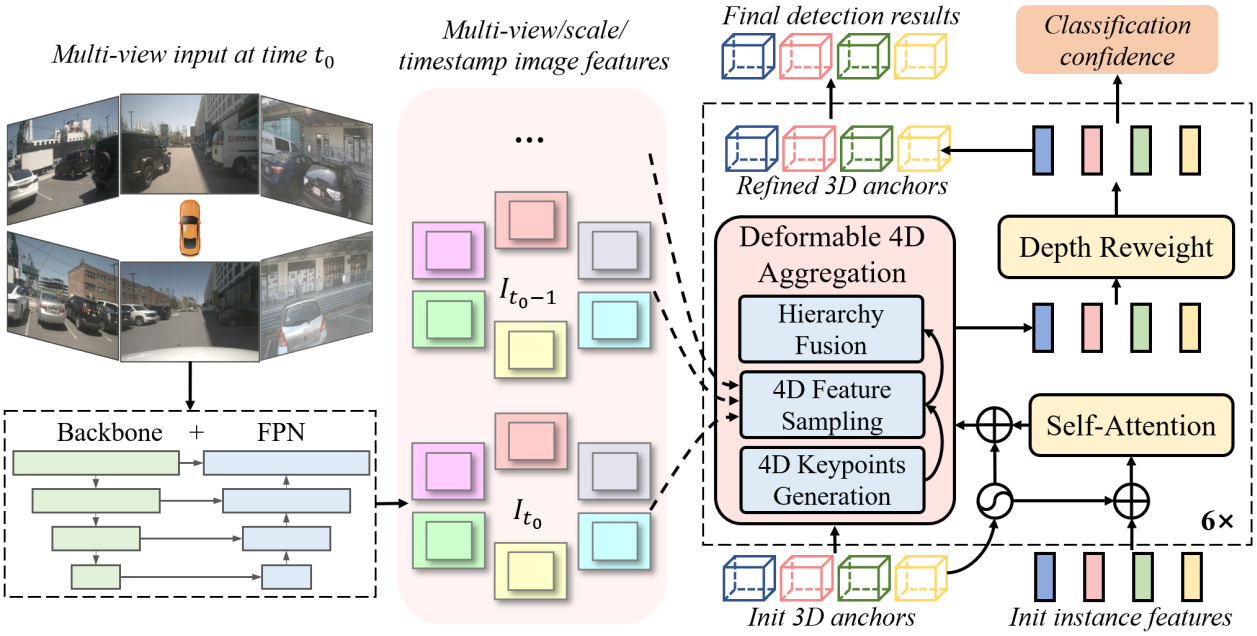


Figure 2. Overall architecture of Sparse4D. Taking multi-view images as input, we first extract multi-timestamp/view/scale feature maps with the image feature encoder. The decoder contains multiple refinement modules with independent parameters, which take image feature maps, instance features and 3D anchors as input, and continuously refines the 3D anchors to obtain accurate detection results.

tive work of sparse methods, which performs feature sampling and fusion based on sparse reference points. Graph DETR3D [5] follows DETR3D and introduces a graph network to achieve better spatial feature fusion, especially for multi-view overlapping regions.

3. Methodology

3.1. Overall Framework

As shown in Fig. 2, Sparse4D conforms to an encoder-decoder structure. The image encoder is used to extract image features with shared weights, which contains a backbone (*e.g.*, ResNet [10] and VoVNet [14]) and a neck (*e.g.*, FPN [20]). Given N view input images at time t, the image encoder extracts multi-view multi-scale feature maps as $I_t = \{I_{t,n,s} | 1 \leq s \leq S, 1 \leq n \leq N\}$. To exploit temporal context, we extract image feature of recent T frames as image feature queue $I = \{I_t\}_{t=t_s}^{t_0}$, where $t_s = t_0 - (T-1)$.

Then, the decoder predicts detection results in an iteratively refinement fashion, which contains a series of refinement modules and a classification head for predicting final classification confidences in the end. Each refinement module takes image feature queue I, 3D anchor boxes $B \in \mathbb{R}^{M \times 11}$ and corresponding instance features $F \in \mathbb{R}^{M \times C}$ as inputs, then outputs refined 3D boxes with updated instance features. Here, M is the number of anchors and C is the feature channel number. The format of an anchor is

