MoCA3D: Monocular 3D Bounding Box Prediction in the Image Plane

Changwoo Jeon[1,2], Rishi Upadhyay[1], and Achuta Kadambi[1]

[1] University of California, Los Angeles
[2] Yonsei University

Abstract. Monocular 3D object understanding has largely been cast as a 2D RoI-to-3D box lifting problem. However, emerging downstream applications require image-plane geometry (e.g., projected 3D box corners) which cannot be easily obtained without known intrinsics, a problem for object detection in the wild. We introduce **MoCA3D**, a **Mo**nocular, **C**lass-**A**gnostic **3D** model that predicts projected 3D bounding box corners and per-corner depths without requiring camera intrinsics at inference time. MoCA3D formulates pixel-space localization and depth assignment as dense prediction via corner heatmaps and depth maps. To evaluate image-plane geometric fidelity, we propose **Pixel-Aligned Geometry (PAG)**, which directly measures image-plane corner and depth consistency. Extensive experiments demonstrate that MoCA3D achieves state-of-the-art performance, improving image-plane corner PAG by 22.8% while remaining comparable on 3D IoU, using up to 57× fewer trainable parameters. Finally, we apply MoCA3D to downstream tasks which were previously impractical under unknown intrinsics, highlighting its utility beyond standard baseline models.

Keywords: Monocular 3D Understanding · Dense Prediction · Pixel-Aligned Geometry

1 Introduction

Monocular 3D object understanding is a long-standing goal in computer vision, with applications in augmented reality [24, 33, 42], robotics [17, 44], and visual servoing/control [8]. Most monocular 3D approaches [5, 22, 23, 58] represent an object as a compact 3D entity in camera/world coordinates, such as an oriented cuboid or a 6D pose. They are typically trained and evaluated in this 3D parameter space, and image-plane geometry is obtained only by projecting these 3D representations back to pixels when needed.

However, an increasing number of downstream applications (e.g., diffusion-based editing, controllable generation) consume geometry primarily in the image plane [3, 13, 32, 38, 49, 52, 56, 57]. In particular, they require accurate *projected 3D bounding box corners* (often with assigned depths) to enforce perspective-consistent layout and spatial control. This mismatch highlights a key limitation of existing monocular 3D pipelines: projected 3D corners are only actionable

when camera intrinsics are known, which are often unavailable or inconsistent in real-world settings.

A natural alternative is to predict the required image-plane geometry directly from the RoI, as most instance lifting models attempt via direct regression of box or pose parameters. However, Yu *et al*. [54] observe that direct corner regression is inherently sparse, providing limited gradient signals and potentially degrading performance; we therefore cast geometry recovery as a dense prediction task.

We propose **MoCA3D**, designed explicitly for *image-plane-aligned* recovery. Given a single RGB image and a tight oracle 2D bounding box, MoCA3D predicts projected 3D box corners and per-corner depths. Importantly, MoCA3D does *not* require camera intrinsics at inference time. When intrinsics are available at evaluation, we can lift corners and depths into a shared 3D representation for fair comparison with monocular 3D detection baselines.

To realize accurate image-plane geometry from only a box, MoCA3D departs from standard RoI-to-vector regression in two key ways. **(i) Leveraging transformer image priors.** Instead of treating the box only as a crop, we encode the box into a spatial prior that conditions dense image tokens, and further inject box embeddings through cross-attention so that every spatial token can directly incorporate the instance specification. **(ii) Dense prediction for corners and depth.** To avoid sparse supervision, we cast recovery as dense prediction: MoCA3D produces eight corner heatmaps and per-corner depth maps, followed by differentiable soft-argmax extraction and depth sampling. This design yields stable gradients across the RoI and encourages pixel-to-geometry alignment by construction.

A second challenge is evaluation: 3D metrics, such as 3D IoU, 3D corner distance, and pose error, can obscure reprojection errors that matter for downstream controls. We therefore introduce **Pixel-Aligned Geometry (PAG)**, a complementary metric suite that directly measures projected-corner error in the image plane (PAG_{uv}) and depth error at the corners (PAG_{d}). Together with a 3D-space corner metric (NHD) and IoU_{3D}, PAG enables geometry-centric benchmarking in both the image plane and 3D space, by mapping diverse outputs to a unified evaluation protocol. We show that MoCA3D substantially improves image-plane geometric fidelity over strong monocular 3D lifting baselines, while remaining lightweight and efficient. Finally, we demonstrate the practical utility of MoCA3D by integrating its outputs into controllable generation pipelines, where accurate reprojection and depth ordering are essential.

Contributions. Our main contributions are:

- We propose **MoCA3D**, a box-conditioned, class-agnostic monocular 3D object geometry model that directly predicts image-plane 3D geometry as projected cuboid corners and per-corner depths, without requiring camera intrinsics at inference.
- We reformulate box-conditioned monocular 3D recovery as a dense prediction problem, using corner heatmaps and depth maps to provide richer supervision than direct RoI-to-3D regression.

– We introduce Pixel-Aligned Geometry (PAG) to evaluate reprojection and depth consistency in image space under an oracle-2D protocol, and show that MoCA3D delivers strong gains in image-plane geometry and efficiency.

2 Related Work

Monocular 3D Object Detection. Early monocular approaches were largely confined to specialized domains, such as autonomous driving [31, 34] or indoor robotics [15]. However, these models struggle with "in the wild" scenarios due to variations in camera intrinsics and a limited vocabulary of object classes. Cube R-CNN [5] addressed these limitations by introducing a camera-agnostic virtual depth space, enabling the first large-scale universal 3D detector trained on the Omni3D benchmark.

Concurrently, *keypoint-based detection* [19,58] recasts object detection as keypoint estimation by predicting sparse landmarks such as box corners or centers as heatmaps and regressing box parameters. This paradigm has also been adopted for monocular 3D detection: CenterNet-style designs regress 3D parameters (depth, dimensions, and orientation) at detected centers [21, 26, 47], while RTM3D [22] predicts perspective keypoints of a 3D bounding box in the image plane and recovers 3D boxes via geometric constraints. In contrast to predicting sparse keypoints and regressing a compact 3D parameter vector, MoCA3D directly predicts projected 3D box corners with per-corner depths via dense heatmap and depth map prediction.

Coordinate Based Representations. A parallel line of research leverages Normalized Object Coordinate Space (NOCS) [48] to bridge the gap between 2D images and 3D geometry. While early NOCS methods were restricted to small-scale indoor datasets, recent works like OmniNOCS [16] and SOCS [46] scale this representation to hundreds of thousands of instances across diverse environments. By predicting dense 3D coordinates and utilizing Perspective-n-Point (PnP) solvers, these methods provide more granular shape information than traditional cuboid-regressing frameworks. Compared to these pipelines, MoCA3D predicts pixel-aligned corner-depth pairs directly in the image plane, rather than estimating object coordinates and solving for pose.

Open-World and Foundation Models. Recent advances have moved toward open-vocabulary 3D detection. Rather than relying solely on 3D supervision, methods such as DetAny3D [55] and 3D-MOOD [51] leverage the rich spatial priors of 2D foundation models (e.g., SAM [35], DINO [39]). These "lifting" strategies allow for zero-shot generalization to novel categories and unconstrained camera configurations, representing the current state-of-the-art for 3D perception in the wild. MoCA3D is complementary and similarly targets open-world settings via class-agnostic, box-conditioned geometry recovery.

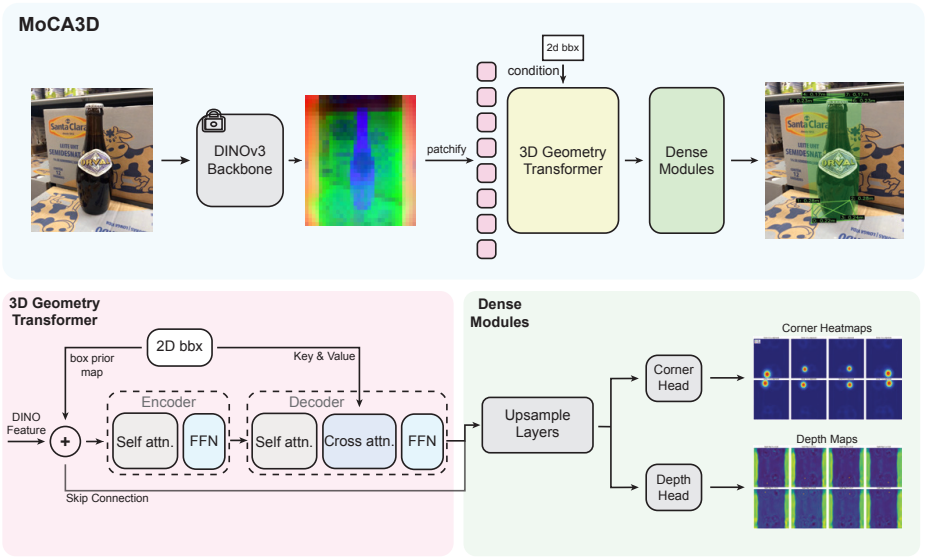

Fig. 1: MoCA3D architecture. Given an RGB image and a tight oracle 2D bounding box, MoCA3D uses a frozen DINOv3 backbone and a box-conditioned 3D Geometry Transformer with dense modules to predict eight corner heatmaps and per-corner depth maps, yielding pixel-aligned projected 3D box corners and depths.

