

# FOMO-3D: Using Vision Foundation Models for Long-Tailed 3D Object Detection

Anonymous Author(s)

Affiliation

Address

email

**Abstract:** In the supplementary materials, we first provide more method details and implementation details. Then, we provide additional experimental results that were not included in the main paper due to space limit. Finally, we provide more qualitative results, on both the *nuScenes* and *Highway* datasets.

## 1 Method Details

### 1.1 Background: Attention

We use the attention [1] mechanism heavily to update object queries with other information in the scene. Here, we provide mathematical details of the attention operation.

Attention takes as input a set of  $N$  object queries  $\mathbf{Q} \in \mathbb{R}^{N \times d}$ , a set of  $M$  keys  $\mathbf{K} \in \mathbb{R}^{M \times r}$  and a set of  $M$  values  $\mathbf{V} \in \mathbb{R}^{M \times s}$  to output

$$\mathbf{A} = \text{softmax}\left(\frac{\tilde{\mathbf{Q}}\tilde{\mathbf{K}}^T}{\sqrt{d_k}}\right)\tilde{\mathbf{V}} \in \mathbb{R}^{N \times d}. \quad (1)$$

$d_k$  is the softmax temperature term, and for brevity  $\tilde{\mathbf{Q}} \in \mathbb{R}^{N \times l}$ ,  $\tilde{\mathbf{K}} \in \mathbb{R}^{M \times l}$ ,  $\tilde{\mathbf{V}} \in \mathbb{R}^{M \times d}$  denote linear projections of  $\mathbf{Q}, \mathbf{K}, \mathbf{V}$  with  $\tilde{\mathbf{Q}} = \mathbf{Q}\mathbf{P}_q$ ,  $\tilde{\mathbf{K}} = \mathbf{K}\mathbf{P}_k$ , and  $\tilde{\mathbf{V}} = \mathbf{V}\mathbf{P}_v$ , and  $\mathbf{P}_q \in \mathbb{R}^{d \times l}$ ,  $\mathbf{P}_k \in \mathbb{R}^{r \times l}$ ,  $\mathbf{P}_v \in \mathbb{R}^{s \times d}$  respectively. Output  $\mathbf{A}$  is designed to capture values that are relevant to queries, based on the similarity between keys and queries. The attention function is general and object queries can absorb different types of information depending on the choice of  $\mathbf{K}$  and  $\mathbf{V}$ . In practice, a multi-head attention (MHA) variation is used for increased expressivity. MHA simply projects  $\mathbf{Q}, \mathbf{K}, \mathbf{V}$  with  $m$  different projections onto latent dimensions of sizes  $k/m, k/m, v/m$ . Then the outputs of attention for each projection are concatenated together. Under popular transformer nomenclature, we also refer to queries, keys, and values generally as tokens.

Transformer layers typically use feed-forward networks (FFN) in conjunction with attention for best results [1, 2]. Following common transformer architectures, we update object queries following:

$$\tilde{\mathbf{A}} = \text{LN}(\mathbf{Q} + \text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V})) \quad (2)$$

$$\mathbf{Q} \leftarrow \text{LN}(\tilde{\mathbf{A}} + \text{FFN}(\tilde{\mathbf{A}})). \quad (3)$$

Here LN denotes layer normalization [3].

### 1.2 Attention-based Refinement Stage

Following the two-stage detection paradigm, in a second refinement stage FOMO-3D employs query-based detection to refine all proposals from the first stage. Different from the frustum-based attentions catered to refining camera proposals in the object frustum region, here we rely on more general attention mechanisms to refine the multi-modal proposals in the BEV space.

Refinement starts with initializing object queries. From the LiDAR proposal branch, each proposal initializes a query using the feature vector from  $\mathbf{F}_{lidar}$  which decoded the box. On the other hand,

queries from the camera proposal branch are taken directly for continued refinement. We then take the union of these proposals and apply non-maximum suppression (NMS) to remove duplicates.

Queries are then refined iteratively through a series of transformer decoder layers. We adopt several kinds of attention layers to model complex correlations between object queries and multi-sensor inputs in an end-to-end manner, which we detail below.

**LiDAR Cross-Attention** adopts the BEV LiDAR feature map  $\mathbf{F}_{lidar}$  and flattens the feature map to obtain a set of key-value LiDAR feature tokens. Concretely, for  $\mathbf{F}_{lidar} \in \mathbb{R}^{V'_x \times V'_y \times D}$ , we flatten it to obtain  $V'_x \times V'_y$  LiDAR feature tokens each with dimension  $D$ , and we set the values to be the same as the keys in the attention operation. Different from the LiDAR-based proposal stage where each proposal is decoded from features from convolution-backbones and subject to receptive field constraints, the attention mechanism enables object queries to attend to spatially distant LiDAR tokens. This allows object queries to incorporate additional information from the input point cloud.

However, since the feature map is a dense representation of the scene, attending to all LiDAR feature tokens is infeasible to scale. For efficiency reasons, we employ *deformable attention* [4] - learning a set of spatial offsets which is added to the object query’s location to derive a sparse set of key-value LiDAR tokens. Specifically, for each object query  $\mathbf{q}_f$  with initial 3D position  $\mathbf{q}_p = (p_x, p_y, p_z)$ , we apply a lightweight MLP to  $\mathbf{q}_f$  to decode a few 2D spatial offsets  $\{(\delta_{x,i}, \delta_{y,i})\}$ , and add to the BEV location  $(p_x, p_y)$  to obtain  $\{(p_x + \delta_{x,i}, p_y + \delta_{y,i})\}$ . We then locate the associated LiDAR tokens at these BEV locations, and apply the attention mechanism in Sec. 1.1 between each query  $\mathbf{q}_f$  and the sampled LiDAR tokens.

**Camera Cross-Attention** incorporates information from cameras, which contains rich semantic cues essential for accurately classifying the object queries. In this attention layer object queries cross attend to OWL tokens  $\mathcal{F}_{owl}$  which capture semantic and contextual information from the image. Note that we attend to the 2D camera features here, different from the previous attention to image features lifted to BEV.

To handle multi-camera inputs, we factorize the attention over each individual camera, and apply adaptive mean pooling to aggregate features across multiple cameras. Specifically, for a proposal with initial 3D position  $\mathbf{q}_p$  with all valid projections  $\{(u_j, v_j)\}$  in camera  $j$ , we first apply an MLP to query feature  $\mathbf{q}_f$  to obtain a set of 2D offsets  $\{(\delta_{u,j,l}, \delta_{v,j,l})\}$  unique to each  $(u_j, v_j)$ . Then, we retrieve the set of OWL tokens at each offset location  $\mathcal{F}_{owl}^{(c_j)} = \{\mathcal{F}_{owl,j}(u_j + \delta_{u,j,l}, v_j + \delta_{v,j,l})\}$ , and MHA at each valid camera yields

$$\mathbf{A}^{(c_j)} = \text{MHA}(\mathbf{Q}, \mathcal{F}_{owl}^{(c_j)}, \mathcal{F}_{owl}^{(c_j)}) \quad (4)$$

We next apply mean pooling among all attention matrices across valid cameras:

$$\mathbf{A} = \text{Mean}(\{\mathbf{A}^{(c_j)}\}). \quad (5)$$

Then we apply  $\mathbf{A}$  to update  $\mathbf{q}_f$  based on Eq. 2.

**Object Self-Attention** designates object queries as the key-value tokens as well. This enables the model to exploit correlations between different traffic participants in the scene. The semantic class, object pose, and geometry can all be improved via object relationship cues. For example, children on the street are often accompanied by adults while parking lot vehicles are usually parked in parallel. However, exploiting object relationship cues is highly reliant on understanding the relative positions between object queries. As positional information has yet to be encoded upstream, we also encode each query with positional embedding  $\mathbf{q}^{(i)} \leftarrow \mathbf{q}^{(i)} + \text{PE}(\mathbf{q}_p^{(i)})$  prior to self-attention. Our positional encoding applies sinusoidal positional encoding from [1] followed by a 3-layer MLP.

Our transformer architecture interleaves multiple repetitions of LiDAR-camera-object attention “blocks”. This facilitates learning complex relationships between diverse traffic participants and multi-sensory inputs. Moreover, after each block, we decode each object query into a detection via a lightweight MLP. This enables more dense supervision on intermediate outputs of the transformer, while also allowing refinement to happen in an iterative manner.

### 76 1.3 Loss Functions

77 For the camera-based proposal branch and refinement stage, we apply DETR-style matching [5]  
 78 to pair ground-truth labels with detections, and compute a box regression loss and a classification  
 79 loss. For the camera-based proposal, we additionally add a frustum-based hard constraint during  
 80 matching. We next detail the matching and losses.