$$\{x, y, z, \ln w, \ln h, \ln l, \sin yaw, \cos yaw, vx, vy, vz\},$$

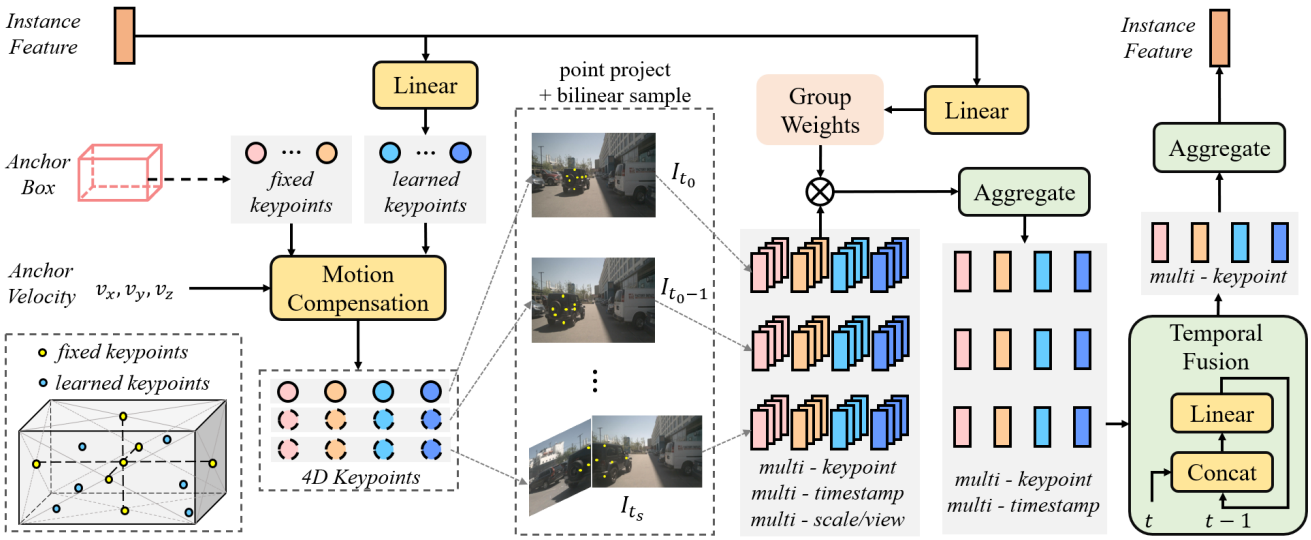
All 3D anchors are set in a unified 3D coordinate system(*e.g.* central LiDAR coordinate).

In each refinement module, we first adopt self-attention to realize the interaction between instances, with the embedding of anchor parameters added before and after. Then, we conduct deformable 4D aggregation (Sec. 3.2) to fuse multi-view, multi-scale, multi-timestamp and multi-keypoint features. Furthermore, we introduce a depth reweight module (Sec. 3.3) to alleviate the ill-posedness issue in image-based 3D detection. Finally, a regression head is used to refine the current anchor via predicting the offset between ground truth and the current anchor.

3.2. Deformable 4D Aggregation

The quality of instance features have a critical impact on the overall sparse perception system. To address this, as demonstrated on Fig. 3, we introduce the deformable 4D aggregation module to obtain high-quality instance features with sparse feature sampling and hierarchy feature fusion.

4D Keypoints Generation. For the m-th anchor instance, we assign K 4D keypoints as $P_m \in \mathbb{R}^{K \times T \times 3}$, which are composed of K_F fixed keypoints and K_L learnable keypoints. As shown in Fig. 3(a), at current timestamp t_0, we first put fixed keypoints P^F_{m,t_0} directly on the stereo center and the six faces center of the anchor box. Then, unlike fixed keypoints, the learnable keypoints vary with different instance features, which allow the neural network to find the most representative feature of each instance. Given



(a) 4D keypoint generation (b) Feature sampling (c) Hierarchy feature fusion

Figure 3. Detailed flowchart of the Deformable 4D Aggregation module. In this module, we extract high-quality instance feature in three steps: (a) for each anchor, generate multiple 4D keypoints; (b) project 4D keypoints to multi-timestamp/view/scale image feature maps and sample corresponding features and (c) hierarchically fuse keypoint features with predicted weights to generate fused instance feature.

instance feature F_m with anchor box embedding added, the learnable keypoints P^L_{m,t_0} are generated by the following formula through a sub-network Φ:

$$D_m = \mathbf{R}_{yaw} \cdot [\textbf{sigmoid}\left(\Phi(F_m)\right) - 0.5] \in \mathbb{R}^{K_L \times 3} \quad (1)$$

$$P^L_{m,t_0} = D_m \times [w_m, h_m, l_m] + [x_m, y_m, z_m] \quad (2)$$

where $\mathbf{R}_{yaw}$ denotes the rotation matrix of yaw.