3 Method

3.1 Overview

Given a single RGB image $I \in \mathbb{R}^{H\times W\times 3}$ and a 2D bounding box $\mathbf{b} = (x_1, y_1, x_2, y_2)$ around an object instance, MoCA3D predicts the output as $\hat{\mathcal{P}} = \{(\hat{\mathbf{p}}_i, \hat{d}_i)\}_{i=1}^{8}$ where $\hat{\mathbf{p}}_i = (\hat{u}_i, \hat{v}_i)$ denotes the image-plane box corner location (in pixels) and $\hat{d}_i$ is the corresponding depth. As illustrated in Fig. 1, MoCA3D consists of three stages. **(i) Backbone feature extraction**, where a frozen ViT foundation model produces a feature map $\mathbf{F}^{\text{dino}} \in \mathbb{R}^{C_0\times h_d\times w_d}$ from I. We use DINOv3 [39] to obtain geometry-aware embeddings. **(ii) Box-conditioned 3D Geometry Transformer**, where the 2D box $\mathbf{b}$ is encoded into a spatial box prior map that conditions the encoder and box embeddings for the decoder, enabling the transformer to refine image tokens within the object extent. **(iii) Dense geometry heads**, where lightweight upsampling layers feed two heads: a corner head predicting eight corner heatmaps and a depth head predicting per-corner depth maps.

3.2 3D Geometry Transformer

We build a box-conditioned transformer (Fig. 1 bottom left) following the canonical formulation of Transformer [45] and the DETR-style encoder-decoder design for vision [7].

Box prior map (relative-coordinate encoding) and Encoder. Starting from the frozen DINOv3 [39] features $\mathbf{F}^{\text{dino}} \in \mathbb{R}^{C_0 \times h_d \times w_d}$, we map them to the transformer hidden size C and inject an explicit per-pixel box prior map $\mathbf{M}(\mathbf{b}) \in \mathbb{R}^{4 \times h_d \times w_d}$ to condition dense tokens on the target instance. Specifically, for each spatial location (x, y), we encode its position relative to the box using four channels. Center-normalized offsets (d_x, d_y) which indicate whether a pixel lies left/right or above/below the box center and box-normalized coordinates (u, v) that measure the pixel's relative location w.r.t. the top-left corner. We project the prior map to the hidden dimension and inject it additively with a learned gate:

$$\mathbf{E}_{\text{prior}} = \psi(\mathbf{M}(\mathbf{b})) \in \mathbb{R}^{C \times h_d \times w_d}, \qquad \alpha = \sigma(\gamma), \qquad \tilde{\mathbf{F}} = \mathbf{F} + \alpha\, \mathbf{E}_{\text{prior}}, \tag{1}$$

where $\psi(\cdot)$ is a learnable projection and γ is a learnable scalar logit. We then add a standard 2D positional encoding and flatten the feature grid into a sequence of $N = h_d w_d$ tokens. An L_e-layer Transformer encoder contextualizes these tokens with multi-head self-attention and feed-forward networks with residual connections.

Decoder and box embeddings. Unlike detection transformers [7,20,25,30,59] that use a small set of object queries to retrieve information from image features, our goal is to refine dense image tokens for pixel-aligned prediction. We therefore reverse the usual conditioning direction: the encoded image tokens act as queries, while the box embeddings provide keys/values, so that every spatial token can directly incorporate the oracle box signal. Concretely, we encode the oracle box $\mathbf{b}$ into a small set of box embeddings $\mathbf{B} \in \mathbb{R}^{C \times N_q}$ (we use N_q=9), then apply an L_d-layer Transformer decoder that utilizes (i) self-attention over the image tokens, (ii) cross-attention where image tokens attend to the box embeddings as keys/values, and (iii) a feed-forward network, all with residual connections.

3.3 Dense Modules

Given the decoder output tokens, we reshape them back to the feature grid to obtain a 2D feature map $\mathbf{D} \in \mathbb{R}^{C \times h_d \times w_d}$. We fuse $\mathbf{D}$ with the box-conditioned backbone feature $\tilde{\mathbf{F}}$ (Eq. (1)) using a lightweight fusion block, $\mathbf{G} = \phi_{\text{fuse}}([\mathbf{D}; \tilde{\mathbf{F}}]) \in \mathbb{R}^{C \times h_d \times w_d}$, where $[\cdot;\cdot]$ denotes channel-wise concatenation. We then progressively upsample $\mathbf{G}$ with two lightweight stages to obtain a higher-resolution feature map $\mathbf{U} \in \mathbb{R}^{C/4 \times h_o \times w_o}$ for dense prediction. In our implementation, the output resolution is $h_o = h_d \times 4$, and the channel dimension is reduced from C to $C/4$ for efficiency.

Dense prediction heads. From $\mathbf{U}$, we predict (i) eight corner heatmaps and (ii) per-corner depth maps using two small convolutional heads:

$$\mathbf{H} = \phi_{\text{hm}}(\mathbf{U}) \in [0,1]^{8 \times h_o \times w_o}, \qquad \mathbf{Z} = \phi_{\text{dep}}(\mathbf{U}) \in \mathbb{R}_+^{8 \times h_o \times w_o}, \tag{2}$$

where $\mathbf{H}$ and $\mathbf{Z}$ denote heatmap and depth map, and ϕ_{hm} and ϕ_{dep} denote corner head and depth head. ϕ_{hm} ends with a sigmoid activation to ensure a bounded heatmap interpretation consistent with the target heatmap values defined in our

loss (Sec. 3.4), facilitating stable alignment between predictions and supervision. In contrast, ϕ_{dep} ends with a softplus [11] to enforce non-negative depths. We initialize the heatmap head bias to encourage low initial activations, which helps stabilize early-stage training.

Soft-argmax [29] corner extraction and depth sampling. We convert each corner heatmap $\mathbf{H}_i$ into a continuous coordinate $\hat{\mathbf{p}}_i \in \mathbb{R}^2$ using a differentiable soft-argmax function. Concretely, soft-argmax first forms a spatial probability distribution by applying a softmax over all locations with an inverse-temperature β:

$$\pi_i(x,y) = \frac{\exp(\beta\,\mathbf{H}_i(x,y))}{\sum_{x',y'}\exp(\beta\,\mathbf{H}_i(x',y'))}, \tag{3}$$

and returns the expected coordinate under π_i:

$$\hat{u}_i = \sum_{x,y} x\,\pi_i(x,y), \qquad \hat{v}_i = \sum_{x,y} y\,\pi_i(x,y), \qquad \hat{\mathbf{p}}_i = (\hat{u}_i, \hat{v}_i). \tag{4}$$

The larger β approaches the hard argmax (more peak-like responses), while smaller β yields smoother averaging, allowing sub-pixel localization with stable gradients. Finally, we sample per-corner depths from $\mathbf{Z}$ at the predicted corner locations using bilinear interpolation.

$$\hat{d}_i = \text{Sample}(\mathbf{Z}_i, \hat{\mathbf{p}}_i)\,, \qquad i = 1, \ldots, 8. \tag{5}$$

For consistency with our evaluation setting, the coordinates are scaled from the heatmap grid back to the image resolution, yielding the final predictions. $\hat{\mathcal{P}} = \{(\hat{\mathbf{p}}_i, \hat{d}_i)\}_{i=1}^{8}$.

3.4 Loss

We supervise MoCA3D with a peak-weighted heatmap regression loss and a coordinate-level refinement loss for corner localization, together with a confidence-weighted depth loss. Let $\{\mathbf{p}_i\}_{i=1}^{8}$ and $\{d_i\}_{i=1}^{8}$ denote the ground-truth projected corners and depths.

Target heatmap generation. By placing a Gaussian-like peak around each ground-truth corner, we construct a soft target heatmap $\mathbf{W} \in [0,1]^{8\times h_o \times w_o}$:

$$\mathbf{W}_i(x,y) = \exp\!\left(-\frac{\|(x,y) - \mathbf{p}_i\|_2^2}{2\sigma_i^2}\right), \tag{6}$$

where $2\sigma_i^2$ is set adaptively based on the object size, following Yu *et al.* [54] (we set $2\sigma_i^2$ as the squared one-fifth of pixel distance between object's 2D center and each corner).

