RefineAny3D: Depth Refinement as Semantic Alignment for Monocular 3D Detection

Zhihao Zhang[1] **Gengwei Zhang**[2] **Tianlong Chen**[2] **Xiaoming Liu**[1,2]
[1]Michigan State University [2]University of North Carolina at Chapel Hill

Abstract

Monocular 3D object detection spans two regimes: closed-set detectors operating within a fixed category vocabulary, and open-vocabulary detectors that localize arbitrary categories by leveraging depth foundation models for 3D geometry. We find that current depth foundation models, despite their strong zero-shot generalization, lack the object-level precision 3D detection demands: substituting a state-of-the-art depth foundation model for a strong detector's predicted depth degrades accuracy. Rather than pushing detectors or depth models to be more accurate end-to-end, we treat object-level depth refinement as a stand-alone task and present RefineAny3D, a vision-language model that corrects depth without ever predicting a numerical value. Our key insight is that depth error has a direct visual signature in image space: when projected onto the image, a correctly placed box tightly encloses the object, while a too-far box projects too small and a too-close box projects too large. Depth refinement thus reduces to a visual alignment problem rather than a metric regression problem, which we instantiate by extending the VLM's vocabulary with action tokens that replace numerical depth output with categorical decisions, and by supervising the model on a large-scale chain-of-thought dataset that grounds each decision in explicit visual evidence. Applied as a single post-hoc step, RefineAny3D delivers consistent gains across closed-set detectors, open-vocabulary detectors, and 3D auto-labeling tools, and generalizes to novel categories, scenes, and cameras without retraining. Code is available at https://github.com/alanzhangcs/RefineAny3D.

1 Introduction

Monocular 3D object detection (Mono3D), which aims to localize and recognize objects in 3D space from a single RGB image, is a fundamental problem with broad applications in autonomous driving [10, 12], robotics [70, 103], and augmented reality [43]. Compared to LiDAR-, radar-, or multi-view-based alternatives [85, 61, 48, 47, 54], Mono3D is more cost-effective and scalable, yet the absence of explicit depth cues makes accurate 3D reasoning inherently challenging. Existing Mono3D detectors broadly fall into two regimes. *Closed-set* detectors [62, 88] are trained and evaluated on a fixed category vocabulary within a single dataset; while accurate in-domain, they cannot handle novel categories. *Open-vocabulary* 3D detectors [83, 82, 87], in contrast, aim to recognize and localize arbitrary categories in unseen domains, a setting where category-specific geometric priors no longer apply and depth must be recovered from generic visual cues alone.

This need has made depth foundation models [55, 69] the de facto source of 3D geometry for open-vocabulary Mono3D, since these models offer strong zero-shot generalization to unseen domains and categories. Yet despite this generalization, their predicted object depth is not precise enough at the level that 3D IoU thresholds demand: substituting a state-of-the-art depth foundation model [69] for a strong detector's predicted depth in fact degrades AP_{3D} by 3.68 on Omni3D [3] (Fig. 1; details in Appendix C). This depth bias is then inherited by every downstream pipeline that builds on these models, including auto-labeling tools [84, 73] that distill their predictions into pseudo-3D

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

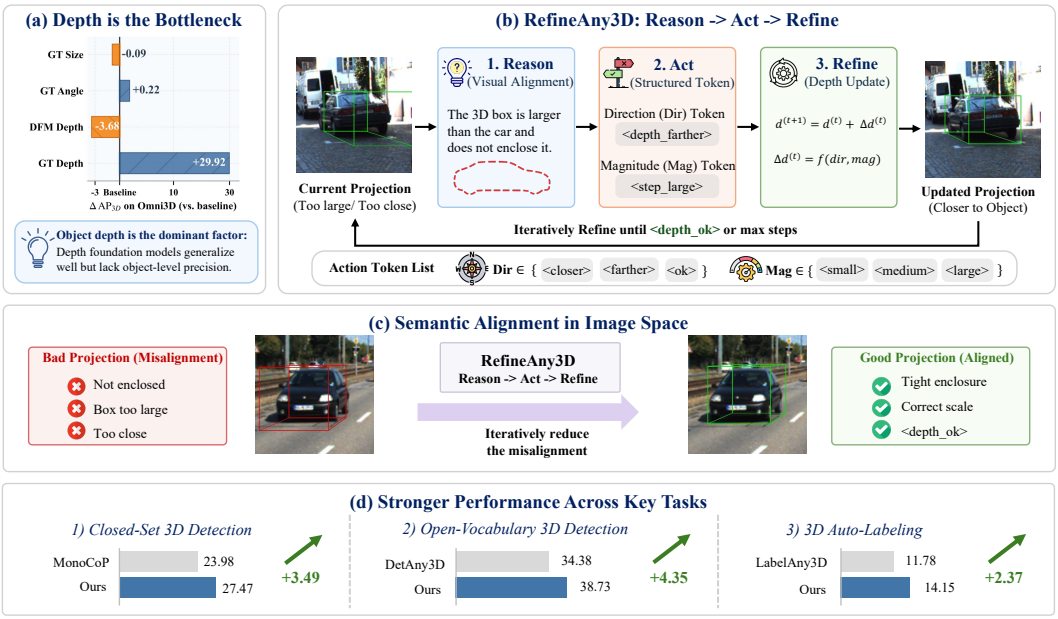


Figure 1: **RefineAny3D: 3D box refinement as semantic alignment in image space. (a)** Object depth is the dominant bottleneck of monocular 3D detection: replacing predicted depth with ground truth boosts AP_{3D} by 29.92, while substituting a depth foundation model (DFM) degrades it by 3.68. Relying on DFM alone is insufficient for accurate object-level depth. **(b)** RefineAny3D reformulates depth refinement as a semantic alignment problem in image space. Given a candidate 3D box projected onto the image, a vision-language model emits action tokens that iteratively adjust the depth until the projection aligns with the target or a step limit is reached. **(c)** Visual signature of depth error: a too-close box projects too large (left), while a correctly placed box tightly encloses the target object (right). **(d)** As a plug-and-play module, RefineAny3D consistently improves closed-set detectors, open-vocabulary detectors, and 3D auto-labeling methods.

annotations. The object-level geometric precision of open-vocabulary Mono3D systems thus remains a fundamental open challenge.

This observation motivates a different approach. Rather than pushing depth foundation models or open-vocabulary 3D detectors to be more accurate end-to-end, we ask: *can the residual object-depth error of any existing 3D pipeline be corrected after the fact, by a generic refinement module*? We formalize this as the task of **3D box refinement**: given an image and a candidate 3D box from any upstream source (a closed-set detector, an open-vocabulary detector, or an auto-labeling tool), output a refined 3D box whose object depth is closer to the ground truth, leaving other attributes untouched. A successful refinement module is plug-and-play, depends on no specific upstream architecture, and generalizes across diverse object categories, scene types, and camera setups.

We introduce RefineAny3D, the first refinement module designed for this task. Our key insight is that the correctness of an object's depth has a direct visual signature: when a candidate 3D box is projected back onto the image, a correctly placed box tightly encloses the target object, while a too-far box projects too small and a too-close box too large. Refining depth therefore reduces to a familiar visual judgment that depends only on object appearance and 2D spatial layout, the kind of reasoning vision-language models (VLMs) are well positioned to make. We thus recast depth refinement from a metric regression problem into a semantic alignment problem in image space.

Realizing this idea is not straightforward, since off-the-shelf VLMs [68, 2, 1] are not trained to reason about projected 3D boxes or to relate small visual misalignments to depth corrections. We therefore curate a large-scale training dataset on top of Omni3D that pairs each candidate box with a chain-of-thought reasoning trace and a corresponding refinement decision, forcing the model to ground its decisions in explicit visual evidence. To make these decisions atomic and unambiguous, we further extend the VLM's vocabulary with a small set of dedicated action tokens, each carrying its own learnable embedding. This sidesteps the VLM's well-known weakness at numerical regression [35,

81]: the model never emits a continuous depth value, only one of a few discrete decisions, which are then chained iteratively to recover fine-grained precision.

Because refinement depends only on the local alignment between a projected box and its target object, RefineAny3D generalizes across categories, scenes, and cameras without retraining. We show that RefineAny3D, applied as a plug-and-play module, consistently improves the depth accuracy of closed-set 3D detectors, open-vocabulary 3D detectors, and 3D auto-labeling tools alike, establishing 3D box refinement as a practical capability that complements existing Mono3D systems.

Our contributions are as follows:

- We formalize 3D box refinement as a new task that operates on top of any upstream Mono3D method, positioning refinement as a capability orthogonal to existing detectors.
- We recast depth refinement from metric regression into semantic alignment in image space, where a VLM judges depth correctness from visual cues rather than predicting metric depth.
- We instantiate RefineAny3D with action tokens, a small set of dedicated VLM-vocabulary tokens that turn depth refinement into a sequence of categorical decisions, sidestepping the VLM's weakness at numerical regression and enabling iterative recovery of fine-grained precision.
- Applied as a plug-and-play module, RefineAny3D consistently improves closed-set detectors, open-vocabulary detectors, and 3D auto-labeling tools across novel categories, scenes, and cameras.

2 Related Work

Monocular 3D Detection. Mono3D methods can be broadly grouped into closed-set and open-vocabulary settings. Closed-set methods are trained and evaluated on a fixed category vocabulary, typically on benchmarks such as KITTI [20] and nuScenes [6]. Early CNN-based detectors adopt center-based prediction [46, 50, 71, 45], exploit 2D–3D geometric consistency [38, 49, 89, 76, 5, 72, 28], introduce depth-equivariant operators [31, 57], build on 2D detector backbones [4, 32, 3], lift image features into 3D voxel grids [41], or add bird's-eye-view segmentation supervision [33], while more recent transformer-based detectors [8, 11, 95, 24, 75, 97, 88, 56, 42, 93] have come to dominate these benchmarks. Others improve generalization across cameras by conditioning the detector on camera intrinsics [94] or by making depth robust to unseen camera heights [34]. In parallel, open-vocabulary methods generalize Mono3D beyond a fixed category set by scaling 3D training data or transferring foundation-model priors. Specifically, Omni3D [3] unifies existing 3D detection datasets into a single large-scale benchmark used to train unified detectors [39], auto-labeling pipelines [84, 73, 25, 64, 36] turn unlabeled images or videos into pseudo-annotated 3D data via 3D-aware foundation models [69], and methods such as OVMono3D [83], 3D-MOOD [82], and DetAny3D [87] couple 2D vision foundation models [65, 44] with depth foundation models [55, 69] to recognize and localize novel categories in unseen domains [74, 18]. Since RefineAny3D operates as a detector-agnostic refinement model, it can be applied to both streams, and our experiments show that object-level depth remains an under-addressed bottleneck rather than a detector-specific limitation.

Visual Prompting and Semantic Alignment. Recent work shows that multimodal models can benefit from explicit visual cues rendered directly on top of the input image. Early visual prompt engineering with red circles [63] demonstrates that simple marks can direct VLM attention to specific regions, while Set-of-Mark prompting [80], ViP-LLaVA [7], and Draw-and-Understand [40] extend this idea to boxes, masks, points, or arrows that make region references easier for MLLMs to parse. Omni-RGPT [21] and VideoRefer [86] further adapt such region cues to video understanding. These methods primarily use overlaid marks to improve visual understanding, grounding, and referring. Although our RefineAny3D follows the broader idea of reasoning over an overlaid visual cue, we use the projected 3D box as an alignment indicator of how the candidate depth should be refined.

Vision-Language Models for 3D Understanding. A growing body of work extends vision-language models (VLMs) toward 3D understanding. One line equips VLMs with a point-cloud encoder [22, 77, 51], often pre-trained to align 3D shapes with image and language representations [90]; RGB images are lifted into point clouds via a 3D foundation model [67] and a point-cloud-aware VLM is trained on top. Another line adopts a dual-backbone design [17, 19, 59] that augments the vision encoder with a 3D or depth foundation model [55, 69], or distills 3D features into the visual representation [96, 23, 37]. A third strengthens the VLM's ability to regress numerical 3D quantities through reinforcement learning or numerical supervision [9, 27, 60, 26, 58, 16]. These approaches

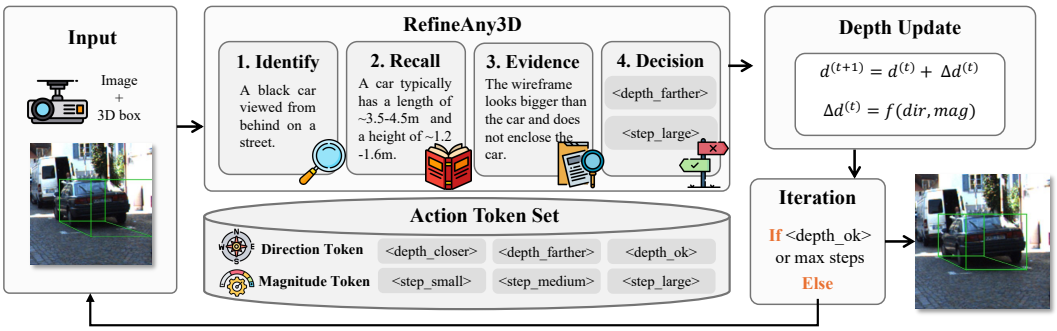


Figure 2: **Overview of RefineAny3D.** RefineAny3D recasts depth refinement as semantic alignment in image space: when a candidate 3D box is projected onto the image, depth error has a direct visual signature, with a too-far box projecting too small and a too-close box too large. We render the projected box as a wireframe overlay on the image, and a vision-language model performs chain-of-thought reasoning over the visual misalignment to emit two action tokens specifying the corrective *direction* (closer / ok / farther) and *magnitude* (small / medium / large). These tokens define an object-relative depth update that is applied to the box, and the loop repeats with the updated wireframe until the model emits ⟨`depth_ok`⟩ or a step cap is reached.

all task the VLM with directly producing 3D output, whether raw coordinates or metric depth. RefineAny3D takes a different route: we avoid VLM 3D prediction altogether and instead reduce 3D refinement to a 2D semantic alignment problem in image space, solved by emitting a small set of discrete decisions. From another perspective, the use of discrete action outputs also appears in vision-language-action (VLA) models for robotic control [104, 30]. Differently, RefineAny3D uses action tokens to represent relative depth-refinement decisions inferred from projected-box alignment, and recovers metric precision for monocular 3D detection through iterative updates.

3 Approach

Overview. RefineAny3D refines the object depth of a candidate 3D box by reformulating depth correction as a 2D semantic alignment problem (Fig. 2). Given a noisy box, RefineAny3D projects it onto the image, queries a vision-language model to judge how the projection misaligns with the target object, and updates the depth based on a discrete action emitted by the model. The same step is applied iteratively, so that fine-grained corrections emerge from chaining categorical decisions rather than regressing a continuous metric quantity. Sec. 3.1 formalizes the alignment view, Sec. 3.2 introduces the discrete action tokens that realize it, and Sec. 3.3 describes the supervision and inference protocols.