Temporal features are crucial for 3D detection and can improve depth estimation accuracy. Therefore, after getting the 3D keypoints of the current frame, we extend them to 4D to prepare for temporal fusion. For a past timestamp t, we first build a constant velocity model to shift each 3D keypoints in the 3D coordinate system of the current frame.

$$P'_{m,t} = P_{m,t_0} - d_t \cdot (t_0 - t) \cdot [vx_m, vy_m, vz_m] \quad (3)$$

where d_t is the time interval between two adjacent frames. Then, we use the ego vehicle motion information to convert $P'_{m,t}$ to the coordinate system of the past t frame.

$$P_{m,t} = \mathbf{R}_{t_0 \to t} P'_{m,t} + \mathbf{T}_{t_0 \to t} \quad (4)$$

where $\mathbf{R}_{t_0 \to t}$ and $\mathbf{T}_{t_0 \to t}$ represent the rotation matrix and translation of the ego vehicle from current frame t_0 to frame t, respectively. In this way, we can finally construct 4D keypoints as $P_m = \{P_{m,t}\}^{t_0}_{t=t_s}$.

Sparse Sampling. Based on the above 4D keypoints P and the image feature maps queue F, sparse features with strong representation ability can be efficiently sampled. First, the 4D keypoints are projected onto the feature maps through the transformation matrix $\mathbf{T}^{\text{cam}}$.

$$P^{\text{img}}_{t,n} = \mathbf{T}^{\text{cam}}_n P_t, 1 \le n \le N \quad (5)$$

Then, we conduct multi-scale feature sampling for each view and each timestamp via bilinear interpolation:

$$f_{m,k,t,n,s} = \textbf{Bilinear}\left(I_{t,n,s}, P^{\text{img}}_{m,k,t,n}\right) \quad (6)$$

where the subscript m, k, t, n and s here indicate the indices of anchor, keypoint, timestamp, camera and feature map scale, respectively. So far, we have obtained multi-keypoints, timestamp, view and scale feature vectors $f_m \in \mathbb{R}^{K \times T \times N \times S \times C}$ for the m-th candidate detection anchor, where C is the number of feature channels.

Hierarchy Fusion. To generate high quality instance feature, we fuse the above features vectors f_m in a hierarchical manner. As shown in Fig. 3(c), for each keypoint, we first aggregate features in different view and scale with predicted weights and then conduct temporal fusion with sequence linear layers. Finally, for each anchor instance, we fuse multi-point features to generate instance feature.

Specifically, given instance feature F_m with anchor box embedding added, we first predict group weighting coefficients through a linear layer Ψ as:

$$W_m = \Psi\left(F_m\right) \in \mathbb{R}^{K \times N \times S \times G} \quad (7)$$

where G is the number of groups to divide features by channels. With this, we can aggregate channels of different groups with different weights, which is similar to group

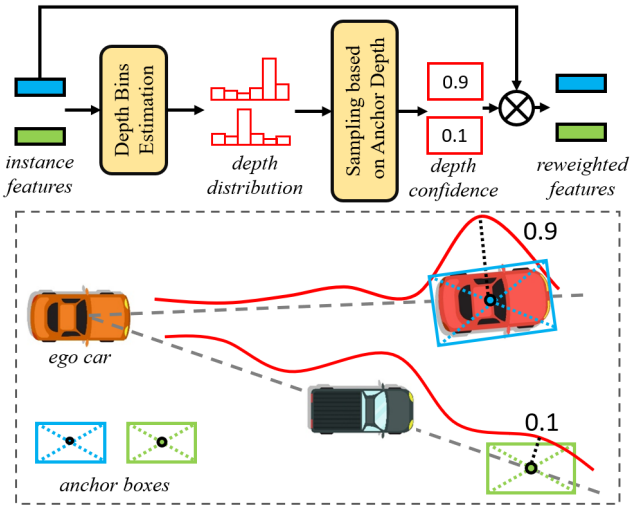


Figure 4. Illustration of depth reweight module. For each anchor instance, we estimate the depth distribution along the ray from the camera to the center of anchor box, then sample the depth confidence in the depth of anchor box. The sampled depth confidence is used to reweight instance feature.

convolution [13]. We sum the weighted feature vectors for each group along the scale and view dimensions, and then concatenate the groups to obtain the new features $f'_{m,k,t}$.

$$f'_{m,k,t,i} = \sum_{n=1}^{N}\sum_{s=1}^{S} W_{m,k,n,s,i} f_{m,k,t,n,s,i} \tag{8}$$

$$f'_{m,k,t} = \left[f'_{m,k,t,1}, f'_{m,k,t,2}, ..., f'_{m,k,t,G}\right] \tag{9}$$

The subscript i above is the index of the group, and $[,]$ represents the concatenate operation. Next, with concatenation operations and a linear layer Ψ_{temp}, the timestamp dimension of the features $f'_{m,k,t}$ will be fused in a sequential manner.

