SpatialPoint: Spatial-aware Point Prediction for Embodied Localization

Qiming Zhu[1,2*], Zhirui Fang[1,2*], Tianming Zhang[1], Chuanxiu Liu[1], Xiaoke Jiang[1†], Lei Zhang[1,3]

[1]Visincept Research
[2]Tsinghua University
[3]International Digital Economy Academy (IDEA)

[*]Work done during an internship at Visincept. [†]Corresponding author.

Abstract

*Embodied intelligence fundamentally requires a capability to determine where to act in 3D space. We formalize this requirement as **embodied localization** — the problem of predicting executable 3D points conditioned on visual observations and language instructions. We instantiate embodied localization with two complementary target types: touchable points, surface-grounded 3D points enabling direct physical interaction, and air points, free-space 3D points specifying placement and navigation goals, directional constraints, or geometric relations. Embodied localization is inherently a problem of embodied 3D spatial reasoning — yet most existing vision-language systems rely predominantly on RGB inputs, necessitating implicit geometric reconstruction that limits cross-scene generalization, despite the widespread adoption of RGB-D sensors in robotics. To address this gap, we propose **SpatialPoint**, a spatial-aware vision-language framework with careful design that integrates structured depth into a vision-language model (VLM) and generates camera-frame 3D coordinates. We construct a 2.6M-sample RGB-D dataset covering both touchable and air points QA pairs for training and evaluation. Extensive experiments demonstrate that incorporating depth into VLMs significantly improves embodied localization performance. We further validate SpatialPoint through real-robot deployment across three representative tasks: language-guided robotic arm grasping at specified locations, object placement to target destinations, and mobile robot navigation to goal positions. Project page: https://qimingzhu-google.github.io/SpatialPoint/.*

1 Introduction

Recent advances in embodied intelligence [1–3] increasingly shift focus from passive visual recognition toward spatially executable perception driven by the great wave of AGI. In many manipulation-oriented and language-conditioned robotic tasks [4, 5], the central requirement is to determine *where* to act in 3D space. We argue that a broad class of embodied behaviors—ranging from grasping and placement to navigation and motion specification—can be unified under a single abstraction: predicting executable 3D points conditioned on visual observations and language instructions. We term this problem **embodied localization**.

We instantiate embodied localization with two complementary target types. **Touchable points** are surface-grounded 3D coordinates that enable direct physical interaction, such as grasping or contact [6, 7]. **Air points** are free-space 3D coordinates that specify placement and navigation goals, directional constraints, or geometric relations beyond visible surfaces [8]. Together, these two types constitute a minimal yet expressive representation of executable actions, unifying perception-supported and reasoning-driven targets within a single framework, as illustrated in Figure 1.

Embodied localization is inherently a problem of embodied 3D spatial reasoning. However, most existing vision-language models (VLMs) [9–11] rely predominantly on RGB inputs, predicting 2D bounding boxes, masks, or other image-level outputs. While such models can produce guesses for 3D queries, they cannot guarantee alignment with real-world metric geometry [12–14]. Monocular RGB cues lack explicit metric depth, often resulting in insufficient geometric fidelity for precise spatial reasoning. Although multi-view reconstruction can recover 3D structure, it is mathematically complex and heavily dependent on accurate camera calibration and relative poses. Consequently, learning 3D spatial structure implicitly from RGB views alone is challenging, prone to overfitting to training scenes, and often exhibits weak cross-scene generalization—despite the widespread adoption of RGB-D sensors in robotics and industrial systems that provide direct metric depth measurements.

To address this gap, we propose **SpatialPoint**, a spatial-aware vision-language framework that directly integrates structured depth into a VLM and generates camera-frame 3D coordinates via the language modeling head. However, incorporating a new depth modality into a pre-trained vision-language model that expects only RGB and language inputs is non-trivial. This is precisely why most existing approaches avoid using depth at the input stage, instead treating it as an auxiliary cue for feature augmentation or post-hoc refinement (see Section 2.3). To tackle this modality integration challenge, SpatialPoint takes RGB and structured depth as parallel inputs: depth is first encoded into a 3-channel representation and processed by a dedicated depth backbone, producing depth tokens that are fused with visual and language features within the multimodal backbone for

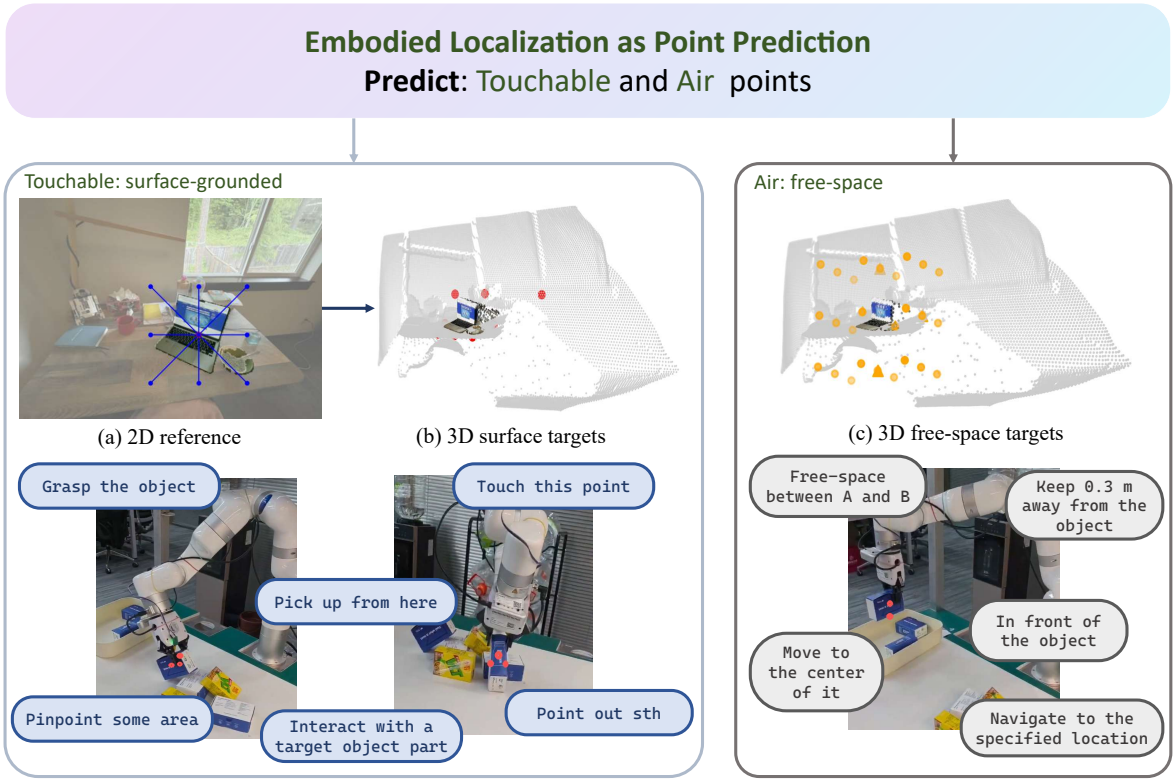


Figure 1: Embodied localization as executable 3D target prediction. We reduce embodied execution to predicting camera-frame 3D points of two complementary types: **touchable points** grounded on observed surfaces, and **air points** located in free space and specified by spatial language.

depth-aware reasoning. To fully activate the depth modality, we further adopt a two-stage training strategy that progressively aligns the depth encoder with the pre-trained VLM. Together, these careful design choices are what enable structured depth to contribute meaningfully to embodied 3D spatial reasoning, rather than introducing noise or degrading the pre-trained capabilities.

To enable large-scale training and evaluation, we construct **SpatialPoint-Data**, a 2.6M-sample RGB-D dataset covering both touchable and air points question-answer pairs. Touchable targets are obtained by lifting 2D interaction annotations into camera-frame 3D coordinates using depth and camera geometry. Air targets are automatically synthesized to cover direction-specified and relation-centric queries with optional metric constraints. We further establish **SpatialPoint-Bench** for unified evaluation over both target types. Extensive experiments demonstrate that incorporating structured depth into VLMs consistently improves embodied localization performance over RGB-only baselines and alternative geometry-input designs, establishing depth tokens as a critical geometric inductive bias for scalable, spatial-aware embodied reasoning.

Finally, we validate SpatialPoint through real-robot deployment across three representative tasks: language-guided robotic arm grasping at specified locations (*touchable points*), object placement to target destinations (*air points*), and mobile robot navigation to goal positions (*air points*). The consistently strong real-world performance demonstrates the practical effectiveness and generalizability of our approach. Our contributions are threefold:

1. We introduce embodied localization as a unified formulation of executable 3D target prediction and instantiate it with two complementary types: touchable points and air points.
2. We construct SpatialPoint-Data (2.6M samples) and SpatialPoint-Bench, enabling large-scale training and standardized evaluation of depth-aware embodied localization.
3. We propose SpatialPoint, an RGB-D extension of a VLM with an explicit depth-token stream that directly generates camera-frame 3D coordinates, achieving consistent gains in spatial reasoning and cross-scene generalization, validated by both offline benchmarks and real-robot experiments.

2 Related Work

2.1 Spatial Understanding in Vision-Language Models

Recent vision-language models (VLMs) have progressed from 2D grounding toward spatial understanding in 3D environments. A common direction is to explicitly inject 3D representations into multimodal frameworks [15, 16], e.g., aligning point-cloud features with large language models for 3D grounding, or improving grounding and relational reasoning via fine-grained reward modeling [17]. Another line preserves strong 2D backbones while enabling 3D reasoning through multi-view inputs or coordinate prompting [18–20].