Peak-weighted heatmap loss (coarse). A naive heatmap regression treats background and peak pixels similarly; however, peak supervision is inherently sparse and can be overwhelmed by the vast background, leading to diffuse responses that are less informative for downstream coordinate extraction. To explicitly prioritize peak alignment, we upweight pixels whose target heatmap value

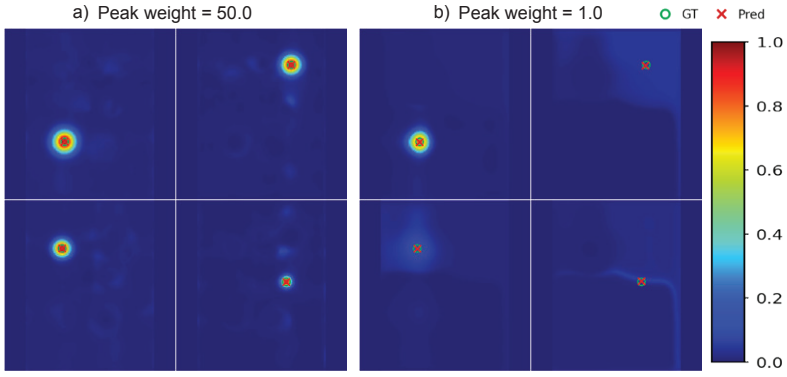

Fig. 2: Heatmap Comparison. Predicted corner heatmaps with (a) peak weight = 50.0 and (b) = 1.0 (uniform). Larger peak weight sharpens and localizes responses near GT corners, improving soft-argmax stability, while uniform weighting yields flatter heatmaps.

exceeds a threshold τ:

$$\mathbf{A}_i(x,y) = \begin{cases} \lambda, & \mathbf{W}_i(x,y) > \tau, \\ 1, & \text{otherwise}, \end{cases} \tag{7}$$

where $\lambda > 1$ is a peak emphasis factor. We then apply robust regression loss:

$$\mathcal{L}_{\text{coarse}} = \frac{\sum_{i,x,y} \mathbf{A}_i(x,y) \cdot \text{SmoothL1}(\mathbf{H}_i(x,y) - \mathbf{W}_i(x,y))}{\sum_{i,x,y} \mathbf{A}_i(x,y) + \epsilon}. \tag{8}$$

As visualized in Fig. 2, increasing the peak weight concentrates gradients around the true corner locations, producing sharper and more localized maxima, while uniform weighting yields flatter and less reliable heatmaps.

Coordinate refinement loss (fine). To encourage accurate sub-pixel localization, we apply a coordinate-level loss on the soft-argmax outputs:

$$\mathcal{L}_{\text{fine}} = \frac{1}{8} \sum_{i=1}^{8} \text{SmoothL1}(\hat{\mathbf{p}}_i - \mathbf{p}_i)\,, \tag{9}$$

Importantly, $\mathcal{L}_{\text{coarse}}$ and soft-argmax make $\mathcal{L}_{\text{fine}}$ more effective: as the peak-weighted coarse loss sharpens H_i, the soft-argmax concentrates around the true mode, producing sub-pixel coordinates and cleaner gradients for refinement.

Confidence-weighted depth loss (pixel-aligned depth supervision). Following Cube R-CNN [5], MoCA3D regresses virtual depth to alleviate scale ambiguity, and supervises per-corner depth with a confidence-weighted loss in which the predicted heatmaps act as per-pixel reliability weights. Let $\bar{d}_i \in \mathbb{R}^{h_o \times w_o}$

denote the ground-truth for that corner (broadcast to the heatmap grid). We define:

$$\mathcal{L}_{\text{depth}} = \frac{\sum_{i,x,y} \mathbf{H}_i(x,y) \cdot \text{SmoothL1}\big(\mathbf{Z}_i(x,y) - \bar{d}_i\big)}{\sum_{i,x,y} \mathbf{H}_i(x,y) + \epsilon}. \tag{10}$$

This formulation explicitly couples localization and depth estimation: pixel-aligned heatmaps indicate where each corner is likely to lie, thereby focusing depth supervision on spatial regions that are consistent with predicted corner location. Consequently, more reliable heatmaps (Fig. 2) not only improve $\hat{\mathbf{p}}_i$, but also provide more accurate and stable sampling for per-corner depth.

Total loss and warm-up. The final objective is:

$$\mathcal{L} = w_{\text{coarse}}\mathcal{L}_{\text{coarse}} + w_{\text{fine}}\mathcal{L}_{\text{fine}} + w_{\text{depth}}\mathcal{L}_{\text{depth}}. \tag{11}$$

We use a warm-up schedule: we first optimize only the coarse heatmap loss, then enable the refinement and depth terms and linearly re-balance the weights over training ($w_{\text{coarse}} : 50\to1$, $w_{\text{fine}} : 0\to2$, $w_{\text{depth}} : 0\to5$).

3.5 MoCA3D-Cube

To demonstrate MoCA3D's adaptability to standard 3D bounding box prediction, we design **MoCA3D-Cube**, a lightweight 3D bounding box adapter. MoCA3D-Cube takes as input (i) MoCA3D's projected corner coordinates $\hat{\mathbf{p}}$ and depth outputs $\hat{d}$ and (ii) a RoI-aligned embedding pooled from the decoder feature map. These features are concatenated and passed to a compact two-layer Cube MLP that regresses a parametric 3D bounding box, including center, size, and rotation with uncertainty, conditioned on camera intrinsics K. For training and supervision, we follow the Cube R-CNN [5] protocol for 3D box parameterization and losses. We initialize from a pretrained MoCA3D and jointly fine-tune the adapter with MoCA3D while keeping the backbone frozen.

4 Experiments

4.1 Experimental Settings

Datasets and Baselines. We evaluate **MoCA3D** on the Omni3D benchmark [5] that provides diverse spatial coverage across multiple domains (urban, indoor, and general) by integrating datasets and annotations from nuScenes [6], KITTI [10], SUN RGB-D [40], ARKitScenes [2], Hypersim [36] and Objectron [1]. However, Omni3D contains instances with missing 2D boxes; we complete them using SAM2 with projected 3D corners as prompts (see supplementary material for preprocessing details).

We compare **MoCA3D** against Cube R-CNN [5], OVMono3D-Lift* [53], and DetAny3D [55]. Although these methods are developed for monocular 3D detection, we benchmark them under a unified protocol that removes detector confounds: given the box as input, and each method performs *box-conditioned*

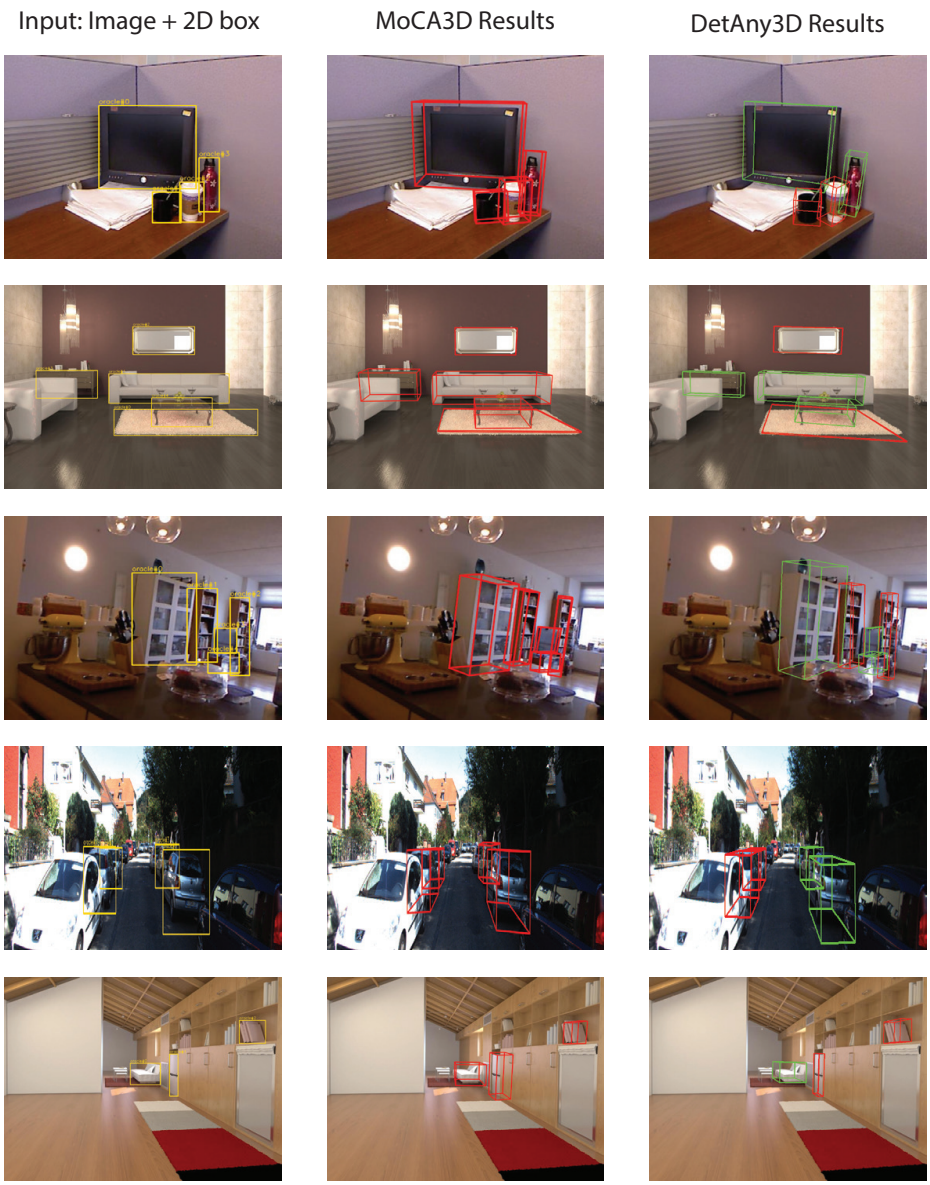

Fig. 3: Qualitative Results. MoCA3D vs. DetAny3D predictions under oracle 2D boxes on samples from the KITTI, Omni3D, and Hypersim datasets. Detections in green have a 3D IoU of less than 0.1, making them low-quality detections.