$$\begin{aligned} f''_{m,k,t_s} &= f'_{m,k,t_s} \\ f''_{m,k,t} &= \Psi_{temp}\left(\left[f'_{m,k,t}, f''_{m,k,t-1}\right]\right) \\ f''_{m,k} &= f''_{m,k,t_0} = \Psi_{temp}\left(\left[f'_{m,k,t_0}, f''_{m,k,t_0-1}\right]\right) \end{aligned} \tag{10}$$

The multi-keypoint features $f''_{m,k}$ after temporal fusion will be summed to complete the final feature aggregation and get the updated instance feature as:

$$F'_m = \sum_{k=1}^{K} f''_{m,k} \tag{11}$$

3.3. Depth Reweight Module

This 3D to 2D transformation (Eq. (5)) has a certain ambiguity, that is, different 3D points may correspond to the same 2D coordinates. For different 3D anchors, the same features may be sampled (see Fig. 4), which increases the difficulty of neural network fitting. To alleviate this problem, we incorporate an explicit depth estimation module Ψ_{depth}, which consists of multiple MLPs with residual connections. For each aggregated feature F'_m, we estimate a discrete depth distribution, and use the depth of center point of 3d anchor box to sample the corresponding confidence C_m, which will be used to reweight the instance feature.

$$C_m = \textbf{Bilinear}\left(\Psi_{depth}(F'_m), \sqrt{x_m^2 + y_m^2}\right) \tag{12}$$

$$F''_m = C_m \cdot F'_m \tag{13}$$

In this way, for those instances whose 3D center points are far from the ground truth in the depth direction, even if the 2D image coordinates are very close to the ground truth, the corresponding depth confidence tends to zero. Thus the corresponding instance feature F''_m is punished after reweighting also tend to $\mathbf{0}$. Incorporating an explicit depth estimation module can help the visual perception system to further improve the perception accuracy. Also, the depth estimation module can be designed and optimized as a separate part to facilitate model performance.

3.4. Training

We sample video clips with T frames to train the detector end to end. The time interval between consecutive frames is randomly sampled in $\{d_t, 2d_t\}$ ($d_t \approx 0.5$). Following DETR3D [41], the Hungarian algorithm is used to match each ground truth with one predicted value. The loss includes three parts: classification loss, bounding box regression loss and depth estimation loss:

$$L = \lambda_1 L_{cls} + \lambda_2 L_{box} + \lambda_3 L_{depth} \tag{14}$$

where λ_1, λ_2 and λ_3 are weight terms to balance the gradient. We adopt focal loss [21] for classification, L_1 loss for bounding box regression, and binary cross entropy loss for depth estimation. In depth reweight module, we directly use the depth of the labeled bounding box center as the ground truth to supervise per-instance depth. Since we only estimate per-instance depth rather than dense depth, the training process gets rid of the dependence on LiDAR data.

4. Experiment

4.1. Datasets and Metrics

We evaluate our method on the nuScenes benchmark. The nuScenes dataset [2] contains data for 1000 scenes, of

DRM	LKP	mAP↑	mAOE↓	NDS↑
✗	✗	0.432	0.408	0.533
✓	✗	0.431	0.381	0.537
✗	✓	0.432	0.379	0.537
✓	✓	**0.436**	**0.363**	**0.541**

Table 1. Ablation study of the influence of Depth Reweight Module (DRM) and Learnable Key Points (LKP) on detection effect.

H	Ego	Object	mAP↑	mATE↓	mAVE↓	NDS↑
0	-	-	0.322	0.747	0.890	0.401
3	✗	✗	0.334	0.759	0.682	0.424
3	✓	✗	**0.376**	**0.678**	0.398	0.488
3	✓	✓	0.373	0.687	**0.329**	**0.495**

Table 2. Ablation study of the influence of Motion Compensation setting. In this experiment, the input image size is set to 320×800 and learnable keypoints are removed. H is number of history frames.

