AmodalBox: Automated 3D Bounding Box Annotation under Partial Visibility

In-Jae Lee[1] **Namtae Kim**[1] **Robert Exley**[2] **Kwonyoung Ryu**[3] **Jaesik Park**[1]
[1]Seoul National University [2]University of Pittsburgh [3]POSTECH

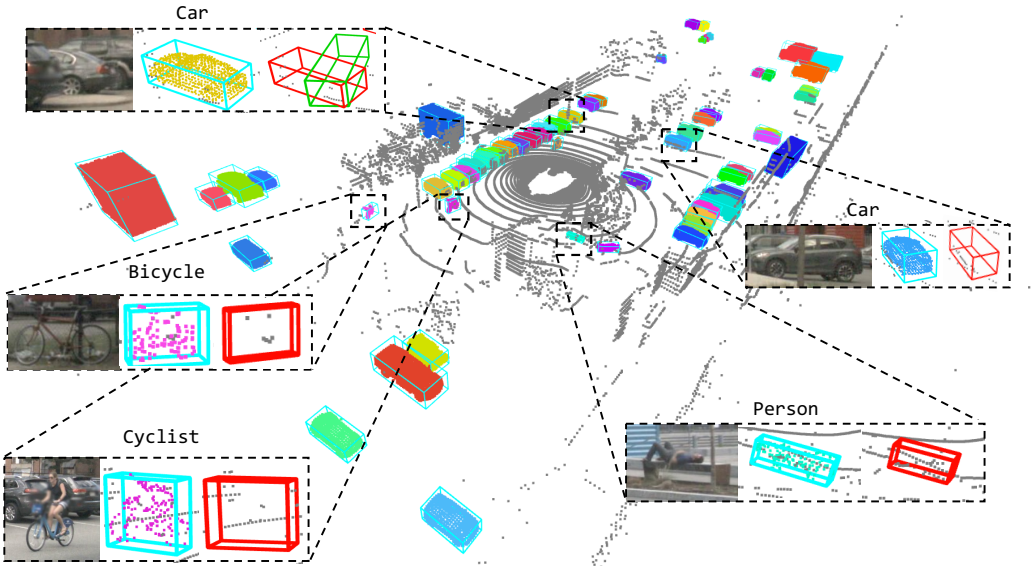


Figure 1: **Annotation from AmodalBox.** For diverse classes in the nuScenes dataset [1], Baseline often struggles under partial visibility, either missing objects or producing mislocalized boxes. In contrast, AmodalBox captures more complete object extents and produces spatially coherent annotations, while human annotations are shown as reference boxes. Best viewed in color and zoomed in.

Abstract: Recent progress in vision foundation models (VFMs) has made automated open-vocabulary 3D bounding box annotation increasingly feasible by jointly leveraging LiDAR point clouds and images. However, partial observations remain a major challenge in sparse driving scenes, where visible LiDAR points often cover only a small fraction of the object's surface and provide insufficient cues for estimating its full spatial extent. As a result, existing methods often produce mislocalized boxes or leave objects unlabeled. To address this limitation, we propose **AmodalBox**, an automated annotation framework that leverages image-generated 3D assets as amodal geometric priors for partially observed objects. By establishing correspondences between input images and rendered views of 3D assets, AmodalBox robustly aligns canonical amodal geometry with partially observed world-space LiDAR point clouds. To improve temporal consistency and reduce computational cost, AmodalBox generates a single asset per instance, estimates its metric scale once at a single frame and propagates it across frames for dynamic instances, and aggregates multi-frame observations for static instances. 3D object detectors trained on annotations from AmodalBox improve performance over strong baselines by 14.1 mAP on the nuScenes dataset and by 12.45 AP at IoU 0.5 for the vehicle class on the Waymo Open Dataset (WOD).

Keywords: Automated Annotation, 3D Object Detection, Multi-modal Fusion, Autonomous Driving

10th Conference on Robot Learning (CoRL 2026), Austin TX, USA.

1 Introduction

3D object detection is a fundamental capability for robotics and autonomous driving. While closed-set 3D detectors [2–5] have achieved strong performance on predefined categories, real-world scenes contain previously unseen objects that may still affect downstream prediction and planning. This motivates open-vocabulary 3D perception [6–9], which aims to recognize and localize object categories beyond a fixed label space [10].

Despite this progress, scaling open-vocabulary 3D detection remains fundamentally limited by the lack of large-scale 3D annotations for diverse object categories. Existing 3D detectors require instance-level bounding boxes that accurately specify object category, position, orientation, and spatial extent. However, obtaining such annotations is expensive and labor-intensive [11, 12]. This bottleneck becomes even more severe at scale, where annotations must cover a much wider diversity of object shapes, sizes, and appearances. Therefore, automated annotation provides a practical path toward scaling open-vocabulary 3D perception.

To alleviate the annotation bottleneck, earlier unsupervised methods [13–15] refine noisy initial pseudo-labels through heuristic scoring and bounding box filtering [6, 7, 13, 16] or computationally expensive self-training [15, 16]. More recent methods [6, 10, 17] leverage VFMs [18–22] to provide semantic information to LiDAR point clouds. However, partial observations remain difficult to handle, since visible evidence alone provides only incomplete constraints on the object's full spatial extent. Existing methods mitigate this issue through temporal refinement, size priors, or noisy point cloud filtering, but these strategies mainly refine boxes within the space supported by observed points. As a result, they may still produce mislocalized boxes or false negatives, as illustrated in Fig. 1. These noisy annotations can in turn degrade 3D object detector training, leading to weaker supervision for occluded objects. This motivates us to propose an annotation pipeline that uses visible observations as alignment cues while leveraging amodal object geometry to infer the full object extent beyond what is directly observed.

In this paper, we propose **AmodalBox**, a novel automated 3D bounding box annotation pipeline for handling partial visibility in real-world driving scenes. Our method aligns partial world-space LiDAR observations with an amodal 3D asset generated in canonical space by an image-to-3D generation model [23]. To guide this 3D alignment, we leverage correspondences obtained from 2D matching [24] between the real image and rendered views of the generated asset. AmodalBox further accounts for instance motion during alignment. For dynamic instances, our method estimates the alignment once at a single anchor frame and propagates it across the sequence, while for static instances, it aggregates LiDAR point clouds from multiple frames before alignment. As a result, AmodalBox improves 3D bounding box localization accuracy and yields more robust alignment under occlusion. Through experiments on the nuScenes [1] dataset and WOD [25], we show that leveraging image-generated amodal geometry yields more reliable 3D annotations and consistently improves downstream 3D object detection performance.

Our contributions are:

- We propose AmodalBox, an open-vocabulary automated 3D bounding box annotation pipeline for driving scenes that explicitly addresses partial visibility and occlusion.
- To tackle partial visibility, we introduce an amodal 3D alignment strategy that leverages image-generated object geometry and correspondences between real images and rendered views of the generated 3D asset to align partial world-space LiDAR observations with canonical 3D assets.
- We further incorporate motion-aware temporal reasoning, using multi-frame aggregation for static instances and anchor-frame propagation for moving instances to improve annotation consistency across frames.
- Training 3D object detectors on annotations from AmodalBox outperforms strong automated annotation baselines by 14.1 mAP on the nuScenes dataset [1] and by 12.45 AP for the vehicle class at IoU 0.5 on WOD [25].

2 Related Work

2.1 Open-vocabulary 3D Object Detection

OpenSight [8] and Find n' Propagate [7] lift 2D proposals from open-vocabulary 2D models into 3D space to generate pseudo 3D boxes, and further improve them through geometric search, alignment, or propagation. OVM3D-Det [26] performs such lifting from a single image, fitting boxes to pseudo-LiDAR obtained from a monocular depth model with category size priors to resolve the scale ambiguity. UP-VL [9] extends motion-based auto-labeling [27] with vision-language cues, producing semantically aligned amodal boxes and distilling point-wise semantic features into a 3D detector. OV-SCAN [6] improves novel object discovery and cross-modal alignment by optimizing 3D box search, filtering semantically inconsistent 3D–2D pairs, and applying a hierarchical alignment head. AnnofreeOD [28] models per-face box uncertainty and learns from noisy pseudo labels through box augmentation and noise-aware regression. FOMO-3D [29] refines 3D boxes from LiDAR and camera branches in BEV space using attention. Finally, OpenBox [10] performs cross-modal instance alignment and generates adaptive boxes according to the physical type of each instance.