3D lifting to predict a 3D box. This enables an apples-to-apples comparison with MoCA3D; both can be mapped into a shared representation using camera in-

trinsics K for evaluation. Cube R-CNN operates in a closed-set setting, whereas OVMono3D and DetAny3D are open vocabulary models. To further isolate the effect of category priors, we additionally train a Cube R-CNN* variant by removing class priors while keeping the remaining training and inference settings unchanged.

Implementation Details. We use DINOv3 ViT-L/16 [39] as the frozen image backbone and train on Omni3D [5] with oracle tight 2D boxes. Inputs are resized to 512×512 using aspect-ratio preserving *letterboxing* (padding), and the same transform is applied to ground-truth 2D boxes and projected corners. Training is performed for 120 epochs on 3 NVIDIA RTX 3090 GPUs (50,000 instances per epoch; $\sim$60 GPU-hours total). For efficiency, we pre-extract and cache backbone features, which requires $\sim$4 GPU-hours. We optimize with AdamW [28] using a learning rate of 2×10^{-4} and a cosine schedule [27] with a 5-epoch warm-up. We apply standard augmentations including box jittering, feature dropout, horizontal flipping, and rotation; further details are provided in the supplementary material.

4.2 Metrics

Following CatFree3D [4], we report Normalized Hungarian Distance (NHD). NHD computes an optimal one-to-one correspondence between the eight corners using Hungarian matching and sums their Euclidean distances, normalized by the ground-truth box diagonal:

$$\mathrm{NHD}(M_{\mathrm{gt}}, M_{\mathrm{pred}}) = \frac{1}{d_{\mathrm{gt}}} \sum_{i=1}^{8} \left\| \mathbf{a}_i - \mathbf{b}_{\pi(i)} \right\|_2 , \tag{12}$$

where π is the optimal assignment and d_{gt} denotes the diagonal length of the ground-truth 3D box M_{gt}.

To evaluate image-plane geometry, we additionally report Pixel-Aligned Geometry (PAG) metrics that operate on the predicted corner-depth pairs. Recall from Sec. 3 that MoCA3D predicts $\hat{\mathcal{P}} = \{(\hat{\mathbf{p}}_i, \hat{d}_i)\}_{i=1}^{8}$, where $\hat{\mathbf{p}}_i \in \mathbb{R}^2$ is the image-plane corner location and $\hat{d}_i$ is its depth; ground truth is denoted by $\mathcal{P} = \{(\mathbf{p}_i, d_i)\}_{i=1}^{8}$. We define the image-plane corner error (in pixels) as:

$$\mathrm{PAG}_{uv} = \frac{1}{8} \sum_{i=1}^{8} \left\| \hat{\mathbf{p}}_i - \mathbf{p}_i \right\|_2 , \tag{13}$$

and the relative depth error (%) as:

$$\mathrm{PAG}_{d} = \frac{100}{8} \sum_{i=1}^{8} \frac{|\hat{d}_i - d_i|}{d_i}, \tag{14}$$

Unless otherwise specified, we compute PAG on images resized to 512×512 for consistent comparison across methods. Together, NHD and IoU_{3D} evaluate

$\mathbf{PAG}_{uv}$ (px) ↓						
Model	kitti	nuscenes	objectron	arkitscenes	sunrgbd	hypersim
Cube R-CNN	10.10	11.73	23.57	22.47	23.17	13.92
Cube R-CNN*	10.38	11.86	31.28	23.75	24.06	14.17
OVMono3D-LIFT*	10.29	11.82	23.55	22.55	22.36	13.02
DetAny3D	4.90	6.75	27.41	17.15	21.06	10.71
MoCA3D	**3.12**	**5.63**	**18.60**	**14.90**	**16.99**	**9.56**
$\mathbf{PAG}_{d}$ (%) ↓						
Model	kitti	nuscenes	objectron	arkitscenes	sunrgbd	hypersim
Cube R-CNN	12.63	12.05	15.03	13.22	13.92	20.66
Cube R-CNN*	13.85	13.04	33.18	14.92	16.61	22.81
OVMono3D-LIFT*	13.20	12.45	12.68	12.60	14.61	19.94
DetAny3D	**4.82**	**6.07**	**12.55**	8.87	**8.71**	**16.81**
MoCA3D	5.04	7.17	14.59	**8.78**	10.08	19.57

Table 1: Image-plane geometry metrics: MoCA3D outperforms or is comparable to prior models across all datasets. All results were computed with oracle 2D bounding boxes and ground truth intrinsics were used to calculate pixel-space coordinates for models other than MoCA3D. Green indicates the best and orange indicates the second-best results. Cube R-CNN* is trained in a class-agnostic setting.

geometric accuracy in 3D bounding box space, while PAG measures pixel-aligned geometry via image-plane corner error (PAG_{uv}) and relative depth error (PAG_{d}). IoU_{3D} is implemented using PyTorch3D [18].

4.3 Comparisons with Baselines

Image-plane Evaluations Table 1 evaluates pixel-aligned geometry under the oracle-2D protocol. Across all six domains, MoCA3D achieves the lowest PAG_{uv}, indicating the most accurate projected corner localization. This suggests that MoCA3D yields stronger pixel-level supervision than detection-style lifting pipelines. Aggregated over Omni3D, MoCA3D improves PAG_{uv} by 22.8.%. MoCA3D ranks second overall on PAG_{d} across the six domains, achieving the best on ARKitScenes [2] (8.78%) and the second on KITTI [10], nuScenes [6], SUN RGB-D [40], and Hypersim [36], while remaining competitive on Objectron [1].

3D Box Evaluation Table 2 reports NHD and IoU_{3D} under oracle-2D protocol. On NHD, MoCA3D achieves the best performance on KITTI [10] and ARKitScenes [2] and is comparable to the strongest 3D lifting baseline (DetAny3D [55]) across domains, indicating accurate 3D corner geometry even without explicitly enforcing a cuboid parameterization.

To compute IoU_{3D}, we rectify MoCA3D's unprojected corners into the closest valid cuboid using a Kabsch-based [14] rigid alignment. Specifically, we unproject $(\hat{\mathbf{p}}_i, \hat{d}_i)$ with K to obtain 3D corner points, then align them to a canonical cuboid template via a Procrustes/Kabsch solve and recover per-axis scale to form

NHD ↓						
Model	kitti	nuscenes	objectron	arkitscenes	sunrgbd	hypersim
Cube R-CNN	0.2811	0.3067	0.2462	0.3590	0.4466	1.0603
Cube R-CNN*	0.3504	0.3758	0.5282	0.4074	0.5642	1.1875
OVMono3D-LIFT*	0.3177	0.3458	0.2132	0.3527	0.4864	1.0216
DetAny3D	0.1911	**0.2428**	**0.2107**	0.2593	**0.3026**	**0.8970**
MoCA3D	**0.1902**	0.3002	0.2519	**0.2529**	0.3381	1.0406

IoU_{3D} ↑						
Model	kitti	nuscenes	objectron	arkitscenes	sunrgbd	hypersim
Cube R-CNN	0.4387	0.4118	0.3501	0.3135	0.2908	0.0895
Cube R-CNN*	0.3738	0.3569	0.2748	0.2534	0.2260	0.0769
OVMono3D-LIFT*	0.3812	0.3755	**0.3706**	0.3060	0.2419	0.0899
DetAny3D	**0.4834**	**0.4311**	0.3587	**0.3826**	**0.3272**	**0.1079**
MoCA3D	0.4735	0.3962	0.3015	0.3180	0.2753	0.0713
MoCA3D-Cube	0.4549	0.3993	0.3163	0.3435	0.2455	0.0899

Table 2: 3D camera space metrics: MoCA3D is competitive with prior work such as DetAny3D despite a significantly smaller parameter count and compute requirements. Metrics were computed using ground truth camera intrinsics and oracle 2D bounding boxes. Green indicates the best and orange indicates the second-best results.

an oriented cuboid. This produces a cuboid with a consistent vertex topology, allowing direct evaluation with PyTorch3D [18] IoU_{3D}. MoCA3D achieves the second-best performance on KITTI [10], and is comparable to Cube R-CNN and OVMono3D-Lift* across the remaining datasets.

Finally, MoCA3D-Cube explicitly adapts image-plane geometry to a parametric box representation. As a result, it consistently improves IoU_{3D} over MoCA3D except on SUNRGBD [40] and KITTI [10] and attains second-best performance on ARKitScenes [2] and Hypersim [36], demonstrating that MoCA3D's representation can be readily mapped to conventional 3D detection outputs when box-level evaluation is desired.

Efficiency of MoCA3D MoCA3D is lightweight (19.0 M trainable parameters, 57× smaller than DetAny3D), runs efficiently at inference time (0.14 s/ex) at CV-Bench [43]. Moreover, in Fig. 4(b), MoCA3D attains the lowest PAG_{uv} while remaining highly efficient, demonstrating a favorable efficiency-performance trade-off.

4.4 Ablations

We conduct controlled ablations on Objectron [1], keeping the overall MoCA3D pipeline unchanged. Table 3 reports PAG (positive delta indicates degradation), and IoU_{3D} (negative delta indicates degradation). In addition, it reports the trainable parameter count and training compute for each variant.