Hybrid approaches further incorporate geometric cues such as depth, lifting, or spatial programs. SpatialRGPT [12]

introduces a depth plugin that leverages monocular (relative) depth to improve direction and distance reasoning; ZSVG3D [21] formulates spatial reasoning as visual programs; and several works lift 2D detections into 3D space for object-level localization [13, 22]. Recent methods also explore more expressive 3D representations—such as 3D Gaussians and structured scene abstractions [23–26]—to support relational reasoning.

Despite these advances, most approaches remain object-centric, focusing on 3D boxes, scene graphs, or coarse abstractions, rather than producing fine-grained *executable* spatial targets for embodied interaction.

2.2 Vision-Language Interfaces for Manipulation Execution

Embodied tasks ultimately require actionable targets in metric 3D space. One line of work [6] predicts language-conditioned interaction points or affordances, emphasizing contact-level grounding for manipulation. Another line represents actions in structured 3D spaces to better couple perception and control, such as voxelized action prediction from RGB-D reconstructions [5] or sequence-level control generation in continuous action spaces [27, 28].

More recently, vision-language-action (VLA) models scale end-to-end policy learning for multi-task generalization. Systems such as RT-1/RT-2 [2, 29], PaLM-E [1], and SayCan [4] integrate language, vision, and control at different levels of abstraction, while open generalist policy backbones—including OpenVLA and Octo [30, 31]—are trained on large embodied datasets. Complementary to policy learning, explicit 3D world modeling offers geometry-grounded action representations; for example, PointWorld [32] predicts 3D point flows in a shared metric space for manipulation.

On the dataset side, RoboAfford [33] scales affordance supervision for manipulation, emphasizing object parts and surface contact regions. At a broader scope, RoboBrain 2.5 [34] aggregates multi-source embodied supervision to support generalist models, including depth-aware coordinate-related signals.

However, existing approaches typically focus either on surface-supported interaction points or on end-to-end action generation, and rarely provide a unified view that covers both *surface-attached* targets and reasoning-required goals in surrounding free space under a consistent vision-conditioned interface.

2.3 Depth Cues for Spatial Reasoning in VLMs and VLAs

In practical robotics, depth is often readily available via low-cost RGB-D sensors [35, 36] (or can be approximated by monocular depth estimation [37]), making metric geometry an economical signal rather than an extra burden. Accordingly, depth cues provide a direct way to inject metric structure into vision-language models and embodied policies, improving direction and distance reasoning beyond what RGB-only cues can reliably support.

For VLMs, SpatialBot [38] explicitly consumes RGB and depth images to enhance depth-aware spatial understanding, while SpatialRGPT [12] introduces a flexible depth plugin that incorporates (typically estimated) depth maps as auxiliary inputs to existing VLM visual encoders and benchmarks. For VLAs, DepthVLA [39] explicitly incorporates depth prediction into policy architectures to improve manipulation robustness, while PointVLA and GeoVLA [40, 41] inject point-cloud or depth-derived geometric features to enhance spatial generalization. Beyond adding depth as an auxiliary cue, 3D-VLA [42] links 3D perception, reasoning, and action through a 3D-centric foundation model, and LLaVA-3D [43] adapts large multimodal models toward 3D outputs via 3D-augmented visual patches.

These efforts collectively support the view that explicit depth cues are valuable for metric spatial reasoning. In contrast to end-to-end action generation, our depth-aware vision-language modeling directly produces *camera-frame 3D target points* and evaluates both *touchable* and *air* targets with complementary metrics.

Table 1: Composition of SpatialPoint-Data and SpatialPoint-Bench. Ratios are computed within each category.

Category	Subtype	#QA	Ratio (%)
Touchable-Data	Object detection	513,395	26.99
	Object pointing	161,681	8.50
	Object affordance	560,836	29.48
	Object reference	346,617	18.22
	Region reference	319,961	16.82
Touchable-Bench	Object affordance	124	36.69
	Spatial affordance	100	29.59
	Object reference	114	33.73
Air-Data	direction only	252,267	35.18
	direction (offset)	241,373	33.66
	body-length	138,615	19.33
	between	26,981	3.76
	between (offset)	57,890	8.07
Air-Bench	direction only	927	37.91
	direction (offset)	855	34.97
	body-length	422	17.26
	between	72	2.94
	between (offset)	169	6.91

3 Methodology

We formulate embodied localization as *depth-aware, language-conditioned 3D point prediction*. Given an RGB image, a depth map, and an instruction, the model generates a list of 3D target points. This section presents the construction of SpatialPoint-Data, which serves as the fuel for introducing a new depth modality into vision-language models; followed by the network architecture and training strategy of SpatialPoint; and finally the evaluation protocol, including the proposed metrics and the benchmark dataset SpatialPoint-Bench. Table 1 summarizes the composition of SpatialPoint-Data across both touchable-point and air-point supervision.

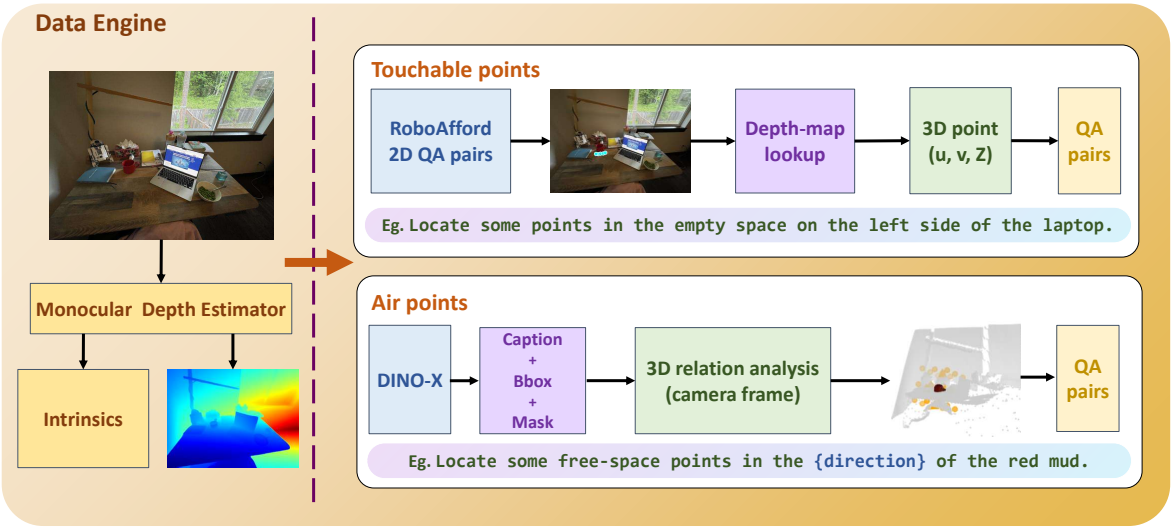


Figure 2: Our data engine. Touchable points are converted from RoboAfford [33] 2D annotations using monocular depth estimation [37]. Air points are generated from objects' 3D relations, which are computed by lifting DINO-X [10] detections (caption/bbox/mask) into the camera frame using the estimated depth map and intrinsics, and then applying geometric computations.

3.1 SpatialPoint-Data and Data Engine

3.1.1 Data source.

Since introducing a new modality as input to VLM is non-trivial and massive data is the best fuel, we build SpatialPoint-Data on RoboAfford-Data [33], due to its large scale, diverse manipulation-related task types, and publicly available RGB imagery. Figure 2 provides an overview of our data engine, as explained in the following text.

Each target is encoded as a triplet (u, v, Z), where (u, v) are pixel coordinates in the image frame and Z is pixel depth in millimeters. In our implementation, u and v are integers in $[0, 1000)$, and Z is an integer-valued depth. Targets are serialized as plain text and learned via standard token prediction, without additional coordinate binning. We then use off-the-shelf monocular depth estimation model [37] to estimate depth for each RGB image.

3.1.2 Touchable points via monocular depth estimation.

RoboAfford provides 2D QA pairs with surface-attached supervision, where each query is associated with valid 2D points on the image. Therefore, we first obtain the 2D target location (u, v) from the 2D annotation, look up the corresponding depth value Z from the estimated depth map at (u, v), and form the unified 3D target (u, v, Z). By applying this lift-from-2D procedure to all touchable points samples, we construct **1.9M** touchable points QA pairs.

3.1.3 Air points via 3D relation analysis.

Air points are not anchored to annotated surface contacts. Instead, we generate QA pairs from RGB images using estimated depth and geometry constraints derived from object-centric 3D context. Concretely, we first parse each image with DINO-X [10] to obtain objects' captions, bounding boxes, and masks. Combining these outputs with the estimated depth map and intrinsics, we lift the scene into the camera frame to obtain object-centric 3D occupancy cues. Based on this geometry, we can directly derive objects' 3D relations via geometric computations. Using the resulting 3D relations, we construct **0.72M** QA pairs over **26** discrete camera-frame directions (six canonical axes and their compositions) and organize them into three groups: (i) *direction* queries that ask for a point in a specified direction (optionally with a metric offset), (ii) *between-object* queries that place a point between two referenced objects, optionally with distance constraints (e.g., *near* one object or with a metric offset), and (iii) *body-length* queries that mirror the direction form but express distance using body-length units, requiring the model to reason about the target object's physical extent.