81 Given  $N$  3D detections and  $M$  ground-truths, we first apply Hungarian-based matching based  
 82 on a cost matrix  $\mathbf{C} \in \mathbb{R}^{N \times M}$  where  $C_{ij}$  indicates the score between the detection  $\mathbf{b}_i =$   
 83  $[x_i, y_i, z_i, l_i, w_i, h_i, \theta_i]$  and ground-truth  $\mathbf{b}_j^* = [x_j^*, y_j^*, z_j^*, l_j^*, w_j^*, h_j^*, \theta_j^*]$ . Specifically,

$$C_{ij} = \lambda_{giou} \text{GIoU}(\mathbf{b}_i, \mathbf{b}_j^*) + \lambda_{l_2} \|\text{Loc}(\mathbf{b}_i) - \text{Loc}(\mathbf{b}_j^*)\|_2, \quad (6)$$

84 where GIoU is the 3D generalized IoU [6],  $\text{Loc}(\mathbf{b}) = [x, y, z, l, w, h]$  is the centroid and dimension  
 85 of the 3D box, and  $\lambda_{giou}$  and  $\lambda_{l_2}$  are hyperparameters.

86 In addition, for the camera proposal branch, we compute whether the ground-truth  $\mathbf{b}_j^*$  falls inside  
 87 the object frustum of detection  $\mathbf{b}_i$  as follows: we first transform both 3D boxes from the reference  
 88 frame to the camera frame with the camera extrinsics matrix  $[\mathbf{R}|\mathbf{t}]$  to obtain  $\mathbf{b}_i^{cam}$  and  $\mathbf{b}_j^{*cam}$ . Then  
 89 for any bounding box  $\mathbf{b}$  in the camera frame, we compute the angles between the box and the camera  
 90 as  $\Phi(\mathbf{b}) = (\arctan(x/z), \arctan(y/z))$ , and then we can derive

$$\text{in\_frustum}(i, j) = (\|\Phi(\mathbf{b}_i^{cam}) - \Phi(\mathbf{b}_j^{*cam})\|_2 < \alpha_\phi) \quad \text{and} \quad (\|z_i^{cam} - z_j^{*cam}\| < \alpha_z) \quad (7)$$

91 where the first condition compares the camera angles between the detection and ground-truth and  
 92 constrains the differences to be under a threshold  $\alpha_\phi$  (set to 0.03 radians in practice), and the second  
 93 condition constrains the depths of the two boxes to be no more than  $\alpha_z$  apart (set to 5 meters for  
 94 *nuScenes* and 30 meters for *Highway*). If  $\text{in\_frustum}(i, j)$  is False, we set the corresponding  $C_{ij}$  to  
 95  $\text{inf}$ .

96 With the computed cost matrix  $\mathbf{C}$ , we conduct Hungarian Matching to assign a ground-truth box  
 97 for each detection. If the associated matching cost is  $\text{inf}$ , we discard the matching and the associ-  
 98 ated detection will remain unmatched. For each matched detection, ground-truth pair  $(\mathbf{b}_i, \mathbf{b}_i^*)$ , we  
 99 compute a box regression loss:

$$\mathcal{L}_{box}(\mathbf{b}_i, \mathbf{b}_i^*) = -\lambda_{giou} \text{GIoU}(\mathbf{b}_i, \mathbf{b}_i^*) \quad (8)$$

$$+ \lambda_{xyz} (\|x_i - x_i^*\| + \|y_i - y_i^*\| + \|z_i - z_i^*\|) \quad (9)$$

$$+ \lambda_{lwh} (\|l_i - l_i^*\| + \|w_i - w_i^*\| + \|h_i - h_i^*\|) \quad (10)$$

100 where  $\lambda_{xyz} = 0.2$  and  $\lambda_{lwh} = 0.04$  in practice.

101 In addition, for each detection  $i$  with object class logits  $\mathbf{c}_i \in \mathbb{R}^C$  where  $C$  is the total number of  
 102 object classes, we set the ground-truth  $\mathbf{c}_i^* \in \mathbb{R}^C$  as follows: if the detection is matched to a ground-  
 103 truth label with class  $1 \leq k \leq C$ , then we set  $\mathbf{c}_i^*$  to be a one-hot vector with 1 at the  $k^{\text{th}}$  position,  
 104 otherwise  $\mathbf{c}_i^*$  is a zero vector. Then

$$\mathcal{L}_{class}(\mathbf{c}_i, \mathbf{c}_i^*) = \text{SigmoidFocalLoss}(\mathbf{c}_i, \mathbf{c}_i^*) \quad (11)$$

105 where the SigmoidFocalLoss first applies sigmoid to the logits  $\mathbf{c}_i$  and then uses focal loss [7].

106 The final loss is

$$\mathcal{L} = \frac{1}{N^*} \sum_i \mathcal{L}_{box}(\mathbf{b}_i, \mathbf{b}_i^*) + \frac{1}{N} \sum_i \mathcal{L}_{class}(\mathbf{c}_i, \mathbf{c}_i^*) \quad (12)$$

107 where  $N^*$  is the number of matched pairs.

## 108 2 Implementation Details

### 109 2.1 OWL: Cropping and Prompting

110 OWL [8] usually preprocesses the input image by resizing it to a square image of fixed dimensions  
 111 (e.g., 960 by 960 pixels for OWL-Medium, and 1008 by 1008 pixels for OWL-Large). If the input

image is a rectangular image, it will pad it to a square and then resize. To avoid information loss from padding, we preprocess our input images preemptively by cropping the rectangular input image into multiple square crops. For *nuScenes*, each input image is 1600 pixels by 900 pixels, and we apply two crops by cropping the leftmost  $900 \times 900$  pixels and rightmost  $900 \times 900$  pixels of the input image, and run inference with the two crops separately. We merge the 2D bounding boxes from the two crops with concatenation and image-based 2D non-maximum suppression with iou threshold 0.85. For camera-based proposal, each box is associated with the feature embedding from the respective crop. For camera-attention during refinement, we zero-pad the feature map of each crop to the original dimension (right-pad for the leftmost crop, and left-pad for the rightmost crop), stack them, and apply a lightweight convolutional network to generate a unified feature map of the same size as the input rectangular image. The convolutional network is:

```

123     x_proj = Conv2d(in=768*2, out=256, kernel=1)(x)
124     x = Conv2d(in=256, out=256, kernel=3)(x_proj)
125     x = GroupNorm(num_groups=8, out=256)(x)
126     x = GELU(x)
127     x = Conv2d(in=256, out=256, kernel=3)(x_proj)
128     x = GroupNorm(num_groups=8, out=256)(x)
129     x = GELU(x)
130     x = x + x_proj

```

To prompt OWL for 2D detection boxes, we use the following prompts for *nuScenes*:

```

132     vehicle.car: ["a car"]
133     vehicle.truck: ["a truck"]
134     vehicle.trailer: ["a trailer"]
135     vehicle.construction: ["a construction vehicle"]
136     vehicle.bicycle: ["a bicycle"]
137     vehicle.motorcycle: ["a motorcycle"]
138     vehicle.bus: ["a bus"]
139     vehicle.emergency: ["a police vehicle", "an ambulance"]
140     pedestrian.adult: ["a person"]
141     pedestrian.child: ["a child"]
142     pedestrian.stroller: ["a stroller"]
143     pedestrian.construction_worker: ["a construction worker"]
144     pedestrian.police_officer: ["a police officer"]
145     pedestrian.personal_mobility: ["a scooter", "a wheelchair"]
146     movable_object.trafficcone: ["a traffic cone"]
147     movable_object.pushable_pullable: ["a drolley", "a wheel barrow",
148                                         "a shopping cart", "a garbage bin"]

```

If an object class corresponds to multiple prompts, then all boxes associated with the prompt belong to this object class. Note that we do not have prompts for barriers and debris because OWL tends to generate a lot of false positives for these two classes. In addition, we found that “a person” is a better prompt for adult.

OWL outputs each 2D box with an affinity score for each prompt. We take the argmax of the affinity scores and assigns the object class associated with the argmax prompt to be the class of the 2D box. Before feeding the list of boxes to the camera proposal branch, we additionally perform per-prompt score-based filtering to filter out low confidence 2D boxes. Specifically, we set the confidence score threshold to 0.2 for car, truck, trailer, bus, construction vehicle, police vehicle, stroller, scooter and wheel barrow, 0.15 for bicycle, motorcycle, wheelchair, traffic cone, drolley and shopping cart, 0.1 for ambulance, person, child, construction worker, police officer, and 0.3 for garbage bin.

## 2.2 Proposal Stage

For *nuScenes*, we use LiDAR points within distance  $[-54, 54]$ ,  $[-54, 54]$  and  $[-5, 3]$  meters for the  $x$ ,  $y$ ,  $z$  directions respectively. For *Highway*, we use a longer  $x$  range  $[0, 235]$  meters along  $x$ . For voxelization of both LiDAR and image point clouds, the voxel size is  $(7.5, 7.5, 20)$  centimeters along  $x$ ,  $y$  and  $z$ . for *nuScenes* and  $15.625\text{cm}$  for *Highway*.