MoCA3D with Depth Anything [23]. We augment MoCA3D with Depth Anything v3 [23] depth cues by fusing mono and metric predictions into a feature

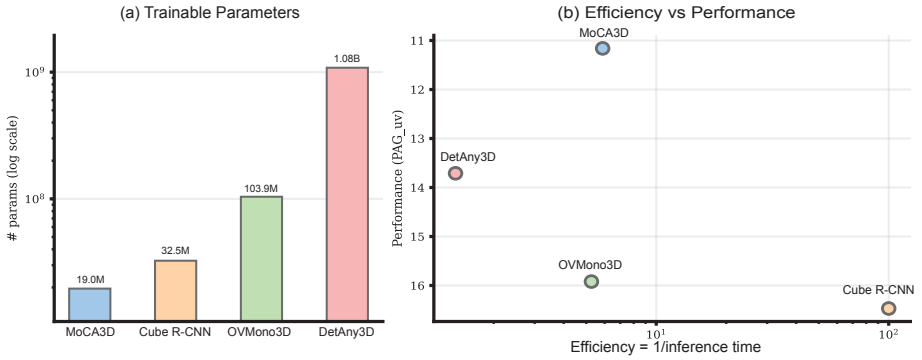

Fig. 4: Efficiency of MoCA3D. We compare (a) trainable parameters and (b) trade-off between efficiency and performance (PAG_{uv}). Efficiency is defined as the inverse of end-to-end inference time per example on CV-Bench [43].

Method	**PAG**$_{uv}$ ↓ (px)	**PAG**$_{d}$ ↓ (%)	**IoU**$_{3D}$ ↑	#**Params** (M)	**Train** (GPU-hrs)
MoCA3D (baseline)	**16.05**	**10.83**	**0.3768**	19.0	27.0
MoCA3D w/ DA	16.14	10.92	0.3682	19.8	–
Δ vs. MoCA3D	**+0.09**	**+0.09**	**-0.0086**		
MoCA3D w/o box prior	16.35	10.88	0.3736	19.0	27.0
Δ vs. MoCA3D	**+0.29**	**+0.05**	**-0.0032**		
Direct Regressor	17.38	14.89	0.3064	22.1	19.2
Δ vs. MoCA3D	**+1.32**	**+4.06**	**-0.0704**		

Table 3: MoCA3D ablations. We report PAG_{uv}, PAG_{d} (lower is better) and IoU_{3D} (higher is better); Δ denotes *method* − MoCA3D.

stream alongside the DINOv3 [39] feature map. This variant shows a marginal degradation on all metrics (+0.09px in PAG_{uv}, +0.09% in PAG_{d}, and -0.0086 in IoU_{3D}), suggesting that MoCA3D's box-conditioned dense geometry learning already provides sufficiently strong cues, while injecting external depth features can introduce redundancy or noise.

Removing box prior map. Disabling box prior map conditioning degrades on all metrics (+0.29px in PAG_{uv}, +0.05% in PAG_{d}, and -0.0032 in IoU_{3D}), indicating that explicit box-relative geometry cues are important for pixel-level corner localization.

Direct regression head. We replace the dense heatmap and depth heads with an RoI-aligned direct regressor that predicts corner (u, v, d) values in a single shot. This yields the largest performance drop (+1.32px in PAG_{uv}, +4.06% in PAG_{d}, and -0.0704 in IoU_{3D}), supporting our design choice that dense, pixel-aligned supervision is critical for stable optimization and precise image-plane geometry recovery.

Fig. 5: Driving scene variation guided by MoCA3D. MoCA3D recovers image-plane vehicle geometry (projected 3D box corners) from a single image and uses it to condition diffusion-based generation for diverse prompt-driven edits.

4.5 Downstream Task: Controllable Driving Scene Generation

Recent controllable street-view synthesis methods [41, 50, 52, 57] rely on explicit 3D geometric controls (e.g., perspective-projected layouts). For instance, PerLDiff [57] leverages perspective projection cues as geometric priors and uses text prompt to manipulate lighting and weather conditions of street scenes. However, these controls are typically sourced from 3D annotations, where projected 3D boxes provide eight 2D corner points per object.

We show that MoCA3D can replace this annotation-dependent geometric input. Given a single real driving image, we run MoCA3D to obtain image-plane geometry for vehicles as projected cuboid corners. We then use the recovered geometry as the object-level control signal in a diffusion-based generation pipeline, while changing the text and class prompt to specify the desired environmental condition. As shown in Fig. 5, this produces diverse driving-scene variations that preserve the underlying viewpoint and vehicle layout, demonstrating that image-plane geometry is sufficient to enable practical, annotation-free controllable generation from raw images.

5 Conclusion and Future Work

In this work, we introduce MoCA3D, a monocular, class-agnostic approach that represents object geometry directly in the image-plane as projected 3D cuboid

corners with depths. By casting recovery as dense prediction, MoCA3D yields stable, accurate pixel-aligned geometry without relying on camera intrinsics. We also introduce Pixel-Aligned Geometry (PAG) to directly evaluate corner and depth consistency in the image plane. MoCA3D achieves state-of-the-art PAG performance while remaining competitive on standard 3D metrics.

MoCA3D currently assumes that all eight corners are visible, and thus struggles with truncated objects. In addition, corner-only geometry leaves orientation ambiguity for symmetric instances. Future work could mitigate these issues via truncation-augmented preprocessing or by incorporating directional cues and auxiliary keypoints.

Acknowledgements

Icon credits. "Inference" icon by Wehape and "Man" icon by Viktor Vorobyev, from The Noun Project (CC BY 3.0).

A Additional Method Details

A.1 Box Prior Encoding and Box-Conditioned Decoder

We provide additional implementation details on how the oracle 2D box is injected into the transformer. Given a normalized box $\mathbf{b} = (x_1, y_1, x_2, y_2)$, we first construct a dense four-channel prior map on the backbone feature grid. For each feature location (x, y), we compute box-relative center offsets and top-left-relative normalized coordinates:

$$d_x = \frac{x - c_x}{w_b}, \qquad d_y = \frac{y - c_y}{h_b}, \qquad u = \frac{x - x_1}{w_b}, \qquad v = \frac{y - y_1}{h_b}, \tag{15}$$

where (c_x, c_y) is the box center and (w_b, h_b) denote the box width and height. In the implementation, (x, y) corresponds to the center of each backbone token on the normalized feature grid, i.e., $(x, y) \in ((0, 1), (0, 1))$. The resulting prior tensor is projected to the transformer hidden dimension using a lightweight 1×1-ReLU-1×1 module, and then fused with the DINOv3 [39] feature map through a learnable gated residual [12] connection.

We also encode the input box as a set of box embeddings for decoder conditioning. Rather than using a single box token, we form 9 geometric keypoints from the box: the four corners, four edge midpoints, and the box center. Their normalized coordinates are transformed with a sine-cosine positional embedding, passed through an MLP, and augmented with learned identity embeddings so that each token preserves its geometric role. These box tokens are then used as keys and values in decoder cross-attention.

Our decoder keeps dense image tokens $\mathbf{X} \in \mathbb{R}^{N \times C}$ as queries and treats box embeddings $\mathbf{B} \in \mathbb{R}^{N_q \times C}$ as conditioning keys/values:

$$\mathrm{Attn}(\mathbf{X}, \mathbf{B}) = \mathrm{softmax}\left(\frac{(\mathbf{X}W_Q)(\mathbf{B}W_K)^\top}{\sqrt{d}}\right)\mathbf{B}W_V, \tag{16}$$

Since cross-attention preserves the query shape, the output remains a dense $N \times C$ token map, which is well suited for pixel-aligned prediction and resembles conditioned generation pipelines such as Stable Diffusion [37], where spatial tokens are modulated by external conditions through attention.

A.2 Virtual Depth Parameterization

Following Cube R-CNN [5], we supervise depth in *virtual depth* space rather than raw metric depth. This is important for Omni3D, which aggregates images captured with diverse cameras and focal lengths. Virtual depth reduces cross-camera scale variation by mapping metric depth into a normalized camera space with fixed virtual focal length f_v and virtual image height H_v.

Given the metric depth d_i, image height H, and focal length f, its virtual depth target is defined as:

$$d_i^v = d_i \cdot \frac{f_v}{f} \cdot \frac{H}{H_v}, \tag{17}$$

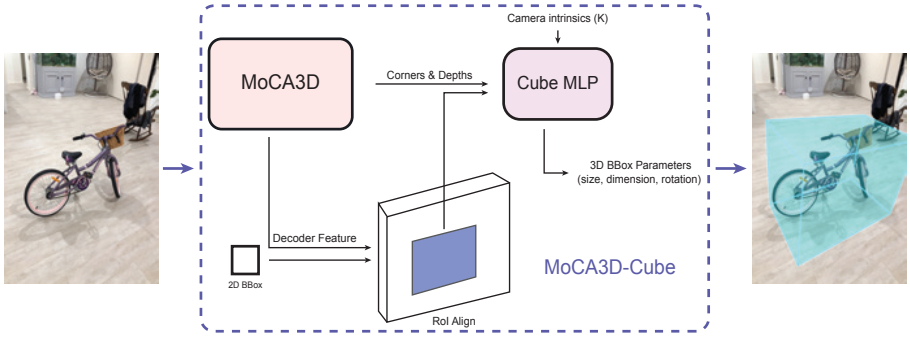

Fig. 6: MoCA3D-Cube Adapter. MoCA3D-Cube combines MoCA3D's predicted corner-depth pairs with an RoI-aligned decoder feature and camera intrinsics K to regress a parametric 3D bounding box. The adapter reuses image-plane geometry as the primary cue while adding a lightweight Cube MLP for conventional box prediction.

where f_v and H_v are fixed hyperparameters (f_v=H_v=512.0 in our implementation) shared across the dataset. Since the conversion depends only on camera- and image-level quantities, all corners share the same scaling.