Preliminaries: Monocular 3D Detection. A Mono3D detector takes an RGB image $\mathbf{I} \in \mathbb{R}^{H\times W\times 3}$ and camera intrinsics $\mathbf{K}$ as input, and outputs a set of 3D boxes $\{\mathbf{b}_i\}_{i=1}^{N}$. Each box is parameterized as $\mathbf{b} = (x, y, z, w, h, l, \theta)$, where (x, y, z) is the object center in the camera coordinate frame, (w, h, l) are the box dimensions, and θ is the yaw angle. We write $d = z$ for the object depth, the component of the center along the camera's optical axis. RefineAny3D takes such a candidate box $\mathbf{b}$ as input and outputs a refined box $\mathbf{b}'$ with corrected depth.

3.1 Depth Refinement as Semantic Alignment

Directly regressing a corrected metric depth is difficult: depth is a continuous, scale-sensitive quantity that depends jointly on camera intrinsics, object geometry, and scene context. Our key observation is that once the candidate 3D box is projected onto the image plane, the camera intrinsics and metric scale are absorbed into the projection itself, and what remains is a purely 2D question: *does the projected box tightly enclose the target object?* If the projection is too small, the box is too far; if too large, too close. The hard metric problem of correcting depth thereby reduces to a semantic alignment task in image space, judged entirely from 2D visual evidence. Sec. 5.5 shows that simpler non-VLM alternatives to this judgment fall short.

To realize this view, we cast depth refinement as a conditional generation problem over a VLM's token sequence. At step t, the candidate box $\mathbf{b}^{(t)}$ is projected onto the image and rendered as a wireframe overlay:

$$\mathbf{I}^{(t)} = \mathrm{render}\Big(\mathbf{I},\ \pi(\mathbf{b}^{(t)};\ \mathbf{K})\Big)\,, \tag{1}$$

where $\pi(\mathbf{b};\mathbf{K})$ projects the eight corners of $\mathbf{b}$ to 2D image coordinates using $\mathbf{K}$. The rendered image $\mathbf{I}^{(t)}$ encodes the current alignment hypothesis: a VLM looking at $\mathbf{I}^{(t)}$ can compare the wireframe with the underlying object directly, without ever computing distances in metric space.

Conditioned on $\mathbf{I}^{(t)}$, the VLM autoregressively generates a single token sequence

$$\mathcal{S}^{(t)} = \big(\mathbf{r}^{(t)},\ \mathbf{a}^{(t)}\big), \tag{2}$$

where $\mathbf{r}^{(t)}$ is a chain-of-thought reasoning trace and $\mathbf{a}^{(t)}$ is a compact action; the concrete form of $\mathbf{a}^{(t)}$ is given in Sec. 3.2. The autoregressive nature of this single generation pass implies the following decomposition by the chain rule:

$$P\Big(\mathcal{S}^{(t)} \mid \mathbf{I}^{(t)}\Big) = \underbrace{P\Big(\mathbf{r}^{(t)} \mid \mathbf{I}^{(t)}\Big)}_{\text{visual reasoning}} \cdot \underbrace{P\Big(\mathbf{a}^{(t)} \mid \mathbf{I}^{(t)},\ \mathbf{r}^{(t)}\Big)}_{\text{action prediction}}, \tag{3}$$

where the action prediction is conditioned on the reasoning trace produced earlier in the same sequence. This decomposition exposes two complementary design choices that drive the rest of the method: the reasoning prefix $\mathbf{r}^{(t)}$ grounds the action in explicit visual reasoning about the current projection, and the action $\mathbf{a}^{(t)}$ is structured as discrete decisions to sidestep the VLM's known weakness at numerical regression.

3.2 From Alignment to Action Tokens

The formulation in Eq. (3) reduces depth refinement to producing an action $\mathbf{a}^{(t)}$ at each step. The most direct realization would be to let the VLM verbalize this action in natural language (*e.g.*, "the box is slightly too far") or, even more directly, regress a numerical correction Δd. Both options inherit the VLM's well-known unreliability at fine-grained numerical or open-ended outputs. We instead represent $\mathbf{a}^{(t)}$ with a small set of *action tokens* that encode the action categorically: a direction token a_d specifying how the box should move, paired with a magnitude token a_m specifying how strongly. We extend the VLM's vocabulary with six new special tokens (three for direction, three for magnitude), each carrying its own learnable embedding tuned for depth adjustment, defining a closed output space that simplifies decoding and supervision.

Direction tokens. The direction token $a_d \in \{\langle\texttt{depth_closer}\rangle,\ \langle\texttt{depth_ok}\rangle,\ \langle\texttt{depth_farther}\rangle\}$ encodes the qualitative outcome of the alignment check: $\langle\texttt{depth_closer}\rangle$ when the projected box is too small (the box should be moved closer to the camera, decreasing depth), $\langle\texttt{depth_farther}\rangle$ when it is too large (the box should be moved farther away, increasing depth), and $\langle\texttt{depth_ok}\rangle$ when the projection already aligns with the target. We map them to scalar signs by

$$\mathrm{dir}(\langle\texttt{depth_closer}\rangle) = -1, \quad \mathrm{dir}(\langle\texttt{depth_ok}\rangle) = 0, \quad \mathrm{dir}(\langle\texttt{depth_farther}\rangle) = +1. \tag{4}$$

Discretizing direction into three classes turns this part of the action into a classification problem that VLMs handle reliably; the precision lost by discretization is recovered through iterative refinement at inference time.

Magnitude tokens. The magnitude token $a_m \in \{\langle\texttt{step_small}\rangle,\ \langle\texttt{step_medium}\rangle,\ \langle\texttt{step_large}\rangle\}$ specifies how strongly to adjust along the chosen direction. A fixed metric step size would be inappropriate here: a $0.5\,\mathrm{m}$ shift is negligible for a distant truck but catastrophic for a nearby cup. We therefore tie magnitudes to the object's own characteristic size $s_{\text{obj}} = (w+h+l)/3$, the arithmetic mean of the box's 3D dimensions, so that step sizes are *relative* to the object scale:

$$\mathrm{mag}(a_m,\ s_{\text{obj}}) = \alpha_{a_m}\, s_{\text{obj}}, \quad \alpha_{\langle\texttt{step_small}\rangle} < \alpha_{\langle\texttt{step_medium}\rangle} < \alpha_{\langle\texttt{step_large}\rangle}. \tag{5}$$

This scale-relative design keeps the same action semantically consistent across objects of different sizes, which is important for generalization across categories and scenes. The full action is then $\mathbf{a}^{(t)} = (a_d^{(t)}, a_m^{(t)})$, and the depth update becomes

$$\Delta d^{(t)} = \mathrm{dir}(a_d^{(t)}) \cdot \mathrm{mag}(a_m^{(t)},\ s_{\text{obj}}). \tag{6}$$

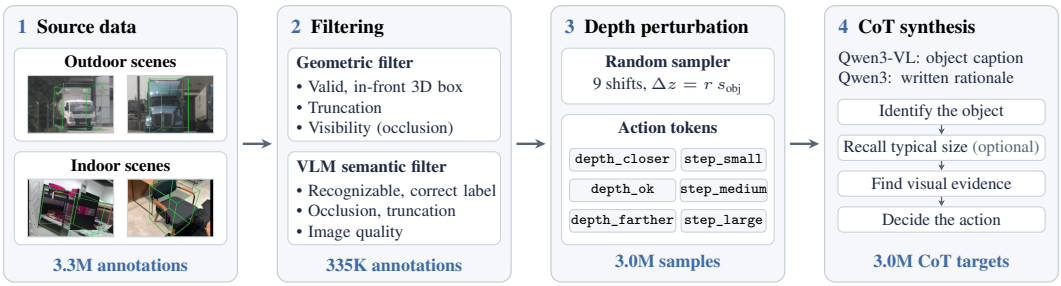


Figure 3: **Data construction pipeline for training RefineAny3D.** Scenes are selected, filtered by geometric and VLM-based checks, perturbed in depth with balanced action-token targets, and paired with chain-of-thought rationales; details in Appendix D.

3.3 Training and Inference

RefineAny3D is trained in two stages, both using single-step supervision: given a noisy box paired with its ground-truth chain-of-thought and target action tokens, the model predicts (a_d, a_m) in one pass. In Stage 1, we initialize each new action-token embedding to the mean embedding of a short phrase describing its alignment semantics, freeze the VLM, and train only the input and output embeddings of these six tokens to stabilize them before joint optimization. In Stage 2, we jointly fine-tune the LLM backbone and the action-token embeddings while freezing the vision encoder, preserving the general visual priors from VLM pretraining that we find essential for generalization across novel categories, scenes, and cameras. At inference, although trained with single-step supervision, RefineAny3D refines a box iteratively: at each step we render the current box onto the image, query the VLM for an action, and update the depth, until the model emits $\langle$`depth_ok`$\rangle$ or a safeguard cap of $T_{\max}$ steps is reached. This train-inference asymmetry is justified because each action is a local decision conditioned only on the current visual state, so single-step supervision generalizes naturally to multi-turn refinement.

4 Data Curation at Scale

Training RefineAny3D requires large-scale triplets of (image, noisy 3D box, refinement target), where each target encodes the visual reasoning that maps a perturbed projection back to a corrective action. We curate this data from Omni3D [3] through a four-stage pipeline (Fig. 3) summarized below; full construction details are in Appendix D.

Source datasets. We build on the Omni3D release, which unifies six existing 3D detection datasets into a single benchmark under a shared camera convention, spanning indoor and outdoor scenes, photographic and synthetic imagery, and fifty object categories. This diversity is essential for RefineAny3D's generalization across categories, scenes, and cameras.

Annotation filtering. The raw release contains many annotations unsuitable as positive training examples, including invalid 3D boxes, severe occlusion, and mis-categorized labels. We apply a two-stage filter, combining geometric checks on Omni3D's metadata fields with a semantic check that queries a VLM to verify object recognizability and label correctness, retaining 335K annotations.

Depth perturbation. For each filtered annotation, we synthesize noisy 3D boxes by perturbing the ground-truth (GT) depth along the camera ray while keeping all other attributes fixed. Perturbation magnitudes are tied to the object's own size s_{obj} to match the object-relative action design in Sec. 3.2, and stratified so that all action-token outcomes are uniformly represented. This yields approximately 3.0M training samples with an action-token distribution balanced by construction.

Chain-of-thought reasoning. Supervising only the final action tokens leaves the model free to take shortcuts that do not generalize. We therefore pair each sample with a chain-of-thought (CoT) prefix (Fig. 4) that the model produces before emitting the action. Each CoT is synthesized by a two-model pipeline: a VLM [1] identifies the target object from the image, and a text-only LLM [79] composes this object identification together with the sample metadata into a reasoning trace consistent with the

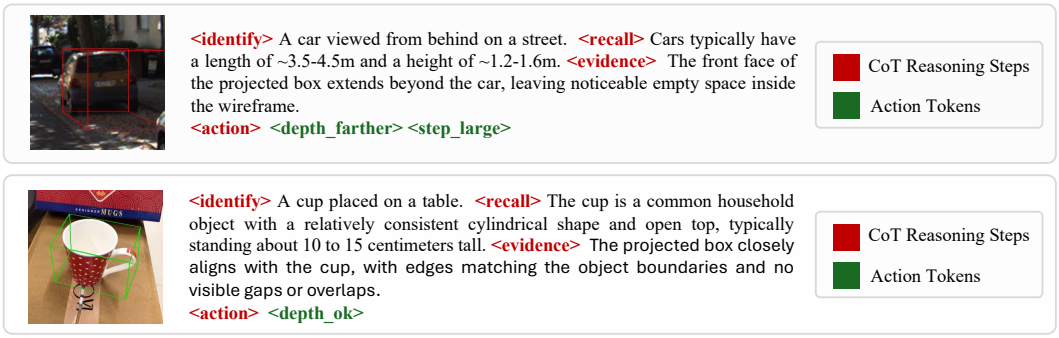


Figure 4: **Examples of CoT-based supervision.** Each training sample pairs a rendered image of a projected 3D wireframe with a chain-of-thought reasoning target that grounds the corrective action in explicit visual evidence before emitting the action tokens.

Table 1: **Evaluation on Refine3D**, our standalone refinement benchmark on Omni3D. RefineAny3D outperforms both off-the-shelf and fine-tuned Qwen3-VL baselines across all three settings.

Method	Standard			Novel Category			Novel Camera		
	DirAcc ↑	FullAcc ↑	DepthErr ↓	DirAcc ↑	FullAcc ↑	DepthErr ↓	DirAcc ↑	FullAcc ↑	DepthErr ↓
Qwen3-VL-8B [1]	28.7	11.4	0.32	27.5	10.9	0.37	28.3	11.1	0.33
Qwen3-VL-8B [1]$_{w/\ Our\ Data}$	75.6	55.4	0.21	56.3	41.2	0.27	68.7	49.8	0.23
RefineAny3D (Ours)	**89.2**	**76.7**	**0.16**	**74.4**	**58.5**	**0.19**	**85.6**	**73.2**	**0.17**

corrective decision. This forces the model to ground every action in explicit visual evidence rather than memorize input-output shortcuts.

5 Experiments

We evaluate RefineAny3D along four complementary axes: as a standalone refiner on a controlled benchmark (Sec. 5.1), and as a plug-and-play module on top of three downstream pipelines, namely closed-set 3D detectors (Sec. 5.2), open-vocabulary 3D detectors (Sec. 5.3), and 3D auto-labeling methods (Sec. 5.4). We further show that simpler non-VLM mechanisms cannot replace the VLM (Sec. 5.5), and verify that fine-tuning for refinement does not erode the underlying VLM's general capability (Appendix H).

5.1 Refine3D: Refinement as a Standalone Capability

Refine3D is a controlled benchmark we construct on top of Omni3D: each test sample pairs a GT 3D box with a synthetically perturbed noisy version, and the task is to recover the GT. We evaluate on three settings: Standard (in-distribution), Novel Category (open-vocabulary split of OVMono3D [83]), and Novel Camera (rescaled images with updated intrinsics). We report DirAcc (direction accuracy), FullAcc (direction and magnitude both correct), and DepthErr (residual gap between refined and GT depths).

Tab. 1 reveals three findings. First, off-the-shelf Qwen3-VL-8B falls below the random-guess baselines ($\sim$28 DirAcc, $\sim$11 FullAcc), indicating systematic bias rather than uninformative priors. Second, fine-tuning Qwen3-VL with our data using plain-text actions lifts Standard performance substantially (75.6/55.4), confirming that the supervision signal is informative. Third, RefineAny3D outperforms both baselines on all settings and generalizes to both shifts: it retains nearly all accuracy under Novel Camera (-3.6 DirAcc, -3.5 FullAcc) thanks to its camera-agnostic alignment design, and under Novel Category its FullAcc (58.5) still exceeds the plain-text variant.

5.2 Improving Closed-Set 3D Detectors

We first ask whether RefineAny3D can improve the depth quality of strong, well-tuned closed-set detectors, even within their own training domain. We apply RefineAny3D as a post-hoc refinement

Table 3: **Improving open-vocabulary 3D detectors on Omni3D [3].** RefineAny3D refines DetAny3D [87] conditioned on ground-truth 2D boxes, the strongest variant of DetAny3D, evaluated across the six Omni3D sub-datasets. RefineAny3D lifts AP_{3D} by $+4.35$ overall through depth refinement alone, demonstrating that the gain holds even on top of an oracle-grade detector.