2.2 Unsupervised 3D Object Detection

Prior work on unsupervised 3D object detection typically bootstraps pseudo labels from LiDAR geometry, motion cues, or temporal consistency. LiDAR-centric methods such as MODEST [16], OYSTER [14], and CPD [13] rely on clustering, temporal refinement, or prototype-based regularization to improve noisy boxes. Multi-modal extensions further incorporate image cues. MI-UP [27], LiSe [15], and UNION [17] combine motion, appearance, or cross-modal cues to improve pseudo-label quality, while CM3D [30] exploits image-based foundation models to generate pseudo 3D boxes for pre-training. PLOT [31] brings temporal aggregation to monocular video, tracking object points across frames and registering them into pseudo-LiDAR for both static and dynamic objects. Despite these advances, most existing methods still derive supervision primarily from visible evidence in images and sparse point clouds, and then refine it through clustering, propagation, or regularization. In contrast, our method introduces an explicit amodal shape prior from image-to-3D generation, making the annotation process less dependent on incomplete observed LiDAR geometry.

2.3 Automated Annotation with 3D Shape Priors

Prior works generate object-level 3D annotations by fitting predefined shape priors to visual and geometric observations. CAD-based methods align retrieved object models with RGB-D scans or RGB videos to obtain object-level shape and pose annotations [32–34], while template-based methods exploit category-specific deformable models for shape-aware annotation [35]. Other methods use implicit or canonical shape priors to recover object pose, shape, and 3D extent, where NOCS [36]-based methods rely on canonical coordinate prediction [37, 38], while SDF-based methods optimize object geometry through differentiable rendering [39, 40]. Segment, Lift and Fit [41] further uses SAM masks from 2D prompts and lifts them into 3D through SDF-based shape fitting with LiDAR and mask projection consistency. Although these methods reduce the need for manual 3D annotations, they rely on predefined CAD repositories, category-specific templates, or SDF/NOCS shape spaces. More recently, LabelAny3D [42] replaces such predefined priors with a generated mesh, whose pose is recovered by PnP from image-to-render matches and whose metric scale is obtained from monocular depth. AmodalBox also uses a generated asset [23, 43] as the shape prior, but lifts the matches to 3D-3D correspondences with LiDAR points, solves scale, yaw, and translation jointly in metric space, and propagates the alignment over time.

3 AmodalBox

We propose AmodalBox, an automated 3D bounding box annotation framework for partially visible objects, as illustrated in Fig. 2. AmodalBox aligns partial world-space LiDAR observations with a canonical amodal 3D asset generated from images. To bridge the two spaces, we use feature

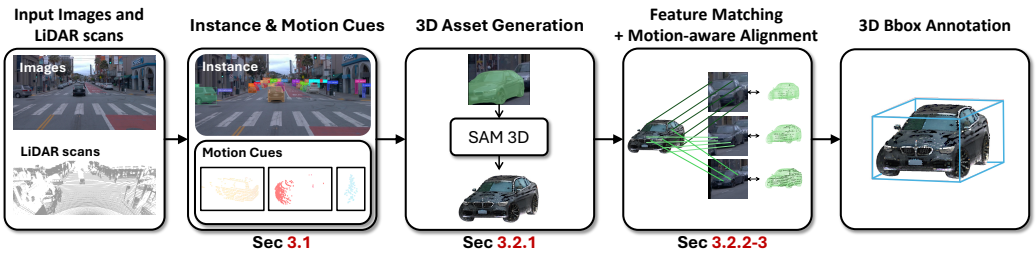


Figure 2: **Overview of AmodalBox**. Given time-synchronized images and LiDAR scans, we first extract instance-level cues and estimate the motion state of each instance using VFMs [18, 20], scene flow [44], and PP scores [16]. We then align the amodal 3D prior [23] with the partial LiDAR point cloud across their coordinate spaces using correspondences obtained from image-level matching. Finally, we generate a 3D bounding box annotation from the aligned amodal asset.

matching between real images and rendered asset views, and perform motion-aware alignment for static and dynamic instances. We first introduce the preliminaries in Sec. 3.1. Sec. 3.2 then presents our cross-modal spatial alignment strategy.

3.1 Preliminaries

Instance, Shape, and Motion Cues Following [10], AmodalBox considers both instance identity and motion state. To extract instance-level cues, AmodalBox processes each input image I_t^n with an open-vocabulary 2D detector f [20] and a trackable open-vocabulary instance segmentation model h [18], using a set of text queries $\mathcal{T}$. We denote the resulting 2D instance cues at frame t and view n by

$$\mathcal{S}_t^n = h(f(I_t^n, \mathcal{T})), \tag{1}$$

where $\mathcal{S}_t^n$ includes the 2D boxes B_t^n, semantic labels C_t^n, instance masks M_t^n, and temporally consistent instance IDs for the corresponding view. Using the calibration matrix between the camera and LiDAR, we gather LiDAR points within each mask to obtain a set of instance-level point clouds, defined as

$$\mathbb{P}_t^n = \{\mathcal{P}_t^{n,k}\}_{k=1}^{K_t^n}, \qquad \mathcal{P}_t^{n,k} \in \mathbb{R}^{M_t^{n,k} \times 3}, \tag{2}$$

where K_t^n denotes the number of instances in view n at frame t, and $M_t^{n,k}$ denotes the number of LiDAR points associated with the k-th instance. Instances observed from multiple cameras are associated through shared LiDAR points and class-wise BEV NMS (Appendix A.7); hereafter we omit the view index n.

In addition, as described in Sec. 3.2, we leverage scene flow [44] on nuScenes [1] and the persistence point score (PP score) [16], which measures point-level ephemerality, on WOD [25], respectively, to improve alignment accuracy. For each instance-level point cloud, we compute point-wise motion cues using the corresponding signal and assign its motion state by majority voting over the associated points.

3.2 Cross-Modal Spatial Alignment

3D Asset Generation Partial visibility makes it difficult to recover the full spatial extent of an object from visible observations alone, often resulting in mislocalized 3D bounding boxes. To overcome this limitation, we leverage 3D Gaussians [45] as an amodal 3D asset using an image-to-3D generation model ψ [23]. In autonomous driving datasets, however, a scene often contains many object instances, and repeatedly using all image observations of the same instance across different frames is computationally expensive and can also produce inconsistent shapes. Therefore, for each instance k, we select the anchor image frame $I_{anchor,k}$ in which its mask has the largest area and feed the corresponding image-mask pair into ψ. This yields an amodal 3D shape in canonical space, represented as a set of 3D Gaussians defined as:

$$\mathcal{G}^k = \psi(I_{anchor,k}, M_{anchor,k}). \tag{3}$$

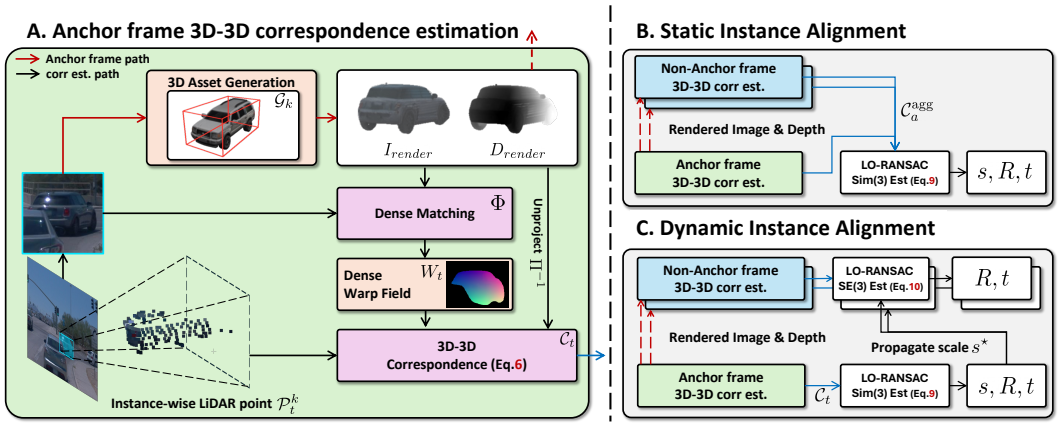


Figure 3: **Motion-aware Alignment.** (A) We generate a 3D asset from the anchor-frame image using [23], render I_{render} and D_{render}, and obtain 3D–3D correspondences via image matching and depth unprojection. (B) Using the anchor-rendered I_{render} and D_{render}, we aggregate neighboring-frame correspondences and estimate the alignment (Eq. (9)). (C) We estimate the anchor scale $s^\star$ (Eq. (9)), propagate it across frames, and align each frame (Eq. (10)).

We then extract the Gaussian centers from $\mathcal{G}^k$ and use them as the amodal point representation, denoted by $A^k = \{\mu_j\}_{j=1}^J$, where $\mu_j \in \mathbb{R}^3$ is the center of the j-th Gaussian. Since A^k is defined in canonical space, we align it with the partial LiDAR point cloud $\mathcal{P}^k_{anchor}$ in world space, which captures only the visible geometry of the object. The problem is therefore formulated as estimating a relative Sim(3) transformation, including scale, rotation, and translation, that best aligns the two representations across modalities.