At inference, MoCA3D predicts virtual depth, which can be converted back to metric depth when camera intrinsics are available:

$$d_i = d_i^v \cdot \frac{f}{f_v} \cdot \frac{H_v}{H}. \tag{18}$$

When camera intrinsics are unavailable at inference time, we assume a canonical focal length $f = f_v$.

A.3 MoCA3D-Cube Adapter Details

As illustrated in Fig. 6, MoCA3D-Cube uses the MoCA3D corner and depth outputs to define geometric priors for box center and size, and the Cube MLP predicts residual updates on top of them. Specifically, the predicted corner-depth pairs are backprojected to 3D corners (X_i, Y_i, z_i) using camera intrinsics, and their center prior is computed as the mean corner location:

$$(X_0, Y_0, z_0) = \frac{1}{8} \sum_i (X_i, Y_i, z_i) \tag{19}$$

The size prior is computed from the mean center-to-corner radius:

$$r = \frac{1}{8} \sum_i \|(X_i, Y_i, z_i) - (X_0, Y_0, z_0)\|_2 \tag{20}$$

Table 4: Image statistics of the Omni3D source datasets.

Dataset	Total	Train	Val	Test
KITTI	7,481	3,321	391	3,769
SUN RGB-D	10,335	4,929	356	5,050
nuScenes	34,149	26,215	1,915	6,019
Objectron	46,644	33,519	3,811	9,314
ARKitScenes	60,924	48,046	5,268	7,610
Hypersim	74,619	59,543	7,386	7,690
Omni3D	234,152	175,573	19,127	39,452

This is converted into a cube prior $\mathbf{s}_0 = (2/\sqrt{3})\, r \cdot (1, 1, 1)$.

The Cube MLP takes two inputs: a geometry feature, formed from per-corner offsets $(x_i^{\text{norm}} - x_0,\ y_i^{\text{norm}} - y_0,\ \log z_i - \log z_0)$ over the 8 corners together with the global prior term $(x_0, y_0, z_0, X_0, Y_0)$, and the RoI feature obtained by applying `roi_align` with an output size 7×7 to the tight 2D box, followed by a 1×1 convolution, ReLU, and global average pooling:

$$\mathbf{r} = \text{GAP}(\text{ReLU}(\text{Conv}_{1\times1}(\text{RoIAlign}(\mathbf{F}_{\text{dec}}, \mathbf{b})))), \qquad \mathbf{o} = \text{MLP}([\mathbf{g}; \mathbf{r}]), \quad (21)$$

where $\mathbf{o} \in \mathbb{R}^{13}$ denotes the predicted residual box parameters, rotation, and uncertainty.

B Data and Preprocessing

B.1 Omni3D and Dataset Notes

Our experiments build on Omni3D [5], a large-scale benchmark for image-based 3D object understanding. After merging semantically overlapping categories across sources, Omni3D contains 234,152 images with roughly 3 million labeled 3D bounding boxes spanning 98 object categories. Table 4 shows the detailed statistics of Omni3D.

All annotations are expressed in a unified camera-centric coordinate system with the camera center as origin and axes defined as +x right, +y down, and +z inward. Each instance is annotated with a category, a 2D box, a 3D centroid, a rotation matrix, and metric box dimensions. Input resolutions range from 370 to 1920 pixels and focal lengths from 518 to 1708 pixels.

B.2 Preprocessing Procedure

2D Box completion with SAM2 [35] and Filtering Criteria Some Omni3D annotations do not provide a valid tight 2D bounding box. To make such instances usable in our box-conditioned setting, we first complete missing `bbox2D_tight` annotations using SAM2. Concretely, for each object whose tight 2D box is missing, we construct a rough box from the min/max coordinates of the projected

3D cuboid corners and use it as a box prompt for SAM2. We then convert the predicted mask into a tight 2D bounding box by taking the extremal mask coordinates, and retain the update only when the predicted region is non-empty and spatially valid.

After box completion, we apply a two-stage procedure. First, we follow the standard Cube R-CNN annotation filtering. Specifically, this step removes ignored, invalid, or severely corrupted annotations according to dataset metadata and geometric validity checks; we refer readers to Omni3D [5] for the full filtering details.

Second, we apply an additional filter. We keep only annotations marked as `Good`, i.e., instances whose projected cuboid corners all lie inside the image, and further discard very small objects using a minimum resized box-area threshold (at least 1024 pixels^2 after resizing the longest image side to 512). We enforce the `Good` constraint because MoCA3D predicts image-plane corners with soft-argmax, making targets with out-of-image corners unsuitable for reliable supervision. After preprocessing, the final training set contains approximately 364K samples. The same filtering procedure is applied to the validation and test sets, and all experiments are conducted on this filtered subset.

Image-Space Canonicalization and Input Preparation Because MoCA3D predicts geometry directly in the image plane, we canonicalize the target corner order using 2D image coordinates rather than a dataset-specific 3D vertex convention. For each instance, we reorder the eight projected cuboid corners by splitting them into the lower and upper four points according to their image-plane vertical coordinate, and then sorting each group from left to right using the horizontal coordinate. The same permutation is applied to the corresponding depth values, and this image-aligned ordering provides a more stable supervision format.

Our dataloader then converts each annotation into image-aligned training targets. Images are resized to 512×512 with aspect-ratio-preserving letterboxing and normalized using ImageNet [9] mean and standard deviation. Projected corners and tight 2D boxes are transformed with the same scale and padding, and are normalized to $[0, 1]$. Corner depths are normalized to virtual depth (Sec. A.2).

Augmentation In the feature-based training pipeline, we do not rely on photometric augmentation. Instead, augmentation is applied primarily in feature space and is synchronized with corresponding geometric targets. Specifically, during training we apply random horizontal flipping with probability 0.4 and random in-plane rotation within $\pm 10^\circ$ to the pre-extracted DINOv3 feature map.

We further use lightweight stochastic perturbations in feature and box space. After a 5-epoch warm-up, Gaussian feature noise with standard deviation 0.01 is added to the DINOv3 feature map, and Gaussian jitter with standard deviation 0.02 is applied to the normalized 2D box coordinates.

In addition, we apply token-level regularization inside the model. During training, spatial feature tokens are randomly masked with probability 0.25, using

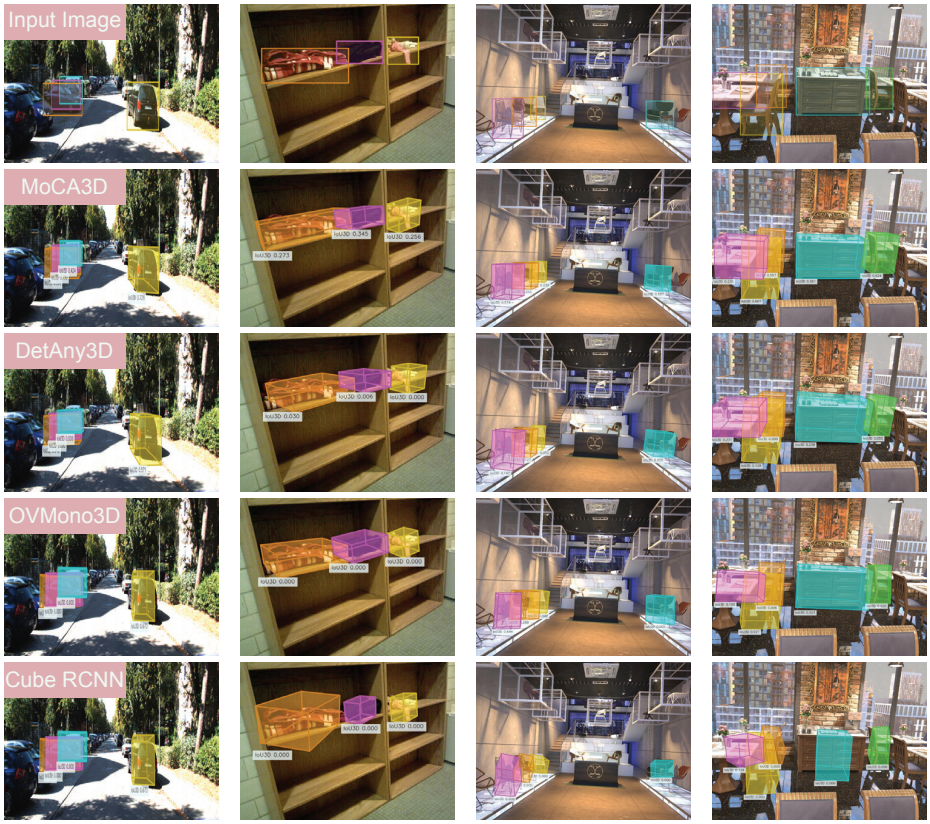

Fig. 7: Qualitative comparisons under oracle 2D boxes. MoCA3D is compared with DetAny3D, OVMono3D, and Cube R-CNN on representative samples. Each prediction is shown together with its 3D IoU score, which is also displayed inside the corresponding image. Across diverse scenes, MoCA3D generally yields more accurate and visually consistent 3D geometry.

either independent random masking or, with probability 0.4, block masking with block size 4×4. When enabled, masked tokens are also excluded from transformer attention.