Method	AP_{3D}^{kit} ↑	AP_{3D}^{nus} ↑	AP_{3D}^{sun} ↑	AP_{3D}^{ark} ↑	AP_{3D}^{obj} ↑	AP_{3D}^{hyp} ↑	AP_{3D} ↑
Cube R-CNN [3]	32.50	30.06	15.33	41.73	50.84	7.48	23.26
OVMono3D [83]	25.45	24.33	15.20	41.60	58.87	7.75	22.98
DetAny3D [87]	31.61	30.97	18.96	46.13	54.42	7.17	24.92
DetAny3D [87]$_{w/\text{ Ground-Truth 2D Box}}$	38.68	37.55	46.14	50.62	56.82	15.98	34.38
RefineAny3D (Ours)	**43.47**	**44.13**	**48.58**	**54.20**	**59.29**	**17.80**	**38.73**

step on top of the recent state-of-the-art MonoCoP [93], evaluated on the KITTI [20] validation set under the standard AP_{3D} protocol at IoU ≥ 0.7.

As shown in Tab. 2, applying RefineAny3D on top of MonoCoP yields consistent gains across all difficulty levels, raising AP_{3D} from $32.06 \rightarrow 35.62$ on *Easy*, $23.98 \rightarrow 27.47$ on *Moderate*, and $20.64 \rightarrow 21.31$ on *Hard*. The improvements are most pronounced on the Easy and Moderate splits, where less occluded objects allow the visual signature of depth misalignment to be read most reliably. The result confirms that even mature closed-set detectors leave object-level depth as an under-addressed source of error, and that RefineAny3D recovers a substantial part of it without changing the underlying detector.

Table 2: **Improving closed-set 3D detectors** at IoU ≥ 0.7. Applying RefineAny3D on top of MonoCoP [93] yields consistent gains.

Method	**Val**, AP_{3D} (↑)		
	Easy	Mod.	Hard
MonoDETR [88]	28.84	20.61	16.38
MonoCD [78]	26.45	19.37	16.38
FD3D [76]	28.22	20.23	17.04
MonoMAE [29]	30.29	20.90	17.61
MonoTAKD [42]	34.36	22.61	19.88
MonoDGP [56]	30.76	22.34	19.02
MonoCoP [93]	32.06	23.98	20.64
RefineAny3D (Ours)	**35.62**	**27.47**	**21.31**

5.3 Improving Open-Vocabulary 3D Detectors

Open-vocabulary detectors face a much harder regime: the upstream detector must localize categories it has never been trained on, and current methods compensate by relying on depth foundation models for 3D geometry, the very source of bias we identified in Sec. 1. We test whether RefineAny3D can correct this propagated bias by applying it on top of the state-of-the-art DetAny3D [87] across the full Omni3D [3] test split. In Tab. 3, we apply RefineAny3D on top of DetAny3D conditioned on ground-truth 2D boxes (34.38 AP_{3D}), the strongest variant of DetAny3D in which 2D localization is already error-free. Even on this oracle baseline, refining only the object depth with RefineAny3D lifts AP_{3D} to 38.73 ($+4.35$), with consistent gains on every Omni3D sub-dataset. The improvement comes entirely from depth correction, since all other box attributes are held fixed, demonstrating that RefineAny3D's gains are not limited to weak detectors: it provides a meaningful and orthogonal correction even on top of an oracle-grade open-vocabulary detector. Fig. 5 illustrates this qualitatively, with most cases corrected in a single decisive step and a few requiring a second pass; we quantify convergence behavior and inference cost in Appendix F, and provide additional comparisons in Appendix G.

5.4 Improving 3D Auto-Labeling Methods

Beyond detection, depth foundation models also propagate their bias into auto-labeling pipelines that distill them into pseudo-3D annotations. To evaluate whether RefineAny3D can correct this propagated bias, we adopt the following protocol: we use LabelAny3D [84] to auto-label a dataset with available GT, refine the resulting pseudo-labels with RefineAny3D, and train a downstream 3D detector [93] on the refined pseudo-labels. Detection accuracy on

Table 4: **Improving 3D auto-labeling methods.** RefineAny3D refines LabelAny3D's pseudo-labels, narrowing the gap to GT-trained accuracy.

Method	Easy	Mod.	Hard
GT [20]	32.06	23.98	20.64
LabelAny3D [84]	13.57	11.78	10.27
RefineAny3D (Ours)	**15.76**	**14.15**	**10.99**

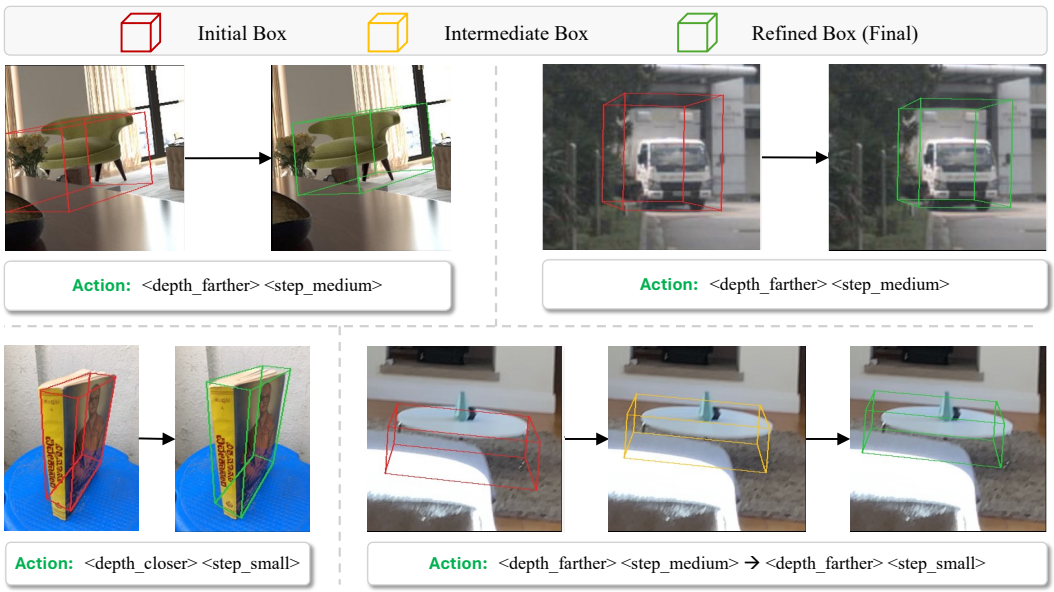


Figure 5: **Single-step and iterative depth refinement.** RefineAny3D refines DetAny3D [87]'s initial prediction (red) toward the refined result (green). Most cases are corrected in a single step; in some cases the model applies multiple action steps until the projected box aligns with the target.

the held-out validation set, compared against a model trained on the original GT, measures how faithfully the auto-labeled data approximates the GT.

Tab. 4 reports KITTI validation AP_{3D} for detectors trained on LabelAny3D's pseudo-labels with and without RefineAny3D refinement. Refining the pseudo-labels lifts the downstream detector by $+2.19/+2.37/+0.72$ on Easy / Moderate / Hard, narrowing the gap to GT-trained accuracy. Because the only change is a refinement pass on the auto-labels themselves, the gain isolates the contribution of RefineAny3D's depth correction to label quality, and shows that refinement directly translates into better trained detectors downstream.

5.5 Is a VLM Necessary?

We compare RefineAny3D with two simpler mechanisms that could in principle judge image-space alignment, both refining only the depth of MonoCoP predictions (Tab. 5). *Geometric projection fitting* searches along the camera ray for the depth whose projected cuboid best overlaps the object's tight 2D box, which we take from the ground truth to favor the baseline. It still lowers AP_{3D}, even with GT dimensions and yaw: a tight 2D box covers only the visible object, whereas the projected cuboid also spans occluded corners and empty volume, so the two objectives disagree; an 8% extent mismatch already shifts a car at 40 m by about 3.3 m. A *DINOv2 action classifier* [53], trained on the same data to emit the same action tokens, degrades accuracy even more: its crop-to-class mapping does not transfer from pure depth perturbations to the coupled errors of a real detector, and every wrong token becomes a metric error. The gains of RefineAny3D therefore come from the VLM's visual judgment rather than from the discrete action space itself; derivations and full details are in Appendix A.

Table 5: **Non-VLM refinement mechanisms.** All methods refine MonoCoP's depth on KITTI val (AP_{3D}, IoU ≥ 0.7). Geometric fitting uses the GT 2D box.

Method	Easy	Mod.	Hard
MonoCoP [93]	32.06	23.98	20.64
Geometric fitting	29.89	21.55	18.36
Geometric fitting (GT dims + yaw)	30.56	21.79	18.43
DINOv2 action classifier	24.71	17.62	12.78
RefineAny3D (Ours)	**35.62**	**27.47**	**21.31**

5.6 Ablation Studies

We ablate three design dimensions central to RefineAny3D: how to surface the corrective action to the VLM, how to initialize the new action-token embeddings, and how to set up the training recipe.

Table 6: **Ablation studies across three design dimensions of RefineAny3D.** Default configurations are highlighted in gray and outperform the alternatives.

Action vocabulary	DirAcc	FullAcc
Plain text	75.6	55.4
Joint tokens	86.8	72.5
Dir + Mag	**89.2**	**76.7**

Initialization	DirAcc	FullAcc
Random	86.8	73.2
Anchor	87.6	74.4
Semantic	**89.2**	**76.7**

Training recipe	DirAcc	FullAcc
No freeze vision	82.7	69.6
No CoT	84.5	70.2
Ours	**89.2**	**76.7**

Each ablation in Tab. 6 compares our default against natural alternatives to support the design choices made in Secs. 3.2 and 3.3. We report **DirAcc** (direction prediction accuracy) and **FullAcc** (both direction and magnitude correct).

Action-token vocabulary. A natural alternative to introducing new tokens is to let the VLM verbalize the action in plain text and parse it back, or to introduce a single special token for each of the seven possible actions (six directional-magnitude combinations plus one standalone `ok` action). Plain text loses 13.6 DirAcc and 21.3 FullAcc compared to ours (Tab. 6, left): free-form generation introduces unnecessary lexical variability, making the supervision target sparse and harder to align consistently across samples. Joint tokens, which dedicate one special token to each action, recover most of the gap on DirAcc (86.8) but still lag on FullAcc (72.5 vs. 76.7): each joint token is supervised on only $\sim\frac{1}{7}$ of training samples, while our factored direction and magnitude tokens each see $\sim\frac{1}{3}$. Factoring direction and magnitude is therefore not just a clean abstraction; it provides substantially denser supervision for each token.

Action-token initialization. The six new tokens carry no meaning at initialization, so how we seed them shapes what the embeddings can become through Stage 1 warm-up. We compare three options: *random*; *anchor*, which copies the embedding of "depth" into all six tokens; and *semantic* (ours), where each token is initialized from a descriptive phrase (Tab. 8 in Appendix B). All three reach similar DirAcc after warm-up (within 2.4 points), but on FullAcc random costs 3.5 points and anchor costs 2.3 relative to semantic (Tab. 6, middle). The asymmetry reflects the structure of the two token groups. The three direction tokens are pairwise distinct categories, and random or identical initializations separate cleanly along this categorical axis during warm-up; DirAcc therefore largely recovers under any initialization. The three magnitude tokens, by contrast, form an *ordinal* progression (small $\rightarrow$ medium $\rightarrow$ large); semantic initialization provides a better inductive bias for this ordinal structure from the start, whereas random or single-anchor initializations must recover it from scratch within the limited warm-up budget. FullAcc requires magnitude to be correct, which is precisely where this gap appears.

Training recipe. Our training recipe makes two choices: it keeps the vision encoder frozen throughout, and it pairs each sample with a chain-of-thought (CoT) rationale rather than supervising on the action tokens alone. Both choices contribute substantially (Tab. 6, right). Unfreezing the vision encoder costs 6.5 DirAcc and 7.1 FullAcc, likely because the encoder drifts away from the broad visual priors acquired during VLM pretraining and over-specializes to the narrow distribution of wireframe overlays. Removing the CoT rationale costs 4.7 DirAcc and 6.5 FullAcc: without CoT, the supervision becomes concentrated almost entirely on the action tokens themselves, losing the denser reasoning supervision that explicitly connects each action to visual evidence and making the model more susceptible to shortcut learning.

6 Conclusion

We presented RefineAny3D, which refines the depth of a monocular 3D box after the fact by asking a vision-language model whether the projected box fits the object in the image. The model answers with a direction and a step size relative to the object's size, and repeating this moves the box until it fits, so no metric depth is ever regressed. As a post-hoc step, it improves MonoCoP on KITTI, DetAny3D with ground-truth 2D boxes on Omni3D, and LabelAny3D pseudo-labels, whereas geometric fitting and a discriminative classifier given the same inputs make the boxes worse. More broadly, our results suggest that a model which cannot output a precise number can still reliably judge whether a rendered candidate is right. The approach does need the object to be visible enough to judge the fit, so heavily occluded or truncated objects remain hard, and its projection assumes a pinhole camera. Future work could combine the visual check with scene context or temporal cues across video frames, and extend it to omnidirectional cameras.