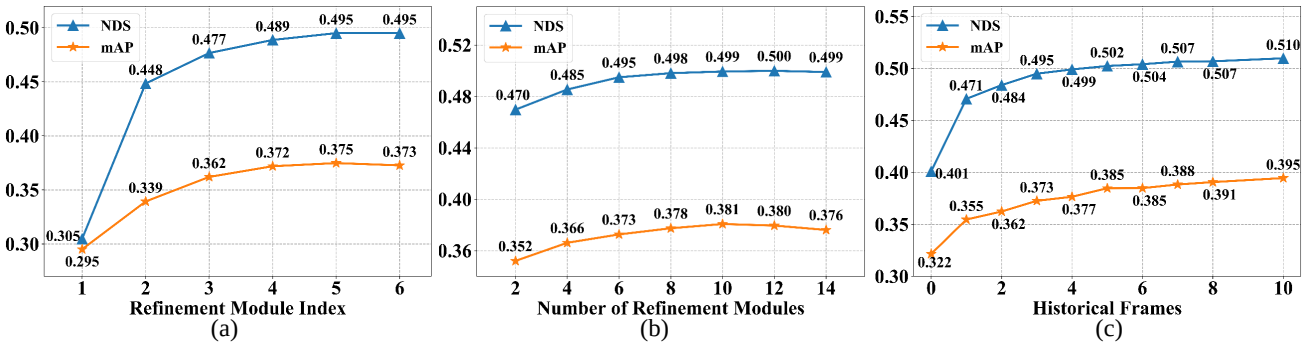


Figure 5. Ablation study of the influence of Refinement Modules and Historical Frame. In this experiment, the input image size is set to 320×800 and learnable keypoints are removed.

which 700, 150, and 150 scenes are used for training, validation, and testing, respectively. Each scene is a 20 second video clip at 2 frames per second. Each frame has image data from 6 cameras, and enough annotations such as the category, 3D bounding box, and ID of objects.

For the 3D detection task, evaluation metrics include mean Average Precision (mAP), mean Average Error of Translation (mATE), Scale (mASE), Orientation (mAOE), Velocity (mAVE), Attribute (mAAE) and nuScenes Detection Score (NDS), where NDS is a weighted average of other metrics. For the object tracking task, Average Multi-Object Tracking Accuracy (AMOTA), Average Multi-Object Tracking Precision (AMOTP) and Recall are the three main evaluation metrics. Please refer to [2, 42] for details.

4.2. Implementation Details

The initial $\{x, y, z\}$ parameters of the 3D anchors are obtained by performing K-Means clustering on the training set, and the other parameters are all initialized with fixed values $\{1, 1, 1, 0, 1, 0, 0, 0\}$. The instance feature uses random initialization. By default, the number of 3D anchors and instance features M is set to 900, the number of cascade refinement modules is 6, the number of feature map scales S from the neck is 4, the number of fixed keypoints K_F is 7, the number of learnable keypoints K_L is 6, the input image size is 640×1600, and the backbone is ResNet101.

Method	mAP↑	NDS↑	FLOPs(G)	Params(M)
DETR3D	0.346	0.425	996.6	53.3
LS-DETR*	0.348	0.397	1087.7	74.0
BEVFormer-S	0.375	0.448	1303.5	68.7
Sparse4D$_{T=1}$	0.382	0.451	1019.2	58.1
Sparse4D$_{T=4}$	**0.436**	**0.541**	1113.8	58.9

Table 3. Comparison of FLOPs and parameter amount of different methods, where all methods share same backbone with 854.3G Flops and 23.3M parameters. * LS-DETR here stands for Lift-Splat + Deformable DETR.

Sparse4D is trained with AdamW optimizer [27]. The initial learning rates of backbone and other parameters are 2e-5 and 2e-4, respectively. The decay strategy is cosine annealing [26]. The initial network parameters come from pre-trained FCOS3D [39]. For the experiments on the nuScenes test set, the network was trained for 48 epochs, and the rest of the experiments were only trained for 24 epochs unless otherwise specified. In order to save GPU memory, we detach the feature maps of all historical frames and the fusion features f'_t of a random part of historical frames during the training phase. CBGS [50] and test time augmentation were not used in all experiments.

Method	Temporal	Backbone	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓	NDS↑
FCOS3D	✗	ResNet101	0.299	0.785	**0.268**	0.557	1.396	**0.154**	0.373
DETR3D	✗	ResNet101	0.349	0.716	**0.268**	**0.379**	0.842	0.200	0.434
BEVDet	✗	ResNet101	0.357	**0.710**	0.270	0.490	0.885	0.224	0.421
BEVFormer-S	✗	ResNet101	0.375	0.725	0.272	0.391	**0.802**	0.200	0.448
Sparse4D$_{T=1}$	✗	ResNet101	**0.382**	**0.710**	0.279	0.411	0.806	0.196	**0.451**
BEVFormer	✓	ResNet101	0.416	0.673	0.274	0.372	0.394	0.198	0.517
BEVDet4D	✓	Swin-Base	0.396	0.619	**0.260**	0.361	0.399	0.189	0.515
PolarFormer-T	✓	ResNet101	0.432	0.648	0.270	**0.348**	0.409	0.201	0.528
BEVDepth	✓	ResNet101	0.412	**0.565**	0.266	0.358	0.331	0.190	0.535
Sparse4D$_{T=4}$	✓	ResNet101	0.436	0.633	0.279	0.363	0.317	**0.177**	0.541
Sparse4D$_{T=9(-6)}$	✓	ResNet101	**0.445**	0.613	0.279	0.378	**0.303**	0.180	0.547
Sparse4D$^{\dagger}_{T=9(-6)}$	✓	ResNet101	0.444	0.603	0.276	0.360	0.309	0.178	**0.550**

Table 4. Results of 3D object detection on nuScenes validation dataset.† indicates the number of training epochs is 48. The subscript $T = X(-Y)$ means that the number of frames used is X, and the feature f'_t of Y historical frames is randomly detached during training.