For the LiDAR-only branch, we follow the group-free wide-512-channel header CenterPoint implementation from the LT3D codebase [9] exactly. We did not apply the hierarchical heuristic as we did not find it to help with performance in our experiments.

For the camera-based branch, we provide more details of the model architecture.

**Query Initialization and Image Point Cloud Encoding** The camera proposal branch first applies OWL to the input image with details specified in Sec. 2.1 to obtain a set of 2D detection boxes with associated OWL tokens that decode each box. Each OWL token is a feature vector  $\in \mathbb{R}^{1024}$ . For each box, we sample the M3D depth map at the 2D box center with nearest neighbor interpolation to obtain the initial depth. As M3D can produce degenerate zero depths, we discard the 2D box if the depth is  $< 0.5$ . The remaining 2D boxes are initialized as queries to lift to 3D.

For each pixel inside any valid 2D detection, we query the M3D depth map and lift the pixel if the associated M3D depth confidence is  $> 0.5$ . The lifted pixels form a 3D pseudo image-based feature point cloud, where each point is associated with an OWL token of dimension 1024. We then process the lifted point cloud with a feature encoder and construct a BEV image-based feature map.

The feature encoder consists of a sparse voxelizer followed by a sparse 3D feature extractor. The voxelizer first voxelizes the 3D point cloud into  $V_x \times V_y \times V_z$  voxels based on the range of interest and voxel sizes specified above. It applies a linear layer to each point-based feature to reduce feature dimension from 1024 to 256, and encodes the xyz position with positional encoding followed by an MLP to add to the reduced feature. To aggregate point features inside each voxel cell, we mean pool all available point features as the voxel feature. The resulting sparse voxel feature grid is of dimension  $V_x \times V_y \times V_z \times 256$ . We use a sparse voxelizer that only keeps track of occupied voxels for memory efficiency.

Then, we apply the sparse 3D encoder to the image-based voxel feature grid. To simplify notation, we use `Conv3d(in=128, out=128, k=3, s=1, bias=False)` to denote a 3D sparse convolution layer with input channels 128, output channels 128, kernel size  $3 \times 3 \times 3$ , stride size  $1 \times 1 \times 1$  and no bias. The sparse 3D encoder consists of `Conv3d(in=256, out=128, k=3, s=1, bias=False)`, followed by `BatchNorm (BN)`, `ReLU`, and two repetitions of `Conv3d(in=128, out=128, k=3, s=1, bias=False)  $\rightarrow$  BN  $\rightarrow$  ReLU  $\rightarrow$  Conv3d(in=128, out=128, k=3, s=1, bias=False)  $\rightarrow$  BN`. For each voxel cell, we further apply a linear layer to the feature with input and output dim 128 and `bias=True`. Next, we squash the 3D voxel grid to BEV by mean pooling the features along the  $z$  dimension. The BEV feature map goes through a dense block of `Conv2d(in=128, out=128, k=3, s=1, bias=False)  $\rightarrow$  BN  $\rightarrow$  Conv2d(in=128, out=128, k=3, s=1, bias=False)  $\rightarrow$  BN  $\rightarrow$  ReLU`, and then a sequence of `Conv2d(in=128, out=256, k=1, s=1, bias=False)  $\rightarrow$  BN  $\rightarrow$  ReLU` to finally decode a BEV feature map of size  $V_x \times V_y \times 256$ .

**Frustum-based Attention** For each feature initialized with OWL tokens  $\in \mathbb{R}^{1024}$  and positional encodings over  $xyz$  and  $uvd$ , we first apply a lightweight MLP with `Linear(in=1024, out=512, bias=False)  $\rightarrow$  LayerNorm (LN)  $\rightarrow$  ReLU  $\rightarrow$  Linear(in=512, out=512, bias=False)  $\rightarrow$  LN  $\rightarrow$  ReLU  $\rightarrow$  Linear(in=512, out=512, bias=False)` to reduce feature dimension to 512. To include classification information from OWL, we also encode the OWL class logits that represent the affinity score between the detection and all prompts. We encode the OWL class logits with an MLP that has a similar architecture to the feature dimension reduction MLP except the input dimension of the very first linear layer is the number of OWL prompts. We add these two encodings as the updated query feature to be used in the attention operations.

To efficiently sample features from the merged LiDAR and image BEV feature map, we sample along frustum rays as detailed in the main paper. We then apply two repetitions of attention blocks, where each block starts with an object-to-object self-attention, and an object-to-sampled-BEV-features cross-attention. The attention layer [1] follows Sec. 1.1 and is a standard pytorch `TransformerEncoderLayer` layer with the attention operation followed by two FFNs. Both the object-to-object self-attention and the object-to-BEV-feature cross-attention layers are a single transformer encoder block with 8 attention heads, input and FFN feature dimension 512, bias=False and dropout=0.1.

At the end of each attention block, we use a lightweight MLP to decode the 3D proposal box parameters. The box decoder is a sequence of `Linear(in=512, out=256, bias=False) → LN → ReLU → Linear(in=256, out=256, bias=False) → LN → ReLU → Linear(in=256, out=8, bias=False)` to decode  $(x, y, z, l, w, h, \sin(\theta), \cos(\theta))$ . After the first attention block, we update the query 3D position with the decoded 3D box, and re-sample the BEV features based on the new 3D position to cross-attend to in the second attention block. After the second attention block, we additionally use a lightweight MLP to decode the proposal object class heatmap  $\mathbf{c} \in \mathbb{R}^C$ . The class decoder is a sequence of `Linear(in=512, out=256, bias=False) → LN → ReLU → Linear(in=256, out=256, bias=False) → LN → ReLU → Linear(in=256, out=C, bias=False)` to output a logit for each of the  $C$  classes. During training, we supervise both sets of the 3D proposal box parameters, and the final class logits.

At the end of the multi-modal proposal stage, we aggregate both LiDAR and camera 3D proposals with IoU-based non-maximum suppression (nms) performed in the 2D BEV space. We use a nms threshold of 0.2. In addition, in practice we found that for *nuScenes* the camera-proposal branch gives the best performance when it is used to lift small and/or rare objects complementary to the LiDAR proposals. As a result, in practice, we remove camera proposals for car, trailer, truck, bus and construction vehicle classes before refinement.

### 2.3 Refinement Stage

In this subsection we provide more details with the transformer layers used in object-LiDAR-camera attention blocks. First, as explained in the main paper, the object query features  $\in \mathbb{R}^{256}$  are initialized from either the LiDAR features or the camera object queries. Both the deformable attention blocks in LiDAR and camera cross-attention employ a linear layer with input dim 256 and output dim 2 to sample 2D offsets. The cross-attention transformer layers have 8 attention heads with input dim 256, FFN feature dim 1024, bias=False, and drop out=0.1. The self-attention transformer layer has 8 attention heads with input dim 256, FFN feature dim 256, bias=False, and drop out=0.1.

After each object-camera-LiDAR attention block, we apply a lightweight box decoder and a class decoder. The box decoder is a sequence of `Linear(in=256, out=256, bias=False) → LN → ReLU → Linear(in=256, out=256, bias=False) → LN → ReLU → Linear(in=256, out=7, bias=False)` to decode  $(x, y, z, l, w, h, \theta)$  box parameters. The class decoder is a sequence of `Linear(in=256, out=256, bias=False) → LN → ReLU → Linear(in=256, out=C, bias=False)` to decode logits for  $C$  classes. During training we supervise both sets of 3D boxes.