References

[1] Shuai Bai, Yuxuan Cai, Ruizhe Chen, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. 2, 6, 7, 17, 21, 23, 27, 31, 32

[2] Shuai Bai, Keqin Chen, Xuejing Liu, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025. 2, 27

[3] Garrick Brazil, Abhinav Kumar, Julian Straub, Nikhila Ravi, Justin Johnson, and Georgia Gkioxari. Omni3D: A large benchmark and model for 3D object detection in the wild. In *CVPR*, 2023. 1, 3, 6, 8, 19, 20, 24, 27, 28, 31

[4] Garrick Brazil and Xiaoming Liu. M3D-RPN: Monocular 3D region proposal network for object detection. In *ICCV*, 2019. 3

[5] Garrick Brazil, Gerard Pons-Moll, Xiaoming Liu, and Bernt Schiele. Kinematic 3D object detection in monocular video. In *ECCV*, 2020. 3

[6] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 3, 31

[7] Mu Cai, Haotian Liu, Siva Karthik Mustikovela, Gregory P Meyer, Yuning Chai, Dennis Park, and Yong Jae Lee. ViP-LLaVA: Making large multimodal models understand arbitrary visual prompts. In *CVPR*, 2024. 3

[8] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 3

[9] Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. SpatialVLM: Endowing vision-language models with spatial reasoning capabilities. In *CVPR*, 2024. 3

[10] Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers. *TPAMI*, 2024. 1

[11] Qiang Chen, Xiaokang Chen, Jian Wang, Shan Zhang, Kun Yao, Haocheng Feng, Junyu Han, Errui Ding, Gang Zeng, and Jingdong Wang. Group DETR: Fast DETR training with group-wise one-to-many assignment. In *ICCV*, 2023. 3

[12] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3D object detection for autonomous driving. In *CVPR*, 2016. 1

[13] Yiwei Chen, Soumyadeep Pal, Yimeng Zhang, Qing Qu, and Sijia Liu. Unlearning isn't invisible: Detecting unlearning traces in LLMs from model outputs. In *ICLR*, 2026. 27

[14] Yiwei Chen, Bingqi Shang, and Sijia Liu. Who built this model? tracing LLM lineage via spectral fingerprints in weight space. In *COLM*, 2026. 28

[15] Yiwei Chen, Yuguang Yao, Yihua Zhang, Bingquan Shen, Gaowen Liu, and Sijia Liu. Safety mirage: How spurious correlations undermine VLM safety fine-tuning and can be mitigated by machine unlearning. In *ICLR*, 2026. 27

[16] An-Chieh Cheng, Yang Fu, Yatai Ji, Ligeng Zhu, Guanqi Zhan, Zhuoyang Zhang, Zhaojing Yang, Song Han, Yao Lu, Pavlo Molchanov, et al. Grounded 3D-aware spatial vision-language modeling. In *CVPR*, 2026. 3

[17] An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. SpatialRGPT: Grounded spatial reasoning in vision-language models. In *NeurIPS*, 2024. 3

[18] Daniel DeTone, Tianwei Shen, Fan Zhang, Lingni Ma, Julian Straub, Richard Newcombe, and Jakob Engel. Boxer: Robust lifting of open-world 2D bounding boxes to 3D. In *ECCV*, 2026. 3

[19] Zhiwen Fan, Jian Zhang, Renjie Li, Junge Zhang, Runjin Chen, Hezhen Hu, Kevin Wang, Huaizhi Qu, Shijie Zhou, Dilin Wang, et al. VLM-3R: Vision-language models augmented with instruction-aligned 3D reconstruction. In *CVPR*, 2026. 3

[20] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *CVPR*, 2012. 3, 8, 16, 26, 31

[21] Miran Heo, Min-Hung Chen, De-An Huang, Sifei Liu, Subhashree Radhakrishnan, Seon Joo Kim, Yu-Chiang Frank Wang, and Ryo Hachiuma. Omni-RGPT: Unifying image and video region-level understanding via token marks. In *CVPR*, 2025. 3

[22] Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3D-LLM: Injecting the 3D world into large language models. In *NeurIPS*, 2023. 3

[23] Wenbo Hu, Jingli Lin, Yilin Long, Yunlong Ran, Lihan Jiang, Yifan Wang, Chenming Zhu, Runsen Xu, Tai Wang, and Jiangmiao Pang. G^2VLM: Geometry grounded vision language model with unified 3D reconstruction and spatial reasoning. In *CVPR*, 2026. 3

[24] Kuan-Chih Huang, Tsung-Han Wu, Hung-Ting Su, and Winston H Hsu. MonoDTR: Monocular 3D object detection with depth-aware transformer. In *CVPR*, 2022. 3

[25] Rui Huang, Henry Zheng, Yan Wang, Zhuofan Xia, Marco Pavone, and Gao Huang. Training an open-vocabulary monocular 3D detection model without 3D data. In *NeurIPS*, 2024. 3

[26] Ting Huang, Zeyu Zhang, and Hao Tang. 3D-R1: Enhancing reasoning in 3D VLMs for unified scene understanding. *arXiv preprint arXiv:2507.23478*, 2025. 3

[27] Wenxuan Huang, Bohan Jia, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and Shaohui Lin. Vision-R1: Incentivizing reasoning capability in multimodal large language models. In *ICLR*, 2026. 3

[28] Changwoo Jeon, Rishi Upadhyay, and Achuta Kadambi. MoCA3D: Monocular 3D bounding box prediction in the image plane. In *ECCV*, 2026. 3

[29] Xueying Jiang, Sheng Jin, Xiaoqin Zhang, Ling Shao, and Shijian Lu. MonoMAE: Enhancing monocular 3D detection through depth-aware masked autoencoders. In *NeurIPS*, 2024. 8

[30] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Pannag Sanketi, Quan Vuong, et al. OpenVLA: An open-source vision-language-action model. In *CoRL*, 2024. 4

[31] Abhinav Kumar, Garrick Brazil, Enrique Corona, Armin Parchami, and Xiaoming Liu. DEVIANT: Depth equivariant network for monocular 3D object detection. In *ECCV*, 2022. 3

[32] Abhinav Kumar, Garrick Brazil, and Xiaoming Liu. GrooMeD-NMS: Grouped mathematically differentiable NMS for monocular 3D object detection. In *CVPR*, 2021. 3

[33] Abhinav Kumar, Yuliang Guo, Xinyu Huang, Liu Ren, and Xiaoming Liu. SeaBird: Segmentation in bird's view with dice loss improves monocular 3D detection of large objects. In *CVPR*, 2024. 3

[34] Abhinav Kumar, Yuliang Guo, Zhihao Zhang, Xinyu Huang, Liu Ren, and Xiaoming Liu. CHARM3R: Towards unseen camera height robust monocular 3D detector. In *ICCV*, 2025. 3

[35] Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. LISA: Reasoning segmentation via large language model. In *CVPR*, 2024. 2

[36] Seokyeong Lee, Sithu Aung, Junyong Choi, Seungryong Kim, Ig-Jae Kim, and Junghyun Cho. PLOT: Pseudo-labeling via object tracking for monocular 3D object detection. In *ECCV*, 2026. 3

[37] Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. In *ICLR*, 2026. 3

[38] Zhuoling Li, Zhan Qu, Yang Zhou, Jianzhuang Liu, Haoqian Wang, and Lihui Jiang. Diversity matters: Fully exploiting depth clues for reliable monocular 3D object detection. In *CVPR*, 2022. 3

[39] Zhuoling Li, Xiaogang Xu, SerNam Lim, and Hengshuang Zhao. UniMODE: Unified monocular 3D object detection. In *CVPR*, 2024. 3

[40] Weifeng Lin, Xinyu Wei, Ruichuan An, Peng Gao, Bocheng Zou, Yulin Luo, Siyuan Huang, Shanghang Zhang, and Hongsheng Li. Draw-and-understand: Leveraging visual prompts to enable MLLMs to comprehend what you want. In *ICLR*, 2025. 3

[41] Feng Liu and Xiaoming Liu. Voxel-based 3D detection and reconstruction of multiple objects from a single image. In *NeurIPS*, 2021. 3

[42] Hou-I Liu, Christine Wu, Jen-Hao Cheng, Wenhao Chai, Shian-Yun Wang, Gaowen Liu, Hugo Latapie, Jhih-Ciang Wu, Jenq-Neng Hwang, Hong-Han Shuai, et al. MonoTAKD: Teaching assistant knowledge distillation for monocular 3D object detection. In *CVPR*, 2025. 3, 8

[43] Luyang Liu, Hongyu Li, and Marco Gruteser. Edge assisted real-time object detection for mobile augmented reality. In *MobiCom*, 2019. 1

[44] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *ECCV*, 2024. 3

[45] Xianpeng Liu, Ce Zheng, Kelvin B Cheng, Nan Xue, Guo-Jun Qi, and Tianfu Wu. Monocular 3D object detection with bounding box denoising in 3D by perceiver. In *ICCV*, 2023. 3

[46] Zechen Liu, Zizhang Wu, and Roland Tóth. SMOKE: Single-stage monocular 3D object detection via keypoint estimation. In *CVPR Workshop*, 2020. 3

[47] Yunfei Long, Abhinav Kumar, Xiaoming Liu, and Daniel Morris. RICCARDO: Radar hit prediction and convolution for camera-radar 3D object detection. In *CVPR*, 2025. 1

[48] Yunfei Long, Abhinav Kumar, Daniel Morris, Xiaoming Liu, Marcos Castro, and Punarjay Chakravarty. RADIANT: Radar-image association network for 3D object detection. In *AAAI*, 2023. 1

[49] Yan Lu, Xinzhu Ma, Lei Yang, Tianzhu Zhang, Yating Liu, Qi Chu, Junjie Yan, and Wanli Ouyang. Geometry uncertainty projection network for monocular 3D object detection. In *ICCV*, 2021. 3, 16

[50] Xinzhu Ma, Yinmin Zhang, Dan Xu, Dongzhan Zhou, Shuai Yi, Haojie Li, and Wanli Ouyang. Delving into localization errors for monocular 3D object detection. In *CVPR*, 2021. 3

[51] Yongsen Mao, Junhao Zhong, Chuan Fang, Jia Zheng, Rui Tang, Hao Zhu, Ping Tan, and Zihan Zhou. SpatialLM: Training large language models for structured indoor modeling. In *NeurIPS*, 2025. 3

[52] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Košecká. 3D bounding box estimation using deep learning and geometry. In *CVPR*, 2017. 16

[53] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. 9, 17

[54] Liang Peng, Junkai Xu, Haoran Cheng, Zheng Yang, Xiaopei Wu, Wei Qian, Wenxiao Wang, Boxi Wu, and Deng Cai. Learning occupancy for monocular 3D object detection. In *CVPR*, 2024. 1

[55] Luigi Piccinelli, Christos Sakaridis, Yung-Hsu Yang, Mattia Segu, Siyuan Li, Wim Abbeloos, and Luc Van Gool. UniDepthV2: Universal monocular metric depth estimation made simpler. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025. 1, 3

[56] Fanqi Pu, Yifan Wang, Jiru Deng, and Wenming Yang. MonoDGP: Monocular 3D object detection with decoupled-query and geometry-error priors. In *CVPR*, 2025. 3, 8

[57] Zequn Qin and Xi Li. MonoGround: Detecting monocular 3D objects from the ground. In *CVPR*, 2022. 3

[58] Huaizhi Qu, Hossein Nourkhiz Mahjoub, Vaishnav Tadiparthi, Kwonjoon Lee, and Tianlong Chen. MonoVLM: Monocular 3D visual grounding with vision language models. In *CVPR*, 2026. 3

[59] Kevin Qu, Haozhe Qi, Mihai Dusmanu, Mahdi Rad, Rui Wang, and Marc Pollefeys. Loc3R-VLM: Language-based localization and 3D reasoning with vision-language models, 2026. 3

[60] Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, et al. VLM-R1: A stable and generalizable R1-style large vision-language model. *arXiv preprint arXiv:2504.07615*, 2025. 3

[61] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D object proposal generation and detection from point cloud. In *CVPR*, 2019. 1

[62] Xuepeng Shi, Qi Ye, Xiaozhi Chen, Chuangrong Chen, Zhixiang Chen, and Tae-Kyun Kim. Geometry-based distance decomposition for monocular 3D object detection. In *ICCV*, 2021. 1, 16

[63] Aleksandar Shtedritski, Christian Rupprecht, and Andrea Vedaldi. What does CLIP know about a red circle? Visual prompt engineering for VLMs. In *ICCV*, 2023. 3

[64] Jan Skvrna and Lukas Neumann. MonoSOWA: Scalable monocular 3D object detector without human annotations. In *ICCV*, 2025. 3

[65] Quan Sun, Yuxin Fang, Ledell Wu, Xinlong Wang, and Yue Cao. EVA-CLIP: Improved training techniques for CLIP at scale. *arXiv preprint arXiv:2303.15389*, 2023. 3

[66] Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, Ziteng Wang, Rob Fergus, Yann LeCun, and Saining Xie. Cambrian-1: A fully open, vision-centric exploration of multimodal LLMs. In *NeurIPS*, 2024. 27

[67] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *CVPR*, 2025. 3

[68] Peng Wang, Shuai Bai, Sinan Tan, et al. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024. 2

[69] Ruicheng Wang, Sicheng Xu, Yue Dong, Yu Deng, Jianfeng Xiang, Zelong Lv, Guangzhong Sun, Xin Tong, and Jiaolong Yang. MoGe-2: Accurate monocular geometry with metric scale and sharp details. In *NeurIPS*, 2025. 1, 3, 20

[70] Tai Wang, Xiaohan Mao, Chenming Zhu, Runsen Xu, Ruiyuan Lyu, Peisen Li, Xiao Chen, Wenwei Zhang, Kai Chen, Tianfan Xue, et al. EmbodiedScan: A holistic multi-modal 3D perception suite towards embodied AI. In *CVPR*, 2024. 1

[71] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. Probabilistic and geometric depth: Detecting objects in perspective. In *CoRL*, 2022. 3

[72] Yifan Wang, Yian Zhao, Fanqi Pu, Xiaochen Yang, Yang Tang, Xi Chen, and Wenming Yang. SPAN: Spatial-projection alignment for monocular 3D object detection. In *CVPR*, 2026. 3

[73] Yuxin Wang, Lei Ke, Boqiang Zhang, Tianyuan Qu, Hanxun Yu, Zhenpeng Huang, Meng Yu, Dan Xu, and Dong Yu. N3D-VLM: Native 3D grounding enables accurate spatial reasoning in vision-language models. *arXiv preprint arXiv:2512.16561*, 2025. 1, 3

[74] Zhenyu Wang, Yali Li, Taichi Liu, Hengshuang Zhao, and Shengjin Wang. OV-Uni3DETR: Towards unified open-vocabulary 3D object detection via cycle-modality propagation. In *ECCV*, 2024. 3

[75] Zizhang Wu, Yuanzhu Gan, Lei Wang, Guilian Chen, and Jian Pu. MonoPGC: Monocular 3D object detection with pixel geometry contexts. In *ICRA*, 2023. 3

[76] Zizhang Wu, Yuanzhu Gan, Yunzhe Wu, Ruihao Wang, Xiaoquan Wang, and Jian Pu. FD3D: Exploiting foreground depth map for feature-supervised monocular 3D object detection. In *AAAI*, 2024. 3, 8

[77] Runsen Xu, Xiaolong Wang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. PointLLM: Empowering large language models to understand point clouds. In *ECCV*, 2024. 3

[78] Longfei Yan, Pei Yan, Shengzhou Xiong, Xuanyu Xiang, and Yihua Tan. MonoCD: Monocular 3D object detection with complementary depths. In *CVPR*, 2024. 8

[79] An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. 6, 23, 32

[80] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in GPT-4V. *arXiv preprint arXiv:2310.11441*, 2023. 3

[81] Senqiao Yang, Tianyuan Qu, Xin Lai, Zhuotao Tian, Bohao Peng, Shu Liu, and Jiaya Jia. LISA++: An improved baseline for reasoning segmentation with large language model. *arXiv preprint arXiv:2312.17240*, 2023. 3

[82] Yung-Hsu Yang, Luigi Piccinelli, Mattia Segu, Siyuan Li, Rui Huang, Yuqian Fu, Marc Pollefeys, Hermann Blum, and Zuria Bauer. 3D-MOOD: Lifting 2D to 3D for monocular open-set object detection. In *ICCV*, 2025. 1, 3

[83] Jin Yao, Hao Gu, Xuweiyi Chen, Jiayun Wang, and Zezhou Cheng. Open vocabulary monocular 3D object detection. In *3DV*, 2026. 1, 3, 7, 8, 24, 27

[84] Jin Yao, Radowan Mahmud Redoy, Sebastian Elbaum, Matthew B. Dwyer, and Zezhou Cheng. LabelAny3D: Label any object 3D in the wild. In *NeurIPS*, 2025. 1, 3, 8

[85] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3D object detection and tracking. In *CVPR*, 2021. 1

