DETR3D: 3D Object Detection from Multi-view Images via 3D-to-2D Queries

Anonymous Author(s)
Affiliation
Address
email

Abstract: We introduce a framework for multi-camera 3D object detection. In contrast to existing works, which estimate 3D bounding boxes directly from monocular images or use depth prediction networks to generate input for 3D object detection from 2D information, our method manipulates predictions directly in 3D space. Our architecture extracts 2D features from multiple camera images and then uses a sparse set of 3D object queries to index into these 2D features, linking 3D positions to multi-view images using camera transformation matrices. Finally, our model makes a bounding box prediction per object query, using a set-to-set loss to measure the discrepancy between the ground-truth and the prediction. This top-down approach outperforms its bottom-up counterpart in which object bounding box prediction follows per-pixel depth estimation, since it does not suffer from the compounding error introduced by a depth prediction model. Moreover, our method does not require post-processing such as non-maximum suppression, dramatically improving inference speed. We achieve state-of-the-art performance on the nuScenes autonomous driving benchmark.

1 Introduction

3D object detection from visual information is a long-standing challenge for low-cost autonomous driving systems. While object detection from point clouds collected using modalities like LiDAR benefits from information about the 3D structure of visible objects, the camera-based setting is even more ill-posed, since we must generate 3D bounding box predictions solely from the 2D information contained in RGB images.

Existing methods [1, 2] typically build their detection pipelines purely from 2D computations. That is, they predict 3D information like object pose and velocity using an object detection pipeline designed for 2D tasks (e.g., CenterNet [1], FCOS [3]), without considering 3D scene structure or sensor configuration. These methods require several post-processing steps to fuse predictions across cameras and to remove redundant boxes, yielding a steep trade-off between efficiency and effectiveness. As an alternative to these 2D-based methods, some methods incorporate more 3D computations into our object detection pipeline by applying a 3D reconstruction method like [4, 5, 6] to create a pseudo-LiDAR or range input of the scene from camera images. Then, they could apply 3D object detection methods to this data as if it were collected directly from a 3D sensor. This strategy, however, is subject to compounding errors [7]: poorly-estimated depth values have a strongly negative effect on the performance of 3D object detection, which also can exhibit errors of its own.

In this paper, we propose a more graceful transition between 2D observations and 3D predictions for autonomous driving, which does not rely on a module for dense depth prediction. Our framework, termed DETR3D (Multi-View 3D Detection), addresses this problem in a top-down fashion. We link 2D feature extraction and 3D object prediction via geometric back-projection with camera transformation matrices. Our method starts from a sparse set of object priors, shared across the dataset and learned end-to-end. To gather scene-specific information, we back-project a set of reference points decoded from these object priors to each camera and fetch the corresponding image features extracted by a ResNet backbone [8]. The features collected from the image features of the reference points then interact with each other through a multi-head self-attention layer [9]. After a series of

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

self-attention layers, we read off bounding box parameters from every layer and use a set-to-set loss inspired by DETR [10] to evaluate performance.

Our architecture does not perform point cloud reconstruction or explicit depth prediction from images, making it robust to errors in depth estimation. Moreover, our method does not require any post-processing, such as non-maximum suppression (NMS), improving efficiency and reducing reliance on hand-designed methods for cleaning its output. On the nuScenes dataset, our method (without NMS) is comparable with prior art (with NMS). In the camera overlap regions, our method significantly outperforms others.

Contributions. We summarize our key contributions as follows:

- We present a streamlined 3D object detection model from RGB images. Different from existing works that combine object predictions from the different camera views in a final stage, our method fuses information from all the camera views in each layer of computation. To the best of our knowledge, this is the first attempt to cast multi-camera detection as 3D set-to-set prediction.
- We introduce a module that connects 2D feature extraction and 3D bounding box prediction via backward geometric projection. It does not suffer from inaccurate depth predictions from a secondary network, and seamlessly uses information from multiple cameras by back-projecting 3D information onto all available frames.
- Similarly to Object DGCNN [11], our method does not require post-processing such as per-image or global NMS, and it is on par with existing NMS-based methods. In the camera overlap regions, our method outperforms others by a substantial margin.
- We will release our code to promote reproducibility and future research.

2 Related Work

2D object detection. RCNN [12] pioneered object detection using deep learning. It feeds a set of pre-selected object proposals into a convolutional neural network (CNN) and predicts bounding box parameters accordingly. Although this method exhibits surprising performance, it is an order of magnitude slower than others because it performs a ConvNet forward pass for each object proposal. To fix this issue, Fast RCNN [13] introduces a shared learnable CNN to process the entire image at a single forward pass. To further improve performance and speed, Faster RCNN [13] includes a region proposal network (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. Mask RCNN [14] incorporates a mask prediction branch to enable instance segmentation in parallel. These methods typically involve multi-stage refinements and can be slow in practice. Different from these multi-stage methods, SSD [15] and YOLO [16] perform dense predictions in a single shot. Although they are significantly faster than the alternatives above, they still rely on NMS to remove redundant box predictions. These methods predict bounding boxes w.r.t. pre-defined anchors. CenterNet [1] and FCOS [3] change the paradigm by shifting from per-anchor prediction to per-pixel prediction, significantly simplifying the common object detection pipeline.