## 3 Additional Experiments Results

***nuScenes* results with hierarchical mAP.** Following previous works [9, 10], we also adopt the hierarchical mAPs ( $mAP_H$ ), which reports mAP with three tiers based on the least common ancestor (LCA) distance: LCA0 is the standard per-class mAP, LCA1 treats each object class as one of the three parent classes (vehicle, pedestrian, movable object) and tolerates misclassification with sibling classes, and LCA2 measures class-agnostic mAP on *nuScenes*. Table 1 showcases per-class hierarchical metrics, comparing ours against previous LT3D methods [9, 10]. FOMO-3D improves the hierarchical metrics for almost all classes in all three LCA tiers, showing that FOMO-3D is not



$mAP_H$	Method	Car	Adult	Truck	CV	Bicycle	MC	Child	CW	Stroller	PP
LCA0	MMF [9]	88.5	86.6	63.4	29.0	58.5	68.2	5.3	35.8	31.6	39.3
	MMLF [10]	86.3	87.7	60.6	35.3	70.0	75.9	8.8	55.9	37.7	<b>58.1</b>
	FOMO-3D	<b>88.9</b>	<b>90.7</b>	<b>65.1</b>	<b>36.6</b>	<b>72.8</b>	<b>80.2</b>	<b>29.8</b>	<b>60.2</b>	<b>40.1</b>	<b>50.0</b>
LCA1	MMF [9]	89.4	87.4	<b>72.4</b>	31.3	61.2	69.7	15.2	52.0	37.7	39.4
	MMLF [10]	86.8	88.3	68.5	37.3	70.4	77.1	16.2	66.0	51.5	<b>58.2</b>
	FOMO-3D	89.4	<b>91.2</b>	70.3	<b>39.2</b>	<b>73.9</b>	<b>81.5</b>	<b>59.2</b>	<b>73.4</b>	<b>54.8</b>	<b>50.1</b>
LCA2	MMF [9]	89.5	87.7	<b>72.5</b>	31.5	62.3	69.9	16.9	56.3	38.8	39.8
	MMLF [10]	86.9	88.6	68.6	37.7	70.9	77.4	16.3	69.0	52.4	<b>58.9</b>
	FOMO-3D	89.5	<b>91.5</b>	70.4	<b>39.7</b>	<b>74.9</b>	<b>81.9</b>	<b>59.5</b>	<b>77.1</b>	<b>58.2</b>	50.6

Table 1: **[nuScenes] Class-specific hierarchical metrics.** CV = Construction Vehicle. MC = Motorcycle. CW = Construction Worker. PP = Pushable-Pullable. Medium and Few classes are in **blue**.

only better at fine-grained classification for almost all object classes at LCA0, but also improves the general localization and detection quality at LCA1 and LCA2 as well.

Method	Frustum-based Loss Constraint	All	Many	Medium	Few
$M_1$		54.0	80.3	59.5	25.8
$M_2$	✓	<b>54.6</b>	79.9	59.6	<b>27.6</b>

Table 2: **[nuScenes]** Ablation for frustum-based matching constraint in camera proposals.

**Effect of frustum-based matching loss.** When training the camera proposals, we add a hard constraint in detection to ground-truth label matching that the ground-truth label must be present in the detection frustum space. We perform an ablation on the *nuScenes* dataset to compare with using regular IoU-based matching without the frustum-based hard constraint. Table 2 shows that with this hard constraint ( $M_1 \rightarrow M_2$ ), the mAP for Few is two points better.

**Ablation table on Highway.** In the main paper, the ablation results for the *Highway* dataset is presented in terms of net gains over the LiDAR-only model in Fig. 5. For completeness, Table 3 shows the detailed numerical metrics for the lidar-only model, lidar-proposal + multi-modal refinement, and multi-modal proposal + multi-modal refinement respectively. The results show that both camera proposal and camera attention during refinement are helpful.

**Alternative fusion strategies.** In the proposal stage of FOMO-3D, we adopt a late-fusion design where we merge the LiDAR-based 3D proposals and camera-based 3D proposals in the end. Alternatively, due to the effectiveness of feature-level fusion in BEVFusion [11], it is also worth exploring feature-level fusion with OWL features in the proposal stage. We therefore designed two alternative proposal-level fusion strategies for exploration.

In the first method, we replace the proposal stage with a dense feature lifting design, where we lift every OWL token in the image with respective M3D depths, fuse with the LiDAR features with concatenation, and decode a single set of 3D proposals to be refined later. This fusion strategy is similar to BEVFusion, except we are only using a single depth from M3D instead of learning and lifting with a depth distribution.

In the second method, instead of lifting with the M3D depth directly, we use a learned depth based on M3D. We first create 20 scale factors that are based on uniform intervals in  $[0.5, 1.5)$ , i.e.,  $s = [0.5, 0.55, 0.6, \dots, 1.45] \in \mathbb{R}^{20}$ . Each scale factor will be multiplied with the original M3D depth

Method	Prop	Refine	Vehicle	Towed	Cone	Person	Cyclist
$M_1$	L	L	86.7	73.5	82.6	58.4	74.8
$M_2$	L	L+C	86.9	73.8	86.0	65.4	77.7
$M_3$	L+C	L+C	<b>87.2</b>	<b>74.2</b>	<b>88.9</b>	<b>68.7</b>	<b>79.3</b>

Table 3: **[Highway]** Ablations.

Method	Proposal Stage Fusion	Learned Depth	All	Many	Medium	Few
$M_1$	Feature-level		54.8	80.2	60.6	26.9
$M_2$	Feature-level	✓	53.3	79.6	59.0	24.7
$M_3$ (FOMO-3D)	Late		54.6	79.9	59.6	27.6

Table 4: [nuScenes] Explorations for alternative proposal-level fusion strategies.

Method	Proposal Stage Fusion	Learned Depth	Vehicle	Towed	Cone	Person	Cyclist
$M_1$	Feature-level		86.1	71.7	86.5	64.6	76.4
$M_2$	Feature-level	✓	87.0	73.7	87.0	64.2	74.7
$M_3$ (FOMO-3D)	Late		<b>87.2</b>	<b>74.2</b>	<b>88.9</b>	<b>68.7</b>	<b>79.3</b>

Table 5: [Highway] Explorations for alternative proposal-level fusion strategies.

to create 20 depth buckets. We then apply a lightweight MLP on each OWL token to learn  $\mathbf{p} \in \mathbb{R}^{20}$ , representing the confidence of each depth bucket. We finally aggregate the learned depth with  $d' = \frac{1}{20} \sum_{i=0}^{19} s_i \cdot p_i \cdot d$  where  $d$  is the original M3D depth, and  $p_i$  is the respective confidence in  $\text{Softmax}(\mathbf{p})$ . We lift every OWL token with the learned depth, fuse with the LiDAR features similar to the first method and decode a single set of proposals. Compared to the first method, the second method has more flexibility to correct depth errors. Unlike BEVFusion, we do not lift the same token with multiple depths due to memory constraints.

Table 4 shows the comparison for these two alternative fusion methods against FOMO-3D on the nuScenes dataset. Interestingly, the feature-level fusion method that uses M3D depth directly ( $M_1$ ) has comparable performance with FOMO-3D, thanks to M3D’s accurate zero-shot depths in the short evaluation range of nuScenes [12]. The learned depth version ( $M_2$ ) is slightly worse, possibly due to the fact that the learned depth introduces more noises to the already accurate M3D initializations. In addition, it’s worth noting that  $M_1$  has noticeably better performance on the Medium group, which implies that directly using all features without any prompting combined with feature-level fusion could be very effective at exploiting 2D foundation models when the depth is very accurate. However, on the Highway dataset with challenging longer evaluation ranges, M3D has more errors especially in the far range. As a result, naively lifting and fusing with M3D depths results in worse performance for  $M_1$  in Table 5, and learned depths over depth buckets  $M_2$  is overall better than  $M_1$ . FOMO-3D outperforms both these baselines, which shows that FOMO-3D’s camera proposal branch with more elaborate 3D lifting and attention-based multi-modal fusion strategies is more robust to M3D depth errors and more effective at lifting 2D foundation model priors to 3D.

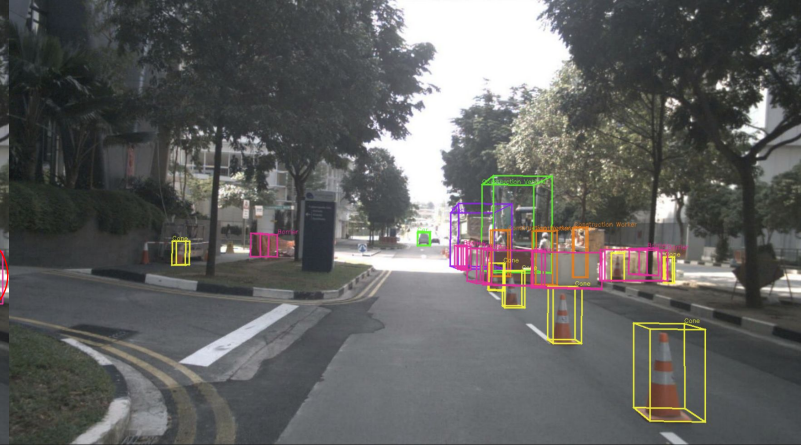
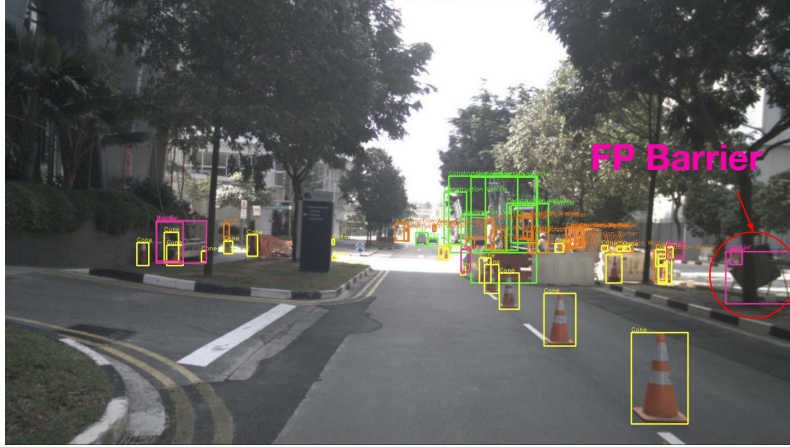
**Additional qualitative results for nuScenes.** Fig. 1, 2, 3, 4 and 5 provide more qualitative examples on the nuScenes dataset. In general, the LiDAR-only model often misses small and/or rare objects such as cone, adult and construction worker. OWL has great zero-shot detections for these small objects, but can have false positives especially when there are true positives in the local region. FOMO-3D is able to recover small and/or rare objects in most cases, with some misclassification and false positive failure cases discussed in Fig. 2 and Fig. 5.