Method	Sparse	Backbone	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓	NDS↑
SRCN3D	✓	VoVNet-99	0.396	0.673	0.269	0.403	0.875	0.129	0.463
DETR3D	✓	VoVNet-99	0.412	0.641	0.255	0.394	0.845	0.133	0.479
Graph-DETR3D	✓	VoVNet-99	0.425	0.621	0.251	0.386	0.790	0.128	0.495
UVTR	✗	VoVNet-99	0.472	0.577	0.253	0.391	0.508	0.123	0.551
BEVDet4D	✗	Swin-Base	0.451	0.511	**0.241**	0.386	**0.301**	0.121	0.569
BEVFormer	✗	VoVNet-99	0.481	0.582	0.256	0.375	0.378	0.126	0.569
PETRv2	✗	VoVNet-99	0.490	0.561	0.243	**0.361**	0.343	**0.120**	0.582
BEVDistill	✗	ConvNeXt-Base	0.496	**0.475**	0.249	0.378	0.313	0.125	0.594
Sparse4D$^{\dagger}_{T=9(-6)}$	✓	VoVNet-99	**0.511**	0.533	0.263	0.369	0.317	0.124	**0.595**

Table 5. Results of 3D object detection on nuScenes test dataset. The superscript † and subscript $T = X(-Y)$ have the same meaning as in Tab. 4. Here the initialization parameters of VoVNet-99 are all pre-trained from DD3D [28] with extra data.

4.3. Ablation Studies and Analysis

Depth Reweight Module and Learnable Keypoints. By adding the depth reweight module or learnable keypoints, we compare and analyze the before-and-after changes in metrics on the nuScenes validation dataset, see Tab. 1. It can be seen that the addition of these two modules has a certain promotion effect on the model performance, and the impact on the metric NDS is similar, 0.33% and 0.35%, respectively. When these two structures are added together, all metrics will increase, among which mAP increases by 0.38% and NDS increases by 0.79%.

Motion Compensation. When generating 4D keypoints, we consider both the ego vehicle motion and the object motion. From Tab. 2, we can see that even without any motion compensation, after adding temporal information, the model performance still improved to a certain extent, in which mAVE increased by 20.8% and NDS increased by 2.3%. However, the overall perceptual performance of this model is still low. After adding ego motion compensation, the model effect is significantly improved, especially mAP and mAVE, which are increased by 4.2% and 28.4% respectively, and the comprehensive metric NDS is increased by 6.4%. On this basis, considering the motion of the object to be detected, the detection accuracy is not improved, but the error of the speed estimation will be reduced by 6.9%, thus increasing the NDS by about 0.7%.

Number of Refinements. The number of iterative refinements also has a significant impact on detection performance. In this regard, we designed two sets of experiments for analysis. In the first set of experiments, we train a model with 6 refinement modules and compute the detection metrics output by each refinement module. As can be seen from

Fig. 5(a), as the times of refinements increases, the overall metrics show an increasing trend, and the growth rate gradually decreases. Compared with the first one, the output accuracy of the second refinement module is significantly increased, but the detection effect between the fifth refinement module and the sixth module is not much different. In the second set of experiments, we train multiple models whose number of refinement modules is increased from 2 to 14. When the number of modules is 10, the NDS is the highest at 38.1%, as shown in Fig. 5(b).

Number of Historical Frames. We train and infer Sparse4D with varying numbers of historical frames and find that model performance continues to grow as the number of frames increases(Fig. 5(c)). Even if the number of frames increases to 10 (equivalent to 5 seconds in history), there is still a small increase compared to 8 frames. There may still be room for improvement in Sparse4D's performance if more frames are added. However, due to the limitations of our training device's memory (V100, 32G), it was not possible to try more frames.

FLOPs and Parameters. In this experiment, the input image size is set to 900x1600 and ResNet101 is used as backbone, and the experimental results are shown in Tab. 3. When $T = 1$, the FLOPs of our model is 1019.2G, and the parameter amount is 58.1M. Compared with DETR3D, the amount of calculation and parameter are only increased by 2.3% and 9.0%, respectively, and the mAP and NDS are increased by 3.6% and 2.6%. Compared with Lift-splat and BEVFormer-S, we have certain advantages in algorithm metrics, FLOPs and parameter amount. After adding 3 history frames, Sparse4D with temporal fusion only increases 9.3% FLOPs and 1.4% parameters, and achieves a very noticeable improvement, 5.4% mAP and 9.0% NDS.