Feature Matching Directly aligning these partial observations with the canonical-space amodal shape A^k using only geometric registration, such as ICP [46], is prone to fall in local minima when the initialization is inaccurate, often resulting in mislocalized bounding boxes. Instead, we use image-space matching as a robust cue. The 3D Gaussian asset is rendered from candidate views [45], and we select the rendered view used for matching according to the view-selection procedure described in Appendix A.2. Visual correspondences between the selected rendered view and the real image provide alignment cues even when the LiDAR observation is sparse. We perform dense matching Φ [24] between the rendered image I_{render} and the corresponding real image I_t, producing a dense warp field from real-image pixels u_t^{real} to rendered-image pixels u_t^{render} as

$$W_t = \Phi(I_t, I_{\text{render}}), \qquad u_t^{\text{render}} = W_t(u_t^{\text{real}}). \tag{4}$$

Although W_t is dense, we do not lift all image pixels into 3D. Instead, the real-side matching candidates are restricted to projected LiDAR pixels. Specifically, each instance-level LiDAR point $\mathbf{x}_j \in \mathcal{P}_t^k$ is projected onto the real image as

$$u_{t,j}^{\text{real}} = \pi_t(\mathbf{x}_j), \tag{5}$$

where $\pi_t(\cdot)$ is the LiDAR-to-image projection. We then sample the dense warp map only at these projected LiDAR pixels and sample the corresponding rendered depth as

$$u_{t,j}^{\text{render}} = W_t\big(u_{t,j}^{\text{real}}\big), \; d_{t,j}^{\text{render}} = D_{\text{render}}\big(u_{t,j}^{\text{render}}\big), \; \mathbf{p}_{t,j}^{\text{amodal}} = \Pi_{\text{render}}^{-1}\big(u_{t,j}^{\text{render}}, d_{t,j}^{\text{render}}\big). \tag{6}$$

Thus, the partial 3D point is the original LiDAR point $\mathbf{x}_j$, while the amodal 3D point is obtained by unprojecting the matched rendered pixel with its rendered depth. After discarding invalid projections, warp samples, and rendered depths, the lifted 3D-3D correspondence set is $\mathcal{C}_t = \{(\mathbf{p}_{t,j}^{\text{amodal}}, \mathbf{x}_j)\}_{j=1}^{L_t}$.

Motion-aware Alignment Given the lifted correspondences $\mathcal{C}_t$, we estimate a transformation from the canonical amodal asset to the world-space partial observation of the target instance. In autonomous driving scenes, most object categories can be reasonably approximated by a single yaw angle, and their coarse physical size can be represented by category-level size priors following OpenBox [10].

Based on this, we formulate the alignment as a scale-constrained 5-DoF problem, consisting of scale $s > 0$, translation $\mathbf{t} \in \mathbb{R}^3$, and yaw rotation $R_{\text{yaw}} \in SO(3)$, so that each correspondence satisfies

$$\mathbf{p}_{t,j}^{\text{partial}} \approx s\, R_{\text{yaw}}(\theta)\, \mathbf{p}_{t,j}^{\text{amodal}} + \mathbf{t}. \tag{7}$$

We estimate this transformation robustly using LO-RANSAC [47], where a closed-form yaw-constrained Umeyama-style estimator is used as the minimal solver for hypothesis generation [48]. Details on the size prior and the solver are given in Appendix A.1.

AmodalBox treats static and dynamic instances differently, since the former benefit from multi-frame aggregation while the latter require temporally stable propagation from a single anchor alignment. For static instances, we match the anchor-rendered view I_{render} with the real image of every frame $t \in \mathcal{N}_a$ in which the instance is observed, where a is the anchor frame of the target instance. Using the ego poses, the matched partial 3D points from each frame are transformed into the anchor frame coordinate system and aggregated as

$$\mathcal{C}_a^{\text{agg}} = \bigcup_{t \in \mathcal{N}_a} \left\{ \left(\mathbf{p}_{t,j}^{\text{amodal}}, E_{t \to a} \mathbf{p}_{t,j}^{\text{partial}} \right) \right\}_{j=1}^{L_t}. \tag{8}$$

The final alignment for static instances is then estimated once from the aggregated correspondence set $\mathcal{C}_a^{\text{agg}}$. For dynamic instances, independently estimating a per-frame yaw-constrained similarity transform, including scale, yaw, and translation, leads to frame-to-frame scale jitter, since partial visibility and sparse LiDAR support can change substantially across frames. We therefore first estimate the metric scale at the anchor frame a together with its yaw and translation as

$$(s^\star, \theta_a^\star, \mathbf{t}_a^\star) = \arg \min_{s>0,\theta,\mathbf{t}} \sum_{j=1}^{L_a} \rho\Big(\Big\| \mathbf{p}_{a,j}^{\text{partial}} - \big(s R_{\text{yaw}}(\theta) \mathbf{p}_{a,j}^{\text{amodal}} + \mathbf{t} \big) \Big\|_2 \Big) + \lambda (s - s_0)^2, \tag{9}$$

where s_0 is the category-level scale prior obtained following OpenBox [10], and $\rho(\cdot)$ is the robust penalty induced by the RANSAC inlier criterion. We then keep $s^\star$ fixed for the remaining frames and update only the yaw and translation at frame τ as

$$(\theta_\tau, \mathbf{t}_\tau) = \arg \min_{\theta,\mathbf{t}} \sum_{j=1}^{L_\tau} \rho\Big(\Big\| \mathbf{p}_{\tau,j}^{\text{partial}} - \big(s^\star R_{\text{yaw}}(\theta) \mathbf{p}_{\tau,j}^{\text{amodal}} + \mathbf{t} \big) \Big\|_2 \Big). \tag{10}$$

The resulting alignment is written as

$$\mathbf{p}_{\tau,j}^{\text{partial}} \approx s^\star R_{\text{yaw}}(\theta_\tau) \mathbf{p}_{\tau,j}^{\text{amodal}} + \mathbf{t}_\tau. \tag{11}$$

We then apply the closeness-to-edge algorithm [49] to tightly fit a bounding box around the asset aligned in world space.

4 Experiments

4.1 Experimental Setup

Dataset & Implementation Details We evaluate on two real-world driving datasets, nuScenes [1] with 700/150 train/val scenes and 10 classes, and WOD [25] with 798/202 train/val scenes and 3 classes. We train and evaluate over the full 360-degree field of view on nuScenes [1], and only within the camera frustum on WOD [25], following [10]. Annotating the training split takes about 50 h on nuScenes and 156 h on WOD with 4 NVIDIA B200 GPUs (Appendix A.6). For the 3D object detector, we use TransFusion [50] and CenterPoint [51] on nuScenes [1], and Voxel R-CNN [52] on WOD [25]. TransFusion and Voxel R-CNN are implemented in OpenPCDet [53], and CenterPoint in MMDetection3D [54]. We refer readers to Appendix A.4 for additional training details.

4.2 Main Result

Following [9, 10], we assess AmodalBox under two scenarios. In Scenario 1, we train 3D object detectors with automated annotations and test them on the human-annotated validation set (Tables 1,

Table 1: **3D object-detection results on the nuScenes [1] validation set.** Ped., M.C., Barr., C.V., and Trail. denote pedestrian, motorcycle, barrier, construction vehicle, and trailer, respectively. † is trained on [50] without semantic alignment head in OV-SCAN [6]. **Bold** and underline indicate the best and the second-best results.

Method	mAP↑	NDS↑	Car	Ped.	Truck	M.C.	Bicycle	Cone	Bus	Barr.	C.V.	Trail.
UP-VL [9]	11.3	17.2	19.4	30.8	7.9	16.8	14.1	15.6	3.1	0.5	4.3	0.4
Find n' Prop. [7]	16.7	22.4	24.3	22.8	8.6	35.8	**34.7**	21.3	11.1	4.4	4.1	0.1
OpenSight [8]	22.9	23.3	25.3	52.5	11.6	30.4	26.1	25.6	5.1	42.2	8.7	0.8
AnnofreeOD [28]	22.4	26.4	50.2	51.2	17.1	27.1	13.4	17.1	26.1	9.6	2.2	0.0
OV-SCAN [6]	31.1	32.8	61.6	60.1	30.3	39.8	31.0	39.6	22.0	18.8	6.8	0.6
OV-SCAN† [6]	28.7	31.4	62.2	59.1	29.8	29.5	26.3	34.6	19.9	17.3	7.1	0.7
AmodalBox (Ours)	**42.8**	**37.6**	**68.4**	**63.2**	**38.5**	**44.9**	29.4	**49.3**	**51.4**	**60.4**	**15.3**	**7.2**

Table 2: **3D object-detection results on WOD [25].** All values denote AP_{3D}. * denotes models trained on the union of OpenBox [10] and AmodalBox annotations after removing duplicates with class-wise NMS. **Bold** and underline indicate the best and second-best results.