**Additional qualitative results for Highway.** Fig. 6, 7, 8 and 9 provide qualitative examples on the Highway dataset. In general, although OWL has impressive zero-shot 2D detection boxes, it oftentimes outputs false positives and confuses vehicle and towed object classes. LiDAR-only detections learn to distinguish between vehicles and towed objects in most cases, but confuse cone with person (due to their similar cylindrical geometries and smaller sizes) and tends to output false positives around spurious LiDAR points. FOMO-3D is able to leverage both 2D and 3D information for correct detection and classification. For truck/trailer confusion, FOMO-3D shares similar failure cases as the LiDAR-only model.



OWL 2D Detection

Ground-Truth Labels



LiDAR-Only 3D Detection

FOMO-3D 3D Detection

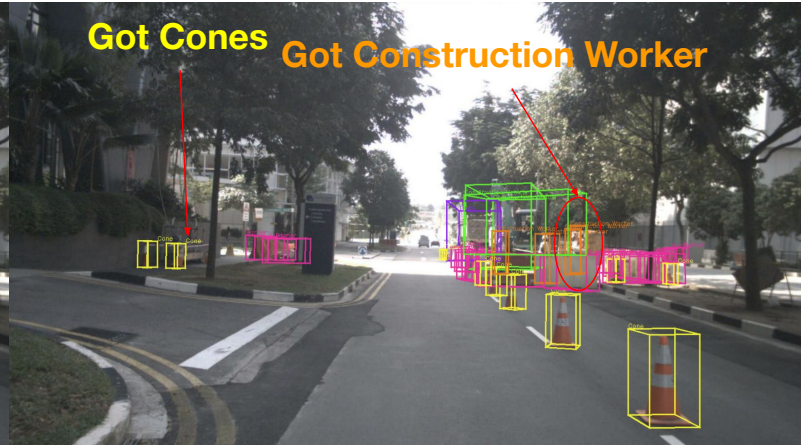
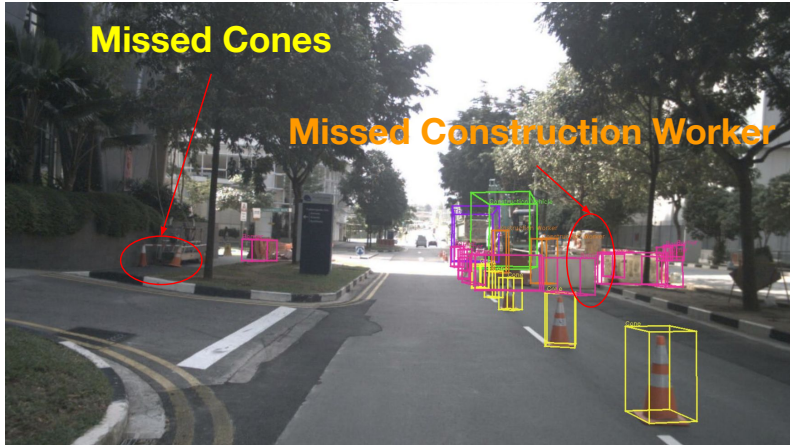
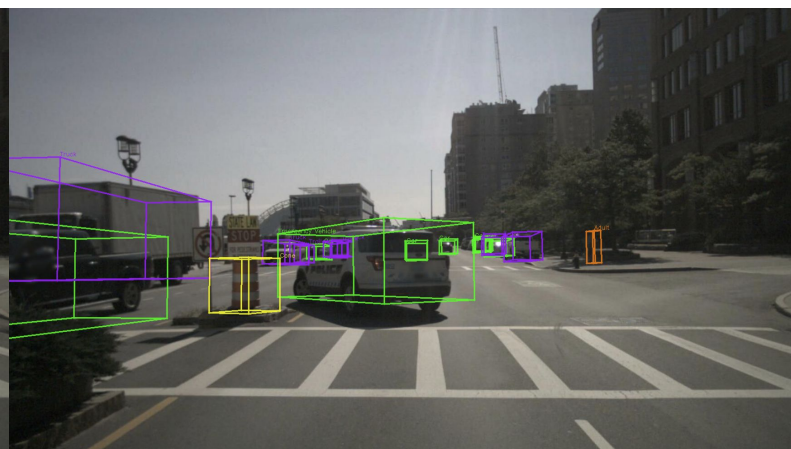
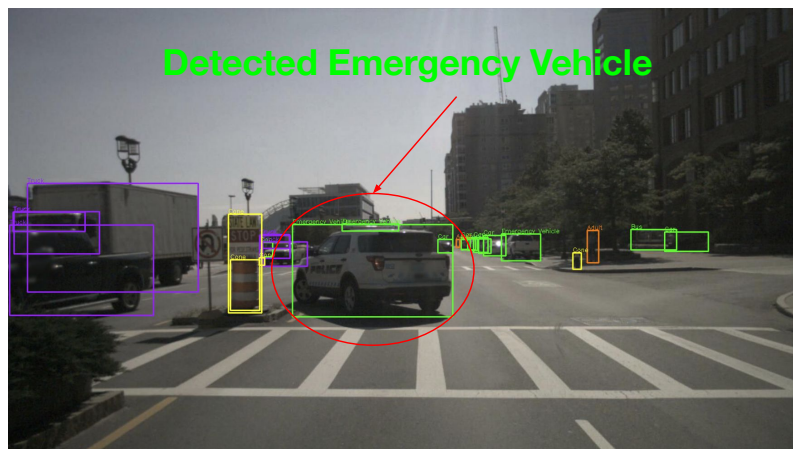


Figure 1: [nuScenes] Qualitative example #1. In this example, we added OWL prompts for barriers to illustrate low-quality zero-shot barrier detections. As shown in the visualization, OWL produces many false positive barriers without capturing most of the true positive barriers. As a result we removed barrier prompting from the camera proposal branch. In addition, the LiDAR-only model misses small objects such as cones and construction workers, but OWL and FOMO-3D are able to detect them successfully.

OWL 2D Detection

Ground-Truth Labels



LiDAR-Only 3D Detection

FOMO-3D 3D Detection

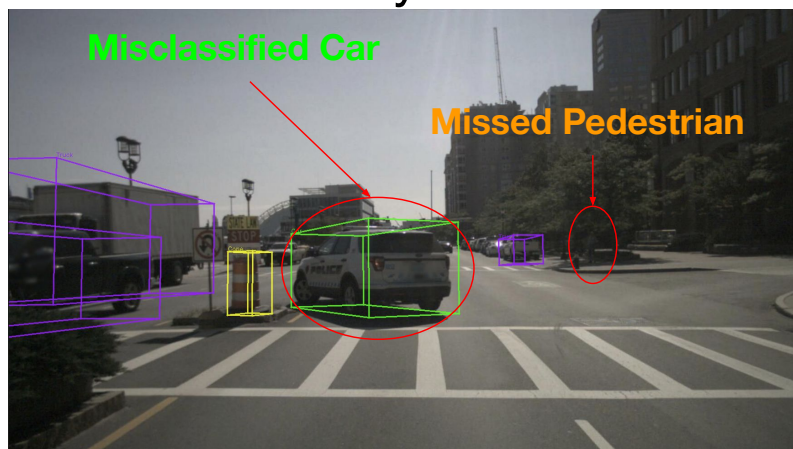
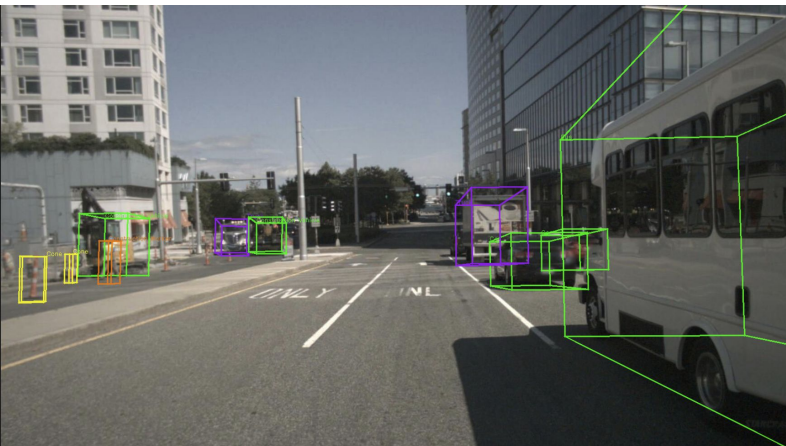
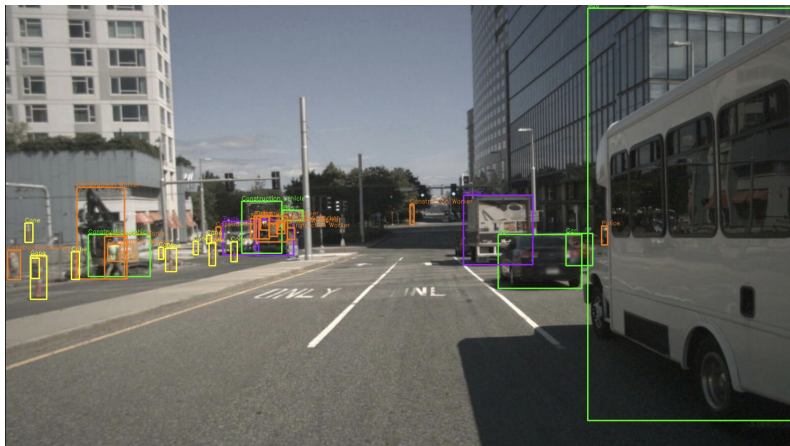