4.4. Main Results

The comparison of the results on the nuScenes validation set is shown in Tab. 4. Among all the non-temporal models, Sparse4D gets the highest mAP and NDS. Compared with the baseline of the sparse methods, DERT3D, we improve mAP and NDS by 3.3% and 1.7%, respectively. Compared to the baseline of the BEV-based methods, BEVFormer, we lead by 0.7% on mAP and 0.3% on NDS. We also compared Sparse4D with other SOTA temporal algorithms, and still obtained the best NDS and mAP. When $T = 4$, Sparse4D outperforms BEVDepth by 2.4% on mAP and 0.6% on NDS. When T is increased from 4 to 9, the mAP and NDS of Sparse4D are improved by 0.9% and 0.6% respectively. Moreover, adding 24 epochs to training can further improve NDS by 0.3%.

We compare Sparse4D with other SOTA algorithms on the nuScenes test set (online leaderboard). As shown in the Tab. 5, with DD3D [28] pre-trained VoVNet-99, Sparse4D achieves 51.1% and 59.5% on mAP and NDS metrics, respectively, outperforming all non-BEV methods including PETRv2. Compared with baseline DETR3D, our method has achieved significant improvements. The mAP and NDS have increased by 9.9% and 11.6%, respectively, greatly improving the competitiveness of sparse methods. In addition, Sparse4D is also superior to the dense BEV based methods including UVTR [16], BEVdet [11], BEVFormer [18] and BEVDistill [1], especially in mAP, which is 1.5% higher than BEVDistill.

Method	AMOTA↑	AMOTP↓	Recall↑
MUTR3D [47]	0.270	1.494	0.411
PolarDETR [4]	0.273	1.185	0.404
SRCN3D	0.398	1.317	0.538
DAMEN-T	0.460	1.155	0.558
QTrack [44]	0.480	1.100	0.583
UVTR-GreedyTrack	**0.519**	1.125	0.599
Sparse4D	**0.519**	**1.078**	**0.633**

Table 6. Results of 3D multi-object tracking on nuScenes test set.

4.5. Extend to 3D Object Tracking

Based on the tracking-by-detection framework [6], Sparse4D is easily extended to a tracker. We use the instance features and bounding boxes output by the last refinement module to extract identity features, and use a lightweight sub-network to estimate the correlation matrix between historical trajectories and current objects. Then, the matching relationship between the historical trajectory and the current object will be obtained using the Hungarian matching algorithm. As shown in Tab. 6, Sparse4D obtains 0.519 AMOTA and 1.078 AMOTP on nuScenes test set, which is ahead of most learning-based methods.

5. Conclusion

In this work, we propose a new method, Sparse4D, which achieves feature-level fusion of multi-timestamp and multi-view through a deformable 4D aggregation module, and uses iterative refinement to achieve 3D box regression. Sparse4D can provide excellent perceptual performance, and it outperforms all existing sparse algorithms and most BEV-based algorithms on the nuScenes leaderboard.

We believe that Sparse4D still has a lot of room for improvement. For example, in the depth reweight module, multi-view stereo (MVS) [15, 45] technology can be added to obtain more accurate depth. Camera parameters can also be considered in the encoder to improve 3D generalization [8, 17]. Therefore, we hope that Sparse4D can become a new baseline for sparse 3D detection. In addition, the framework of Sparse4D can also be extended to other tasks, such as HD map construction, occupancy estimation, 3D reconstruction, etc.

References

[1] Anonymous. BEVDistill: Cross-modal BEV distillation for multi-view 3d object detection. In *Submitted to The Eleventh International Conference on Learning Representations*, 2023. under review. 8

[2] 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. 5, 6

[3] 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. 2

[4] Shaoyu Chen, Xinggang Wang, Tianheng Cheng, Qian Zhang, Chang Huang, and Wenyu Liu. Polar parametrization for vision-based surround-view 3d detection. *arXiv preprint arXiv:2206.10965*, 2022. 8

[5] Zehui Chen, Zhenyu Li, Shiquan Zhang, Liangji Fang, Qinhong Jiang, and Feng Zhao. Graph-detr3d: Rethinking overlapping regions for multi-view 3d object detection. *arXiv preprint arXiv:2204.11582*, 2022. 1, 3

[6] Gioele Ciaparrone, Francisco Luque Sánchez, Siham Tabik, Luigi Troiano, Roberto Tagliaferri, and Francisco Herrera. Deep learning in video multi-object tracking: A survey. *Neurocomputing*, 381:61–88, 2020. 8

[7] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6569–6578, 2019. 2

[8] Jose M Facil, Benjamin Ummenhofer, Huizhong Zhou, Luis Montesano, Thomas Brox, and Javier Civera. Camconvs: Camera-aware multi-scale convolutions for single-view depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11826–11835, 2019. 8