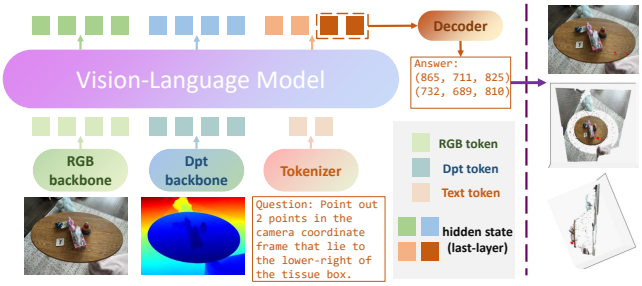


Figure 3: Model overview. We add a dedicated depth encoder by duplicating the original visual backbone and feeding it a three-channel depth map to obtain depth tokens, wrapped by <dpt_start> and <dpt_end>. RGB/depth/text tokens form one causal sequence, and the LM head decodes structured (u, v, Z) point lists.

3.2 Model Architecture

Given an RGB image I, a depth map D, and an instruction T, the input sequence is composed of RGB visual tokens, depth visual tokens, and text tokens. RGB and depth tokens

Table 2: Touchable points results on RoboAfford-Eval. We report overall accuracy and per-category accuracy for object affordance prediction (OAP), object affordance recognition (OAR), and spatial affordance localization (SAL). For methods that output (u, v, Z) targets, we additionally report depth MAE (mm) against the monocular depth reference used in our lifted-3D evaluation, with an inside/outside breakdown; otherwise depth metrics are marked as "–".

Method	Overall↑	OAP↑	OAR↑	SAL↑	MAE$_Z$ (in)↓	MAE$_Z$ (out)↓	MAE$_Z$ (all)↓
RoboPoint[6]	0.447	0.350	0.557	0.442	–	–	–
RoboAfford-Qwen[33]	0.593	0.453	0.661	0.579	–	–	–
RoboAfford-Qwen++[44]	0.634	0.631	0.705	0.558	–	–	–
RoboBrain 2.5[34]	0.741	0.673	0.876	0.671	205.1	255.9	217.9
Qwen2.5-VL-7B[45]	0.161	0.083	0.195	0.219	–	–	–
Qwen3-VL-Inst-2B[11]	0.319	0.307	0.445	0.190	–	–	–
Qwen3-VL-Inst-4B[11]	0.503	0.573	0.540	0.375	567.1	577.9	574.8
Ours	**0.790**	**0.785**	**0.885**	**0.689**	**9.3**	**54.4**	**17.2**

Table 3: Overall air points evaluation on SpatialPoint-Bench (2445 queries).

Method	DirPt	MetPt@5cm	MeanErr (cm)
RoboBrain 2.5[34]	0.0804	0.0637	30.3412
Qwen3-VL-Inst-4B[11]	0.0532	0.0896	54.7086
Ours (epoch 1)	0.4886	0.2587	8.5008
Ours (epoch 2)	**0.5088**	0.2907	7.3034
Ours (epoch 3)	0.5071	**0.3347**	**6.8084**

are enclosed by dedicated delimiters. Conditioned on this RGB-D prefix, the model predicts a variable-length list of camera-centric targets in the structured form $[(u, v, Z), \ldots]$.

Depth map encoding. Inspired by SpatialBot [38], we encode the single-channel depth map into a three-channel uint8 depth image, making depth compatible with the vision tokenizer while preserving geometric structure.

Dual backbones for RGB and depth. Since the depth map constitutes a new visual modality, we introduce a dedicated depth backbone while keeping the rest of the model unchanged. To maximize modality alignment, we directly duplicate the RGB visual backbone and allocate it exclusively to depth, using the same architecture but separate parameters. Both backbones operate on the same patch grid, yielding token sequences that are naturally aligned for subsequent joint reasoning. To make the depth backbone effective without destabilizing the pretrained components, we adopt a two-stage training strategy. In Stage 1, we freeze all modules except the depth backbone and train only this backbone with a $10\times$ larger learning rate. In Stage 2, we unfreeze the full model and perform joint finetuning with the standard learning rate.

Causal multimodal fusion and structured decoding. We follow the standard causal VLM interface that organizes non-text modalities as explicit segments in a single token sequence. In typical VLMs, image patch tokens are inserted into the prompt and bracketed by dedicated boundary markers to make the visual prefix explicit without changing the underlying attention pattern. Following the same design principle, we introduce a symmetric pair of special tokens, `<dpt_start>` and `<dpt_end>`, to delimit the depth segment, within which we place the depth-backbone patch tokens. These non-overlapping marker spans keep RGB and depth explicit in the multimodal prefix while preserving the original causal architecture. Conditioned on the RGB–depth prefix and the instruction tokens, the model autoregressively generates a bracketed list of 3D targets with a fixed syntax (e.g., `[(123, 456, 789), ...]`). We then deterministically parse the generated string into (u, v, Z) tuples for evaluation and downstream geometric computation.

3.3 SpatialPoint-Bench

Evaluation data. We evaluate on **SpatialPoint-Bench**, whose images are sourced from the real-scene split of RoboAfford-Eval [33]. For each image, we predict a dense depth map using the same monocular depth estimator as in training, and use it to express both predictions and targets in the unified (u, v, Z) format.

For touchable points evaluation, RoboAfford-Eval [33] provides a ground-truth valid 2D region mask for each query. By pairing this mask with the estimated depth at the corresponding locations, we obtain surface-attached (u, v, Z) targets and a well-defined touchable points benchmark.

For air points evaluation, we generate air points QA pairs by following the same geometry-constrained synthesis procedure used for training (Section 3.1.3).

Touchable points: 2D accuracy from region masks. Each model response yields a set of predicted points $\{(\hat{u}_i, \hat{v}_i, \hat{Z}_i)\}_{i=1}^{N}$. Given the valid region mask M, per-query 2D accuracy is defined as the fraction of predicted $(\hat{u}_i, \hat{v}_i)$ that fall inside the valid region:

$$\mathrm{Acc}_{2D} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\big[M(\hat{u}_i, \hat{v}_i) = 1\big]. \tag{1}$$

Touchable points: depth error (MAE) against depth reference. We evaluate depth by comparing $\hat{Z}_i$ against the depth reference value $Z^{\mathrm{ref}}(\hat{u}_i, \hat{v}_i)$ at the predicted location, and report mean absolute error (mm):

$$\mathrm{MAE}_Z = \frac{1}{N} \sum_{i=1}^{N} \left| \hat{Z}_i - Z^{\mathrm{ref}}(\hat{u}_i, \hat{v}_i) \right|. \tag{2}$$

Table 4: Point-level micro results by category. MetPt is computed on direction-correct points in metric-offset queries (denominator: dir-passed points in has_metric samples).

	Direction↑			Between↑			Body-length↑		
Method	**DirPt**	**MetPt**	**FullPt**	**DirPt**	**MetPt**	**FullPt**	**DirPt**	**MetPt**	**FullPt**
Qwen3-VL-Inst-4B[11]	0.0573	0.1111	0.0050	0.0108	0.2222	0.0024	0.0627	0.0175	0.0011
RoboBrain 2.5[34]	0.0817	0.0827	0.0070	0.0543	0.0333	0.0018	0.0904	0.0455	0.0041
Ours (epoch 1)	0.4856	0.3096	0.1571	0.3825	0.1833	0.0701	**0.5637**	0.1959	0.1104
Ours (epoch 2)	**0.5264**	0.2864	0.1523	0.4031	**0.2711**	**0.1093**	0.4964	0.3093	0.1535
Ours (epoch 3)	0.5161	**0.3764**	**0.1867**	**0.4371**	0.2123	0.0928	0.5097	**0.3143**	**0.1602**

We additionally report MAE_Z on predictions *inside* the mask, *outside* the mask, and *overall*.

Air points: direction correctness. We evaluate whether a prediction satisfies the language-specified geometric relation in the camera frame. Let $\mathbf{c}$ denote the anchor (proxy-box center) of the referenced object, and $\mathbf{p}$ be the predicted 3D point converted by back-projecting pixel (u, v, Z) in the image to 3D space. For *direction* queries with a direction unit vector $\mathbf{d}$, we mark direction as correct if the prediction lies within a conic sector, i.e., $\angle(\mathbf{p}-\mathbf{c},\ \mathbf{d}) \le \alpha$, where α is a fixed angular tolerance.

For *between-object* queries, we define a cylindrical corridor along the segment connecting the two object anchors $\mathbf{c}_A$ and $\mathbf{c}_B$. A prediction is marked correct if it lies within a radius-ρ cylinder around the segment and projects onto the segment (within endpoints), capturing the "in-between" constraint.

Occupation exclusion. To enforce "not occupied by other objects", we reject predictions that fall inside any other object's inflated proxy box (constructed from lifted mask point clouds under estimated depth).

Air points: distance bias error (conditional). For queries that include an explicit distance constraint, we evaluate distance only when the direction or relationship constraint is satisfied. Let $r = \|\mathbf{p}-\mathbf{c}\|_2$ be the predicted displacement magnitude (with anchor $\mathbf{c}$ chosen per query type) and r^* be the required magnitude. We report distance bias as

$$\mathrm{Bias}_{dist} = |r - r^*|. \tag{3}$$

Body-length definition. To reduce sensitivity to mask noise, we define the body-length scale from the coarse 3D proxy box: the body length is set to half of the proxy box diagonal length. Accordingly, r^* in Equation (3) is specified in this normalized scale for body-length queries, and in metric units for metric-offset queries.