(a) Scenario 1

Method	Vehicle $IoU_{0.5}$ / $IoU_{0.7}$	Pedestrian $IoU_{0.3}$ / $IoU_{0.5}$	Cyclist $IoU_{0.3}$ / $IoU_{0.5}$
CPD [13]	30.30 / 20.90	14.28 / 11.22	3.47 / 3.08
OpenBox [10]	70.49 / 32.41	57.95 / 17.11	**20.81** / 2.15
AmodalBox	82.94 / 36.16	55.92 / 6.41	12.11 / 9.33
AmodalBox*	**84.37 / 47.88**	**65.18 / 20.66**	13.16 / **9.63**

(b) Scenario 2

Method	Vehicle $IoU_{0.5}$ / $IoU_{0.7}$	Pedestrian $IoU_{0.3}$ / $IoU_{0.5}$	Cyclist $IoU_{0.3}$ / $IoU_{0.5}$
OpenBox [10]	26.38 / 5.68	38.53 / **8.49**	8.75 / 0.24
AmodalBox	**42.19** / **9.21**	26.96 / 1.66	9.10 / **2.97**
AmodalBox*	32.73 / 7.76	**40.74** / 8.43	**10.17** / 2.79

2a and 3), and in Scenario 2, we directly compare the annotations with human labels on the training set (Tables 2b, 4 and 5).

Results on nuScenes As shown in Table 1, AmodalBox substantially improves both mAP and NDS over prior methods, without additional strategies such as data augmentation [7, 8, 28] or detector architecture modification [6, 9], and improves AP for 9 out of 10 classes. We attribute this broad improvement to the amodal shape priors used for box generation. Under partial visibility, object points are often sparse or fragmented, so clustering-based post-processing [55, 56] either misses objects or fits boxes only to their visible regions, whereas AmodalBox generates boxes from the recovered amodal extent even when the visible points cover only a small portion of the object. The large gains on bus, construction vehicle, and trailer support this explanation, as these large objects are the most affected by visible-only box fitting under partial observation. The only exception is bicycle, which is known to be challenging in nuScenes [1] due to its thin structure and few LiDAR returns, making the partial observations less reliable for amodal alignment.

Comparison with OpenBox on nuScenes AmodalBox outperforms OpenBox [10] by a large margin on vehicles (+18.6 AP) and cyclists (+19.9 AP) but is lower on pedestrians. As analyzed in Appendix A.5, this gap stems almost entirely from extremely sparse instances. Among ground-truth pedestrians, 79% have at most 15 LiDAR points, where too few correspondences remain for alignment, whereas with at least 16 points our recall reaches 96–99%, and our overall precision is higher than that of OpenBox (94.9 vs. 78.8). A density-conditioned fallback that tightly fits the observed points [10] only for such sparse pedestrians lifts pedestrian AP from 46.7 to 62.0 with marginal changes to the other classes (AmodalBox‡).

Table 3: **Comparison with OpenBox [10] on nuScenes [1] under its 3-class protocol.** OpenBox numbers are as reported.

Method	Vehicle	Ped.	Cyclist
OpenBox [10]	40.9	**62.7**	5.2
AmodalBox	59.5	46.7	25.1
AmodalBox‡	**60.1**	62.0	**26.6**

Results on WOD As shown in Table 2, AmodalBox improves the Vehicle class by a large margin at both IoU thresholds and the Cyclist class at IoU 0.5, while it falls behind OpenBox [10] for Pedestrian and for Cyclist at IoU 0.3. Vehicles benefit most because their full length is often unobserved, whereas OpenBox extends an initial box and selects the orientation of static objects by IoU, which becomes

Table 4: **Ablation studies on 50 nuScenes [1] training scenes.** (a) Incremental ablation of the proposed modules. (b) Correspondence source and category size prior; "Vol. ratio" is the box volume over the class-prior volume (p99: 99th percentile; $>20\times$: % of boxes above $20\times$). Defaults in **bold**.

(a) Module ablation

	Partial LiDAR	3D Asset Gen & Matching	Motion-aware	mAP↑ / NDS↑
(1)	✓			3.57 / 5.76
(2)	✓	✓		24.62 / 26.30
(3)	✓	✓	✓	**31.40 / 31.13**

(b) Correspondence and scale prior

				Vol. ratio	
Correspondence	λ	mAP↑	NDS↑	p99↓	$>20\times$
ICP [46]	1	21.80	20.72	15.88	0.54
Dense matching	**1**	**31.40**	**31.13**	4.41	0.05
Dense matching	0	31.99	31.65	12.65	0.72
Dense matching	0.1	32.24	31.93	7.57	0.23
Dense matching	10	29.77	29.03	3.30	0.00

Table 5: **Robustness analysis on 50 nuScenes [1] training scenes.** (a) Each corruption is applied to the generated assets. Cov. is recall at a 2 m center distance. (b) Oracle states come from human annotation tracks. Flips are applied to the oracle states. D→S flips the given fraction of dynamic tracks to static (S→D the reverse).

(a) 3D asset quality sensitivity

Asset corruption	mAP↑	Cov.↑	mATE↓	mASE↓
Amodal half removed	19.3	41.2	0.826	0.708
Same-class swap	21.2	39.7	0.658	0.609
Appearance flattening	24.8	43.5	0.593	0.538
Anisotropic extent (30%)	29.0	50.7	0.597	0.597
None (ours)	**31.4**	**51.1**	**0.518**	**0.499**

(b) Motion-state sensitivity

Motion state	mAP↑	NDS↑
Estimated (ours)	31.40	31.13
Oracle	**33.50**	**32.01**
D→S 10/20/40%	32.79/31.49/30.04	31.78/31.09/30.55
S→D 10/20/40%	33.08/33.16/32.51	31.72/31.91/31.27

inaccurate under occlusion. Pedestrians are short and deformable, so they gain little from amodal shape recovery, and tightly fitting the observed points as in OpenBox can be more advantageous. The gain from mixing both annotations (AmodalBox*) and the effect of the density-conditioned fallback on nuScenes (Table 3) both support this interpretation.

4.3 Ablation and Robustness Analysis

We conduct the following analyses on 50 randomly sampled scenes from the nuScenes [1] training set, whose sparse LiDAR observations provide a challenging setting for annotation, and evaluate the generated annotations directly against human labels.

Module Ablation We analyze the effect of each proposed module by adding each module incrementally, as shown in Table 4a. Comparing (1) and (2), mAP and NDS increase by 21.1 and 20.5 points, respectively. This indicates that relying only on partial LiDAR point clouds tends to produce mis-sized boxes, whereas using amodal priors enables more reliable box annotation, improving both mAP and NDS, which reflects localization and size quality. Comparing (2) and (3), we also observe additional performance gains. We conjecture that repeatedly generating boxes for the same instance in each frame can cause box-size jittering, especially for dynamic objects, because the generated amodal priors may vary across frames. For static instances, aggregating points across frames makes the alignment more robust to outliers and improves box accuracy.

Correspondence and Scale Prior Table 4b examines the two design choices inside 3D asset generation and matching. Replacing dense matching with geometry-only ICP [46], initialized with the same yaw candidates and size prior, reduces mAP by 9.6 and inflates the tail of the box-volume distribution (p99 from 4.41 to 15.88), which is the local-minimum failure that dense matching avoids. The size prior in Eq. (9) does not drive the accuracy gain, since removing it ($\lambda = 0$) gives 32.0 mAP while over-weighting it ($\lambda = 10$) hurts, but it suppresses rare scale divergence that the center-distance-based nuScenes AP is blind to, reducing the fraction of boxes above $20\times$ the class volume from 0.72% to 0.05%, so we keep $\lambda = 1$ as a safeguard.