C Additional Experimental Results

C.1 Qualitative Comparisons

Fig. 7 presents qualitative comparisons under the oracle-2D setting on samples from Omni3D. We compare MoCA3D with DetAny3D [55], OVMono3D [53], and Cube R-CNN [5]. The predicted 3D boxes are visualized together with their 3D IoU scores, which are also shown inside each image for direct reference. Overall, MoCA3D produces more accurate and visually consistent projected cuboids,

Fig. 8: Additional driving scene generation results. Additional examples of controllable driving-scene editing using MoCA3D-recovered image-plane geometry.

especially for challenging cross-domain cases involving scale variation, clutter, and indoor layouts.

C.2 Driving Scene Generation Results

Fig. 8 provides additional qualitative results for controllable driving-scene generation. We follow the same pipeline used in the main paper, where MoCA3D first recovers image-plane vehicle geometry from a single real image, and the projected cuboid corners are then used as control signals in the generation pipeline. By keeping the geometric control fixed while changing only the text prompt, the pipeline produces diverse scene variations such as heavy rain and night glare while preserving viewpoint, placement, and road layout. Even under substantial appearance changes, the results remain consistent with the source image.

C.3 Failure Cases

Fig. 9 illustrates representative failure cases of MoCA3D. A common failure mode arises when an object is truncated by the image boundary. Since MoCA3D

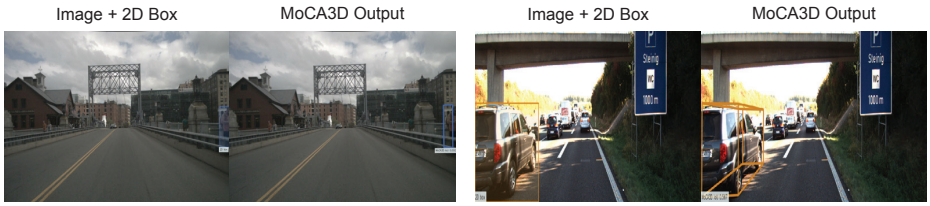

Fig. 9: Failure Cases. We show representative failure cases of MoCA3D. For each example, the left panel shows the input image with the tight oracle 2D box, and the right panel shows the corresponding MoCA3D prediction.

predicts pixel-aligned geometry directly in the image plane, its predictions are naturally constrained by the visible image extent. As a result, when a portion of the object lies outside the frame, the predicted corners may collapse toward the boundary or yield inaccurate 3D geometry.

This behavior highlights a boundary case of image-plane geometry prediction rather than a failure of the overall representation. Incorporating explicit reasoning for partially unobserved object extent is a promising direction for future work.

D Code and Reproducibility

All results in this appendix correspond to the same models, training protocol, and experimental setup described in the main paper. The code for MoCA3D and MoCA3D-Cube, including the core training and evaluation scripts used in our experiments, will be released publicly upon publication.

D.1 Implementation Details

Unless otherwise noted, MoCA3D is trained in feature mode with pre-extracted DINOv3 features rather than end-to-end RGB backbone optimization. The model uses a DINOv3 image size of 512, predicts heatmaps at resolution 128×128, and applies soft-argmax with $\beta = 100$. A 1×1 projection maps the 1024-channel DINOv3 feature to the transformer dimension $d = 384$. Both the transformer encoder and decoder use 4 layers with 6 attention heads and feed-forward dimension 1024. The model predicts corner heatmaps and corner depth maps, and uses ReLU activation with dropout 0.15 throughout. The depth head uses a hidden dimension of 256, bottleneck dimension of 64, and minimum depth of 10^{-3}.

Training is performed for 120 epochs with 5 warm-up epochs, effective batch size 192, learning rate 2×10^{-4}, and weight decay 0.02. We use decoder and box-embedding learning-rate multipliers of 1.5 and 2.0, respectively. The training epoch length is fixed to 50,000 samples, validation is performed every 5 epochs, and checkpoints are saved every 60 epochs.

D.2 Details of Ablation Models

MoCA3D with Depth-Anything This variant augments MoCA3D with depth predictions from Depth Anything3 [23], namely a monocular depth output and a metric depth output. All notations in this section follow the main paper. Concretely, given an input image I, we first extract:

$$\mathbf{D}^{\text{mono}} = f_{\text{mono}}(I), \qquad \mathbf{D}^{\text{metric}} = f_{\text{metric}}(I), \tag{22}$$

where $f_{\text{mono}}, f_{\text{metric}}$ denote the two Depth-Anything branches, and the two depth outputs are embedded by a dedicated depth embedder:

$$\mathbf{E}^{\text{da}} = \phi_{\text{da}}(\mathbf{D}^{\text{mono}}, \mathbf{D}^{\text{metric}}), \tag{23}$$

where $\mathbf{E}^{\text{da}} \in \mathbb{R}^{C \times h_d \times w_d}$. Additionally, we inject the box prior map into $\mathbf{E}^{\text{da}}$:

$$\tilde{\mathbf{E}}^{\text{da}} = \mathbf{E}^{\text{da}} + \alpha \mathbf{E}_{\text{prior}}, \tag{24}$$

so that both the appearance stream and the auxiliary depth stream are aligned to the same box-relative geometry.

Unlike the baseline MoCA3D, this variant replaces the encoder with a depth-aware fusion module that integrates appearance and auxiliary depth tokens. After adding standard 2D positional encodings and flattening both $\tilde{\mathbf{F}}$ and $\tilde{\mathbf{E}}^{\text{da}}$ into token sequences, the module applies a decoder-like layer structure consisting of self-attention, cross-attention, and a feed-forward network. In particular, the DINOv3 token stream serves as the query sequence, while the depth-aware tokens are used as keys and values in cross-attention. The fused token sequence is then reshaped back to the feature grid and passed to the same decoder and dense heads as in MoCA3D. Therefore, this ablation modifies only the feature fusion stage, enabling a controlled comparison of whether auxiliary depth cues improve geometry recovery.

Direct Regression Model This ablation replaces MoCA3D's dense upsampling and prediction heads with a direct regression head. Instead of predicting eight heatmaps and eight depth maps, the direct model first fuses the decoder feature and the skip feature by channel-wise concatenation followed by a fusion block. Given the fused feature map, we apply RoI-Align to the input 2D box with output size of 7×7. The cropped feature is then channel-wise compressed, flattened, and passed through a two-layer MLP to directly regress an output vector.

The output is interpreted as the image-plane 3D geometry itself: the first 16 values correspond to the projected eight corners, and the remaining 8 values correspond to the assigned depths. Compared with MoCA3D, this variant removes the dense pixel-aligned formulation, and the task is reduced to an RoI-to-vector regression, making this ablation a clean control experiment for testing whether MoCA3D's gains arise from its dense geometry prediction design.

References

1. Ahmadyan, A., Zhang, L., Ablavatski, A., Wei, J., Grundmann, M.: Objectron: A large scale dataset of object-centric videos in the wild with pose annotations. CVPR (2021)
2. Baruch, G., Chen, Z., Dehghan, A., Dimry, T., Feigin, Y., Fu, P., Gebauer, T., Joffe, B., Kurz, D., Schwartz, A., Shulman, E.: ARKitscenes - a diverse real-world dataset for 3d indoor scene understanding using mobile RGB-d data. In: NeurIPS Datasets and Benchmarks Track (Round 1) (2021)
3. Bhat, S.F., Mitra, N., Wonka, P.: Loosecontrol: Lifting controlnet for generalized depth conditioning. In: ACM SIGGRAPH 2024 Conference Papers. pp. 1–11 (2024)
4. Bian, W., Wang, Z., Vedaldi, A.: Catfree3d: Category-agnostic 3d object detection with diffusion. In: 2025 International Conference on 3D Vision (3DV). pp. 101–111. IEEE (2025)
5. Brazil, G., Kumar, A., Straub, J., Ravi, N., Johnson, J., Gkioxari, G.: Omni3D: A large benchmark and model for 3D object detection in the wild. In: CVPR. IEEE, Vancouver, Canada (June 2023)
6. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: CVPR (2020)
7. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European conference on computer vision. pp. 213–229. Springer (2020)
8. Choi, C., Baek, S.M., Lee, S.: Real-time 3d object pose estimation and tracking for natural landmark based visual servo. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 3983–3989. IEEE (2008)
9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009). https://doi.org/10.1109/CVPR.2009.5206848
10. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: CVPR (2012)
11. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. pp. 315–323. JMLR Workshop and Conference Proceedings (2011)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
13. He, X., Zhou, S., Venkateswaran, T., Zheng, K., Wan, Z., Kadambi, A., Wang, X.E.: Morphosim: An interactive, controllable, and editable language-guided 4d world simulator. arXiv preprint arXiv:2510.04390 (2025)
14. Kabsch, W.: A solution for the best rotation to relate two sets of vectors. Foundations of Crystallography **32**(5), 922–923 (1976)
15. Kim, J., Lee, G., Kim, J.S., Kim, H.J., Kim, K.: Monocular 3d object detection for an indoor robot environment. In: 2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN). pp. 438–445. IEEE (2020)
16. Krishnan, A., Kundu, A., Maninis, K.K., Hays, J., Brown, M.: Omninocs: A unified nocs dataset and model for 3d lifting of 2d objects. In: European Conference on Computer Vision. pp. 127–145. Springer (2024)