4 Experiments

In this section, we begin with training details and evaluation metrics, then present the main experimental results, followed by ablation studies and qualitative visualizations. Beyond offline evaluation, we further conduct real-robot experiments to validate practical effectiveness; the experimental setup, results, and video demonstrations are available in the supplementary material.

4.1 Experimental Setup

Dataset. Our network is trained with SpatialPoint-Data which includes approximately **1.9M** touchable points samples and **0.72M** air points samples. Our offline evaluation is conducted on SpatialPoint-Bench.

Model training. We build upon Qwen3-VL-Inst-4B [11] and introduce an explicit depth-token stream while keeping the original causal multimodal transformer unchanged (see Figure 3). We adopt a two-stage optimization scheme for the newly introduced depth branch. We first warm up the depth branch by training only the duplicated depth vision backbone while freezing the VLM, the RGB vision backbone, and the multimodal transformer; during this stage, we use a learning rate that is **10**$\times$ the base learning rate. We then unfreeze all components and jointly fine-tune the entire model with AdamW and cosine learning-rate decay to perform instruction-tuned point generation under mixed surface and air points supervision. Unless otherwise specified, we train for 1 epoch. All experiments are run on 8 GPUs with per-GPU batch size 4 and gradient accumulation 2. During inference, we decode the coordinate list from the LM head with $\texttt{top-}p$ sampling ($p = 0.9$), $\texttt{top-}k$ sampling ($k = 50$), and temperature 0.1.

To study longer adaptation on air points targets, we additionally continue fine-tuning on the air points subset for 1/2/3 epochs under the same architecture and recipe, and report the corresponding checkpoints in Tables 3 and 4.

4.2 Main Results

We report results on touchable points and air points following the evaluation protocols in Section 3.3. All air points metric results are reported conditional on relation correctness, since distance constraints are meaningful only when the intended geometric relation is satisfied.

4.2.1 Touchable Points.

Table 2 summarizes performance on RoboAfford-Eval for touchable points. We report the official 2D accuracy

Figure 4: Surface-target qualitative comparison on RoboAfford-Eval(touchable-point).

across the three categories—*Object Affordance Recognition* (OAR): identifying objects based on attributes such as category, color, size, and spatial relations, *Object Affordance Prediction* (OAP): localizing functional parts of objects to support specific actions, such as the handle of a teapot for grasping, and *Spatial Affordance Localization* (SAL): detecting vacant areas in the scene for object placement and robot navigation—as well as the overall average. Our experiments have demonstrated that our method achieves the sota performance among the current methods. For methods that generate (u, v, Z) outputs, we additionally report depth MAE (mm) against the monocular depth *reference* used in our evaluation pipeline, with an inside/outside breakdown; methods that do not output depth are marked as "–".

4.2.2 Air points.

Table 3 summarizes overall air points performance, while Table 4 reports point-level micro results by category. **DirPt/MetPt** are point-level micro accuracies. **MetPt** and **MeanErr** are computed on direction-correct points in metric-offset queries (i.e., conditional on direction correctness). **FullPt** measures joint direction+metric success on metric-offset queries. We report DirPt for relation correctness (direction-cone or between-cylinder), and for distance-constrained queries we additionally report MetPt@5cm, MeanErr (cm), and the joint success rate FullPt requiring both correct relation and distance satisfaction. Following our protocol, MetPt@5cm and MeanErr are computed only on relation-correct points in distance-constrained queries.

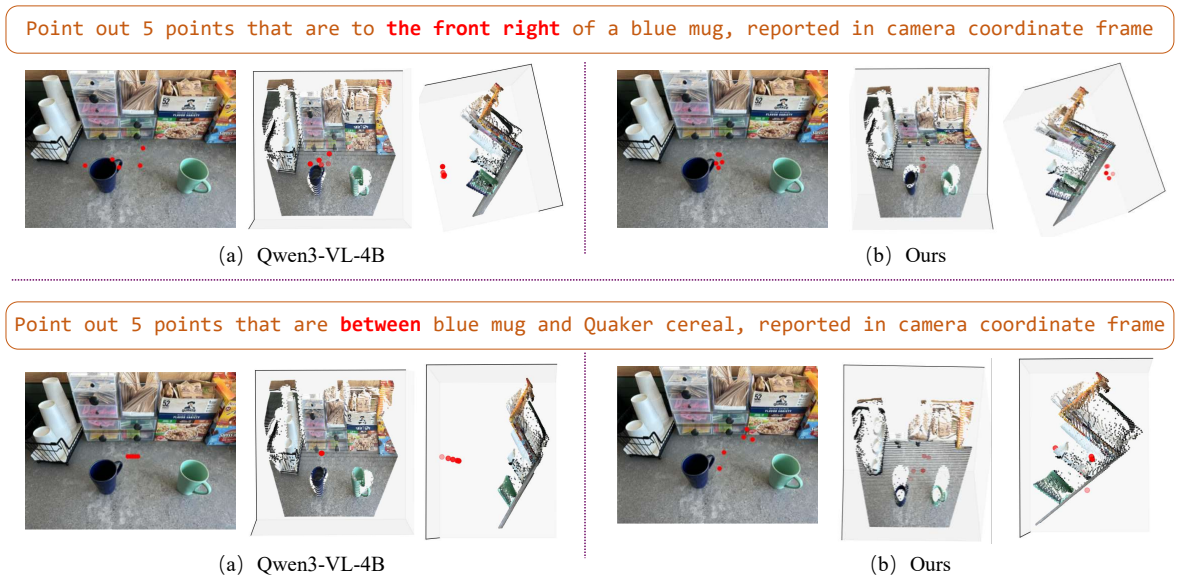


Figure 5: Free-space qualitative comparison on our benchmark. Under the same queries, our model better satisfies air points relation constraints, compared to Qwen3-VL, demonstrating more reliable air points target prediction.

Table 5: Touchable points ablations on RoboAfford-Eval. We vary the extra 3D input modality (depth map vs. point cloud), whether using a dual-branch backbone, whether depth is encoded in the SpatialBot-style (Depth→3ch) when applicable, and whether special depth tokens <dpt_start>/<dpt_end> are used. Depth MAE is reported only when valid depth predictions are available under our lifted-3D evaluation; otherwise it is marked as "–".

Model	Extra input	Dual-branch	SpatialBot enc.	Special tokens	Overall Acc↑	Depth MAE (mm)↓
Ours e1	Depth map	✓	✓	✓	**0.790**	**17.2**
T1	–	×	×	×	0.676	227.1
T2	–	×	×	✓	0.698	191.9
T3	Depth map	×	✓	×	0.708	49.2
T4	Depth map	×	✓	✓	0.707	22.1
T5	Depth map	✓	×	✓	0.736	38.1
T6	Point cloud	✓	–	✓	0.443	23.3

Table 6: Air points ablations on SpatialPoint-Bench (overall). We fix the dual-branch backbone and special depth tokens (used only when an extra input is provided), and only vary the input representation.

Variant	Epoch	Input	DirPt↑	MetPt@5cm↑	FullPt↑	MeanErr (cm)↓
Ours1	1	SpatialBot-encoded depth	0.4886	0.2587	0.1300	8.5008
Ours2	2	SpatialBot-encoded depth	**0.5088**	0.2907	0.1456	7.3034
Ours3	3	SpatialBot-encoded depth	0.5071	0.3347	0.1641	6.8084
A1	1	rgb-only	0.2007	0.0810	0.0181	18.6502
A2	2	rgb-only	0.3050	0.1050	0.0344	22.5311
A3	3	rgb-only	0.2556	0.0617	0.0174	26.3364
A4	1	Depth-3ch	0.4398	0.1521	0.0693	11.6903
A5	2	Depth-3ch	0.3980	0.1403	0.0574	14.1135
A6	3	Depth-3ch	0.4200	0.1667	0.0714	12.8000
A7	1	Point cloud (XYZ)	0.4667	0.4532	0.2138	7.24
A8	2	Point cloud (XYZ)	0.4712	0.4602	0.2362	6.47
A9	3	Point cloud (XYZ)	0.4549	0.4674	0.2284	5.25

For body-length constraints, the required offset is instantiated as a metric distance via the object-specific body-length scale derived from proxy geometry, enabling a unified 5 cm tolerance across query types. We include checkpoints fine-tuned for 1/2/3 epochs on the air points subset to study the effect of longer air points adaptation. Results show consistent improvement with more training epochs.

4.3 Ablations and Analysis

We ablate key design choices in our depth-aware formulation, including (A) whether depth is fed to the network, (B) alternative ways to feed depth, (C) the dual backbones design for depth tokens. Unless otherwise specified, all variants follow the same training data and evaluation protocols as in Section 4.1 and Section 3.3. Table 5 and Table 6 summarize ablation results on touchable and air points targets, respectively. For air points targets, we analyze two complementary axes: **relation correctness** and **distance fidelity conditioned on correct relation**.