Set-based object detection. DETR [10] casts object detection as a set-to-set problem. It employs a Transformer [9] to capture feature and object interactions. DETR learns to assign predictions to a set of ground-truth boxes; thus, it does not require post-processing to filter out redundant boxes. One critical drawback of DETR, however, is that it requires a significant amount of training time. Deformable DETR [17] analyzes DETR's slow convergence and proposes a deformable self-attention module to localize features and accelerate training. Concurrently, [18] attributes the slow convergence of DETR to the set-based loss and the Transformer cross attention mechanism. They propose two variants, TSP-FCOS and TSP-RCNN, to overcome these problematic aspects. SparseR-CNN [19] incorporates set prediction into a RCNN-style pipeline; it outperforms multi-stage object detection without NMS. OneNet [20] studies an interesting phenomenon: dense-based object detectors can be made NMS-free after they are equipped with a minimum-cost set loss. For 3D domains, Object DGCNN [11] studies 3D object detection from point clouds. It models 3D object detection as message passing on a dynamic graph, generalizing the DGCNN framework to predict a set of objects. Similar to DETR, Object DGCNN is also NMS-free.

Monocular 3D object detection. An early method for 3D detection from RGB images is Mono3D [21], which uses semantic and shape cues to select from a collection of 3D proposals, using scene constraints and additional priors at training time. [22] uses the birds-eye-view (BEV)

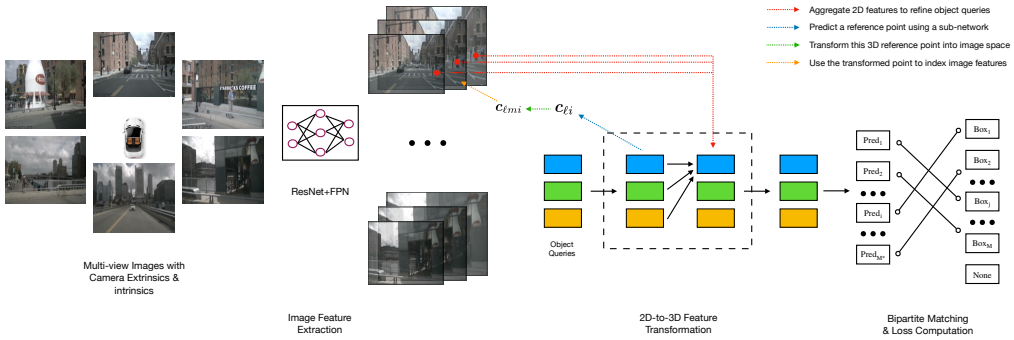

Figure 1: Overview of our method. The inputs to the model are a set of multi-view images, which are encoded by a ResNet and a FPN. Then, our model operates on a set of sparse object queries in which each query is decoded to a 3D reference point. 2D features are transformed to refine the object queries by projecting the 3D reference point into the image space. Our model makes per-query predictions and uses a set-to-set loss.

for monocular 3D detection, and [23] leverages 2D detections for 3D bounding box regression via the minimization of 2D-3D projection error. The use of 2D detectors as a starting point for 3D computation recently has become a standard approach [24, 25]. Other works also explore advances in differentiable rendering [26] or 3D keypoint detection [27, 28, 1] to enable state-of-the-art 3D object detection performance. All these methods operate in a monocular setting, and extensions to multiple cameras are done by independently processing each frame before merging the outputs in a post-processing stage.

3 Multi-view 3D Object Detection

3.1 Overview

Our architecture inputs RGB images collected from a set of cameras whose projection matrices (the combination of intrinsics and relative extrinsics) are known, and it outputs a set of 3D bounding box parameters for the objects in the scene. In contrast to past approaches, we build our architecture based on a few high-level desiderata:

- We incorporate 3D information into intermediate computations within our architecture, rather than performing purely 2D computations in the image plane.
- We do not estimate dense 3D scene geometry, avoiding associated reconstruction errors.
- We avoid post-processing steps such as NMS.