[86] Yuqian Yuan, Hang Zhang, Wentong Li, Zesen Cheng, Boqiang Zhang, Long Li, Xin Li, Deli Zhao, Wenqiao Zhang, Yueting Zhuang, et al. VideoRefer suite: Advancing spatial-temporal object understanding with video LLM. In *CVPR*, 2025. 3

[87] Hanxue Zhang, Haoran Jiang, Qingsong Yao, Yanan Sun, Renrui Zhang, Hao Zhao, Hongyang Li, Hongzi Zhu, and Zetong Yang. Detect anything 3D in the wild. In *ICCV*, 2025. 1, 3, 8, 9, 27, 28, 29

[88] Renrui Zhang, Han Qiu, Tai Wang, Ziyu Guo, Ziteng Cui, Yu Qiao, Hongsheng Li, and Peng Gao. MonoDETR: Depth-guided transformer for monocular 3D object detection. In *ICCV*, 2023. 1, 3, 8

[89] Yunpeng Zhang, Jiwen Lu, and Jie Zhou. Objects are different: Flexible monocular 3D object detection. In *CVPR*, 2021. 3

[90] Zhihao Zhang, Shengcao Cao, and Yu-Xiong Wang. TAMM: TriAdapter multi-modal learning for 3D shape understanding. In *CVPR*, 2024. 3

[91] Zhihao Zhang, Yiwei Chen, Weizhan Zhang, Caixia Yan, Qinghua Zheng, Qi Wang, and Wangdu Chen. Tile classification based viewport prediction with multi-modal fusion transformer. In *ACM MM*, 2023. 28

[92] Zhihao Zhang, Haipeng Du, Shouqin Huang, Weizhan Zhang, and Qinghua Zheng. VR-Former: 360-degree video streaming with FoV combined prediction and super resolution. In *ISPA/BDCloud/SocialCom/SustainCom*, 2022. 28

[93] Zhihao Zhang, Abhinav Kumar, Girish Chandar Ganesan, and Xiaoming Liu. Unleashing the power of chain-of-prediction for monocular 3D object detection. In *CVPR*, 2026. 3, 8, 9, 16, 26

[94] Zhihao Zhang, Abhinav Kumar, and Xiaoming Liu. Towards intrinsic-aware monocular 3D object detection. In *CVPR*, 2026. 3

[95] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. DETRs beat YOLOs on real-time object detection. In *CVPR*, 2024. 3

[96] Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3D-VLA: A 3D vision-language-action generative world model. In *ICML*, 2024. 3

[97] Yunsong Zhou, Hongzi Zhu, Quan Liu, Shan Chang, and Minyi Guo. MonoATT: Online monocular 3D object detection with adaptive token transformer. In *CVPR*, 2023. 3

[98] Jie Zhu, Girish Chandar Ganesan, and Xiaoming Liu. DepthAgent: Towards better universal depth estimation via sample-wise expert selection. *arXiv preprint arXiv:2605.23281*, 2026. 28

[99] Jie Zhu, Xiao Guo, Yiyang Su, Anil Jain, and Xiaoming Liu. FusionAgent: A multimodal agent with dynamic model selection for human recognition. In *CVPR*, 2026. 28

[100] Jie Zhu, Yiyang Su, Minchul Kim, Anil Jain, and Xiaoming Liu. A quality-guided mixture of score-fusion experts framework for human recognition. In *ICCV*, 2025. 28

[101] Jie Zhu, Yiyang Su, and Xiaoming Liu. Can textual reasoning improve the performance of MLLMs on fine-grained visual classification? In *CVPR Findings*, 2026. 23

[102] Jie Zhu, Ivy Zhang, Minchul Kim, and Xiaoming Liu. Subtoken vision transformer for fine-grained recognition. *arXiv preprint arXiv:2607.09086*, 2026. 28

[103] Menglong Zhu, Konstantinos G Derpanis, Yinfei Yang, Samarth Brahmbhatt, Mabel Zhang, Cody Phillips, Matthieu Lecce, and Kostas Daniilidis. Single image 3D object detection and pose estimation for grasping. In *ICRA*, 2014. 1

[104] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *CoRL*, 2023. 4

Table 7: **Comparison with non-VLM refinement mechanisms.** All methods refine MonoCoP [93] predictions on KITTI validation (AP_{3D} at IoU ≥ 0.7) and modify only the object depth. Both geometric-fitting variants are given the *ground-truth* tight 2D box as the fitting target, an oracle input RefineAny3D never receives.

Method	Box attributes	Easy	Mod.	Hard	Δ vs. MonoCoP
MonoCoP [93] (unrefined)	predicted	32.06	23.98	20.64	—
Geometric projection fitting	predicted	29.89	21.55	18.36	$-2.17/-2.43/-2.28$
Geometric projection fitting	GT dims + yaw	30.56	21.79	18.43	$-1.50/-2.19/-2.21$
DINOv2 ViT-S/14 action classifier	predicted	24.71	17.62	12.78	$-7.35/-6.36/-7.86$
RefineAny3D (Ours)	predicted	**35.62**	**27.47**	**21.31**	$+3.56/+3.49/+0.67$

A Why Simpler Alternatives Fall Short

RefineAny3D uses a VLM to judge image-space alignment. Two simpler mechanisms could in principle serve the same role: a purely geometric procedure that fits depth by projection, and a small discriminative network trained to emit the same action tokens. We implement both and find that neither recovers the improvement, for different reasons. All experiments in this section refine MonoCoP [93] predictions on KITTI [20] validation under the AP_{3D} protocol at IoU ≥ 0.7 used in Tab. 2, and modify only the object depth.

A.1 Geometric Projection Fitting

Why the fitting objective is biased. Given predicted dimensions, yaw, and the camera ray, one may recover depth by searching for the value whose projected cuboid best overlaps the object's tight 2D box. This resembles the alignment task we assign to the VLM, but the two objectives are not equivalent: a tight 2D box encloses the *visible* object pixels, whereas the projected cuboid also spans occluded corners and empty volume outside the visible silhouette. The mismatch is not a second-order effect. Consider a front-facing, on-axis cuboid with center depth $z^\star$, height H, and length L, under focal length f. Its projected envelope height at depth z is $h_{\text{cub}}(z) = fH/(z - L/2)$, while the annotated tight box has height $h_{2D} = \rho\, h_{\text{cub}}(z^\star)$ for some extent ratio ρ that is not 1 in general. Matching heights yields a recovered depth $\hat{z}$ satisfying

$$\hat{z} - z^\star = \left(\tfrac{1}{\rho} - 1\right)\left(z^\star - \tfrac{L}{2}\right), \tag{7}$$

so $\rho < 1$ pushes the recovered depth farther, $\rho > 1$ pulls it closer, and only $\rho = 1$ is exact—a condition that accurate 2D localization does not imply. At detection scale this residual is decisive: an 8% extent mismatch ($\rho = 0.92$) displaces a car centered at 40 m by roughly 3.3 m, far exceeding the longitudinal tolerance of the $IoU_{3D} \geq 0.7$ criterion at that range. A visible-object mask inherits the same mismatch, since it too covers only the visible extent. The same difficulty appears in prior Mono3D work: Deep3DBox [52] recovers translation from this projected-box constraint, and later methods [49, 62] model the uncertainty of geometry-derived depth rather than treating it as exact.

Setup. We implement the baseline as a one-dimensional search along the original predicted camera ray, selecting the depth that maximizes the 2D IoU between the tight 2D box and the axis-aligned envelope of the projected cuboid, with dimensions, yaw, and confidence held fixed. We use the *ground-truth* tight 2D box as the fitting target, an oracle input RefineAny3D itself never receives, since a predicted 2D box would only add localization noise to this target; the setting therefore favors the baseline. We report two variants: one using the detector's predicted dimensions and yaw, and one in which these are replaced by their GT values, isolating how much of the baseline's behavior is attributable to attribute noise. For the oracle variant, each prediction is matched to the same-class GT object with the highest 3D IoU, falling back to the object with the nearest projected center when no same-class GT overlaps it. A sanity check confirms the implementation: fitting the GT projected envelope with GT attributes recovers the GT depth exactly.

Results. Geometric fitting *lowers* AP_{3D} by $2.17/2.43/2.28$ on Easy / Moderate / Hard (Tab. 7), while RefineAny3D improves the same predictions by $3.56/3.49/0.67$. Supplying GT dimensions and yaw recovers only $0.67/0.24/0.07$ AP, leaving the baseline $1.50/2.19/2.21$ below the unrefined detector.

Attribute noise is therefore not the primary cause: the degradation traces to the fitting objective itself, as Eq. (7) predicts.

A.2 A Lightweight Discriminative Action Classifier

Setup. The second alternative keeps our action formulation but replaces the decision model. We train a classifier built on a pretrained DINOv2 ViT-S/14 [53] encoder with two lightweight heads predicting the direction and magnitude tokens. It receives the same object crop with the projected wireframe that RefineAny3D sees, and is trained on the same depth-perturbation data with the same target tokens. The vocabularies, object-relative step sizes, depth-update rule, and iterative procedure are identical, so the decision model is the only component that differs.

Results. Despite the pretrained encoder and identical supervision, the classifier reduces AP_{3D} by $7.35/6.36/7.86$ on Easy / Moderate / Hard (Tab. 7), well below the unrefined detector, while RefineAny3D improves the same predictions by $3.56/3.49/0.67$. The predefined discrete action space is therefore not by itself sufficient to reproduce the improvement: what distinguishes the two methods is the visual judgment used to select among the actions, not the actions themselves.

We attribute this to a distribution shift that the refinement loop amplifies. Training samples perturb depth on otherwise ground-truth boxes, so the classifier observes misalignments caused by depth alone; real detector outputs instead carry coupled residual errors in depth, dimensions, orientation, and projected position, which produce a considerably wider range of image–wireframe relationships. Because every predicted token pair is converted directly into a metric update, a decision model that does not transfer under this shift does more than fail to help: it moves initially reasonable predictions away from the target, consistent with the degradation we observe.

A.3 Discussion: What the VLM Contributes

Neither alternative can reliably assess image–wireframe consistency under realistic detector residuals, but they fail through different mechanisms.

Geometric fitting is limited by objective mismatch. It substitutes a surrogate, rectangular overlap between a tight 2D box and the projected cuboid envelope, for the judgment we actually want, namely whether the projected cuboid is consistent with the visible object. Maximizing that surrogate can favor a depth with better 2D agreement but worse 3D localization, and the failure persists under GT 2D conditioning and oracle attributes.

The classifier is limited by generalization. It optimizes the right objective over the right action space, yet a task-specific discriminative mapping from crop to class does not survive the shift from controlled perturbations to the coupled residual errors of a real detector, and under this formulation every mispredicted token is spent as a metric error.

RefineAny3D pairs broad visual priors with a constrained correction interface. The VLM is never asked to regress metric depth. It evaluates the relationship between the visible object and the projected cuboid (contour agreement, surface enclosure, internal projected structure, and the direction in which the mismatch points) rather than apparent size alone, and expresses the result through a small set of direction and magnitude tokens. The pretrained multimodal representation supports interpreting varied visual evidence, while the constrained vocabulary converts that judgment into bounded, object-relative metric updates. A further asymmetry is that RefineAny3D can *abstain*: the ⟨`depth_ok`⟩ action leaves 37.8% of objects unchanged (Appendix F), whereas a fitting procedure always commits to its argmax. The improvement therefore does not arise from discretization alone, but from pairing the discrete interface with a decision model that generalizes the alignment judgment to realistic detector outputs.

B Implementation Details

Model architecture. We build RefineAny3D on top of Qwen3-VL-8B-Instruct [1], extending its tokenizer with six special action tokens that factor the corrective action into three direction tokens and three magnitude tokens (Tab. 8). Each new embedding row is initialized via *semantic initialization*: we tokenize a short phrase describing the token's meaning (right column of Tab. 8) and mean-pool the corresponding sub-token embeddings of the frozen base model, placing each new token in a

region of the embedding space that already encodes its intended concept. Magnitudes correspond to object-relative shifts of $0.20\, s_{\text{obj}}$, $0.55\, s_{\text{obj}}$, and $1.10\, s_{\text{obj}}$ for small, medium, and large, where $s_{\text{obj}} = (w + h + l)/3$ is the object's mean linear extent.

Table 8: Action-token vocabulary, inference contract, and semantic initialization. Each new token's embedding is initialized by mean-pooling the sub-token embeddings of its init phrase.

Token	Meaning at inference	Init phrase
`<depth_closer>`	move box towards camera (decrease center z)	*"move closer to camera"*
`<depth_farther>`	move box away from camera (increase center z)	*"move farther from camera"*
`<depth_ok>`	no correction needed	*"no depth correction needed"*
`<step_small>`	apply $0.20 \times s$ m (midpoint of $[0.10, 0.30)$)	*"small step size"*
`<step_medium>`	apply $0.55 \times s$ m (midpoint of $[0.30, 0.80)$)	*"medium step size"*
`<step_large>`	apply $1.10 \times s$ m (representative for $[0.80, \infty)$)	*"large step size"*

Training data and infrastructure. We train on the curated dataset of approximately 3.0M (image, target-tokens) samples (Secs. D.3 and D.4). All training is performed on 8 NVIDIA H100 (80 GB) GPUs in bfloat16 with DeepSpeed ZeRO-3 and FlashAttention-2, using AdamW with cosine learning-rate decay and standard auto-regressive cross-entropy loss applied to both the chain-of-thought tokens and the action tokens. Stage 1 takes approximately 1 hour and Stage 2 takes approximately 90 hours per configuration.

B.1 Two-Stage Training Protocol

Single-step supervision. Both training stages use single-step supervision. Each training sample consists of (i) an input image with the candidate 3D box rendered as a wireframe overlay (Sec. D.3), (ii) a target chain-of-thought reasoning trace (Sec. D.4), and (iii) the GT action tokens (a_d, a_m) corresponding to the perturbation slot. We do not unroll the iterative update at training time. Each action is a local decision conditioned only on the current visual state $\mathbf{I}^{(t)}$ and is therefore independent of the trajectory leading up to it; teaching the model a mapping from any current visual state to its locally correct action is sufficient for multi-turn inference to chain such decisions correctly.

Stage 1: action-token warm-up. The six action tokens are newly added to the VLM's vocabulary. Directly co-training them with the full VLM is unstable, since the loss must simultaneously shape both the new embeddings and the LLM's use of them, and the LLM cannot meaningfully attend to embeddings that are still essentially random. We therefore freeze every transformer block and train only the new token embedding rows (`embed_tokens` and `lm_head`) for one epoch on a stratified subsample, using AdamW with peak learning rate 5×10^{-3}, cosine schedule, and 5% warmup. This stage is intentionally short and has no goal beyond stabilizing the new embeddings before joint optimization.

Stage 2: joint fine-tuning. With the action-token embeddings warmed up, we fully fine-tune the language tower together with the action-token embeddings while keeping the vision encoder frozen. Freezing the vision encoder is a deliberate design choice: the encoder retains the general visual priors learned during VLM pretraining, which the trained RefineAny3D relies on to generalize across novel categories, scenes, and cameras at test time. We empirically observed that unfreezing the vision encoder during Stage 2 hurts generalization on Refine3D, particularly on the novel-scene split. We train for one epoch over the full curated dataset using AdamW with peak learning rate 1×10^{-5}, cosine schedule, 3% warmup, and gradient clipping at 1.0.