(A) Feed depth or not. We remove all depth inputs to train and evaluate the model with RGB tokens and text tokens as input. This variant tests whether depth cues are essential for 3D point prediction and geometric consistency. The RGB-only variant drops on both touchable and air points benchmarks, indicating that explicit depth cues are important for executable 3D target prediction. (T1 & T2 for touchable points in Table 5 and A1 - A3 for air points in Table 6)

(B) Alternative ways to feed depth. We provide geometry as a lifted point cloud (from monocular depth) and convert it into a three-channel representation by directly quantizing (x, y, z) values into three channels. The resulting three-channel "geometry map" is fed to the model as an alternative geometry input, enabling a controlled comparison between *depth-token fusion* and *point-set-derived geometry input* under the same VLM interface. Compared with alternative geometry inputs, our depth-token fusion provides a simple and effective interface that improves relation correctness and metric precision. (T3-T6 & A4-A9)

Notably, point-cloud input shows competitive distance accuracy on air points but weaker direction correctness, and performs substantially worse than the baseline on touchable points. Despite these limitations, point-cloud geometry offers complementary advantages worth further exploration in

future work.

(C) Shared vs. dual backbones. We compare using a *shared* backbone for both RGB and encoded depth images versus duplicating the backbone with separate parameters. This ablation isolates whether a dedicated depth branch improves geometry token quality and downstream performance. The dual-backbone design consistently outperforms the shared-backbone variant. (T1-T4)

4.4 Qualitative Comparison

We provide qualitative comparisons on RoboAfford-Eval to complement quantitative results. Figure 4 compares touchable-points predictions between our model and Qwen3-VL, where we visualize predicted points in the image (and depth-lifted 3D when applicable) to illustrate localization quality. Figure 5 compares air points predictions, highlighting whether generated targets satisfy the intended spatial relation (direction/between) and distance constraints.

5 Conclusion

We introduced a minimalist execution interface for embodied tasks by formulating them as *language-conditioned 3D target point prediction* with two complementary target types: **touchable** and **air points**. Using open-source RGB images with monocular depth estimates, we constructed large-scale supervision and benchmarks under a unified (u, v, Z) camera-centric representation. Building on Qwen3-VL with an explicit depth-token stream, our model directly decodes structured point lists with the LM head and achieves consistent gains on both touchable and air points benchmarks under our evaluation protocols.

Current limitations include reliance on monocular depth estimates, which may degrade in textureless or reflective regions. Future work could extend embodied localization to dynamic scenes, integrate trajectory-level planning, and leverage world models to support richer spatial reasoning across embodied tasks.

References

[1] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[2] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[3] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on robot learning*, pages 894–906. PMLR, 2022.

[4] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[5] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

[6] Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. *arXiv preprint arXiv:2406.10721*, 2024.

[7] Yaoyao Qian, Xupeng Zhu, Ondrej Biza, Shuo Jiang, Linfeng Zhao, Haojie Huang, Yu Qi, and Robert Platt. Thinkgrasp: A vision-language system for strategic part grasping in clutter. *arXiv preprint arXiv:2407.11298*, 2024.

[8] Chenyang Ma, Kai Lu, Ta-Ying Cheng, Niki Trigoni, and Andrew Markham. Spatialpin: Enhancing spatial reasoning capabilities of vision-language models through prompting and interacting 3d priors. *Advances in neural information processing systems*, 37:68803–68832, 2024.

[9] 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 *European conference on computer vision*, pages 38–55. Springer, 2024.

[10] Tianhe Ren, Yihao Chen, Qing Jiang, Zhaoyang Zeng, Yuda Xiong, Wenlong Liu, Zhengyu Ma, Junyi Shen, Yuan Gao, Xiaoke Jiang, et al. Dino-x: A unified vision model for open-world object detection and understanding. *arXiv preprint arXiv:2411.14347*, 2024.

[11] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[12] 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. *Advances in Neural Information Processing Systems*, 37:135062–135093, 2024.

[13] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7429–7439, 2025.

[14] 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.

[15] Jianing Yang, Xuweiyi Chen, Shengyi Qian, Nikhil Madaan, Madhavan Iyengar, David F Fouhey, and Joyce Chai. Llm-grounder: Open-vocabulary 3d visual grounding with large language model as an agent. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7694–7701. IEEE, 2024.

[16] Yilun Chen, Shuai Yang, Haifeng Huang, Tai Wang, Runsen Xu, Ruiyuan Lyu, Dahua Lin, and Jiangmiao Pang. Grounded 3d-llm with referent tokens. *arXiv preprint arXiv:2405.10370*, 2024.

[17] Siming Yan, Min Bai, Weifeng Chen, Xiong Zhou, Qixing Huang, and Li Erran Li. Vigor: Improving visual grounding of large vision language models with fine-grained reward modeling. In *European Conference on Computer Vision*, pages 37–53. Springer, 2024.

[18] Zoey Guo, Yiwen Tang, Ray Zhang, Dong Wang, Zhigang Wang, Bin Zhao, and Xuelong Li. Viewrefer: Grasp the multi-view knowledge for 3d visual grounding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15372–15383, 2023.

[19] Runsen Xu, Zhiwei Huang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. Vlm-grounder: A vlm agent for zero-shot 3d visual grounding. *arXiv preprint arXiv:2410.13860*, 2024.

[20] Dingning Liu, Cheng Wang, Peng Gao, Renrui Zhang, Xinzhu Ma, Yuan Meng, and Zhihui Wang. 3daxisprompt: Promoting the 3d grounding and reasoning in gpt-4o. *Neurocomputing*, 637:130072, 2025.

[21] Zhihao Yuan, Jinke Ren, Chun-Mei Feng, Hengshuang Zhao, Shuguang Cui, and Zhen Li. Visual programming for zero-shot open-vocabulary 3d visual grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20623–20633, 2024.

[22] Rong Li, Shijie Li, Lingdong Kong, Xulei Yang, and Junwei Liang. Seeground: See and ground for zero-shot open-vocabulary 3d visual grounding. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 3707–3717, 2025.

[23] Zhenyang Liu, Yikai Wang, Sixiao Zheng, Tongying Pan, Longfei Liang, Yanwei Fu, and Xiangyang Xue. Reasongrounder: Lvlm-guided hierarchical feature splatting for open-vocabulary 3d visual grounding and reasoning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 3718–3727, 2025.

[24] Tatiana Zemskova and Dmitry Yudin. 3dgraphllm: Combining semantic graphs and large language models for 3d scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8885–8895, 2025.

[25] Anh Thai, Songyou Peng, Kyle Genova, Leonidas Guibas, and Thomas Funkhouser. Splattalk: 3d vqa with gaussian splatting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4712–4721, 2025.

[26] Zhangyang Qi, Zhixiong Zhang, Ye Fang, Jiaqi Wang, and Hengshuang Zhao. Gpt4scene: Understand 3d scenes from videos with vision-language models. *arXiv preprint arXiv:2501.01428*, 2025.

[27] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[28] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[29] 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 *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

[30] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[31] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[32] Wenlong Huang, Yu-Wei Chao, Arsalan Mousavian, Ming-Yu Liu, Dieter Fox, Kaichun Mo, and Li Fei-Fei. Pointworld: Scaling 3d world models for in-the-wild robotic manipulation. *arXiv preprint arXiv:2601.03782*, 2026.

[33] Yingbo Tang, Lingfeng Zhang, Shuyi Zhang, Yinuo Zhao, and Xiaoshuai Hao. Roboafford: A dataset and benchmark for enhancing object and spatial affordance learning in robot manipulation. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 12706–12713, 2025.

[34] Huajie Tan, Enshen Zhou, Zhiyu Li, Yijie Xu, Yuheng Ji, Xiansheng Chen, Cheng Chi, Pengwei Wang, Huizhu Jia, Yulong Ao, et al. Robobrain 2.5: Depth in sight, time in mind. *arXiv preprint arXiv:2601.14352*, 2026.

[35] Chi Chen, Bisheng Yang, Shuang Song, Mao Tian, Jianping Li, Wenxia Dai, and Lina Fang. Calibrate multiple consumer rgb-d cameras for low-cost and efficient 3d indoor mapping. *Remote Sensing*, 10(2):328, 2018.

[36] Lukas Rustler, Vojtech Volprecht, and Matej Hoffmann. Empirical comparison of four stereoscopic depth sensing cameras for robotics applications. *IEEE Access*, 2025.

[37] Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

[38] Wenxiao Cai, Iaroslav Ponomarenko, Jianhao Yuan, Xiaoqi Li, Wankou Yang, Hao Dong, and Bo Zhao. Spatialbot: Precise spatial understanding with vision language models. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9490–9498. IEEE, 2025.

[39] Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Zhuoguang Chen, Tao Jiang, and Hang Zhao. Depthvla: Enhancing vision-language-action models with depth-aware spatial reasoning. *arXiv preprint arXiv:2510.13375*, 2025.

[40] Chengmeng Li, Junjie Wen, Yaxin Peng, Yan Peng, and Yichen Zhu. Pointvla: Injecting the 3d world into vision-language-action models. *IEEE Robotics and Automation Letters*, 11(3):2506–2513, 2026.

[41] Lin Sun, Bin Xie, Yingfei Liu, Hao Shi, Tiancai Wang, and Jiale Cao. Geovla: Empowering 3d representations in vision-language-action models. *arXiv preprint arXiv:2508.09071*, 2025.

[42] 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. *arXiv preprint arXiv:2403.09631*, 2024.

[43] Chenming Zhu, Tai Wang, Wenwei Zhang, Jiangmiao Pang, and Xihui Liu. Llava-3d: A simple yet effective pathway to empowering lmms with 3d-awareness. *arXiv preprint arXiv:2409.18125*, 2024.