We address these desiderata using a new set prediction module, which links 2D feature extraction and 3D box prediction by alternating between 2D and 3D computations. Our model contains three critical components, illustrated in Figure 1. First, following common practice in 2D vision, it extracts features from the camera images using a shared ResNet [8] backbone. Optionally, these features are enhanced by a feature pyramid network (FPN) [29] (§3.2). Second, a detection head (§3.3)—our main contribution—links the computed 2D features to a set of 3D bounding box predictions in a geometry-aware manner (§3.3). Each layer of the detection head starts from a sparse set of object queries, which are learned from the data. Each object query encodes a 3D location, which is projected to the camera planes and used to collect image features via bilinear interpolation. Similarly to DETR [10], we then use multi-head attention [9] to refine the object queries by incorporating object interactions. This layer is repeated multiple times, alternating between feature sampling and object query refinement. Finally, we evaluate a set-to-set loss [30, 10] to train the network (§3.4).

3.2 Feature Learning

Our model starts with a set of images $\mathcal{I} = \{\mathbf{im}_1, \ldots, \mathbf{im}_K\} \subset \mathbb{R}^{\mathrm{H_{im}} \times \mathrm{W_{im}} \times 3}$ (captured by surrounding cameras), camera matrices $\mathcal{T} = \{T_1, \ldots, T_K\} \subset \mathbb{R}^{3\times 4}$, ground-truth bounding boxes $\mathcal{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_j, \ldots, \boldsymbol{b}_M\} \subset \mathbb{R}^9$, and categorical labels $\mathcal{C} = \{c_1, \ldots, c_j, \ldots, c_M\} \subset \mathbb{Z}$. Each $\boldsymbol{b}_j$ contains position, size, heading angle, and velocity in the birds-eye view (BEV); our model aims to

predict these boxes and their labels from the these images. We *do not* use point clouds, which are usually captured by high-end LiDAR.

These images are encoded with a ResNet [8] and a FPN [29] into four sets of features $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4$. Each set $\mathcal{F}_k = \{\boldsymbol{f}_{k1}, \ldots, \boldsymbol{f}_{k6}\} \subset \mathbb{R}^{H \times W \times C}$ corresponds to a level of features of the 6 images. These multi-scale features provide rich information to recognize objects of different sizes. Next, we detail our approach to transform these 2D features into 3D using a novel set prediction module.

3.3 Detection Head

Existing methods for detecting objects from camera input typically employ a bottom-up approach, which predicts a dense set of bounding boxes per image, filters redundant boxes between the images, and aggregates predictions across cameras in a post-processing step. This paradigm has two crucial drawbacks: dense bounding box prediction requires accurate depth perception, which itself is a challenging problem; and NMS-based redundancy removal and aggregation are non-parallelizable operations that introduce significant inference overhead. We address these issues using a top-down object detection head described below.

Analogously to [11, 17], DETR3D is *iterative*; it uses L layers with set-based computations to produce bounding box estimates from 2D feature maps. Each layer includes the following steps:

1. predict a set of bounding box centers associated with object queries;
2. project these centers into all the feature maps using the camera transformation matrices;
3. sample features via bilinear interpolation and incorporate them into object queries; and
4. describe object interactions using multi-head attention.

Motivated by DETR [10], each layer $\ell \in \{0, \ldots, L-1\}$ operates on a set of *object queries* $\mathcal{Q}_\ell = \{\boldsymbol{q}_{\ell 1}, \ldots, \boldsymbol{q}_{\ell M^*}\} \subset \mathbb{R}^C$, producing a new set $\mathcal{Q}_{\ell+1}$. A reference point $\boldsymbol{c}_{\ell i} \in \mathbb{R}^3$ is decoded from a object query $\boldsymbol{q}_{\ell i}$ as follows:

$$\boldsymbol{c}_{\ell i} = \Phi^{\text{ref}}(\boldsymbol{q}_{\ell i}), \tag{1}$$

where Φ^{ref} is a neural network. $\boldsymbol{c}_{\ell i}$ can be thought of a hypothesis for the center of the i-th box. Next, we acquire image features corresponding to $\boldsymbol{c}_{\ell i}$ to refine and predict the final bounding box. Then, $\boldsymbol{c}_{\ell i}$ (or more accurately its homogeneous counterpart $\boldsymbol{c}^*_{\ell i}$) is projected into each one of the images using the camera transformation matrices:

$$\boldsymbol{c}^*_{\ell i} = \boldsymbol{c}_{\ell i} \oplus 1 \qquad\qquad \boldsymbol{c}_{\ell m i} = T_m \boldsymbol{c}^*_{\ell i}, \tag{2}$$

where $\oplus$ denotes concatenation, and $\boldsymbol{c}_{\ell m i}$ is the projection of the reference point onto the m-th camera. To remove the effects of the feature map size and gather features across different levels, we normalize $\boldsymbol{c}_{\ell m i}$ to $[-1, 1]$. Next, the images features are collected by

$$\boldsymbol{f}_{\ell k m i} = f^{\text{bilinear}}(\mathcal{F}_{km}, \boldsymbol{c}_{\ell m i}), \tag{3}$$

where $\boldsymbol{f}_{\ell k m i}$ is the feature for i-th point from k-th level of m-th camera at ℓ-th layer.