B.2 Multi-Turn Inference

Iterative refinement loop. At inference, RefineAny3D is applied iteratively to refine an initial noisy box $\mathbf{b}^{(0)}$ produced by an upstream detector. At each step t, the loop performs four operations in sequence:

1. Render the current box $\mathbf{b}^{(t)}$ as a wireframe on the input image to obtain $\mathbf{I}^{(t)} = \text{render}(\mathbf{I},\ \pi(\mathbf{b}^{(t)}; \mathbf{K}))$.

Table 9: Oracle study under the Omni3D evaluation protocol. Each row replaces one attribute of Cube R-CNN's predictions with either the ground truth or the output of a depth foundation model.

Variant	AP_{3D}	Δ vs. baseline
Baseline (Cube R-CNN)	36.00	—
Oracle dimensions	35.91	-0.09
Oracle angle	36.22	$+0.22$
Foundation depth (MoGe-2)	32.32	-3.68
Oracle depth	65.92	$+29.92$

2. Query the VLM with $\mathbf{I}^{(t)}$ and a fixed instruction prompt; the model first emits a chain-of-thought reasoning trace, then the action tokens $(a_d^{(t)}, a_m^{(t)})$.
3. Compute the depth update $\Delta d^{(t)} = \mathrm{dir}(a_d^{(t)}) \cdot \mathrm{mag}(a_m^{(t)},\ s_{\text{obj}})$ and update the box: $d^{(t+1)} = d^{(t)} + \Delta d^{(t)}$, with $(\hat{x}^{(t+1)}, \hat{y}^{(t+1)})$ re-derived along the camera ray to keep the box's image location consistent with the new depth.
4. Continue if $a_d^{(t)} \in \{\langle\texttt{depth_closer}\rangle, \langle\texttt{depth_farther}\rangle\}$; terminate if $a_d^{(t)} = \langle\texttt{depth_ok}\rangle$ or if the maximum number of iterations $T_{\max}$ is reached. We use $T_{\max} = 2$ in all main experiments.

The chain-of-thought reasoning trace is generated at every step but used only for action-token decoding; it is discarded after the action is parsed.

Why train-inference asymmetry is consistent. RefineAny3D is trained on single-step supervision but deployed in a multi-turn loop. This asymmetry is consistent because the action prediction at each step depends only on the current visual state $\mathbf{I}^{(t)}$, not on the trajectory of past actions. The conditional distribution $P(a^{(t)} \mid \mathbf{I}^{(t)})$ that the model learns at training is exactly the distribution it is queried with at each inference step, regardless of how many refinement iterations have already occurred. Single-step supervision therefore teaches a mapping that generalizes naturally to arbitrarily many inference steps, and multi-turn inference simply chains these per-step decisions until the model itself signals convergence via $\langle\texttt{depth_ok}\rangle$.

Decoding. We use greedy decoding for both the chain-of-thought reasoning and the action tokens, with the action-token positions restricted to the six special tokens via constrained decoding. This guarantees that each action prediction is one of the valid (a_d, a_m) combinations and prevents the model from emitting natural-language tokens at action positions.

Termination behavior. The over-representation of $\langle\texttt{depth_ok}\rangle$ in the training distribution (Sec. D.3) gives the trained model a mild conservative bias: under uncertainty, it tends to terminate refinement rather than apply an aggressive correction. Empirically, refinement uses 1.62 VLM calls per object on average under the safeguard cap $T_{\max} = 2$, and 15.8% of objects reach the cap; these correspond to cases where the model oscillates between $\langle\texttt{depth_closer}\rangle$ and $\langle\texttt{depth_farther}\rangle$ near the alignment boundary.

C Oracle Study Details

We provide additional details for the oracle study reported in Fig. 1. We use Cube R-CNN [3] trained on Omni3D as the baseline detector and report AP_{3D} under the standard Omni3D evaluation protocol.

For each predicted box, we find its corresponding GT box via greedy assignment based on 3D IoU, with a matching threshold of 0.5. Each GT box can match at most one prediction. Predictions that do not match any GT are kept unmodified, and unmatched GT boxes contribute to recall as usual. Confidence scores from the original detector are always preserved so that AP ranking is unaffected.

For each matched (prediction, GT) pair, we replace one or more attributes of the prediction with the corresponding GT values:

- *Oracle dimensions* replaces $(\hat{w}, \hat{h}, \hat{l})$ with (w^*, h^*, l^*).
- *Oracle angle* replaces $\hat{\theta}$ with θ^*.

- *Oracle depth* replaces the predicted object depth with the GT depth. Because changing depth alone while keeping the predicted $(\hat{x}, \hat{y})$ fixed would cause the box to drift in the image, we re-derive $(\hat{x}, \hat{y})$ from the camera ray of the original prediction and the new depth, so the projected box stays at the same image location and only depth changes.

- *Foundation depth* replaces the predicted object depth with a depth derived from MoGe-2 [69], a state-of-the-art depth foundation model. Since MoGe-2 predicts the depth of the visible object *surface* rather than the object *center* that 3D detection requires, we apply a geometric correction rather than reading MoGe-2's output directly. Specifically, we (i) project the predicted 3D center onto the image plane to locate its corresponding pixel, (ii) read MoGe-2's depth at that pixel as a surface-depth estimate, and (iii) add an offset—computed from the predicted object dimensions and orientation—that accounts for the distance, along the camera ray, between the surface entry point of the box and its center. This converts the per-pixel surface depth produced by MoGe-2 into a center depth comparable to what the detector regresses. As with oracle depth, $(\hat{x}, \hat{y})$ is then re-derived along the camera ray to keep the box's image location fixed.

All other attributes remain unchanged in each variant.

Tab. 9 reports the full numbers. The unmodified Cube R-CNN baseline reaches an AP_{3D} of 36.00. Replacing the predicted dimensions or yaw angle with their ground-truth values changes AP by less than 0.3 points in either direction, indicating that these two attributes are already well predicted by the in-domain detector. Replacing the predicted depth with the ground-truth depth boosts AP to 65.92 (a $+29.92$ gain), nearly doubling the baseline. Replacing the predicted depth with the surface-to-center-corrected depth from MoGe-2, in contrast, reduces AP to 32.32 (a 3.68 drop), worse than the baseline detector's own predictions.

Two observations emerge from these numbers. First, object depth is the single dominant bottleneck of monocular 3D detection: among the three 3D attributes we ablate, only depth shows a substantial gap to perfect prediction. The detector's predicted dimensions and angles are already accurate enough that perfecting them yields negligible gain, while perfecting depth alone nearly doubles the baseline AP. This motivates RefineAny3D's design of refining only the depth and leaving other attributes untouched. Second, depth foundation models do not yet provide the precision needed to close this bottleneck. Although MoGe-2 demonstrates strong zero-shot performance on unseen scenes, its object-level depth estimates—even after the surface-to-center correction described above—remain less accurate than those of an in-domain detector. We attribute this to a mismatch in objective: depth foundation models are trained with dense, pixel-level supervision and are not optimized to be metrically precise at the object level. Under tight 3D IoU thresholds, even small object-level biases translate into substantial AP drops.

D Data Curation Details

We construct our training set (Fig. 3) from public 3D-annotated images in three stages: source dataset selection (Sec. D.1), per-annotation filtering (Sec. D.2), and depth-perturbation sample synthesis (Sec. D.3). The first two stages yield 335,167 filtered annotations with reliable 3D boxes, visually unambiguous objects, and trustworthy category labels; the third stage synthetically perturbs each annotation along its camera ray to generate approximately 3.0M (image, target-tokens) training samples that span all action-token outcomes.

D.1 Source Datasets

We build on the Omni3D release [3], which re-annotates six existing datasets into a shared COCO-style format under a common camera convention ($+x$ right, $+y$ down, $+z$ into the screen). The six sources are intentionally heterogeneous: KITTI and nuScenes provide outdoor driving scenes; Objectron contributes hand-held captures of common indoor objects; SUN RGB-D and ARKitScenes provide indoor scans; and Hypersim contributes photorealistic synthetic renderings. In total, this yields 3,303,211 annotations across 234,152 images, spanning 50 unified categories (96 raw category names before the Omni3D taxonomy merge).

Table 10: Per-dataset retention through the two-stage filtering pipeline. The first "%" column reports retention from raw to after the geometric filter; the second reports retention from after the geometric filter to after the VLM-based semantic filter.

Dataset	Anns (raw)	After geom.	%	After VLM	%
KITTI	50,514	10,932	21.6	10,023	91.7
nuScenes	356,438	150,240	42.2	90,440	60.2
Objectron	54,214	41,578	76.7	39,898	96.0
SUN RGB-D	78,514	31,444	40.0	23,278	74.0
ARKitScenes	476,501	94,642	19.9	53,677	56.7
Hypersim	2,287,030	137,196	6.0	117,851	85.9
Total	**3,303,211**	**466,032**	**14.1**	**335,167**	**71.9**

D.2 Annotation Filtering

The raw release contains many annotations unsuitable as positive training examples, including invalid 3D boxes, behind-camera projections, tiny or heavily truncated objects, severe occlusion, mis-categorized labels, and rendering artifacts. We apply a two-stage filter: a geometric stage that uses metadata fields available in Omni3D, followed by a VLM-based semantic stage that catches errors no geometric rule can detect.

Geometric and metadata filter. An annotation is retained only if it is marked as a valid 3D box, projects in front of the camera, has a 2D footprint of at least 32×32 pixels, and satisfies $\texttt{truncation} \leq 0.05$ and $\texttt{visibility} \geq 0.8$ (relaxed to ≥ 0.6 for nuScenes, whose visibility field is discretized into coarse bins). When both `bbox2D_tight` and `bbox2D_proj` are available, we additionally require their 2D IoU to exceed 0.85, and we reject annotations whose geometric truncation $1 - \mathrm{area}(\texttt{bbox2D_trunc})/\mathrm{area}(\texttt{bbox2D_proj})$ exceeds 0.05. After this stage, $466{,}032$ annotations across $162{,}002$ images remain (14.1% of the raw pool); per-dataset retention is reported in Tab. 10.

VLM-based semantic filter. Geometric rules cannot detect mis-categorized labels, semantic occlusion by foreground clutter, blurry captures, or rendering artifacts. We add a semantic stage that queries a vision-language model on each surviving annotation. For each annotation we extract a 336×336 RGB crop centered on the object using `bbox2D_tight` (or `bbox2D_proj` when unavailable) expanded by a $1.5\times$ context margin, with the original 2D box rendered in red as a visual anchor. We prompt Qwen3-VL-30B-A3B-Instruct [1] to return a structured judgment of whether the object is recognizable, its predicted category among a 54-class closed set, agreement with the claimed category, occlusion severity, image-edge truncation, bounding-box tightness, and overall image quality. We retain an annotation when the object is visible, the predicted and claimed categories are not in clear conflict, and none of occlusion, truncation, box tightness, or image quality is rated in its worst category. This stage retains 71.9% of the geometrically filtered pool, leaving $335{,}167$ annotations.

D.3 Depth-Perturbation Training Set

To turn the $335{,}167$ filtered annotations into supervision for RefineAny3D, we synthetically perturb each GT box along the camera ray through its center to generate (image, target-tokens) training pairs that span all action-token outcomes.

Per-annotation sampling schedule. For each filtered annotation we draw 9 perturbation samples, one per slot of a sampling schedule (Tab. 11) that exactly covers the action-token vocabulary. Each sample moves the box's center along this ray so that its depth changes by $\Delta z = r \cdot s_{\text{obj}}$, keeping the projected center fixed, where $s_{\text{obj}} = (w + h + l)/3$ is the object's mean linear extent and r is a slot-specific shift ratio sampled uniformly from a fixed range. Positive Δz pushes the box farther from the camera, so the corrective action is $\langle\texttt{depth_closer}\rangle$; negative Δz pulls the box toward the camera, so the corrective action is $\langle\texttt{depth_farther}\rangle$. Lateral position and orientation are kept at GT throughout, so every visible misalignment is purely a depth misalignment. The schedule allocates three slots to the $\langle\texttt{depth_ok}\rangle$ outcome—an exact-zero shift and two narrow dead-zones around zero—reflecting that depth values inside this band are visually indistinguishable from the GT and should be reported as already aligned.

Table 11: Perturbation sampling schedule. Each annotation produces 9 samples, one per slot. The shift ratio r is drawn uniformly from the listed range, and the actual center shift is $\Delta z = r \cdot s_{\text{obj}}$.

Slot	Direction	Magnitude	Shift ratio r
0	⟨depth_closer⟩	⟨step_small⟩	$[+0.10,\ +0.30)$
1	⟨depth_closer⟩	⟨step_medium⟩	$[+0.30,\ +0.80)$
2	⟨depth_closer⟩	⟨step_large⟩	$[+0.80,\ +1.50]$
3	⟨depth_farther⟩	⟨step_small⟩	$[-0.30,\ -0.10)$
4	⟨depth_farther⟩	⟨step_medium⟩	$[-0.80,\ -0.30)$
5	⟨depth_farther⟩	⟨step_large⟩	$[-1.50,\ -0.80]$
6	⟨depth_ok⟩	—	0
7	⟨depth_ok⟩	dead-zone +	$[0,\ +0.10)$
8	⟨depth_ok⟩	dead-zone −	$(-0.10,\ 0]$

Table 12: Action-token distribution in the perturbation training set. The six direction–magnitude combinations are matched in count, and ⟨depth_ok⟩ is over-represented to encourage conservative termination at inference.

Token combination	# samples	Share
⟨depth_closer⟩ ⟨step_small⟩	≈ 335K	11.1%
⟨depth_closer⟩ ⟨step_medium⟩	≈ 335K	11.1%
⟨depth_closer⟩ ⟨step_large⟩	≈ 335K	11.1%
⟨depth_farther⟩ ⟨step_small⟩	≈ 335K	11.1%
⟨depth_farther⟩ ⟨step_medium⟩	≈ 335K	11.1%
⟨depth_farther⟩ ⟨step_large⟩	≈ 335K	11.1%
⟨depth_ok⟩	≈ 1.0M	33.3%
Total	≈ 3.0M	100.0%

Inference contract for magnitude tokens. At inference, each magnitude token corresponds to a fixed corrective shift used by the iterative refinement step. We use the bucket midpoint of each slot's shift-ratio range, so that ⟨step_small⟩ applies $0.20 \cdot s_{\text{obj}}$, ⟨step_medium⟩ applies $0.55 \cdot s_{\text{obj}}$, and ⟨step_large⟩ applies $1.10 \cdot s_{\text{obj}}$ as a representative value for the open-tail $[0.80, \infty)$ regime.

Image rendering. Each of the nine perturbation samples reuses the same 336×336 crop computed in Sec. D.2 from the unperturbed 2D box. On top of this fixed crop we render a wireframe of the *perturbed* cuboid: the eight corners of the GT box are translated with its center along the camera ray, projected through the original intrinsics $\mathbf{K}$, mapped into crop pixels, and connected by the twelve cuboid edges in green. Holding the crop region fixed across all nine slots is intentional: the only visual difference between samples of the same object is the wireframe itself, which prevents the model from exploiting crop-region cues as a shortcut for inferring the depth direction.