17. Labbé, Y., Carpentier, J., Aubry, M., Sivic, J.: Cosypose: Consistent multi-view multi-object 6d pose estimation. In: European conference on computer vision. pp. 574–591. Springer (2020)
18. Lassner, C., Zollhöfer, M.: Pulsar: Efficient sphere-based neural rendering. arXiv:2004.07484 (2020)
19. Law, H., Deng, J.: Cornernet: Detecting objects as paired keypoints. In: Proceedings of the European conference on computer vision (ECCV). pp. 734–750 (2018)
20. Li, F., Zhang, H., Liu, S., Guo, J., Ni, L.M., Zhang, L.: Dn-detr: Accelerate detr training by introducing query denoising. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 13619–13627 (2022)
21. Li, P., Zhao, H.: Monocular 3d detection with geometric constraint embedding and semi-supervised training. IEEE Robotics and Automation Letters **6**(3), 5565–5572 (2021)
22. Li, P., Zhao, H., Liu, P., Cao, F.: Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving. In: European Conference on Computer Vision. pp. 644–660. Springer (2020)
23. Lin, H., Chen, S., Liew, J.H., Chen, D.Y., Li, Z., Shi, G., Feng, J., Kang, B.: Depth anything 3: Recovering the visual space from any views. arXiv preprint arXiv:2511.10647 (2025)
24. Liu, L., Li, H., Gruteser, M.: Edge assisted real-time object detection for mobile augmented reality. In: The 25th annual international conference on mobile computing and networking. pp. 1–16 (2019)
25. Liu, S., Li, F., Zhang, H., Yang, X., Qi, X., Su, H., Zhu, J., Zhang, L.: Dab-detr: Dynamic anchor boxes are better queries for detr. arXiv preprint arXiv:2201.12329 (2022)
26. Liu, Z., Wu, Z., Tóth, R.: Smoke: Single-stage monocular 3d object detection via keypoint estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. pp. 996–997 (2020)
27. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016)
28. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
29. Luvizon, D.C., Tabia, H., Picard, D.: Human pose regression by combining indirect part detection and contextual information. Computers & Graphics **85**, 15–22 (2019)
30. Meng, D., Chen, X., Fan, Z., Zeng, G., Li, H., Yuan, Y., Sun, L., Wang, J.: Conditional detr for fast training convergence. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 3651–3660 (2021)
31. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3d bounding box estimation using deep learning and geometry. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 7074–7082 (2017)
32. Pandey, K., Guerrero, P., Gadelha, M., Hold-Geoffroy, Y., Singh, K., Mitra, N.J.: Diffusion handles enabling 3d edits for diffusion models by lifting activations to 3d. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7695–7704 (2024)
33. Park, Y., Lepetit, V., Woo, W.: Multiple 3d object tracking for augmented reality. In: 2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality. pp. 117–120. IEEE (2008)
34. Qin, Z., Wang, J., Lu, Y.: Monogrnet: A general framework for monocular 3d object detection. IEEE transactions on pattern analysis and machine intelligence **44**(9), 5170–5184 (2021)

35. Ravi, N., Gabeur, V., Hu, Y.T., Hu, R., Ryali, C., Ma, T., Khedr, H., Rädle, R., Rolland, C., Gustafson, L., Mintun, E., Pan, J., Alwala, K.V., Carion, N., Wu, C.Y., Girshick, R., Dollár, P., Feichtenhofer, C.: Sam 2: Segment anything in images and videos. arXiv preprint arXiv:2408.00714 (2024), https://arxiv.org/abs/2408.00714
36. Roberts, M., Ramapuram, J., Ranjan, A., Kumar, A., Bautista, M.A., Paczan, N., Webb, R., Susskind, J.M.: Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In: ICCV (2021)
37. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10684–10695 (2022)
38. Sajnani, R., Vanbaar, J., Min, J., Katyal, K.D., Sridhar, S.: Geodiffuser: Geometry-based image editing with diffusion models. In: Proceedings of the Winter Conference on Applications of Computer Vision. pp. 472–482 (2025)
39. Siméoni, O., Vo, H.V., Seitzer, M., Baldassarre, F., Oquab, M., Jose, C., Khalidov, V., Szafraniec, M., Yi, S., Ramamonjisoa, M., Massa, F., Haziza, D., Wehrstedt, L., Wang, J., Darcet, T., Moutakanni, T., Sentana, L., Roberts, C., Vedaldi, A., Tolan, J., Brandt, J., Couprie, C., Mairal, J., Jégou, H., Labatut, P., Bojanowski, P.: DINOv3 (2025), https://arxiv.org/abs/2508.10104
40. Song, S., Lichtenberg, S.P., Xiao, J.: Sun rgb-d: A rgb-d scene understanding benchmark suite. In: CVPR (2015)
41. Swerdlow, A., Xu, R., Zhou, B.: Street-view image generation from a bird's-eye view layout. IEEE Robotics and Automation Letters **9**(4), 3578–3585 (2024)
42. Tekin, B., Sinha, S.N., Fua, P.: Real-time seamless single shot 6d object pose prediction. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 292–301 (2018)
43. Tong, S., Brown, E., Wu, P., Woo, S., Middepogu, M., Akula, S.C., Yang, J., Yang, S., Iyer, A., Pan, X., Wang, A., Fergus, R., LeCun, Y., Xie, S.: Cambrian-1: A fully open, vision-centric exploration of multimodal llms (2024)
44. Tremblay, J., To, T., Sundaralingam, B., Xiang, Y., Fox, D., Birchfield, S.: Deep object pose estimation for semantic robotic grasping of household objects. arXiv preprint arXiv:1809.10790 (2018)
45. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
46. Wan, B., Shi, Y., Xu, K.: Socs: Semantically-aware object coordinate space for category-level 6d object pose estimation under large shape variations. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 14065–14074 (2023)
47. Wang, G., Wu, J., Tian, B., Teng, S., Chen, L., Cao, D.: Centernet3d: An anchor free object detector for point cloud. IEEE Transactions on Intelligent Transportation Systems **23**(8), 12953–12965 (2021)
48. Wang, H., Sridhar, S., Huang, J., Valentin, J., Song, S., Guibas, L.J.: Normalized object coordinate space for category-level 6d object pose and size estimation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
49. Wu, Z., Rubanova, Y., Kabra, R., Hudson, D.A., Gilitschenski, I., Aytar, Y., Van Steenkiste, S., Allen, K.R., Kipf, T.: Neural assets: 3d-aware multi-object scene synthesis with image diffusion models. Advances in Neural Information Processing Systems **37**, 76289–76318 (2024)

50. Yang, K., Ma, E., Peng, J., Guo, Q., Lin, D., Yu, K.: Bevcontrol: Accurately controlling street-view elements with multi-perspective consistency via bev sketch layout. arXiv preprint arXiv:2308.01661 (2023)
51. Yang, Y.H., Piccinelli, L., Segu, M., Li, S., Huang, R., Fu, Y., Pollefeys, M., Blum, H., Bauer, Z.: 3d-mood: Lifting 2d to 3d for monocular open-set object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7429–7439 (2025)
52. Yang, Z., Guo, X., Ding, C., Wang, C., Wu, W., Zhang, Y.: Instadrive: Instance-aware driving world models for realistic and consistent video generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 25410–25420 (2025)
53. Yao, J., Gu, H., Chen, X., Wang, J., Cheng, Z.: Open vocabulary monocular 3d object detection. arXiv preprint arXiv:2411.16833 (2024)
54. Yu, Y., He, X., Zhao, C., Yu, J., Yang, J., Hu, R., Shen, Y., Zhu, X., Zhou, X., Peng, S.: Boxdreamer: Dreaming box corners for generalizable object pose estimation. arXiv preprint arXiv:2504.07955 (2025)
55. Zhang, H., Jiang, H., Yao, Q., Sun, Y., Zhang, R., Zhao, H., Li, H., Zhu, H., Yang, Z.: Detect anything 3d in the wild. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5048–5059 (2025)
56. Zhang, H., Alaluf, Y., Ma, S., Kadambi, A., Wang, J., Aberman, K.: Instantrestore: Single-step personalized face restoration with shared-image attention. In: Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers. SIGGRAPH Conference Papers '25, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3721238.3730628, https://doi.org/10.1145/3721238.3730628
57. Zhang, J., Sheng, H., Cai, S., Deng, B., Liang, Q., Li, W., Fu, Y., Ye, J., Gu, S.: Perldiff: Controllable street view synthesis using perspective-layout diffusion model. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 26306–26315 (2025)
58. Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. arXiv preprint arXiv:1904.07850 (2019)
59. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020)