Figure 2: **[nuScenes]** Qualitative example #2. In this example, OWL correctly detects and classifies an adult and an emergency vehicle. The LiDAR-only model misses the adult (because it is small and sparsely observed by LiDAR) and misclassifies the emergency vehicle as car due to lack of semantics. FOMO-3D is able to successfully detect the adult thanks to OWL, but fails to retain the original OWL emergency vehicle classification.



OWL 2D Detection

Ground-Truth Labels



LiDAR-Only 3D Detection

FOMO-3D 3D Detection

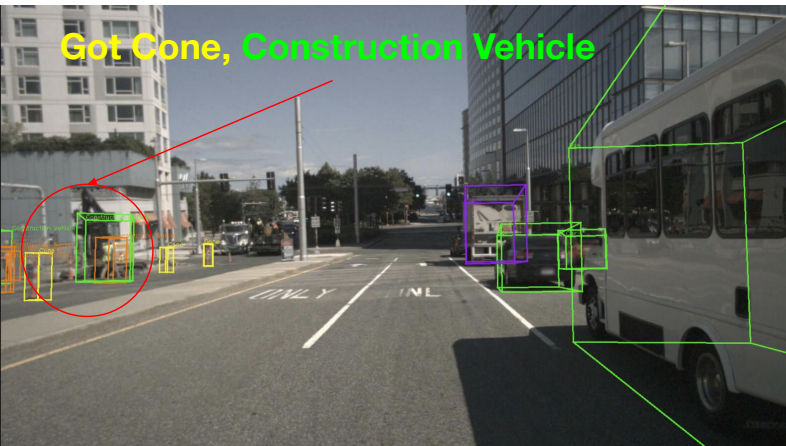
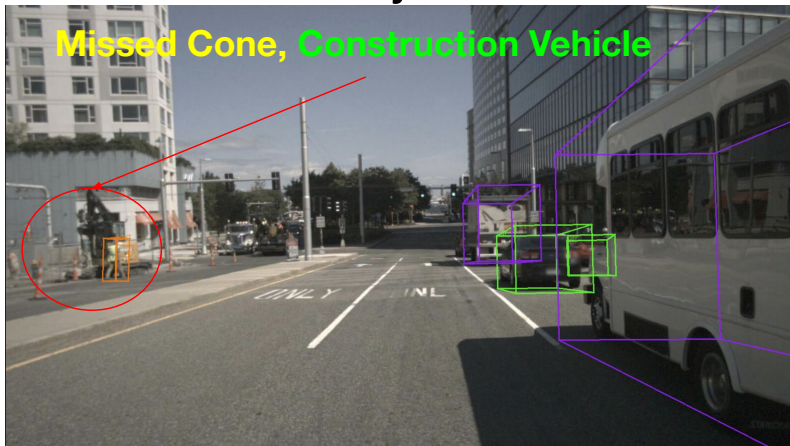
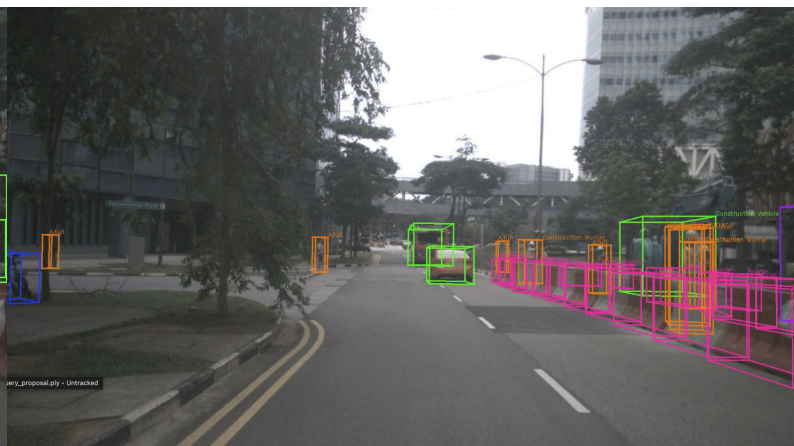
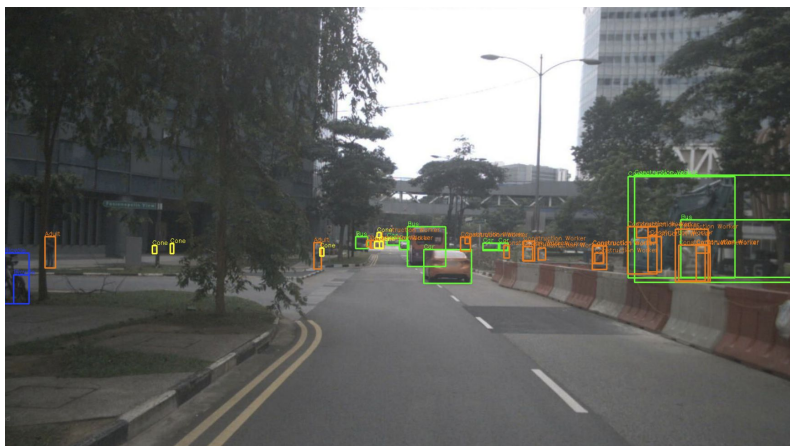


Figure 3: [nuScenes] Qualitative example #3. In this example, the LiDAR-only model fails to detect a cone and a construction vehicle, while FOMO-3D is able to detect and classify them successfully.

OWL 2D Detection

Ground-Truth Labels



LiDAR-Only 3D Detection

FOMO-3D 3D Detection

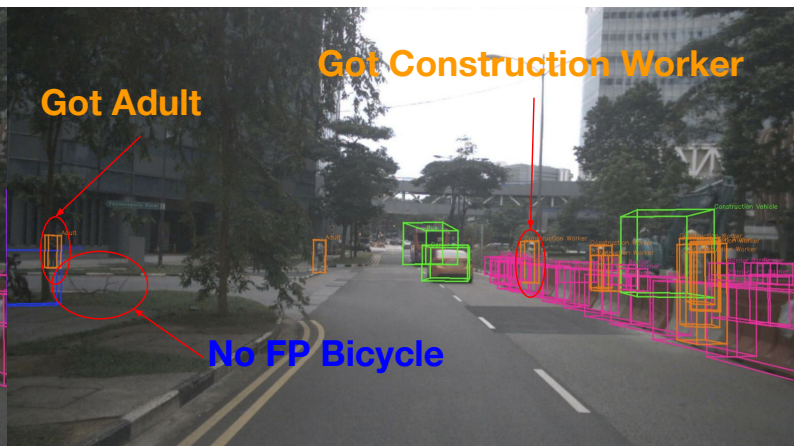
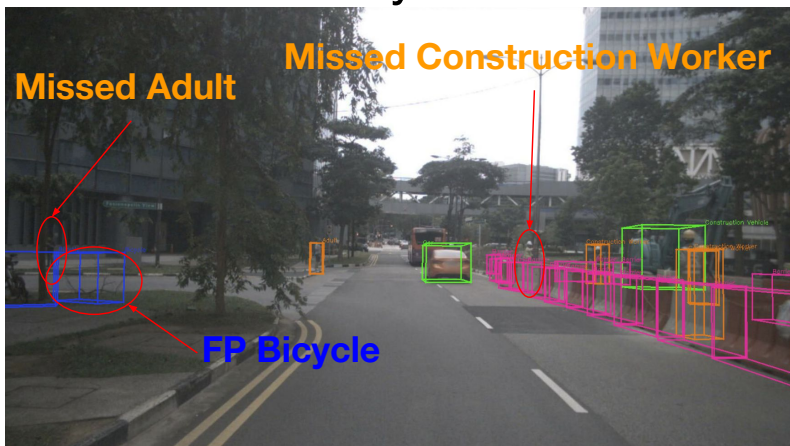