[44] Xiaoshuai Hao, Yingbo Tang, Lingfeng Zhang, Yanbiao Ma, Yunfeng Diao, Ziyu Jia, Wenbo Ding, Hangjun Ye, and Long Chen. Roboafford++: A generative ai-enhanced dataset for multimodal affordance learning in robotic manipulation and navigation. *arXiv preprint arXiv:2511.12436*, 2025.

[45] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *ArXiv*, abs/2502.13923, 2025. URL https://api.semanticscholar.org/CorpusID:276449796.

[46] UFactory. xArm 7 Collaborative Robot Arm — Product Specification. https://www.ufactory.cc/xarm-collaborative-robot-arm, 2023. 7-DOF collaborative robot arm with 700 mm reach and 3.5 kg payload.

[47] AgileX Robotics. TRACER 2.0: Indoor Two-Wheel Differential AGV. https://global.agilex.ai/products/tracer-2-0, 2024. 150 kg payload, 2 m/s maximum speed, 400 W motor power.

[48] AgileX Robotics. PiPER: 6-DOF Lightweight Robotic Arm. https://global.agilex.ai/products/piper, 2024. 1.5 kg payload, 626 mm reach, 0.1 mm repeatability.

SpatialPoint: Spatial-aware Point Prediction for Embodied Localization

Supplementary Material

A More Detailed Task Definitions and Evaluation Protocol

A.1 Touchable and Air Points

Touchable points are surface-grounded 3D targets for direct physical interaction, while air points are free-space 3D targets specified by spatial language. Together, they provide a unified representation of action targets for both object contact and free-space interaction.

A valid air point must satisfy two constraints: it must match the instructed spatial relation and lie in unoccupied free space rather than on or inside occupied objects. This makes air-point prediction more challenging than direction-only grounding because the model must reason about both relative geometry and free-space validity.

A.2 Unified Output Representation

For both touchable and air points, the model predicts targets in the unified format (u, v, Z), where (u, v) denotes the image-plane location and Z denotes metric depth. This representation keeps the prediction visually anchored while remaining directly executable in 3D.

Compared with full camera-frame 3D coordinates, (u, v, Z) is more image-aligned: (u, v) specifies the visually grounded location, while Z provides the depth needed to recover the final 3D point.

A.3 Detailed Geometric Evaluation for Air Points

As stated in the main paper, all distance-related metrics for air points are evaluated *only when the predicted point already satisfies the queried spatial relation*, since metric offsets are meaningful only for relation-valid predictions.

Object center. We use a unified object-center definition across all air-point categories. Each referenced object is lifted into 3D and represented by an object-level 3D proxy derived from its occupancy cue. The center of this proxy serves as the reference point for direction evaluation, between-object evaluation, and body-length-based metric computation.

Direction queries. For direction queries, we define a cone whose apex is the referenced object center and whose axis is the queried camera-frame direction. A prediction is considered relation-correct if it lies within 30° of the cone axis, corresponding to a full angular aperture of 60°. This tolerance allows moderate ambiguity between neighboring directions and avoids an overly brittle criterion.

Between-object queries. For between-object queries, relation correctness is evaluated with respect to the line segment connecting the two referenced object centers. A prediction is considered relation-correct only if: (i) its projection onto the segment lies between 10% and 90% of the segment length; and (ii) its perpendicular distance to the segment is at most 10 cm. This criterion encourages predictions that are genuinely between the two objects rather than merely close to the connecting line.

Body-length queries. For body-length queries, one body length is approximated as half of the diagonal length of the target object's 3D proxy box. A query such as *two body lengths in front of the mug* is therefore converted into a target metric distance equal to twice this object-specific unit along the queried direction.

Occupancy validity. A valid air-point prediction must lie in free space. Predictions that fall inside occupied object volume are rejected, even if they satisfy the queried relation. This requires the model to reason about both spatial relations and basic scene occupancy.

B Zero-Shot Generalization to Real-World Robot Manipulation and Navigation

We provide additional real-world robot results to complement the main-paper experiments on picking, placement, and navigation. We emphasize that in both demonstrations, **SpatialPoint** operates without any fine-tuning on the target scene, highlighting its cross-scene generalizability.

Pick-and-Place. We deploy SpatialPoint on an uFactory xArm7 robotic arm [46] in a tabletop setting with multiple boxes. SpatialPoint serves two roles: (1) predicting the grasping contact point on the target box via natural language instruction (*touchable point*, Figure 6(a)), and (2) predicting the target placement position for the grasped object via natural language instruction (*air point*, Figure 6(b)). We refer the reader to our project page, where the complete manipulation sequence video (`1-robotarm-pick-place.mp4`) is provided; in this video, the robot arm localizes the target box and places it at the specified destination.

Mobile Manipulation. We deploy SpatialPoint on a mobile robot built upon AGILEX Tracer 2.0 [47] and AGILEX Piper [48]. SpatialPoint supports three sequential subtasks within a single scene: (1) predicting the approach position near the target object to enable grasping (*touchable point*, Figure 7(a)), (2) localizing the target object for interaction (*touchable point*, Figure 7(b)), and (3) predicting the placement position for the grasped object (*air point*, Figure 7(c)).

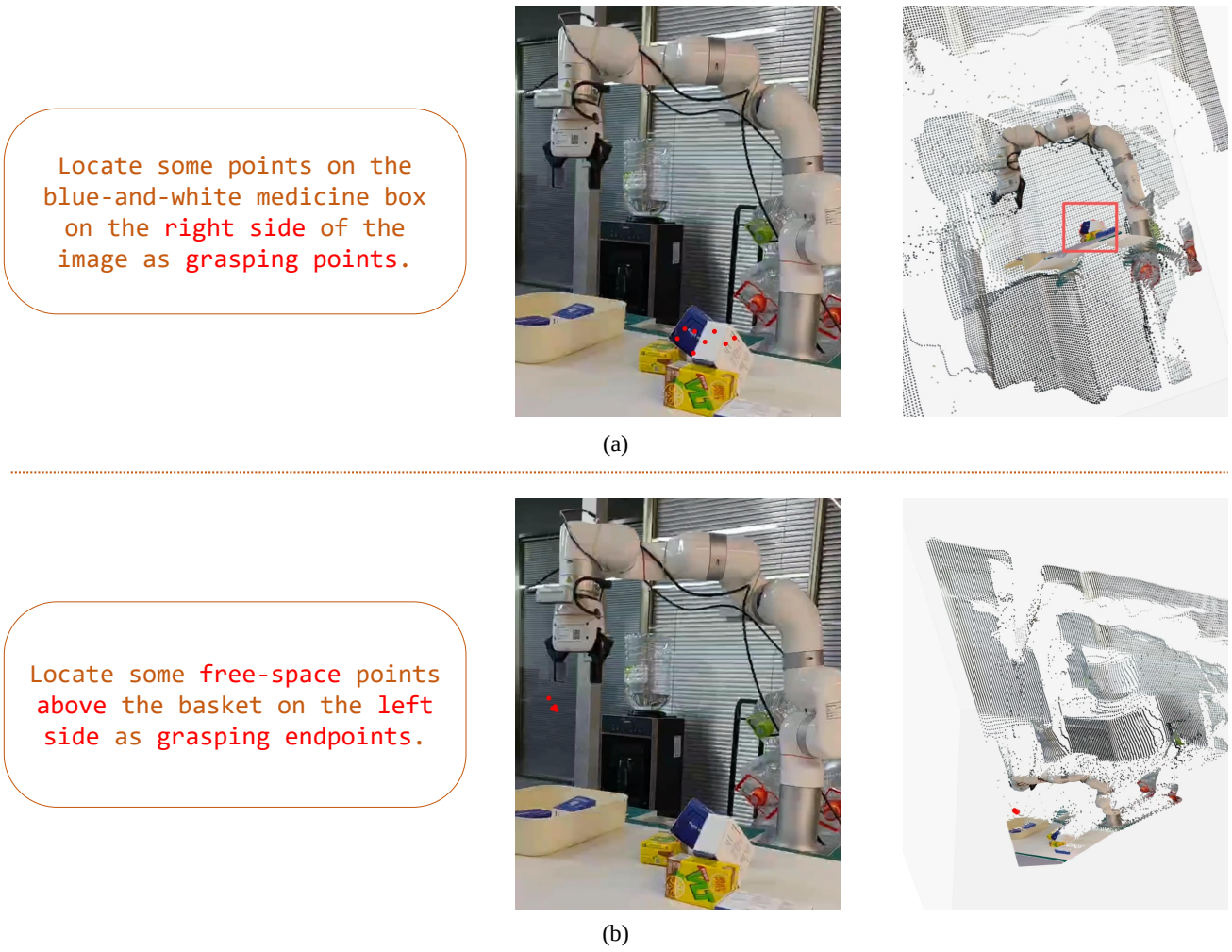


Figure 6: Zero-shot real-world results on robot pick-and-place.

We refer the reader to our project page, where the complete process video (`2-mobile-navigation.mp4`) is provided; in this video, the mobile robot navigates to the target location and retrieves the bottle.

Long-Horizon Task: Package Reordering. We apply SpatialPoint to a long-horizon manipulation task involving the reordering of multiple packages within a multi-compartment open shelf. The packages are placed in individual compartments, and a uFactory xArm7 robotic arm [46] is instructed to reorder them according to human voice or text commands. SpatialPoint supports two sequential subtasks: (1) predicting the grasping point on the target package (*touchable point*, Figure 8(a)), and (2) predicting the placement destination within the specified compartment following the human instruction (*air point*, Figure 8(b)). We refer the reader to our project page, where the complete manipulation sequence video (`3-longhorizon-pack-reranking.mp4`) is provided.