Robustness Table 5 examines how errors in the generated assets and in the estimated motion states propagate through the pipeline. Appearance corruptions, such as flattened textures or an asset

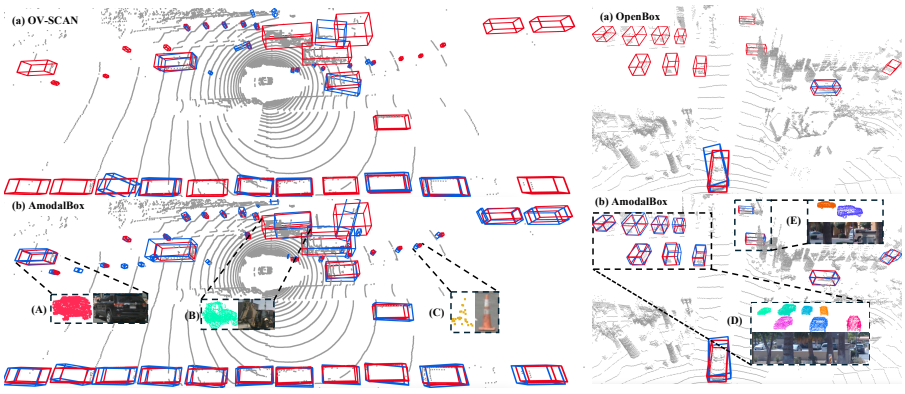


Figure 4: **Visualization of annotations** on nuScenes [1] (left) and WOD [25] (right). Red boxes are human annotations, blue boxes are automated annotations, and the colored (non-gray) points are the aligned amodal asset. Best viewed in color and zoomed in.

swapped with another instance of the same class, mainly reduce coverage, which indicates that dense matching rejects assets inconsistent with the image rather than propagating them into boxes. Geometric corruptions pass through this gate because matching verifies appearance rather than metric extent, so a 30% anisotropic distortion keeps coverage but raises mASE, and removing half of the amodal geometry is the most harmful case since it discards the unobserved extent that the method relies on. Motion-state errors show a similar asymmetry, as misclassifying dynamic tracks as static costs 3.5 mAP at a 40% error rate whereas the reverse costs only 1.0, because the static branch aggregates the correspondences of a moving object across frames.

4.4 Qualitative Result

Fig. 4 compares AmodalBox with OV-SCAN [6] on nuScenes [1] and with OpenBox [10] on WOD [25]. On nuScenes [1], AmodalBox achieves higher recall and more accurate box localization than OV-SCAN [6]. In (A) and (B), OV-SCAN [6] fails to produce boxes due to its selective alignment strategy, whereas AmodalBox annotates these objects by recovering their amodal shapes. In (C), AmodalBox correctly annotates the static traffic cone by explicitly distinguishing the instance motion state. On WOD [25], in (D) and (E), OpenBox [10] produces no annotations despite the presence of points, because the points are removed during its context-aware refinement, whereas AmodalBox first aligns an amodal asset to the observed point cloud and annotates these objects.

5 Conclusion

We proposed AmodalBox, an automated open-vocabulary 3D bounding box annotation pipeline that addresses partial visibility in driving scenes by aligning partial world-space LiDAR observations with amodal 3D assets generated from images in a motion-aware manner. Experiments on two datasets demonstrate that AmodalBox operates more effectively than prior methods under partial visibility, and our ablation and robustness analyses examine the effect of each module and the propagation of errors across stages. We believe that AmodalBox will advance research toward more robust 3D annotation pipelines.

Limitations Our method has several limitations. Since AmodalBox relies on VFMs outputs, reflected objects on glass surfaces introduce false positives. In addition, adverse weather can degrade 2D matching and lead to incorrect 3D alignment, while inaccurate scene flow may misclassify dynamic objects as static, causing false positives to persist across frames. Finally, small, thin, or deformable objects with few LiDAR returns benefit little from amodal alignment.

Acknowledgments

This work was supported by IITP grant (RS-2026-25517417: Development of Compression, Reconstruction, and Rendering Technologies for Free-viewpoint Media (30%), RS-2025-02303703: Realworld Multi-space Fusion and 6DoF Free-viewpoint Immersive Visualization for Extended Reality (30%), RS-2025-25442338: AI Star Fellowship Support Program (Seoul National University) (35%), RS-2021-II211343: AI Graduate School Program (Seoul National University) (5%)) funded by the Korea government (MSIT).

References

[1] 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. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[2] 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 *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[3] I.-J. Lee, S. Hwang, Y. Kim, W. Kim, S. Kim, and D. Kum. CRAB: Camera-radar fusion for reducing depth ambiguity in backward projection based view transformation. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[4] Y. Kim, J. Shin, S. Kim, I.-J. Lee, J. W. Choi, and D. Kum. CRN: Camera radar net for accurate, robust, efficient 3D perception. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[5] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Y. Qiao, and J. Dai. BEVFormer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *Proc. European Conference on Computer Vision (ECCV)*, 2022.

[6] A. Chow, E. Riddell, Y. Wang, S. Sedwards, and K. Czarnecki. OV-SCAN: Semantically consistent alignment for novel object discovery in open-vocabulary 3D object detection. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[7] D. Etchegaray, Z. Huang, T. Harada, and Y. Luo. Find n' Propagate: Open vocabulary 3D object detection in urban scenes. In *Proc. European Conference on Computer Vision (ECCV)*, 2024.

[8] H. Zhang, J. Xu, T. Tang, H. Sun, X. Yu, Z. Huang, and K. Yu. OpenSight: A simple open-vocabulary framework for LiDAR-based object detection. In *Proc. European Conference on Computer Vision (ECCV)*, 2024.

[9] M. Najibi, J. Ji, Y. Zhou, C. R. Qi, X. Yan, S. Ettinger, and D. Anguelov. Unsupervised 3D perception with 2D vision-language distillation for autonomous driving. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[10] I.-J. Lee, M. Kim, K. Ryu, P. Musacchio, and J. Park. OpenBox: Annotate any bounding boxes in 3D. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[11] H. Zhang, H. Jiang, Q. Yao, Y. Sun, R. Zhang, H. Zhao, H. Li, H. Zhu, and Z. Yang. Detect anything 3D in the wild. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[12] W. Huang, J. Zhang, S. Li, T. Jia, J. Duan, Y. Cheng, J. Cho, M. Wallingford, R. Soraki, C. D. Kim, S. Liu, D. Clay, T. Anderson, W. Han, A. Farhadi, B. Hariharan, Z. Ren, and R. Krishna. WildDet3D: Scaling promptable 3D detection in the wild. *arXiv preprint arXiv:2604.08626*, 2026.

[13] H. Wu, S. Zhao, X. Huang, C. Wen, X. Li, and C. Wang. Commonsense prototype for outdoor unsupervised 3D object detection. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[14] L. Zhang, A. J. Yang, Y. Xiong, S. Casas, B. Yang, M. Ren, and R. Urtasun. Towards unsupervised object detection from LiDAR point clouds. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[15] R. Zhang, H. Zhang, H. Yu, and Z. Zheng. Approaching outside: Scaling unsupervised 3D object detection from 2D scene. In *Proc. European Conference on Computer Vision (ECCV)*, 2024.

[16] Y. You, K. Luo, C. P. Phoo, W.-L. Chao, W. Sun, B. Hariharan, M. Campbell, and K. Q. Weinberger. Learning to detect mobile objects from LiDAR scans without labels. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[17] T. Lentsch, H. Caesar, and D. M. Gavrila. UNION: Unsupervised 3D object detection using object appearance-based pseudo-classes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[18] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer. SAM 2: Segment anything in images and videos. In *Proc. International Conference on Learning Representations (ICLR)*, 2025.

[19] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment anything. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[20] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, et al. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *Proc. European Conference on Computer Vision (ECCV)*, 2024.

[21] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen, X. Wang, X. Zhai, T. Kipf, and N. Houlsby. Simple open-vocabulary object detection. In *Proc. European Conference on Computer Vision (ECCV)*, 2022.

[22] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jégou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

[23] W. Feng, M. Wu, Z. Chen, C. Yang, H. Qin, Y. Li, X. Liu, G. Fan, Z. An, L. Huang, Y. Zhang, M. Magno, and Y. Xu. Fast-SAM3D: 3Dfy anything in images but faster. In *Proc. International Conference on Machine Learning (ICML)*, 2026.

[24] J. Edstedt, Q. Sun, G. Bökman, M. Wadenbäck, and M. Felsberg. RoMa: Robust dense feature matching. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[25] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[26] R. Huang, H. Zheng, Y. Wang, Z. Xia, M. Pavone, and G. Huang. Training an open-vocabulary monocular 3D detection model without 3D data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[27] M. Najibi, J. Ji, Y. Zhou, C. R. Qi, X. Yan, S. Ettinger, and D. Anguelov. Motion inspired unsupervised perception and prediction in autonomous driving. In *Proc. European Conference on Computer Vision (ECCV)*, 2022.

[28] B. Sun, Y. Liu, H. He, Y. Tian, and F.-Y. Wang. AnnofreeOD: Detecting all classes at low frame rates without human annotations. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[29] A. J. Yang, J. Tu, N. Dvornik, E. Li, and R. Urtasun. FOMO-3D: Using vision foundation models for long-tailed 3D object detection. In *Proc. Conference on Robot Learning (CoRL)*, 2025.

[30] M. Khurana, N. Peri, J. Hays, and D. Ramanan. Shelf-supervised cross-modal pre-training for 3D object detection. In *Proc. Conference on Robot Learning (CoRL)*, 2024.

[31] S. Lee, S. Aung, J. Choi, S. Kim, I.-J. Kim, and J. Cho. PLOT: Pseudo-labeling via object tracking for monocular 3D object detection. In *Proc. European Conference on Computer Vision (ECCV)*, 2026.