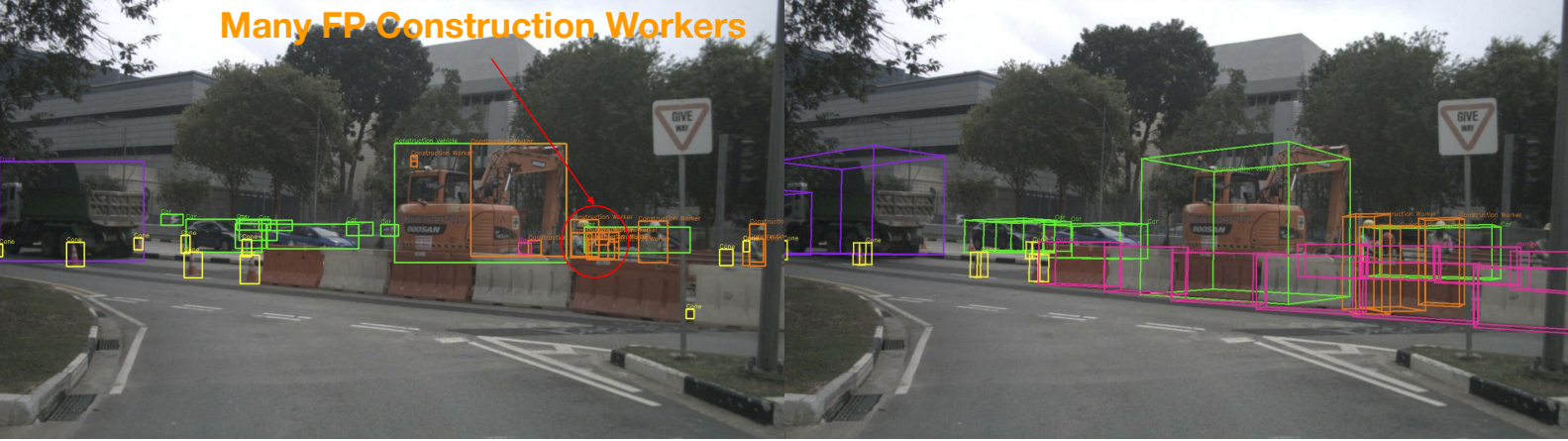
Figure 4: [nuScenes] Qualitative example #4. In this example, the LiDAR-only model fails to detect an adult and a construction worker and also outputs a false positive bicycle, while FOMO-3D is able to detect them successfully while rejecting the false positive.



OWL 2D Detection

Ground-Truth Labels

Many FP Construction Workers



LiDAR-Only 3D Detection

FOMO-3D 3D Detection

Many FP Construction Workers

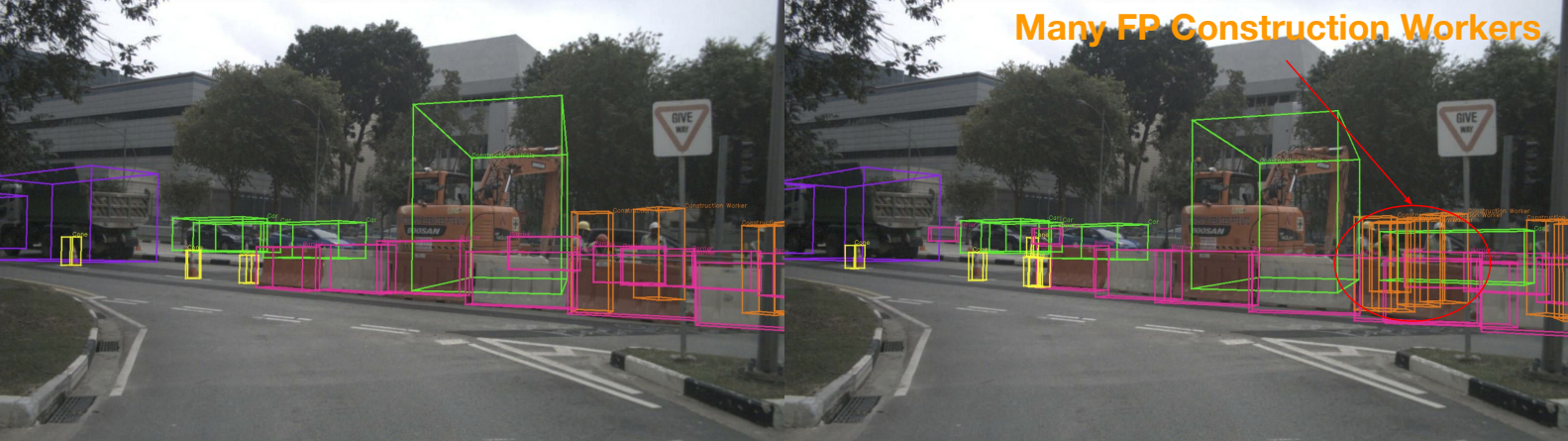


Figure 5: [nuScenes] Qualitative example #5. This example shows a failure case of OWL and FOMO-3D. OWL produces many duplicated construction workers. While FOMO-3D lifts them to the correct construction worker size, it fails to reject some duplicated false positives in the final detections. This is a challenging case because these false positives are adjacent to true positives in the image. False positives can be generated in the camera proposal stage directly from OWL proposals. In the refinement stage, many proposals which project onto the same image region could pick up features describing construction workers, and multiple proposals can attend to the same image features to generate these false positives. Designing a more optimal false positive and duplicate removal method is a future direction to improve FOMO-3D.

## OWL 2D Detection

## Ground-Truth Labels



## LiDAR-Only 3D Detection

## FOMO-3D 3D Detection

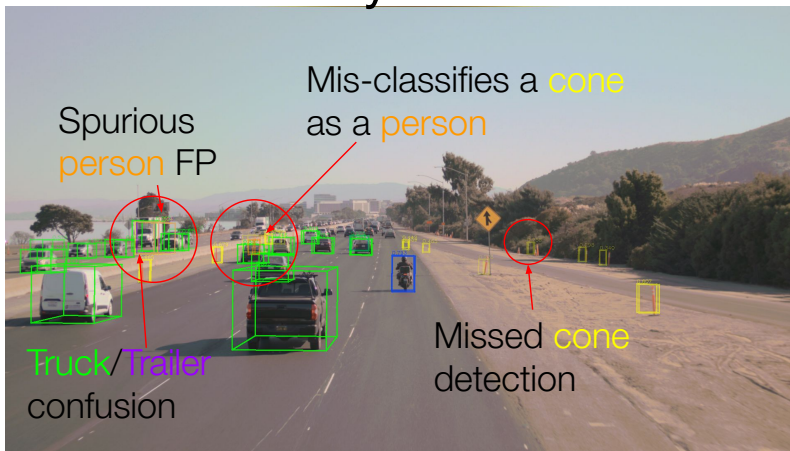
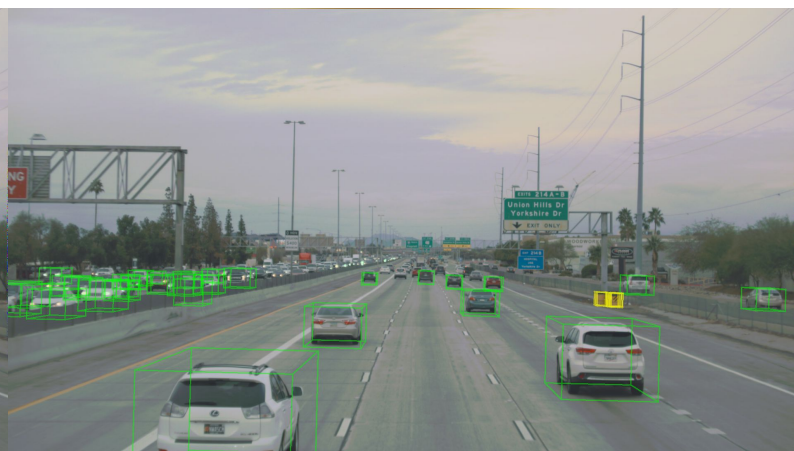


Figure 6: **[Highway]** Qualitative example #1. **Green** denotes vehicle. **Purple** denotes towed object. **Blue** denotes cyclists. **Orange** denotes person. **Yellow** denotes cone. Opacity of the boxes reflects detection confidence (higher confidence corresponds to more solid lines). FP=False Positive. FOMO-3D is able to correct various errors in OWL and LiDAR-only detections, but still mis-classifies a truck as trailer.