C Additional Experimental Results

This section presents additional qualitative comparisons and ablation results. We extend the qualitative comparison by adding RoboBrain2.5 [34] as an extra baseline alongside Qwen3-VL-4B [11] and our model, and we further report a supplementary air-point ablation under SpatialBot-style depth encoding.

C.1 More Qualitative Comparisons

Compared with the main paper, we additionally include RoboBrain2.5 [34] as a baseline and present three-model qualitative comparisons under identical instructions.

Figure 9 shows additional touchable-point examples, while Figure 10 shows additional air-point examples.

C.2 Supplementary Ablations on Air Points

Owing to space limitations, the main paper reports only the default air-point ablation setting. Here we include additional ablation results under SpatialBot-style depth encoding to separately show the effects of the dual-backbone design and special depth tokens. All variants use the same SpatialBot-style depth representation and are trained for 3 epochs.

Table 7 summarizes the results. Compared with the condensed presentation in the main paper, these additional results provide a more explicit breakdown of the two architectural choices, since the default setting there already includes both components.

Overall, the trends are consistent with those reported in the main paper and further clarify the contribution of

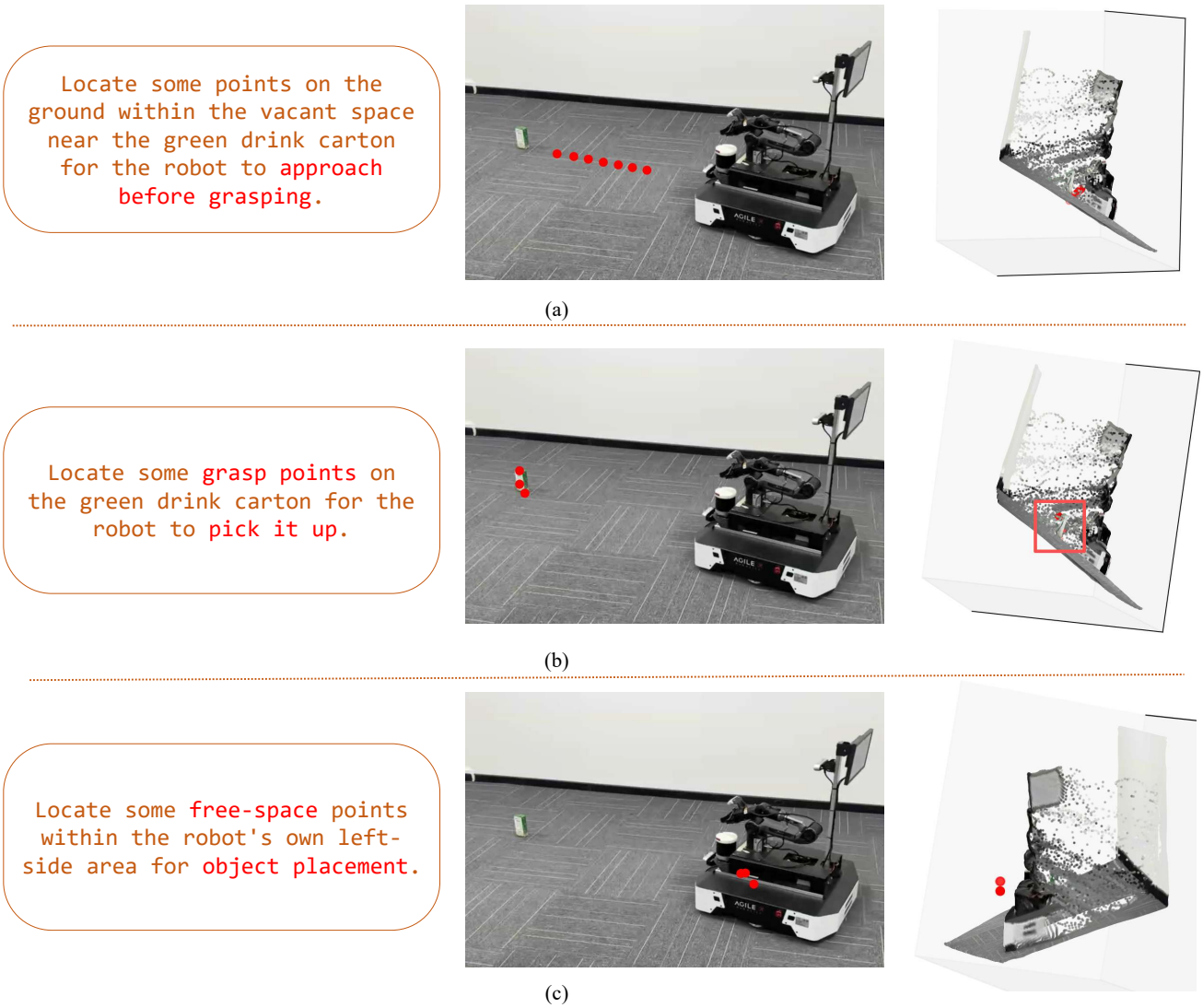


Figure 7: Zero-shot real-world robot results on mobile navigation and picking.

the dual-backbone design and special depth tokens under SpatialBot-style depth encoding.

D Qualitative Analysis of Failure Cases

D.1 Failure Analysis on Touchable Points

We observe three common failure patterns in touchable-point prediction, corresponding to progressively more severe errors in localization, constraint satisfaction, and target grounding.

Fine-grained localization error. In some cases, the model identifies the correct target region but predicts a slightly shifted point, as illustrated in Figure 11(a) and (b). Such errors are common near object boundaries or thin structures, where even a small image-plane deviation can lead to a noticeable 2D mismatch and a larger 3D error after depth lookup. This also helps explain why in-mask depth errors are typically smaller than out-of-mask ones: small boundary shifts can already destabilize both image-plane alignment and the associated depth value.

Fine-grained constraint error. A second failure mode occurs when the model captures the coarse spatial relation but fails to satisfy the finer constraint in the instruction, as shown in Figure 11(c). For example, when the instruction refers to a vacant region on one side of an object, the prediction may fall on the correct side but not in a truly vacant or interaction-valid area. These cases suggest that touchable-point prediction requires not only coarse directional understanding, but also finer discrimination of locally valid target regions.

Relational grounding error. A third and more severe failure mode arises when the model misidentifies the referred target or misgrounds the underlying spatial relation, as shown in Figure 11(d). In such cases, the prediction is anchored to the wrong object or relational target rather than being a small local deviation.

(a)

(b)

Figure 8: Zero-shot real-world robot results on long-horizon task.

Table 7: Supplementary air-point ablations under SpatialBot-style depth encoding.

Variant	Epoch	Dual Backbone	Special Tokens	DirPt↑	MetPt@5cm↑	FullPt↑	MeanErr (cm)↓
Ours1	1	✓	✓	0.4886	0.2587	0.1300	8.5008
Ours2	2	✓	✓	**0.5088**	0.2907	0.1456	7.3034
Ours3	3	✓	✓	0.5071	**0.3347**	**0.1641**	**6.8084**
A10	1	×	×	0.4088	0.1885	0.0795	11.2143
A11	2	×	×	0.4413	0.1916	0.0846	11.2934
A12	3	×	×	0.4403	0.2130	0.0937	10.2762
A13	1	×	✓	0.4315	0.1913	0.0854	11.5730
A14	2	×	✓	0.4576	0.2453	0.1134	10.9660
A15	3	×	✓	0.4650	0.2482	0.1164	9.7431
A16	1	✓	×	0.4252	0.1911	0.0827	10.5795
A17	2	✓	×	0.4543	0.2299	0.1043	10.8764
A18	3	✓	×	0.4609	0.2454	0.1133	10.0685

D.2 Failure Analysis on Air Points

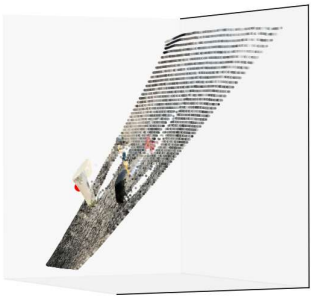
Air-point prediction is more challenging than touchable-point prediction because the target lies in free space and must satisfy both spatial and occupancy constraints. We therefore focus on representative relation-level failures and leave metric errors aside, as they are harder to interpret in isolation.

Relational direction misinterpretation. In this case, the model recognizes the referenced object but fails to place the target in the instructed relational direction, as shown in Figure 12(a). This indicates that air-point prediction requires not only identifying the correct reference object, but also accurately grounding directional relations in 3D.

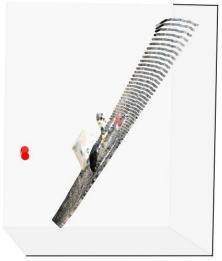
Weak image support for free-space targets. In this case, the queried free-space target is geometrically well defined in camera-frame 3D space but only weakly supported by the visible 2D image, especially near image boundaries, as shown in Figure 12(b). This highlights a key difficulty of air-point prediction: a target may be clear in 3D yet hard to infer from the image plane alone.

Occupancy-awareness failure. In this case, the model captures the coarse relational cue but predicts a point that still falls inside or too close to occupied object space, as shown in Figure 12(c). This indicates that successful air-point prediction requires not only correct directional reasoning, but also stronger awareness of local 3D occupancy and free-space feasibility.