A given reference point is not necessarily visible in all the camera images, so we need some heuristics to filter invalid points. To that end, we define a binary value $\sigma_{\ell k m i}$, which is determined based on whether a reference point is projected outside an image plane. The final feature $\boldsymbol{f}_{\ell i}$ and object query in next layer $\boldsymbol{q}_{(\ell+1)i}$ are given by

$$\boldsymbol{f}_{\ell i} = \frac{1}{\sum_k \sum_m \sigma_{\ell k m i} + \epsilon} \sum_k \sum_m \boldsymbol{f}_{\ell k m i} \sigma_{\ell k m i} \qquad \text{and} \qquad \boldsymbol{q}_{(\ell+1)i} = \boldsymbol{f}_{\ell i} + \boldsymbol{q}_{\ell i}, \tag{4}$$

where ϵ is a small number to avoid division by zero. Finally, for each object query $\boldsymbol{q}_{\ell i}$, we predict a bounding box $\hat{\boldsymbol{b}}_{\ell i}$ and its categorical label $\hat{c}_{\ell i}$ with two neural networks Φ^{reg}_ℓ and Φ^{cls}_ℓ:

$$\hat{\boldsymbol{b}}_{\ell i} = \Phi^{\text{reg}}_\ell(\boldsymbol{q}_{\ell i}) \qquad \text{and} \qquad \hat{c}_{\ell i} = \Phi^{\text{cls}}_\ell(\boldsymbol{q}_{\ell i}). \tag{5}$$

We compute the loss for the predictions $\hat{\mathcal{B}}_\ell = \{\hat{\boldsymbol{b}}_{\ell 1}, \ldots, \hat{\boldsymbol{b}}_{\ell j}, \ldots, \hat{\boldsymbol{b}}_{\ell M^*}\} \subset \mathbb{R}^9$ and $\hat{\mathcal{C}}_\ell = \{\hat{c}_{\ell 1}, \ldots, \hat{c}_{\ell j}, \ldots, \hat{c}_{\ell M}\} \subset \mathbb{Z}$ from every layer during training. During inference, we only use the outputs from the last layer.

3.4 Loss

Following [30, 10], we use a set-to-set loss to measure the discrepancy between the prediction set $(\hat{\mathcal{B}}_\ell, \hat{\mathcal{C}}_\ell)$ and the ground-truth set $(\mathcal{B}, \mathcal{C})$. This loss consists of two parts: a focal loss [31] for the class labels and a L^1 loss for the bounding box parameters. For notational convenience, we drop the ℓ subscript in $\hat{\mathcal{B}}_\ell$ and $\hat{\mathcal{C}}_\ell$. The number of ground-truth boxes M is typically smaller than the number of predictions M^*, so we pad the set of ground-truth boxes with $\varnothing$s (no object) up to M^* for ease of computation. We establish a correspondence between the ground-truth and the prediction via a bipartite matching problem: $\sigma^* = \arg\min_{\sigma \in \mathcal{P}} \sum_{j=1}^{M} -1_{\{c_j \neq \varnothing\}} \hat{p}_{\sigma(j)}(c_j) + 1_{\{c_j = \varnothing\}} \mathcal{L}_{\text{box}}(\boldsymbol{b}_j, \hat{\boldsymbol{b}}_{\sigma(j)})$, where $\mathcal{P}$ denotes the set of permutations, $\hat{p}_{\sigma(j)}(c_j)$ is the probability of class c_j for the prediction with index $\sigma(j)$, and $\mathcal{L}_{\text{box}}$ is the L_1 loss for bounding box parameters. We use the Hungarian algorithm [32] to solve this assignment problem, as in [30, 10], yielding the set-to-set loss $\mathcal{L}_{\text{sup}} = \sum_{j=1}^{N} -\log \hat{p}_{\sigma^*(j)}(c_j) + 1_{\{c_j = \varnothing\}} \mathcal{L}_{\text{box}}(\boldsymbol{b}_j, \hat{\boldsymbol{b}}_{\sigma^*(j)})$.

4 Experiments

We present our results as follows: first, we detail the dataset, metrics, and implementation in §4.1; then we compare our method to existing works in §4.2; we benchmark the performance of different models in camera overlap regions in §4.3; we compare to a forward prediction model in §4.4; and we provide additional analysis and ablations in §4.5.

4.1 Implementation Details

Dataset. We test our method on the nuScenes dataset [33]. nuScenes consists of 1,000 sequences; each sequence is roughly 20s long, with a sampling rate of 20 frames/second. Each sample contains images from 6 cameras `[front_left, front, front_right, back_left, back, back_right]`. Camera parameters including intrinsics and extrinsics are available. nuScenes provides annotations every 0.5s; in total there are 28k, 6k, and 6k annotated samples for training, validation, and testing, respectively. 10 from the total 23 classes are available to compute the metrics.