Geometric rejection. A perturbation is rejected and re-sampled if any cuboid corner falls behind the camera or if the projected cuboid becomes degenerately small in the crop, guarding against samples in which the perturbed box is invisible or unreasonably tiny.

Final perturbation pool. The full pipeline produces approximately 3.0M (image, target-tokens) training samples from the 335,167 distinct objects. By construction, the action-token distribution is uniform over the seven distinct token combinations: each of the six ⟨depth_closer⟩/⟨depth_farther⟩ direction-magnitude pairs appears in $\frac{1}{9}$ of the samples, and ⟨depth_ok⟩ accounts for the remaining $\frac{3}{9} \approx 33\%$ (Tab. 12). The deliberate over-representation of ⟨depth_ok⟩ introduces a mild conservative bias at inference: under uncertainty, the model prefers to terminate refinement, which costs at most one extra iteration while preventing unnecessary perturbation of already-aligned boxes.

Visualization. Fig. 6 shows three sample crops drawn from a single object: slot 6 (⟨depth_ok⟩), slot 1 (⟨depth_closer⟩ ⟨step_medium⟩), and slot 4 (⟨depth_farther⟩ ⟨step_medium⟩). The three samples share the same crop region but differ only in the rendered wireframe, illustrating how a depth perturbation manifests visually as a wireframe that aligns with the object, sits in front of it, or sits behind it.

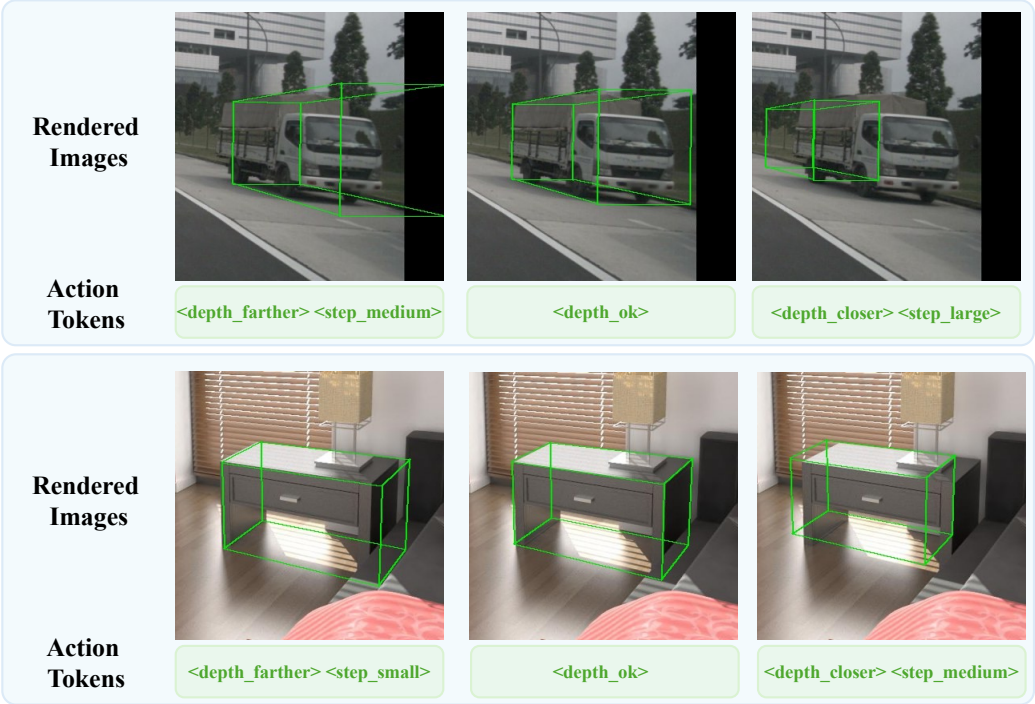


Figure 6: **Examples of generated perturbed data.**

D.4 Chain-of-Thought Reasoning Targets

In preliminary experiments, fine-tuning the VLM to emit only action tokens led to a noticeable drop in its general instruction-following ability. We therefore pair every training sample with a short chain-of-thought (CoT) prefix that the model produces before emitting the action tokens. Keeping the CoT short is also consistent with recent evidence that longer textual reasoning lowers MLLM accuracy on fine-grained visual classification [101]. The CoT acts as a language-anchored intermediate signal: the model articulates which object it is looking at, what visual cue it is reading, and which direction the cue implies, before committing to a discrete action.

Generator choice. We synthesize each CoT with a two-model pipeline that splits the work by what each model can reliably do. A VLM (Qwen3-VL-235B-A22B-Instruct [1]) handles *object identification and description*: given the image crop, it produces a short caption naming the target object and noting salient appearance cues (*e.g.*, "a black car viewed from behind on a street"), a task on which zero-shot VLMs are reliable. A text-only LLM (Qwen3-235B-A22B-Instruct-2507 [79]) then takes this caption together with the sample metadata (object category, GT direction, GT magnitude token) and composes the full CoT, including the optional world-knowledge size recall, the visual-evidence framing of the misalignment, and the corrective decision. We deliberately keep the alignment judgment itself out of the VLM at this stage: reliably telling a well-fitted 3D wireframe from a poorly-fitted one is precisely the capability we aim to teach RefineAny3D, so soliciting that judgment from a zero-shot VLM would inject unreliable supervision into training.

Reasoning structure. Each CoT follows a fixed four-step structure that mirrors how a human annotator would reason about a perturbed wireframe.

- *Identify.* The model first names the object visible in the crop, optionally with a brief scene hint (*e.g.*, "A chair viewed slightly from above.").
- *Recall* (optional, skipped in 30% of samples). The model recalls a generic, world-knowledge size range for the object category in approximate units (*e.g.*, "Chairs typically span around 60 cm wide and 90–110 cm tall."). Precise metric values are forbidden, and the recalled range is decoupled from the actual GT.

- *Visual evidence.* The model describes the misalignment between wireframe and object using either the size cue (*e.g.*, "The wireframe is noticeably smaller than the chair's outline."), the enclosure cue (*e.g.*, "The chair's near surface protrudes in front of the wireframe's near face."), or both, depending on the framing label drawn for this sample. Specific camera-relative distances are forbidden.
- *Decision.* The model concludes by stating the implied depth error and the corrective action, consistent with the GT direction and magnitude token (*e.g.*, "This indicates the box is too far from the camera, requiring a closer correction by a medium step.").

Visual cue diversification. Depth error is observable through two complementary visual cues, and we deliberately diversify the CoTs across both so that the trained VLM learns each independently:

- *Size cue.* A box that is too far appears smaller than the object; too close, larger.
- *Enclosure cue.* A box that is too far fails to enclose the object, with the object's near surface protruding past the wireframe; too close, the wireframe encloses the object loosely, leaving empty volume.

For each sample we draw a framing label from $\{$*size-only*, *enclosure-only*, *mixed*$\}$ with weights $0.4/0.4/0.2$, and the prompt for non-mixed framings forbids the opposite framing's vocabulary, ensuring each CoT exercises a clean single-cue signal.

Training-time use. At training time, each sample's target string is the concatenation of the generated reasoning prose and the canonical `Action` line, and the cross-entropy loss is applied uniformly over both. At inference, the reasoning prose is generated first and discarded; only the parsed `Action` tokens drive the iterative refinement step.

E Refine3D Benchmark Details

Refine3D is a controlled benchmark we construct on top of Omni3D [3] to evaluate 3D box refinement as a standalone capability. Each test sample consists of a ground-truth 3D box paired with a synthetically perturbed noisy version, and the task is to refine the noisy box back toward the GT. The benchmark spans two distribution-shift axes plus an in-distribution reference, which jointly probe the generalization properties RefineAny3D aims to achieve. We describe the construction of each split and the unified evaluation protocol below.

E.1 Test-Sample Construction

For each 3D annotation in the underlying source data, we generate noisy boxes following the same depth-perturbation protocol as the training set (Sec. D.3): the GT box's center is shifted along the camera ray by $\Delta z = r \cdot s_{\text{obj}}$, with r drawn uniformly from each of the perturbation slots in the schedule. All other attributes (lateral position, dimensions, orientation) are held at GT, so each test sample isolates a depth misalignment of known direction and magnitude. Test samples are stratified across the perturbation schedule so that each direction-magnitude combination is equally represented, preventing any setting from being dominated by samples of a particular noise pattern.

E.2 Standard

The Standard split serves as the in-distribution reference. Both training and test pools use the same Omni3D category vocabulary and camera intrinsics distribution. Training samples are drawn from Omni3D's standard train split; test samples are drawn from Omni3D's standard test split, following the construction protocol in Sec. E.1.

E.3 Novel Category

We follow the open-vocabulary monocular 3D detection setup of OVMono3D [83], which partitions the Omni3D category vocabulary into a base set and a novel set. During training, the data curation pipeline (Appendix D) drops every annotation belonging to a novel category, so the trained RefineAny3D has never been supervised on these classes. At test time, we draw test samples (per Sec. E.1) from Omni3D's standard test split, restricted to the held-out novel categories.

E.4 Novel Camera

To probe robustness to unseen camera intrinsics without changing the scene content, we synthetically rescale the original Omni3D test images while updating the camera intrinsics consistently. For each test image $\mathbf{I}$ with intrinsics $\mathbf{K}$, we apply an isotropic scale factor s to obtain a rescaled image $\mathbf{I}_s$ and the rescaled intrinsics

$$\mathbf{K}_s = \mathrm{diag}(s, s, 1)\,\mathbf{K},$$

which preserves the geometric validity of the 3D-to-2D projection but shifts the distribution of projected box sizes away from the training distribution.

Image rescaling alters the effective focal length while keeping the underlying scene unchanged: from the model's perspective, the same physical object now appears at a different projected size on the image plane. Because all other factors (object identity, scene composition, occlusion) are held constant, any drop in performance can be attributed unambiguously to camera-induced distribution shift. Training uses the unscaled Omni3D train split; test samples are drawn from the rescaled Omni3D test split.

E.5 Evaluation Metrics

We report three metrics that jointly characterize refinement quality.

Direction accuracy (DirAcc). The fraction of test samples for which the refiner predicts the correct corrective direction token $a_d \in \{\langle\texttt{depth_closer}\rangle, \langle\texttt{depth_ok}\rangle, \langle\texttt{depth_farther}\rangle\}$ at the first refinement step. This metric isolates the qualitative correctness of the refiner's first-step decision from the magnitude prediction, providing a clean proxy for the model's underlying alignment reasoning. Higher is better.

Full action accuracy (FullAcc). The fraction of test samples for which the refiner predicts both the correct direction a_d and the correct magnitude a_m at the first refinement step. FullAcc strictly extends DirAcc by additionally requiring the magnitude bucket (small, medium, or large) to match the ground-truth bucket induced by the perturbation. Higher is better.

Depth error (DepthErr). The mean absolute residual between the refined and GT object depths after the iterative refinement loop terminates:

$$\mathrm{DepthErr} = \frac{1}{N}\sum_{i=1}^{N}\left|d_i^{(\text{refined})} - d_i^{*}\right|,$$

where d_i^{*} is the GT depth and $d_i^{(\text{refined})}$ is the depth after refinement. To prevent objects at different scales from dominating the average, the per-sample residual is normalized by s_{obj} before averaging. Lower is better.

Why three metrics together. DirAcc and FullAcc characterize the per-step prediction quality at the token level, while DepthErr captures the end-task refinement outcome after the iterative loop. A method can score high on DirAcc yet still leave residual depth error through biased magnitudes; conversely, a method can produce low DepthErr by chance through cancellation of errors across iterations without making genuinely correct per-step decisions. Reporting all three lets readers distinguish whether a method is failing to choose the direction, to choose the step size, or to converge to the right answer.

F Inference Analysis

RefineAny3D is trained with single-step supervision but applied iteratively at inference (Sec. 3.3): at each step, the model emits an action token that updates the candidate box's depth, until either the model commits to $\langle\texttt{depth_ok}\rangle$ or a safeguard cap of $T_{\max}$ VLM calls is reached. We use $T_{\max} = 2$ in all main experiments. Two natural questions arise: how quickly does the loop converge in practice, and what is the actual inference cost in a deployment setting? We answer both below.

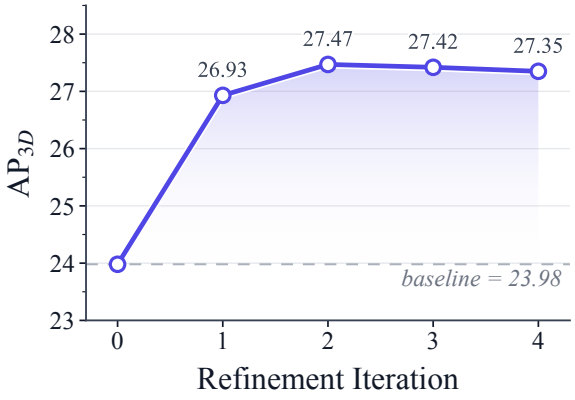


Figure 7: **Iterative refinement convergence on KITTI Moderate.** AP_{3D} as a function of forced iteration count K. A single refinement step recovers most of the gain ($+2.95$ AP_{3D} over the baseline); the second step adds another $+0.54$, after which performance plateaus and slightly drifts down.

F.1 Convergence Behavior

We sweep the iteration count $K \in \{0, 1, 2, 3, 4\}$ on KITTI [20] validation Moderate, refining MonoCoP [93]'s predictions as the upstream detector. $K = 0$ corresponds to the unrefined MonoCoP baseline, and $K \geq 1$ forces the loop to run exactly K steps regardless of self-termination.

Fig. 7 shows three regimes. *First step* ($K = 0 \rightarrow 1$): AP_{3D} jumps from 23.98 to 26.93, a $+2.95$ gain that accounts for $\sim 85\%$ of the total improvement. *Second step* ($K = 1 \rightarrow 2$): an additional $+0.54$ AP_{3D} ($26.93 \rightarrow 27.47$), addressing the subset of objects whose alignment cannot be diagnosed from a single look at the wireframe. *Beyond two steps* ($K = 2 \rightarrow 4$): AP_{3D} plateaus and drifts down slightly ($27.47 \rightarrow 27.35$), as a small fraction of already-aligned boxes is over-corrected. Based on this trade-off we set $T_{\max} = 2$, which captures the peak accuracy at the lowest inference cost.

F.2 Iteration-Count Distribution under Self-Termination

With self-termination via $\langle$`depth_ok`$\rangle$ enabled, each object is processed within $T_{\max} = 2$ VLM calls. On KITTI validation, 37.8% of objects exit on the first call (the model judges the upstream prediction already aligned and applies no update), 46.4% exit on the second call after one applied update, and the remaining 15.8% apply two updates and reach the cap. The mean number of VLM calls per object is 1.62 under $T_{\max} = 2$, indicating that the safeguard cap is exercised only on the minority of harder cases.