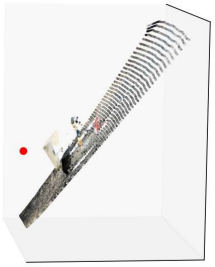
Select the leftmost object in the image.



(a) Ours

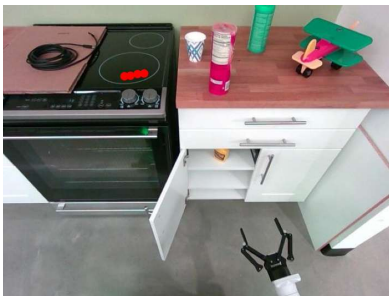


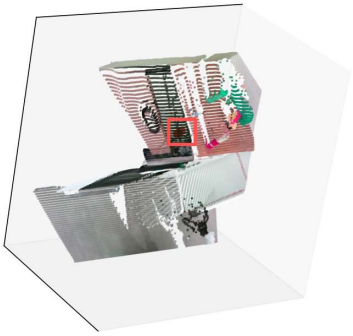
(b) Qwen3-VL (4B)



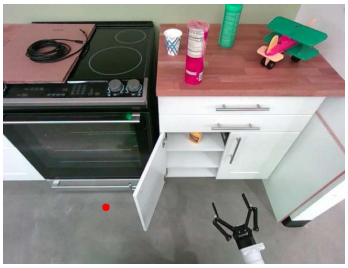
(c) RoboBrain2.5 (8B)

Locate several points within a vacant area on the front portion of the stove.

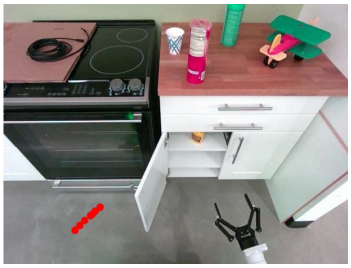


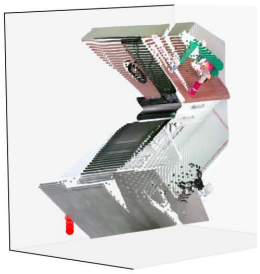


(a) Ours



(b) Qwen3-VL-4B

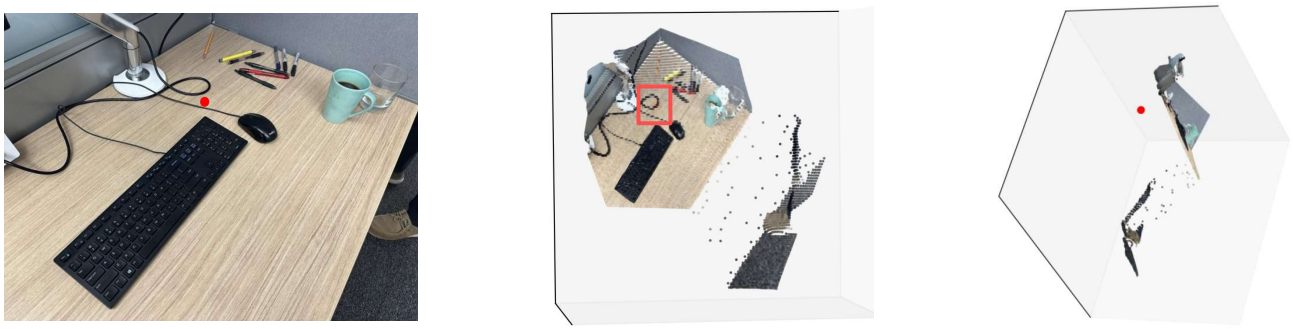




(c) RoboBrain2.5 (8B)

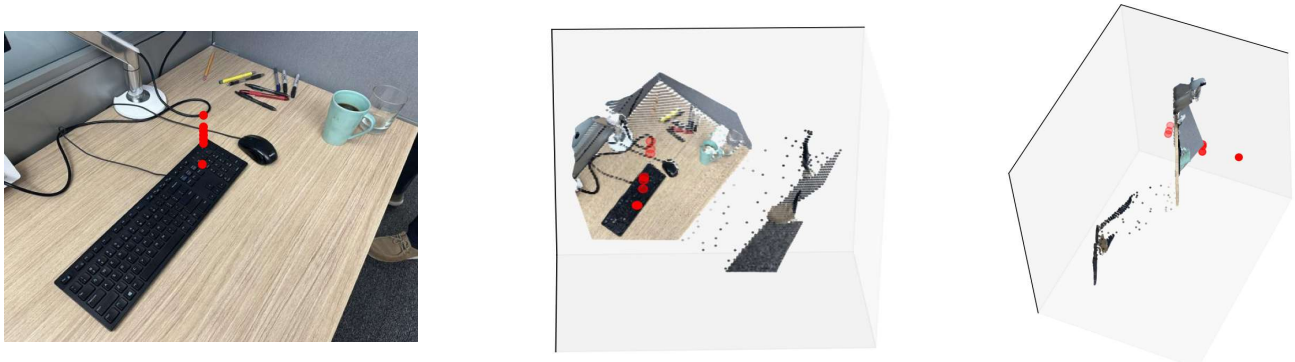
Figure 9: Additional qualitative comparisons on touchable points prediction. We compare 3 models under the same instructions.

Point out some points that are to the lower front left of a yellow pencil, about 3 body lengths away. (camera frame)



(a) Ours

(b) Qwen3-VL (4B)



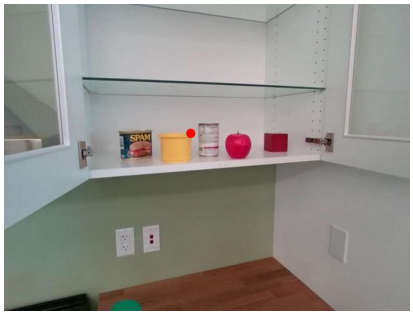
(c) RoboBrain2.5 (8B)

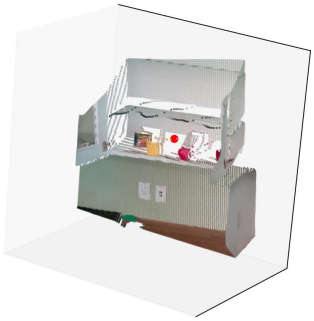
Figure 10: Additional qualitative comparisons on air points prediction. We compare 3 models under the same instructions.

What part of a mug should be gripped to lift it?

(a)

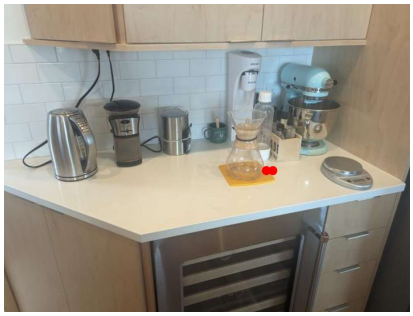
What container can be used to store and dispense liquids or chemicals, often featuring a handle for easy carrying?





(b)

Pinpoint several spots in the vacant area that lies to the right of the glass container.



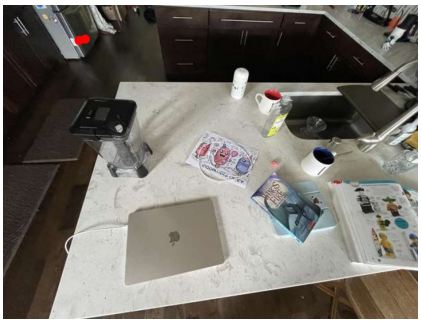
(c)

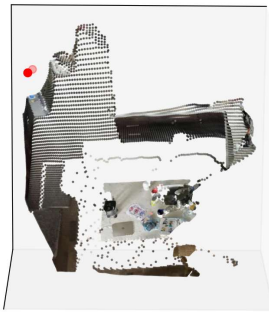
Spot several points on the can right next to the apple in the image.

(d)

Figure 11: Failure cases on touchable-point prediction. (a) and (b) Fine-grained localization errors. (c) Fine-grained constraint error. (d) Relational grounding error.

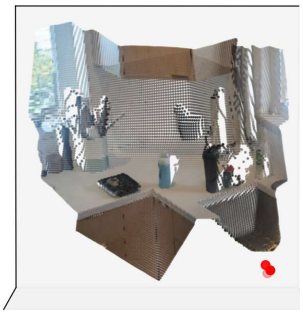
Point out 4 points that are between a white mug and a silver refrigerator. (camera frame)





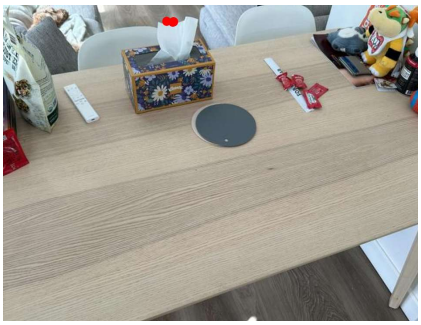
(a)

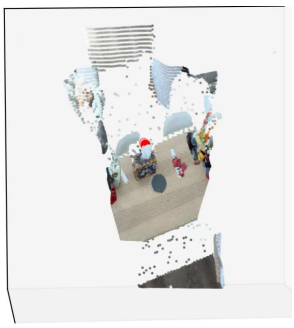
Point out 4 points that are to the lower front right of the sink. (camera frame)



(b)

Point out 6 points that are above a tissue box. (camera frame)





(c)

Figure 12: Failure cases on air-point prediction. (a) Relational direction misinterpretation. (b) Weak image support for free-space targets. (c) Occupancy-awareness failure.