Metrics. We follow the official evaluation protocol provided by nuScenes. We evaluate average translation error (ATE), average scale error (ASE), average orientation error (AOE), average velocity error (AVE), and average attribute error (AAE). These metrics are true positive metrics (TP metrics) and computed in the physical unit. In addition, we measure mean average precision (mAP). To capture all aspects of the detection task, a consolidated scalar metric–the nuScenes Detection Score (NDS) [33]–is defined as $\text{NDS} = \frac{1}{10}[5\text{mAP} + \sum_{\text{mTP} \in \mathbb{TP}}(1 - \min(1, \text{mTP}))]$.

Model. Our model consists of a ResNet [8] feature extractor, a FPN, and a DETR3D detection head. We use ResNet101 with deformable convolutions [34] in the 3rd stage and 4th stage. The FPN [29] takes features output by the ResNet and produces 4 feature maps whose sizes are $1/8$, $1/16$, $1/32$, and $1/64$ of the input image sizes. The DETR3D detection head consists of 6 layers, where each layer is a combination of a feature refinement step and a multi-head attention layer. The hidden dimension of the DETR3D detection head is 256. Finally, two sub-networks predict bounding box parameters and a class label per object query; each sub-network consists of two fully connected layers with hidden dimensions 256. We use LayerNorm [35] in the detection head.

Training & inference. We use AdamW [36] to train the whole pipeline. The weight decay is 10^{-4}. We use an initial learning rate 10^{-4}, which is decreased to 10^{-5} and 10^{-6} at 8th and 11th epochs. The model is trained for 12 epochs in total on 8 RTX 3090 GPUs and the per-GPU batch size is 1. The training procedure takes roughly 18 hours. We do not use any post-processing such as NMS during inference. For evaluation, we use the nuScenes evalutation toolkit.

4.2 Comparison to Existing Works

We compare to previous state-of-the-art methods CenterNet [1] and FCOS3D [2]. CenterNet is an anchor-free 2D detection method that makes dense predictions in a high resolution feature map. FCOS3D employs a FCOS [3] pipeline to make per-pixel predictions. These methods both turn 3D object detection into a 2D problem, and in doing so ignore scene geometry and sensor configuration. To perform multi-view object detection, these methods have to process each image independently,

Table 1: Comparisons to recent works. Our method is robust to the usage of NMS. ∗: CenterNet uses a customized backbone DLA [37]. ‡: this model is trained with depth weight 1.0 and initialized from a FCOS3D checkpoint; the checkpoint is trained on the same dataset with depth weight 0.2. §: with test-time augmentation. ¶: with test-time augmentation, more epochs, and model ensemble. For details, see [2]. †: our model is also initialized from a FCOS3D backbone; the detection head is initialized randomly.

Method	NDS ↑	mAP ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓	NMS
CenterNet ∗	0.328	0.306	0.716	0.264	0.609	1.426	0.658	✓
FCOS3D	0.373	0.299	0.785	0.268	0.557	1.396	0.154	✓
FCOS3D ‡	0.393	0.321	0.746	0.265	0.503	1.351	0.160	✓
FCOS3D §	0.402	0.326	0.743	0.259	0.441	1.341	0.163	✓
FCOS3D ¶	0.415	0.343	0.725	0.263	0.422	1.292	0.153	✓
DETR3D (Ours)	0.374	0.303	0.860	0.278	0.437	0.967	0.235	-
DETR3D (Ours) †	0.425	0.346	0.773	0.268	0.383	0.842	0.216	-

Table 2: Comparisons in Overlap Region. ‡: this model is trained with depth weight 1.0 and initialized from a FCOS3D checkpoint; the checkpoint is trained on the same dataset with depth weight 0.2. For details, see [2]. †: our model is also initialized from a FCOS3D backbone; the detection head is initialized randomly.

Method	NDS ↑	mAP ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓	NMS
FCOS3D	0.317	0.213	0.841	0.276	0.604	1.122	0.173	✓
FCOS3D‡	0.329	0.229	0.816	0.272	0.571	1.084	0.195	✓
DETR3D (Ours)	0.356	0.231	0.825	0.280	0.400	0.863	0.223	-
DETR3D (Ours) †	0.384	0.268	0.807	0.273	0.453	0.788	0.184	-

and use both per-image and global NMS to remove redundant boxes in each view and in the overlap regions respectively. As shown in Table 1, our method is on a par with these methods even though we do not use any post-processing. Our method performs worse than FCOS3D in terms of mATE. We suspect this is because FCOS3D directly predicts bounding box depth, which leads to strong supervision on object translation. Also, FCOS3D uses disentangled heads for different bounding box parameters, which can increase performance.