F.3 Inference Latency

We measure absolute latency on a single NVIDIA H100 (80 GB) GPU with bfloat16 precision and FlashAttention-2. The wireframe rendering step is negligible (< 5 ms on CPU), and one VLM forward over the rendered object crop takes ~ 320 ms on average. Combined with the empirical mean of 1.62 VLM calls per object, the average per-object refinement latency is ~ 520ms. At the typical KITTI validation density of ~ 5.4 objects per image (measured on the validation split), this corresponds to an end-to-end refinement overhead of ~ 2.8s per image.

This overhead is non-trivial relative to the upstream detector itself but acceptable in offline use cases such as auto-labeling, dataset curation, and post-hoc accuracy improvement, where throughput is not the bottleneck. Two factors keep the cost bounded. First, refinement operates on localized object crops rather than the full image, so cost scales with the number of high-confidence proposals rather than image resolution. Second, the $\langle$`depth_ok`$\rangle$ self-termination signal lets the model skip refinement entirely on $\sim 38\%$ of objects, keeping the empirical mean below $T_{\max}$.

Table 13: **Improving open-vocabulary 3D detectors on Omni3D [3] under a realistic 2D-box source.** RefineAny3D refines DetAny3D [87] conditioned on Cube R-CNN's [3] predicted 2D boxes, complementing the oracle-2D evaluation in Sec. 5.3.

Method	AP_{3D}^{kit} ↑	AP_{3D}^{nus} ↑	AP_{3D}^{sun} ↑	AP_{3D}^{ark} ↑	AP_{3D}^{obj} ↑	AP_{3D}^{hyp} ↑	AP_{3D} ↑
Cube R-CNN [3]	32.50	30.06	15.33	41.73	50.84	7.48	23.26
OVMono3D [83]	25.45	24.33	15.20	41.60	58.87	7.75	22.98
DetAny3D [87]	31.61	30.97	18.96	46.13	54.42	7.17	24.92
RefineAny3D (Ours)	**33.94**	**32.29**	**21.07**	**47.83**	**56.36**	**9.05**	**26.96**

Table 14: **General visual capability on CV-Bench [66].** RefineAny3D retains nearly all of the underlying Qwen3-VL-8B's capability, indicating that fine-tuning for refinement does not erode general visual reasoning.

Method	Depth	Distance	Relation	Count	Overall
Qwen2.5-VL-7B [2]	86.33	75.83	88.92	64.85	78.17
Qwen3-VL-8B [1]	94.17	87.00	93.54	72.59	85.94
RefineAny3D (Ours)	93.83	84.00	92.77	71.07	84.53

G More Results on Improving Open-Vocabulary 3D Detectors

Sec. 5.3 reports our main open-vocabulary result on top of DetAny3D [87] conditioned on ground-truth 2D boxes, an oracle setup chosen to isolate the contribution of depth refinement. To complement that stress-test, we also evaluate RefineAny3D under a more realistic configuration in which DetAny3D is conditioned on 2D boxes predicted by Cube R-CNN [3], the standard closed-set detector used in the Omni3D benchmark. This setup reflects how an open-vocabulary refinement pipeline would be deployed in practice: an upstream 2D detector proposes object regions, a 3D detector lifts them to 3D, and RefineAny3D refines the resulting depth.

Tab. 13 reports AP_{3D} on each of the six Omni3D sub-datasets. Applying RefineAny3D on top of DetAny3D with Cube R-CNN's predicted 2D boxes lifts overall AP_{3D} from 24.92 to 26.96 (+2.04), with consistent improvements across every sub-dataset. The gain is smaller in absolute terms than in the oracle-2D setting (+4.35) because the upstream Cube R-CNN 2D detections themselves carry localization error that propagates into the 3D prediction and limits the headroom available to depth refinement. Even so, the consistent positive gains under noisy 2D conditioning indicate that RefineAny3D's refinement signal does not require clean upstream input: it transfers from the oracle setting to a fully realistic pipeline without retraining.

H Preserving General Visual Capability

A practical concern is that fine-tuning may erode the VLM's general visual competence. Fine-tuning is known to affect a model beyond its target task: supervised safety fine-tuning of VLMs can reinforce spurious correlations between surface text patterns and the intended responses [15], and weight updates such as machine unlearning leave traces that are detectable from model outputs even on unrelated inputs [13]. On CV-Bench [66] (Tab. 14), RefineAny3D retains nearly all of the underlying Qwen3-VL-8B's capability: the overall score changes by only 1.41 points (85.94 → 84.53), with no sub-task degrading by more than 3 points. This corroborates freezing the vision encoder throughout training (Sec. 3.3), which preserves the base model's broad visual priors while adding refinement as a new capability on top.

I Limitations and Future Work

RefineAny3D assumes the visual signature of depth misalignment is readable from the image; for heavily occluded or truncated objects this cue weakens and refinement becomes less reliable, particularly under tight 3D IoU thresholds. Future work could combine visual alignment with complementary cues such as scene context or temporal consistency. In addition, our projection $\pi(\mathbf{b}; \mathbf{K})$ assumes a pinhole camera; extending the alignment view to omnidirectional 360° imagery,

which is increasingly common in immersive video applications [92, 91], is another natural direction. Since the alignment cue can be subtle for small or distant objects, finer tokenization of discriminative image regions [102] may further sharpen the judgment. Finally, because RefineAny3D is agnostic to the upstream source, it could be combined with per-sample selection or fusion of multiple upstream detectors and depth experts before refinement, as recent work does for universal depth estimation [98] and for human recognition [100, 99].

J More Visualization

We provide additional qualitative examples of RefineAny3D's refinement behavior in Fig. 8, complementing the main-text examples in Fig. 5. The examples span the six Omni3D [3] sub-datasets and cover diverse object categories, scene types, and initial misalignment patterns. In each row, the leftmost column shows DetAny3D [87]'s initial prediction (red wireframe) overlaid on the input image, and the rightmost column shows RefineAny3D's refined result (green wireframe). For cases that converge in more than one step, the intermediate columns show the projection after each successive action token.

Across these examples, two qualitative patterns are visible. First, the projection tightens around the target object monotonically over iterations: each step either commits to $\langle$`depth_ok`$\rangle$ when the alignment is judged sufficient or moves the projection in the visually correct direction (closer or farther). Second, harder cases tend to involve objects with weak alignment cues, such as small objects, occlusion, or unusual viewpoints, which is where multiple steps are most useful: the first step provides a coarse correction, and subsequent steps make finer adjustments once the projection is roughly in place.

K Broader Impact

RefineAny3D is a refinement module for monocular 3D object detection, a perception capability with broad applications in autonomous driving, robotics, augmented reality, and assistive technologies. By improving the object-level depth precision of existing 3D detectors, RefineAny3D can make downstream perception systems safer and more reliable. In autonomous driving, more accurate object localization translates directly into better trajectory prediction and collision avoidance, particularly in the long-tail open-vocabulary regime where current detectors are weakest. In indoor robotics and AR, more precise 3D geometry enables tighter manipulation and more stable scene understanding, which are critical for assistive applications such as service robots for elderly care or visually impaired users. RefineAny3D's plug-and-play design lowers the barrier to adopting these improvements, since practitioners can integrate it on top of existing detectors without retraining the underlying perception stack. Since RefineAny3D is fine-tuned from an open-weight VLM, its lineage to the base model can in principle be traced from the weights alone [14], which supports attribution and license compliance when such checkpoints are redistributed. At the same time, more accurate 3D localization from a single camera could also be used in surveillance or tracking systems without the consent of the people being observed, and a refined box should not be treated as ground truth in safety-critical settings: RefineAny3D can still fail on heavily occluded or truncated objects, so deployed systems should keep independent safety checks rather than rely on refined depth alone.

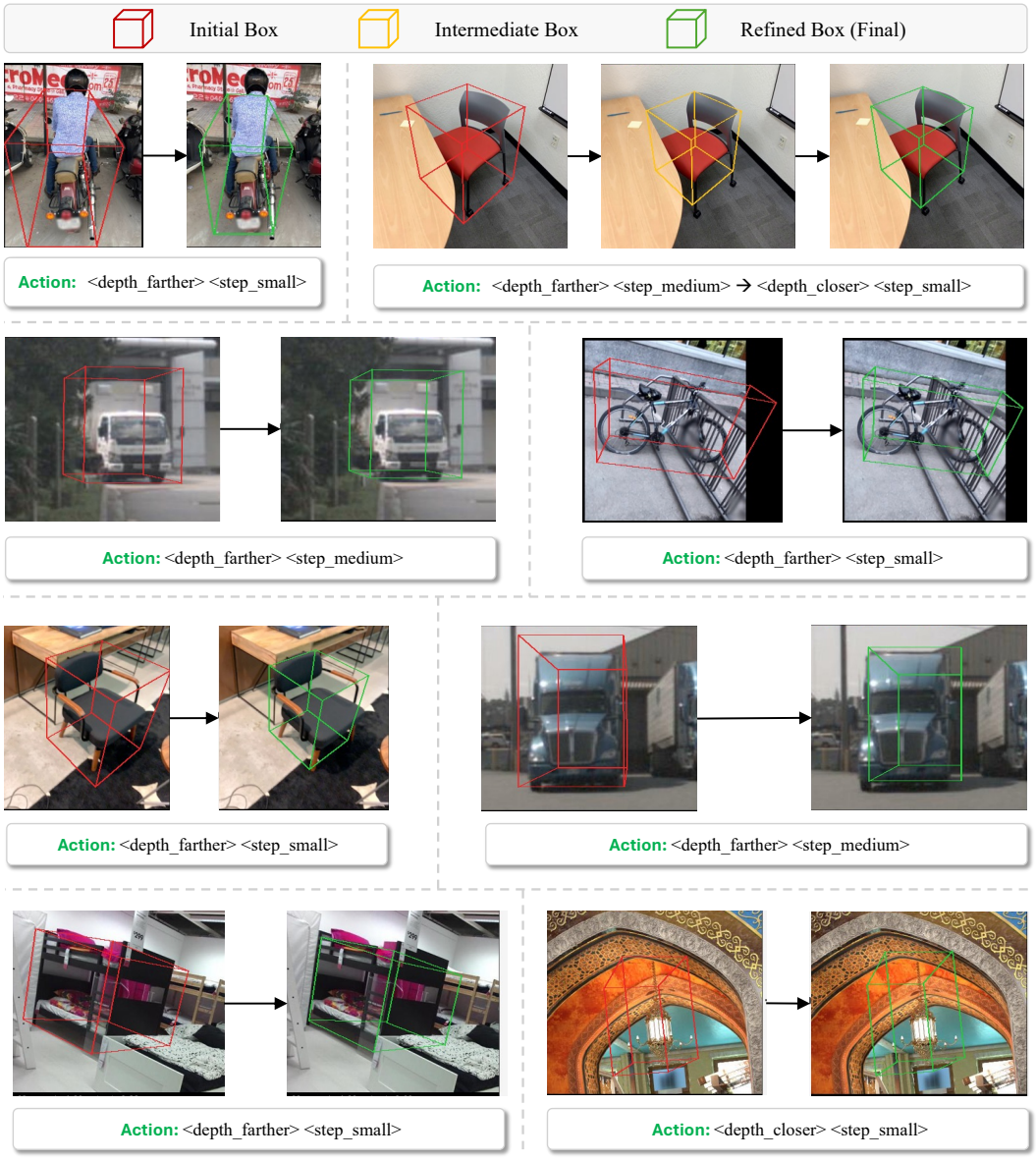


Figure 8: **Additional qualitative examples of single-step and iterative depth refinement on Omni3D.** RefineAny3D refines DetAny3D [87]'s initial prediction (red) toward the refined result (green). Most cases are corrected in a single step; harder cases require multiple action steps until the projected box aligns with the target.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Sec. 1 clearly state our four contributions: formalizing 3D box refinement as a stand-alone task (Sec. 1), recasting it as semantic alignment (Sec. 3), instantiating it with action tokens (Sec. 3), and demonstrating consistent improvements as a plug-and-play module (Sec. 5). All claims are supported by the experimental results in Sec. 5.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix I discusses the main limitations: the visual cue weakens for heavily occluded or truncated objects, and the projection assumes a pinhole camera. In addition, Sec. 5.1 acknowledges that performance degrades more under Novel Category than Novel Camera shifts, and Sec. 5.6 reports honest comparisons including cases where alternative designs achieve comparable results on certain metrics.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include formal theoretical results requiring proofs. The factorization in Sec. 3 is a standard application of the chain rule for autoregressive sequence modeling.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Secs. 3.2 and 3.3 fully describe the architecture and training procedure. Appendix B provides detailed implementation specifications including hyperparameters, optimizer settings, and hardware. Appendix E documents the Refine3D benchmark construction.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We release code, model checkpoints, and the curated training and benchmark data, together with instructions to reproduce the main results, at `https://github.com/alanzhangcs/RefineAny3D`.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Sec. 3.3 describes the two-stage training procedure. Appendix B reports detailed hyperparameters: learning rates (5×10^{-3} for Stage 1, 1×10^{-5} for Stage 2), AdamW optimizer with cosine schedule, gradient clipping, batch size, and training duration. Data splits are specified in Sec. 5 and Appendix E.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We do not report error bars due to the high computational cost of training the full model (approximately 90 hours on 8 H100 GPUs per Stage 2 run). We mitigate variability by evaluating on large stratified test sets that aggregate over many samples, and by reporting consistent improvements across multiple datasets and settings.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix B reports that all training uses 8 NVIDIA H100 (80 GB) GPUs with bfloat16 precision, DeepSpeed ZeRO-3, and FlashAttention-2. Stage 1 takes approximately 1 hour and Stage 2 approximately 90 hours. Inference latency is measured on a single H100 (Appendix F).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics. We use only publicly available datasets (Omni3D and its source datasets) and pretrained models (Qwen3-VL and Qwen3) under their respective licenses, and our work does not involve human subjects or sensitive personal data.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix K discusses both. Our method improves monocular 3D perception, which has positive applications in autonomous driving, robotics, and assistive technologies. Potential negative impacts include misuse for surveillance applications. As the method refines existing 3D detectors rather than introducing new perception capabilities, the marginal risk over existing detection systems is limited.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse?

    Answer: [N/A]

    Justification: Our model and curated data pose no high risk for misuse beyond what is already enabled by existing publicly available 3D detection systems and pretrained VLMs. The training data is derived from established academic datasets that are already publicly available under their respective licenses.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets (Omni3D [3], KITTI [20], nuScenes [6]) and pretrained models (Qwen3-VL [1]) used in the paper are properly cited. Each asset is used in accordance with its respective license.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We introduce two new assets: (i) a curated training dataset of $\sim$3M depth-perturbation samples with chain-of-thought annotations; and (ii) the Refine3D benchmark, documented in Appendix D and Appendix E, respectively. Both are released with documentation at https://github.com/alanzhangcs/RefineAny3D.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or human subjects research. All annotations used in training are derived programmatically from existing public 3D datasets.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve research with human subjects.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research?

Answer: [Yes]

Justification: LLMs are core components of the method: RefineAny3D fine-tunes Qwen3-VL-8B [1], and Qwen3-VL and Qwen3 [79] are used for annotation filtering and chain-of-thought synthesis (Sec. 4 and Appendix D). We additionally use an LLM for grammar checking.