OWL 2D Detection

Ground-Truth Labels



LiDAR-Only 3D Detection

FOMO-3D 3D Detection

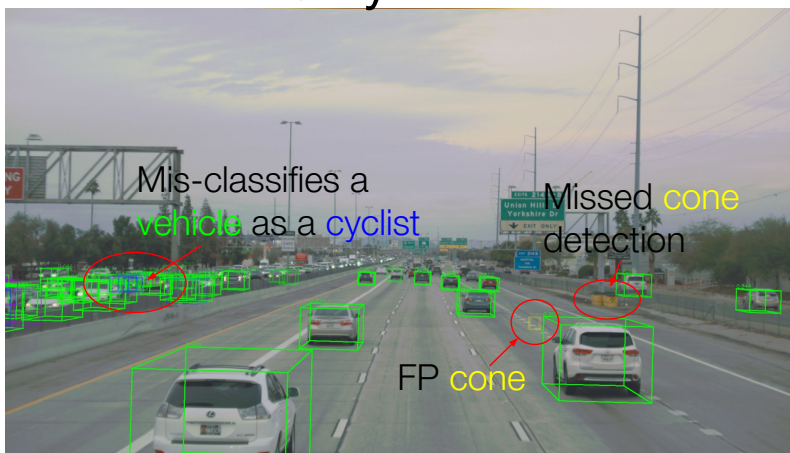
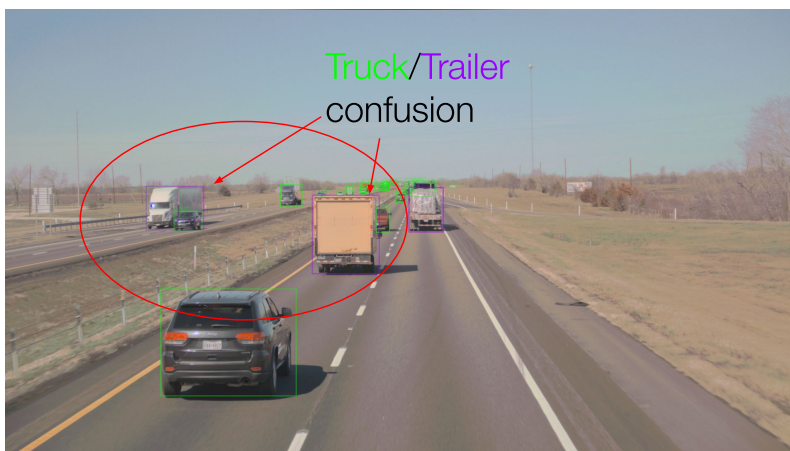


Figure 7: **[Highway]** Qualitative example #2. **Green** denotes vehicle. **Purple** denotes towed object. **Blue** denotes cyclists. **Orange** denotes person. **Yellow** denotes cone. Opacity of the boxes reflects detection confidence (higher confidence corresponds to more solid lines). FP=False Positive. This example shows a very dense traffic scene on the highway. OWL has impressive zero-shot 2D detections, but it also outputs a few false positives. LiDAR-only model mis-classifies a vehicle as a cyclist, and also draws a false positive cone. FOMO-3D is able to correctly classify in most cases without having many false positives. All three detectors miss two cones on the right.

## OWL 2D Detection

## Ground-Truth Labels



## LiDAR-Only 3D Detection

## FOMO-3D 3D Detection

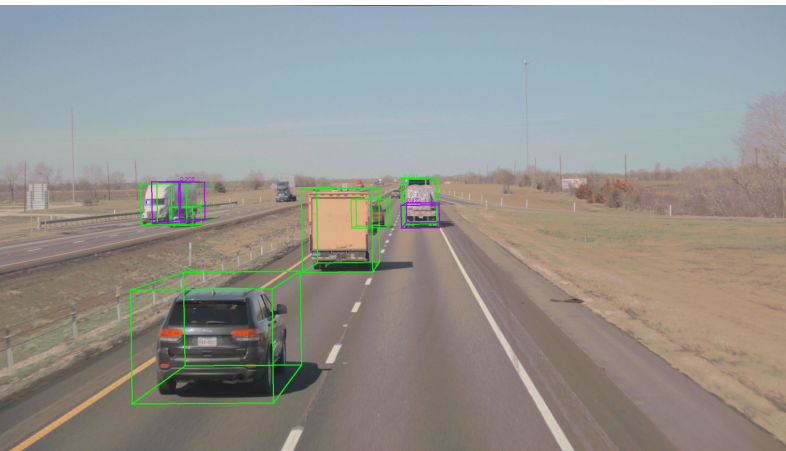
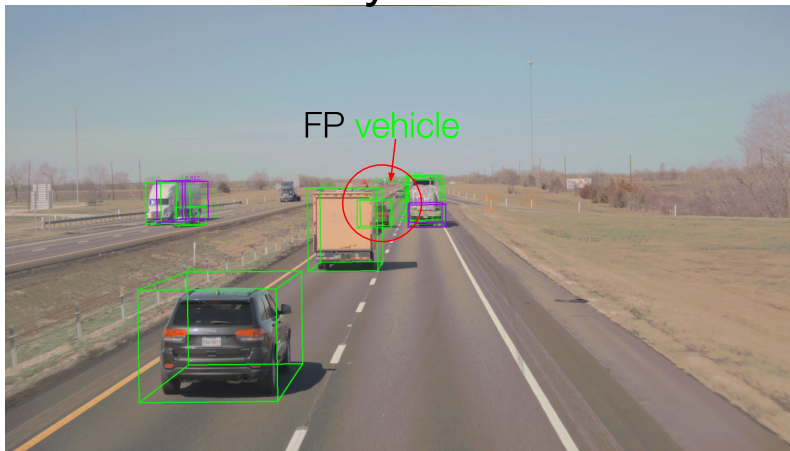


Figure 8: **[Highway]** Qualitative example #3. **Green** denotes vehicle. **Purple** denotes towed object. **Blue** denotes cyclists. **Orange** denotes person. **Yellow** denotes cone. Opacity of the boxes reflects detection confidence (higher confidence corresponds to more solid lines). FP=False Positive. Unlike OWL, FOMO-3D is able to utilize 3D LiDAR information and the training data to distinguish between vehicles and towed objects. FOMO-3D is able to effectively fuse 3D and 2D information and learn from the training data. It also outputs fewer false positives compared to the LiDAR-only model.

OWL 2D Detection



Ground-Truth Labels



LiDAR-Only 3D Detection



FOMO-3D 3D Detection



Figure 9: **[Highway]** Qualitative example #4. **Green** denotes vehicle. **Purple** denotes towed object. **Blue** denotes cyclists. **Orange** denotes person. **Yellow** denotes cone. Opacity of the boxes reflects detection confidence (higher confidence corresponds to more solid lines). FP=False Positive. In this example, the LiDAR-only model misses the person on the right, while OWL is able to detect it. With effective multi-modal fusion, FOMO-3D successfully detects the person.



## References

- [1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [2] S. Casas, B. Agro, J. Mao, T. Gilles, A. Cui, T. Li, and R. Urtasun. Detra: A unified model for object detection and trajectory forecasting. In *European Conference on Computer Vision*, pages 326–342. Springer, 2024.
- [3] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [4] Z. Xia, X. Pan, S. Song, L. E. Li, and G. Huang. Vision transformer with deformable attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4794–4803, June 2022.
- [5] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, page 213–229, Berlin, Heidelberg, 2020. Springer-Verlag. ISBN 978-3-030-58451-1. doi:10.1007/978-3-030-58452-8\_13. URL [https://doi.org/10.1007/978-3-030-58452-8\\_13](https://doi.org/10.1007/978-3-030-58452-8_13).
- [6] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. Generalized intersection over union. June 2019.
- [7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.
- [8] M. Minderer, A. A. Gritsenko, and N. Houlsby. Scaling open-vocabulary object detection. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=mQPNcBWjGc>.
- [9] N. Peri, A. Dave, D. Ramanan, and S. Kong. Towards long tailed 3d detection. *CoRL*, 2022.
- [10] Y. Ma, N. Peri, S. Wei, A. Dave, W. Hua, Y. Li, D. Ramanan, and S. Kong. Long-tailed 3d detection via multi-modal fusion, 2024. URL <https://arxiv.org/abs/2312.10986>.
- [11] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. Rus, and S. Han. Bevfusion: Multi-task multi-sensor fusion with unified bird’s-eye view representation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.
- [12] M. Hu, W. Yin, C. Zhang, Z. Cai, X. Long, H. Chen, K. Wang, G. Yu, C. Shen, and S. Shen. Metric3d v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation. 2024.