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 2

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[11] Junjie Huang and Guan Huang. Bevdet4d: Exploit temporal cues in multi-camera 3d object detection. *arXiv preprint arXiv:2203.17054*, 2022. 2, 8

[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] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 5

[14] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019. 3

[15] 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. *arXiv preprint arXiv:2209.10248*, 2022. 2, 8

[16] Yanwei Li, Yilun Chen, Xiaojuan Qi, Zeming Li, Jian Sun, and Jiaya Jia. Unifying voxel-based representation with transformer for 3d object detection. *arXiv preprint arXiv:2206.00630*, 2022. 8

[17] 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. *arXiv preprint arXiv:2206.10092*, 2022. 1, 2, 8

[18] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Qiao Yu, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. *arXiv preprint arXiv:2203.17270*, 2022. 1, 2, 8

[19] Tingting Liang, Hongwei Xie, Kaicheng Yu, Zhongyu Xia, Zhiwei Lin, Yongtao Wang, Tao Tang, Bing Wang, and Zhi Tang. Bevfusion: A simple and robust lidar-camera fusion framework. *arXiv preprint arXiv:2205.13790*, 2022. 1

[20] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 3

[21] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 5

[22] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 2

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

[24] 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. 2

[25] Zechen Liu, Zizhang Wu, and Roland Tóth. Smoke: Single-stage monocular 3d object detection via keypoint estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 996–997, 2020. 2

[26] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 6

[27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6

[28] 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. 7, 8

[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. *arXiv preprint arXiv:2210.02443*, 2022. 1, 2

[30] Jonah Philion and Sanja Fidler. 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. 2

[31] Rui Qian, Divyansh Garg, Yan Wang, Yurong You, Serge Belongie, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. End-to-end pseudo-lidar for image-based 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5881–5890, 2020. 2

[32] Cody Reading, Ali Harakeh, Julia Chae, and Steven L Waslander. Categorical depth distribution network for monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8555–8564, 2021. 2

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 2

[34] Thomas Roddick, Alex Kendall, and Roberto Cipolla. Orthographic feature transform for monocular 3d object detection. *arXiv preprint arXiv:1811.08188*, 2018. 2

[35] Yining Shi, Jingyan Shen, Yifan Sun, Yunlong Wang, Jiaxin Li, Shiqi Sun, Kun Jiang, and Diange Yang. Srcn3d: Sparse r-cnn 3d surround-view camera object detection and tracking for autonomous driving. *arXiv preprint arXiv:2206.14451*, 2022. 1, 2

[36] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14454–14463, 2021. 2

[37] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020. 2

[38] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019. 2

[39] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. Fcos3d: Fully convolutional one-stage monocular 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 913–922, 2021. 2, 6

[40] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8445–8453, 2019. 2

[41] 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 *Conference on Robot Learning*, pages 180–191. PMLR, 2022. 1, 2, 5

[42] Xinshuo Weng and Kris Kitani. A baseline for 3d multi-object tracking. *arXiv preprint arXiv:1907.03961*, 1(2):6, 2019. 6

[43] Xinshuo Weng and Kris Kitani. Monocular 3d object detection with pseudo-lidar point cloud. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2

[44] Jinrong Yang, En Yu, Zeming Li, Xiaoping Li, and Wenbing Tao. Quality matters: Embracing quality clues for robust 3d multi-object tracking. *arXiv preprint arXiv:2208.10976*, 2022. 8

[45] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European conference on computer vision (ECCV)*, pages 767–783, 2018. 8

[46] Renrui Zhang, Han Qiu, Tai Wang, Xuanzhuo Xu, Ziyu Guo, Yu Qiao, Peng Gao, and Hongsheng Li. Monodetr: Depth-aware transformer for monocular 3d object detection. *arXiv preprint arXiv:2203.13310*, 2022. 2

[47] Tianyuan Zhang, Xuanyao Chen, Yue Wang, Yilun Wang, and Hang Zhao. Mutr3d: A multi-camera tracking framework via 3d-to-2d queries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4537–4546, 2022. 8

[48] Yunpeng Zhang, Wenzhao Zheng, Zheng Zhu, Guan Huang, Jie Zhou, and Jiwen Lu. A simple baseline for multi-camera 3d object detection. *arXiv preprint arXiv:2208.10035*, 2022. 2

[49] Yunpeng Zhang, Zheng Zhu, Wenzhao Zheng, Junjie Huang, Guan Huang, Jie Zhou, and Jiwen Lu. Beverse: Unified perception and prediction in birds-eye-view for vision-centric autonomous driving. *arXiv preprint arXiv:2205.09743*, 2022. 1

[50] 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. 6

[51] 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