4.3 Comparison in Overlap Regions

A great challenge lies in the overlap regions where objects are more likely to be cut off. Our method considers all cameras simultaneously, while FCOS3D predicts bounding boxes per camera individually. To further demonstrate the advantages of fused inference, we calculate the metrics for boxes falling into the camera overlaps. To compute the metrics, we select boxes whose 3D center is visible to multiple cameras. On the validation set, there are 18,147 such boxes, 9.7% of the total. Table 2 shows the results; our method outperforms FCOS3D remarkably in terms of NDS scores in this setting. This confirms that our integrated prediction approach is more effective.

4.4 Comparison to pseudo-LiDAR Methods

Another way to perform 3D object detection is by generating pseudo-LiDAR point clouds from multi-view images using a depth prediction model. On the nuScenes dataset, there are no publicly available pseudo-LiDAR works for us to make a direct comparison. Hence, we implement a baseline ourselves to verify that our approach is more effective than explicit depth prediction. We use a pre-trained PackNet [6] network to predict dense depth maps from all six cameras and then convert these depth maps into point clouds using the camera transformations. We also experimented with a self-supervised PackNet model with velocity supervision (as in the original paper), but we found that ground-truth depth supervision yielded more realistic point clouds and therefore used a supervised model as baseline. For 3D detection, we employ the recently-proposed CenterPoint architecture [38]. Conceptually, this pipeline is a variant of pseudo-LiDAR [39]. Table 3 shows the results; we conclude that this pseudo-LiDAR method underperforms ours significantly even when depth estimates are generated by a state-of-the-art model. One possible explanation is that pseudo-LiDAR object detectors suffer from compounding errors introduced by inaccurate depth prediction, that in turn is known to overfit to training data and generalizes poorly to other distributions [7].

Table 3: Comparisons to pseudo-LiDAR Methods.

Method	NDS ↑	mAP ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓	NMS
pseudo-LiDAR	0.1600	0.0480	-	-	-	-	-	✓
MVOD (Ours)	0.374	0.303	0.860	0.278	0.437	0.967	0.235	-

Table 4: Evaluation on detection results from different layers.

Layer ↑	NDS ↑	mAP ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓
0	0.380	0.302	0.855	0.280	0.435	0.910	0.231
1	0.410	0.335	0.791	0.275	0.408	0.887	0.217
2	0.420	0.343	0.782	0.271	0.395	0.851	0.214
3	0.420	0.346	0.778	0.268	0.390	0.874	0.218
4	0.424	0.346	0.777	0.268	0.389	0.855	0.217
5	0.425	0.346	0.773	0.268	0.383	0.842	0.216

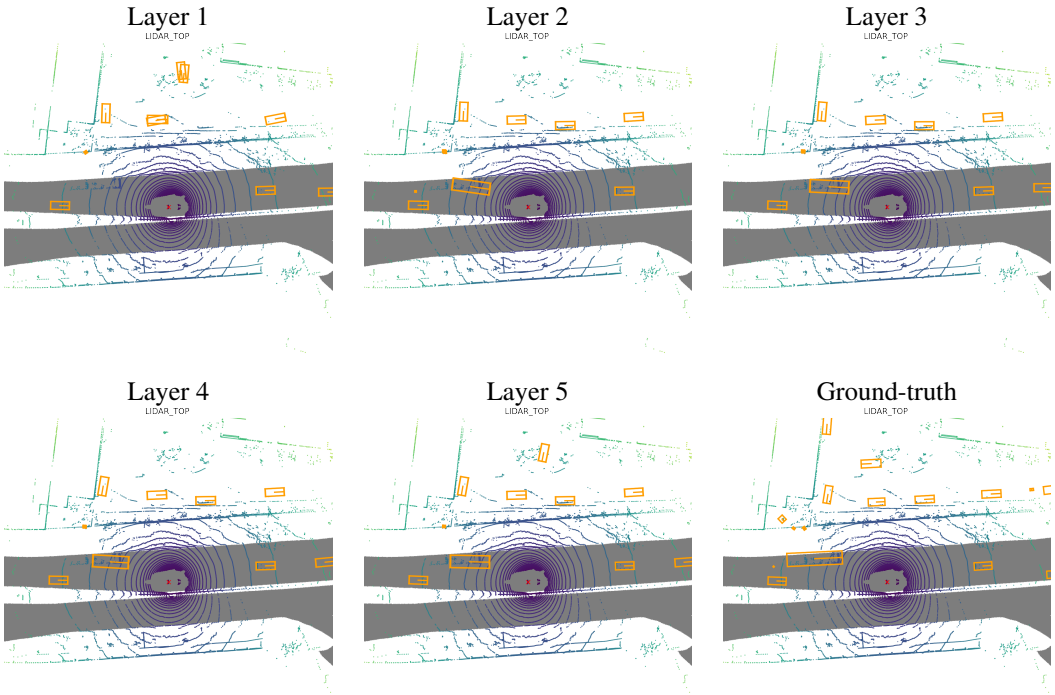

Figure 2: Detection results from layer 1 to layer 5 in the DETR3D head. We visualize the bounding boxes in the BEV and overlay the point clouds from `lidar_top`. The predictions get closer to the ground-truth in the deeper layers.