[32] A. Avetisyan, M. Dahnert, A. Dai, M. Savva, A. X. Chang, and M. Nießner. Scan2CAD: Learning CAD model alignment in RGB-D scans. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[33] S. Ainetter, S. Stekovic, F. Fraundorfer, and V. Lepetit. Automatically annotating indoor images with CAD models via RGB-D scans. In *Proc. IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023.

[34] K.-K. Maninis, S. Popov, M. Nießner, and V. Ferrari. CAD-Estate: Large-scale CAD model annotation in RGB videos. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[35] Z. Liu, D. Zhou, F. Lu, J. Fang, and L. Zhang. AutoShape: Real-time shape-aware monocular 3D object detection. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[36] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas. Normalized object coordinate space for category-level 6D object pose and size estimation. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[37] Z. Min, B. Zhuang, S. Schulter, B. Liu, E. Dunn, and M. Chandraker. NeurOCS: Neural NOCS supervision for monocular 3D object localization. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[38] A. Krishnan, A. Kundu, K.-K. Maninis, J. Hays, and M. Brown. OmniNOCS: A unified NOCS dataset and model for 3D lifting of 2D objects. In *Proc. European Conference on Computer Vision (ECCV)*, 2024.

[39] S. Zakharov, W. Kehl, A. Bhargava, and A. Gaidon. Autolabeling 3D objects with differentiable rendering of SDF shape priors. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[40] Z. Liu, H. Sakuma, and M. Okutomi. VSRD: Instance-aware volumetric silhouette rendering for weakly supervised 3D object detection. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[41] J. Li, T. Sun, Z. Wang, E. Xie, B. Feng, H. Zhang, Z. Yuan, K. Xu, J. Liu, and P. Luo. Segment, lift and fit: Automatic 3D shape labeling from 2D prompts. In *Proc. European Conference on Computer Vision (ECCV)*, 2024.

[42] J. Yao, R. M. Redoy, S. Elbaum, M. B. Dwyer, and Z. Cheng. LabelAny3D: Label any object 3D in the wild. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[43] SAM 3D Team, X. Chen, F.-J. Chu, P. Gleize, K. J. Liang, A. Sax, H. Tang, W. Wang, M. Guo, T. Hardin, X. Li, A. Lin, J. Liu, Z. Ma, A. Sagar, B. Song, X. Wang, J. Yang, B. Zhang, P. Dollár, G. Gkioxari, M. Feiszli, and J. Malik. SAM 3D: 3Dfy anything in images. *arXiv preprint arXiv:2511.16624*, 2025.

[44] Q. Zhang, X. Zhu, Y. Zhang, Y. Cai, O. Andersson, and P. Jensfelt. DeltaFlow: An efficient multi-frame scene flow estimation method. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[45] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.

[46] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[47] O. Chum and J. Matas. Locally optimized RANSAC. In *Pattern Recognition (DAGM)*, volume 2781 of *Lecture Notes in Computer Science*, pages 236–243. Springer, 2003.

[48] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.

[49] X. Zhang, W. Xu, C. Dong, and J. M. Dolan. Efficient L-shape fitting for vehicle detection using laser scanners. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[50] X. Bai, Z. Hu, X. Zhu, Q. Huang, Y. Chen, H. Fu, and C.-L. Tai. TransFusion: Robust LiDAR-camera fusion for 3D object detection with transformers. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[51] T. Yin, X. Zhou, and P. Krähenbühl. Center-based 3D object detection and tracking. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[52] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li. Voxel R-CNN: Towards high performance voxel-based 3D object detection. In *Proc. AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[53] OpenPCDet Development Team. OpenPCDet: An open-source toolbox for 3D object detection from point clouds. https://github.com/open-mmlab/OpenPCDet, 2020.

[54] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d, 2020.

[55] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. International Conference on Knowledge Discovery and Data Mining (KDD)*, 1996.

[56] R. J. G. B. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In *Proc. Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2013.

[57] OpenAI. ChatGPT. https://chat.openai.com/, 2026. Large language model, accessed June 5, 2026.

[58] Z. Teed and J. Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *Proc. European Conference on Computer Vision (ECCV)*, 2020.

A Appendix

A.1 Motion-aware Alignment

Problem Formulation Given the 3D-3D correspondence set $\mathcal{C}_t = \{(\mathbf{p}_{t,j}^{\text{amodal}}, \mathbf{p}_{t,j}^{\text{partial}})\}_{j=1}^{L_t}$ defined in Sec. 3.2, we estimate the transformation from the canonical amodal asset to the world-space partial observation. Since most object categories can be reasonably approximated with only a yaw rotation, we formulate the alignment as a yaw-constrained, scale-regularized 5-DoF problem:

$$\mathbf{p}_{t,j}^{\text{partial}} \approx s\, R_{\text{yaw}}(\theta)\, \mathbf{p}_{t,j}^{\text{amodal}} + \mathbf{t}, \tag{12}$$

where $s > 0$ is the scale, $R_{\text{yaw}}(\theta) \in SO(3)$ is the yaw-only rotation matrix, and $\mathbf{t} \in \mathbb{R}^3$ is the translation. To estimate the yaw-only rotation, we project the 3D correspondences onto the BEV axes and solve a 2D Procrustes problem.

Hypothesis Generation Let $\bar{\mathbf{p}}_j^{\text{amodal}} \in \mathbb{R}^2$ and $\bar{\mathbf{p}}_j^{\text{partial}} \in \mathbb{R}^2$ denote the corresponding BEV-plane coordinates of $\mathbf{p}_j^{\text{amodal}}$ and $\mathbf{p}_j^{\text{partial}}$, respectively. We first compute their centroids

$$\mu_{\text{amodal}} = \frac{1}{L_t}\sum_{j=1}^{L_t} \bar{\mathbf{p}}_{t,j}^{\text{amodal}}, \qquad \mu_{\text{partial}} = \frac{1}{L_t}\sum_{j=1}^{L_t} \bar{\mathbf{p}}_{t,j}^{\text{partial}}, \tag{13}$$

and the centered coordinates

$$\tilde{\mathbf{p}}_{t,j}^{\text{amodal}} = \bar{\mathbf{p}}_{t,j}^{\text{amodal}} - \mu_{\text{amodal}}, \qquad \tilde{\mathbf{p}}_{t,j}^{\text{partial}} = \bar{\mathbf{p}}_{t,j}^{\text{partial}} - \mu_{\text{partial}}. \tag{14}$$

The BEV cross-covariance matrix is then

$$C = \frac{1}{L_t}\sum_{j=1}^{L_t} \tilde{\mathbf{p}}_{t,j}^{\text{amodal}} \tilde{\mathbf{p}}_{t,j}^{\text{partial}\top}. \tag{15}$$

Using the singular value decomposition $C = U\Sigma V^\top$, the BEV rotation is given by

$$R_{\text{bev}} = V \begin{bmatrix} 1 & 0 \\ 0 & \det(VU^\top) \end{bmatrix} U^\top. \tag{16}$$

The BEV translation is computed as

$$\mathbf{t}_{\text{bev}} = \mu_{\text{partial}} - s\, R_{\text{bev}}\, \mu_{\text{amodal}}, \tag{17}$$

while the vertical translation is estimated robustly using the median

$$t_z = \text{median}_j \left(\left[\mathbf{p}_{t,j}^{\text{partial}}\right]_z - s\, \left[\mathbf{p}_{t,j}^{\text{amodal}}\right]_z \right). \tag{18}$$

Local Optimization To regularize the scale, we further introduce a class-dependent prior. Let $(l, w, h)_{\text{class}}$ denote the coarse class-size prior and let d_{src} be the diagonal length of the robust extent of the Gaussian centers in A^k. We define the prior scale as

$$d_{\text{src}} = \left\| \text{extent}_{q_{\text{low}}, q_{\text{high}}}(A^k) \right\|_2, \qquad d_{\text{prior}} = \|(l, w, h)_{\text{class}}\|_2, \tag{19}$$

$$s_0 = \frac{d_{\text{prior}}}{d_{\text{src}}}. \tag{20}$$

Instead of relying solely on the data-driven scale estimate, we combine it with the class prior:

$$s \approx \frac{s_{\text{data}} + \lambda s_0}{1 + \lambda}, \tag{21}$$

where λ controls the prior strength. We solve this problem using LO-RANSAC [47] with closed-form alignment estimation inspired by Umeyama [48]. Overall, this can be viewed as a robust approximation to

$$\min_{s,\theta,\mathbf{t}} \sum_j \rho\Big(\Big\| \mathbf{p}_{t,j}^{\text{partial}} - \big(s\, R_{\text{yaw}}(\theta)\, \mathbf{p}_{t,j}^{\text{amodal}} + \mathbf{t} \big) \Big\|_2 \Big) + \lambda (s - s_0)^2, \tag{22}$$

where $\rho(\cdot)$ is induced implicitly by the RANSAC inlier criterion.