4.5 Ablation & Analysis

We provide a visualization of object query refinement in Figure 2. We visualize bounding boxes decoded from the object queries in each layer. The predicted bounding boxes get closer to the ground-truth as we go into deeper layers in the model.Also, the leftmost figure shows the learned object query priors shared by all data. We also provide quantitative results in Table 4, which shows that iterative refinement indeed improves performance significantly. This suggests that iterative refinement is both beneficial and necessary to fully leverage our proposed architecture. Furthermore, we provide ablations on the number of object queries in Table 5; increasing the number queries consistently improves the performance until it gets saturated at 900. Finally, Table 6 shows the results with different backbones.

We also provide qualitative results in Figure 3 to facilitate an intuitive understanding of model performance. We project the predicted bounding boxes into 6 cameras as well as a BEV perspective. In general, our method generates reasonable results and even detects relatively small objects. However, our method still exhibits substantial translation error (in line with results in Table 4.2): Although our model avoids explicit depth prediction, depth estimation is still a core challenging in this problem.

Table 5: Results with different number of queries.

# queries	30	100	300	600	900	1200	1500
mAP ↑	0.201	0.313	0.338	0.347	0.346	0.340	0.346
NDS ↑	0.331	0.408	0.415	0.420	0.425	0.415	0.420

Table 6: Results with different backbone.

Backbone ↑	NDS ↑	mAP ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓
ResNet50	0.373	0.302	0.811	0.282	0.493	0.979	0.212
ResNet101	0.425	0.346	0.773	0.268	0.383	0.842	0.216
DLA34	0.394	0.312	0.829	0.276	0.450	0.844	0.221

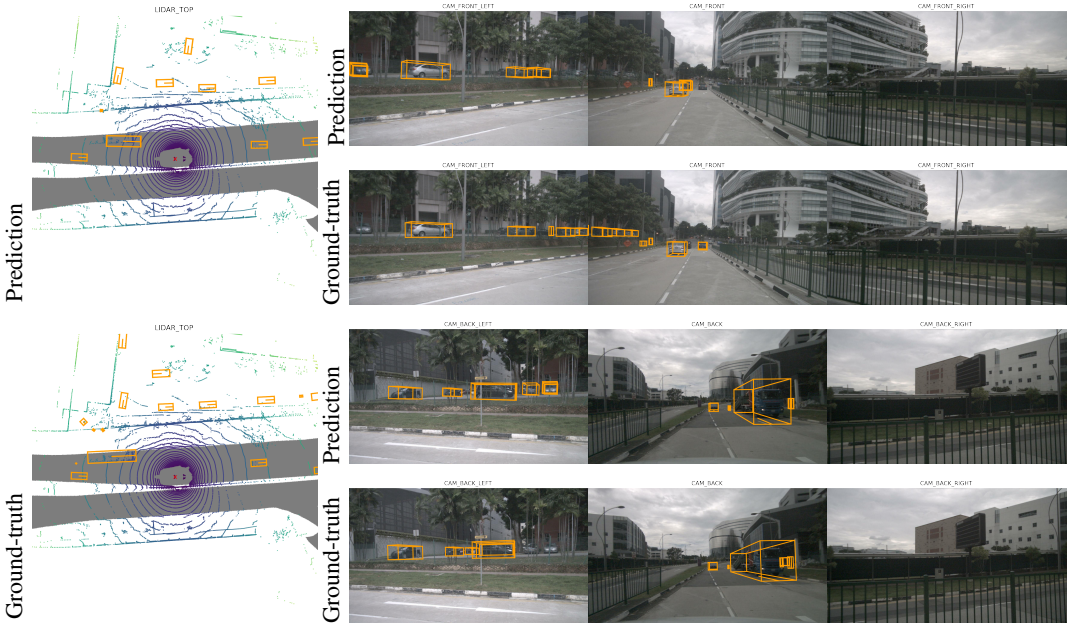

Figure 3: We visualize DETR3D predictions in both BEV and image views. Our model is capable of detecting rather small objects and even objects that were not annotated as ground-truth (cars in CAM_BACK_LEFT). Some failure cases include the far ahead car in CAM_FRONT, that was not detected.

5 Conclusion

We propose a new paradigm to address the ill-posed inverse problem of recovering 3D information from 2D images. In this setting, the input signal lacks essential information for models to make effective predictions without priors learned from data. While other methods either operate solely on 2D computations or use additional depth networks to reconstruct the scene, ours operates in 3D space and uses backward projection to retrieve image features as needed. The benefits of our approach are two-fold: (1) it eliminates the need for middle-level representations (e.g., predicted depth maps or point clouds), which can be a source of compounding errors; and (2) it uses information from multiple cameras by projecting the same 3D point onto all available frames.