Table 6: Training configurations for 3D object detectors.

Config	Voxel R-CNN [52]	TransFusion [50]
Optimizer	`AdamW`	`AdamOneCycle`
Base learning rate	$1e{-2}$	$1e{-3}$
Weight decay	$1e{-3}$	$1e{-2}$
Momentum	0.9	0.9
Momentum range	$[0.95, 0.85]$	$[0.9, 0.8052631]$
Learning rate decay	0.1	0.1
Learning rate clip	$1e{-7}$	$1e{-7}$
Gradient norm clip	10	35
Batch size	16	16
Epoch	20	20

A.2 Rendered View Selection

We select the rendered view by constructing a render bank around the initial object orientation predicted by [23]. Specifically, let θ_{initial} denote the initial yaw angle estimated by [23]. For each candidate, we apply an additional yaw offset $\Delta\theta$ around the vertical axis and render the asset at yaw $\theta_{\text{initial}} + \Delta\theta$. Each rendered candidate produces RGB-D images, which are matched to the real image using RoMa [24]. The resulting 2D correspondences are lifted to 3D to form 3D-3D correspondences and are evaluated through robust alignment. We select the candidate render view that yields the largest number of RANSAC inliers. For static objects, we use a narrower offset set, $\Delta\theta \in \{-30°, -15°, 0°, 15°, 30°\}$, and aggregate correspondences from multiple frames in the anchor ego frame before the final alignment. For dynamic objects, we use a wider offset set, $\Delta\theta \in \{-45°, -22.5°, 0°, 22.5°, 45°\}$, to account for larger pose uncertainty. After selecting the anchor render view, subsequent frames are aligned while keeping the anchor scale fixed. Only nearby yaw-offset bins are searched to preserve temporal consistency.

A.3 Category-level Size Priors

Following OpenBox [10], we used ChatGPT-5.5 [57] to estimate category-level 3D bounding box size priors. Specifically, for each object category, we queried the model with the following prompt and used the returned dimensions as approximate class-wise priors.

> **Prompt**: `What are the typical 3D bounding box dimensions for [class]?`
>
> **Response**: `The typical 3D bounding box size for [class] is approximately given as Length × Width × Height in meters, based on common autonomous driving datasets such as nuScenes, KITTI, and the Waymo Open Dataset.`

A.4 Implementation details

Training and 3D Object Detector Details For a fair comparison, we train Voxel R-CNN [52] on WOD [25], following [10, 13], and TransFusion [50] on nuScenes [1], following [6]. We use the default OpenPCDet [53] configurations shown in Table 6, without modifying any hyperparameters.
Automated Annotation Details The hyperparameters used for motion-aware alignment are listed in Table 7. In addition, for the mixed annotation with NMS used in Table 2b, we use the NMS implementation provided by OpenPCDet [53]. The NMS threshold is set to 0.01, and NMS is performed in a class-wise manner.

Table 7: Hyperparameters for yaw-constrained scale-regularized LO-RANSAC alignment.

Hyperparameter	Value	Description
q_{low}	0.05	Lower quantile for robust source extent
q_{high}	0.95	Upper quantile for robust source extent
λ	1.0	Strength of the class-size scale prior
Inlier threshold	0.3 m	RANSAC inlier distance threshold
Minimum inliers	10	Minimum inliers for accepting a hypothesis
Number of iterations	1000	Maximum LO-RANSAC iterations
Minimum iterations	1000	Minimum iterations before stopping
Hypotheses per step	64	Number of hypotheses evaluated per batch
LO iterations	5	Number of local optimization refinements
LO minimum inliers	10	Minimum inliers required for local optimization

Table 8: **Analysis of small, deformable, and thin objects on nuScenes [1].** (a) Quality of the raw AmodalBox annotations per super-class (Cov.: recall of human boxes at 2 m; Prec.: precision; ATE/ASE: translation and scale errors of matched boxes), next to the detector AP of Table 3. (b) Pedestrian recall at 2 m as a function of the number of LiDAR points inside the human box. "GT" is the share (%) of ground-truth pedestrians in each bin.

(a) Per-class annotation quality

Class	Cov.↑	Prec.↑	ATE↓	ASE↓	Det. AP↑
Vehicle	62.3	81.0	0.35	**0.25**	59.5
Pedestrian	47.8	**94.9**	**0.26**	0.40	46.7
Cyclist	60.7	82.3	0.31	0.48	25.1

(b) Pedestrian recall by LiDAR density

LiDAR points	GT (%)	AmodalBox	OpenBox [10]
1–5	46	15.6	**40.3**
6–15	33	63.0	**66.2**
16–40	12	**96.3**	74.3
>40	9	**98.6**	67.9
Precision	–	**94.9**	78.8

A.5 Small, Deformable, and Thin Objects

Table 8a evaluates the raw annotations per super-class on nuScenes. The two weaker classes fail in different ways. Pedestrians are recall-limited, in that the boxes we do produce are the most accurate of all classes (highest precision, lowest ATE), but coverage is the lowest. Table 8b locates the missing recall in extremely sparse instances, which make up 79% of the ground-truth pedestrians. With at most 15 LiDAR points, too few correspondences survive the lifting step for a reliable alignment, whereas with at least 16 points AmodalBox recovers 96–99% of the pedestrians, well above OpenBox [10]. Cyclists are instead scale-limited. Their coverage matches vehicles, but ASE is the largest, and a detector trained on them inherits the error, since the thin, non-convex bicycle-plus-rider geometry is poorly captured by a single generated asset.

Density-conditioned fallback Motivated by this analysis, AmodalBox‡ in Table 3 adds a single rule. For pedestrian instances with at most 15 LiDAR points, amodal alignment is skipped and the box is fitted tightly to the observed points with the category size prior, following [10]. This lifts pedestrian AP from 46.7 to 62.0 while leaving vehicle and cyclist AP essentially unchanged (59.5 to 60.1 and 25.1 to 26.6), so that AmodalBox‡ matches or exceeds OpenBox on all three classes.

A.6 Runtime Breakdown

Table 9 breaks down the annotation time of the training splits by stage. The 2D instance and motion cues are computed once per frame, and 3D asset generation and rendering are executed once per track at its anchor frame, so their cost is amortized over all frames of the track. The remaining matching, alignment, and box fitting are performed for every frame of every track and account for roughly two thirds of the total time. This stage is independent across instances and can be parallelized directly.

Table 9: **Annotation runtime breakdown** for the training splits on 4 NVIDIA B200 GPUs. "Unit" denotes the granularity at which each stage runs; asset generation is executed once per track rather than per frame.

Stage	Unit	nuScenes [1]	WOD [25]
2D instance & motion cues (Sec. 3.1)	frame	∼4 h	∼18 h
Asset generation & rendering	track	∼11 h	∼38 h
Matching, alignment & box fitting	track-frame	∼35 h	∼100 h
Total	split	∼50 h	∼156 h

Table 10: **Effect of cross-view association and duplicate removal on 50 nuScenes [1] training scenes.** "Box/frame" is the average number of annotated boxes per keyframe.

Cross-view association & NMS	mAP↑	NDS↑	Box/frame
Off	29.2	29.6	18.7
On (AmodalBox)	**31.4**	**31.1**	13.8

A.7 Cross-view Instance Association

Instance IDs produced by the tracker are consistent only within a camera view, whereas an object may be visible in two adjacent cameras simultaneously or move from one camera to another over time. We therefore first remap the per-camera track IDs to be unique within a scene. At each keyframe, tracks from different cameras are associated whenever their masks share a sufficient number of LiDAR points, and the associations are propagated over time, which covers both simultaneous multi-view observations and camera hand-offs. Residual duplicate boxes are removed by class-wise BEV NMS that keeps the box enclosing more LiDAR points. Table 10 shows that duplicate removal reduces the number of boxes by about a quarter (18.7 to 13.8 per keyframe) and improves mAP by 2.2 points.

A.8 Failure Cases

As discussed in Sec. 5, the limitations of AmodalBox can be broadly categorized into three cases. In Fig. 5, (A) shows a failure case caused by Grounding DINO [20], where a vehicle reflected on glass is incorrectly detected. Since the point clouds inside the falsely detected box correspond to a flat glass surface, which resembles the side surface of a vehicle, the amodal 3D asset can still be aligned with the LiDAR point cloud, resulting in false-positive boxes. One possible solution is to adopt context-aware refinement, as in OpenBox [10], to filter instance-level point clouds from the entire cluster. Another possible direction is to prompt background classes, as in UP-VL [9], and suppress alignment when they overlap with foreground instances.