Beyond the direct application of our work to 3D object detection for autonomous driving, there are several venues that warrant future investigation. For example, single point projection creates a limited receptive field in the retrieved image feature maps, and sampling multiple points for each object query would incorporate more information for object refinement. Furthermore, the new detection head is input-agnostic, and including other modalities such as LiDAR/RADAR would enhance performance and robustness. Finally, generalizing our pipeline to other domains such as indoor navigation and object manipulation would increase its scope of application and reveal additional ways for further improvement.

References

[1] X. Zhou, D. Wang, and P. Krähenbühl. Objects as points. *ArXiv*, abs/1904.07850, 2019.

[2] T. Wang, X. Zhu, J. Pang, and D. Lin. FCOS3D: Fully convolutional one-stage monocular 3d object detection. *arXiv:2104.10956*, 2021.

[3] Z. Tian, C. Shen, H. Chen, and T. He. FCOS: Fully convolutional one-stage object detection. In *Proc. Int. Conf. Computer Vision (ICCV)*, 2019.

[4] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow. Digging into self-supervised monocular depth prediction. In *ICCV*, 2019.

[5] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv:1907.10326*, 2019.

[6] V. Guizilini, R. Ambrus, S. Pillai, A. Raventos, and A. Gaidon. 3d packing for self-supervised monocular depth estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[7] A. Simonelli, S. Rota Bulò, L. Porzi, P. Kontschieder, and E. Ricci. Demystifying pseudo-lidar for monocular 3d object detection, 12 2020.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition Recognition (CVPR)*, 2016.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[10] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

[11] Anonymous. Object dgcnn: 3d object detection using dynamic graphs. 2021.

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[13] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Neural Information Processing Systems (NeurIPS)*, 2015.

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *The International Conference on Computer Vision (ICCV)*, 2017.

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *The European Conference on Computer Vision (ECCV)*, 2016.

[16] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[17] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai. Deformable {detr}: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=gZ9hCDWe6ke.

[18] Z. Sun, S. Cao, Y. Yang, and K. Kitani. Rethinking transformer-based set prediction for object detection. *ArXiv*, abs/2011.10881, 2020.

[19] P. Sun, R. Zhang, Y. Jiang, T. Kong, C. Xu, W. Zhan, M. Tomizuka, L. Li, Z. Yuan, C. Wang, and P. Luo. SparseR-CNN: End-to-end object detection with learnable proposals. *arXiv:2011.12450*, 2020.

[20] P. Sun, Y. Jiang, E. Xie, Z. Yuan, C. Wang, and P. Luo. OneNet: Towards end-to-end one-stage object detection. *arXiv: 2012.05780*, 2020.

[21] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2156, 2016.

[22] T. Roddick, A. Kendall, and R. Cipolla. Orthographic feature transform for monocular 3d object detection. *arXiv:1811.08188*, 2018.

[23] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017.

[24] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE international conference on computer vision*, pages 1521–1529, 2017.

[25] J. Ku, A. D. Pon, and S. L. Waslander. Monocular 3d object detection leveraging accurate proposals and shape reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11867–11876, 2019.

[26] D. Beker, H. Kato, M. A. Morariu, T. Ando, T. Matsuoka, W. Kehl, and A. Gaidon. Monocular differentiable rendering for self-supervised 3d object detection. *arXiv:2009.14524*, 2020.

[27] I. Barabanau, A. Artemov, E. Burnaev, and V. Murashkin. Monocular 3d object detection via geometric reasoning on keypoints. *arXiv:1905.05618*, 2019.

[28] Z. Liu, Z. Wu, and R. 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.

[29] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[30] R. Stewart, M. Andriluka, and A. Y. Ng. End-to-end people detection in crowded scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition CVPR*, pages 2325–2333. IEEE Computer Society, 2016. doi:10.1109/CVPR.2016.255. URL https://doi.org/10.1109/CVPR.2016.255.

[31] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal Loss for Dense Object Detection. In *The International Conference on Computer Vision (ICCV)*, 2017.

[32] H. W. Kuhn and B. Yaw. The hungarian method for the assignment problem. *Naval Res. Logist. Quart*, pages 83–97, 1955.

[33] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv:1903.11027*, 2019.

[34] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *The International Conference on Computer Vision (ICCV)*, 2017.

[35] L. J. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *CoRR*, 2016.

[36] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[37] F. Yu, D. Wang, E. Shelhamer, and T. Darrell. Deep layer aggregation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[38] T. Yin, X. Zhou, and P. Krähenbühl. Center-based 3d object detection and tracking. *CVPR*, 2021.

[39] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *CVPR*, 2019.