In (B), the failure is caused by inaccurate scene flow or PP score [16] estimation, which leads to an incorrect prediction of the instance motion state. As shown by the orange box, a dynamic truck is temporarily misclassified as static, resulting in a false-positive box. To mitigate this issue, one could improve motion estimation by incorporating optical flow [58] at the image level.

(C) shows a failure case under adverse weather conditions, where 2D-2D matching incorrectly associates the rear wheel in the real image with the front wheel in the rendered image, as highlighted by the orange box. This erroneous correspondence propagates to incorrect 3D-3D alignment. A possible remedy is to estimate correspondence reliability using local feature similarity, geometric consistency, and reprojection error, and discard low confidence matches before lifting them to 3D.

A.9 Analysis for partial visibility

To further analyze the effectiveness of our method under partial visibility, we conduct modality-specific evaluations, as shown in Table 11 and Fig. 6. Since partial visibility is not explicitly defined

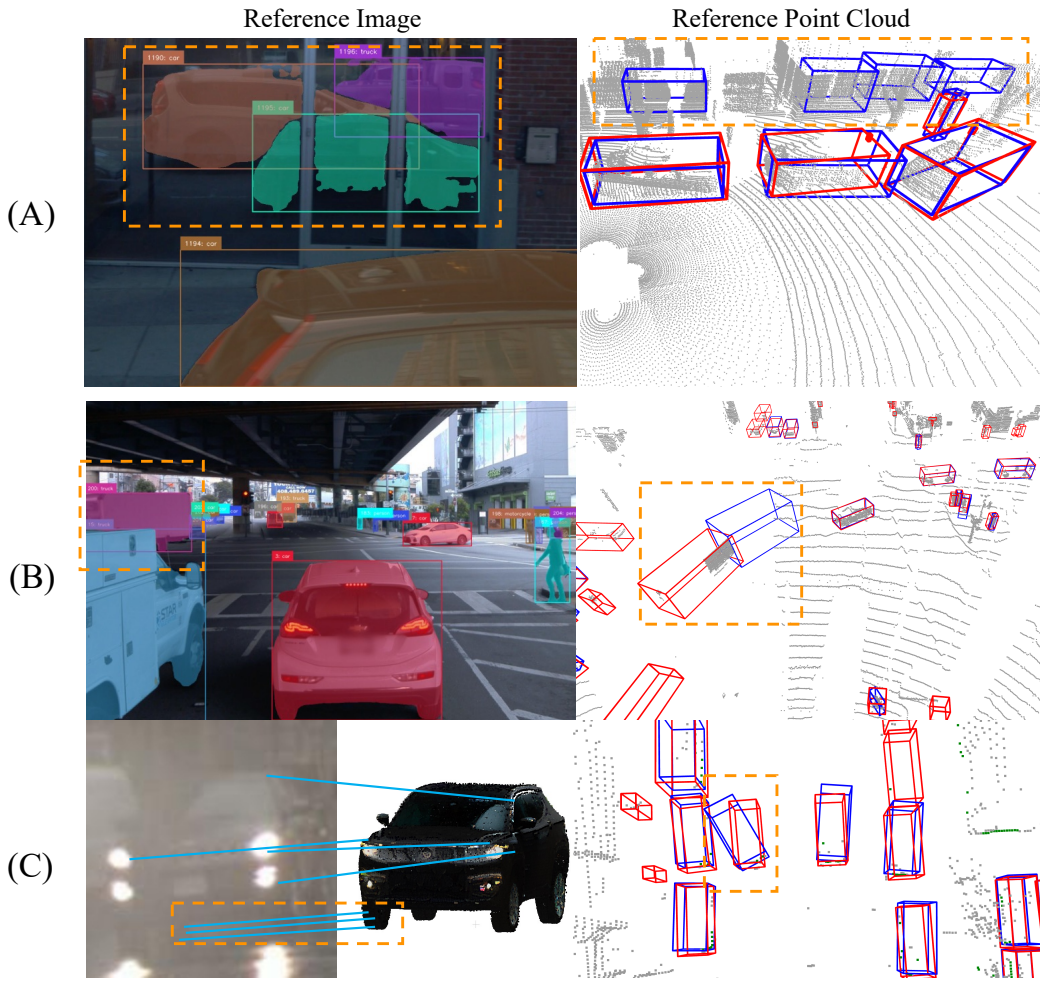


Figure 5: **Visualization of Failure Cases**. Each row shows a representative failure case. (A) illustrates a failure caused by vision foundation models (VFMs), while (B) shows a failure caused by incorrect object-motion estimation. (C) shows a failure case in adverse weather, where inaccurate matching leads to erroneous 3D-3D alignment. Red boxes denote human annotations, and blue boxes denote AmodalBox. Best viewed in color and zoomed in.

Table 11: **Comparison by distance and LiDAR visibility on the nuScenes [1] validation set.** We use the number of LiDAR points inside each human-annotated 3D box as a proxy for LiDAR visibility. For each class, boxes with zero LiDAR points are excluded, and the remaining boxes are split into class-specific tertiles: $(0, p_{33}]$ for *Low*, $(p_{33}, p_{66}]$ for *Mid*, and (p_{66}, ∞) for *High*. Metrics are computed on each distance-visibility subset using the nuScenes detection protocol.

Method	mAP↑											
	0-10m			10-20m			20-30m			30-40m		
	Low	Mid	High	Low	Mid	High	Low	Mid	High	Low	Mid	High
OV-SCAN [6]	2.18	4.75	54.15	3.05	11.67	43.04	2.88	18.05	27.46	2.63	15.41	10.45
AmodalBox	**7.37**	**11.27**	**64.92**	**7.27**	**22.77**	**60.48**	**6.55**	**32.20**	**46.77**	**4.17**	**25.87**	**23.81**
Method	NDS↑											
	0-10m			10-20m			20-30m			30-40m		
	Low	Mid	High	Low	Mid	High	Low	Mid	High	Low	Mid	High
OV-SCAN [6]	10.48	15.24	47.34	13.81	22.67	37.61	12.96	23.33	28.21	13.36	19.50	16.04
AmodalBox	**18.51**	**17.64**	**51.44**	**17.28**	**26.27**	**45.51**	**16.51**	**30.90**	**38.86**	**13.76**	**26.01**	**24.81**

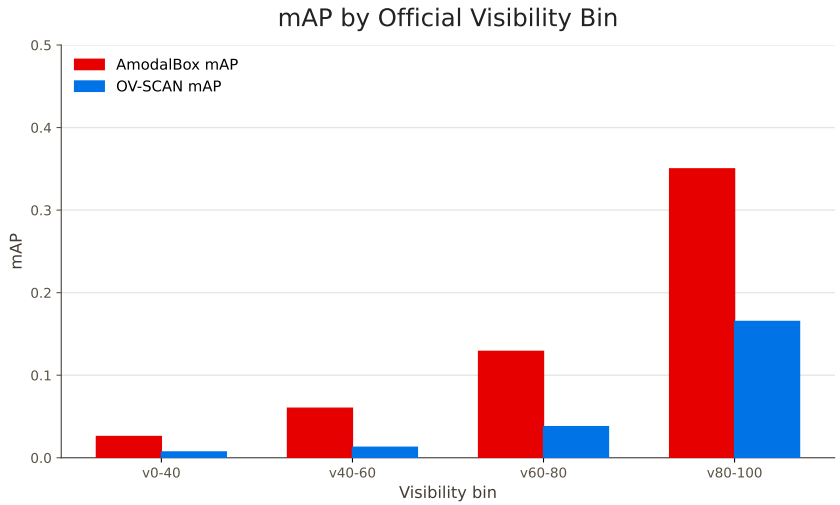


Figure 6: **Camera visibility-conditioned mAP on nuScenes [1].** Camera visibility is defined using the official nuScenes [1] visibility annotations and grouped into four bins corresponding to 0-40%, 40-60%, 60-80%, and 80-100% visible object fraction.

for LiDAR, we first divide objects by distance. In addition, because annotation difficulty varies across classes depending on the number of observed LiDAR points, we further split objects within each distance range into three difficulty levels based on the percentile of the number of points.

The results show that detectors trained with AmodalBox consistently outperform those trained with OV-SCAN [6] across all distances and difficulty levels. In particular, even in the *Low* setting, where only a small number of points are observed and annotation is therefore challenging, AmodalBox shows improvement in mAP. This suggests that our annotations are more effective than OV-SCAN in training detectors to handle partially observed point clouds.

Furthermore, AmodalBox also improves performance across all image-based visibility levels defined by nuScenes [1]. We conjecture that this improvement comes from the different treatment of low-visibility instances. OV-SCAN [6] relies on selective alignment and tends to filter out instances with small or low-quality masks, whereas AmodalBox uses a generation model [23] that is more robust to mask quality to construct amodal 3D assets and derive 3D bounding boxes. As a result, AmodalBox can provide more reliable annotations even under severe partial visibility, leading to higher mAP.