# KNOWING WHERE TO GO: GENERALIZING MOBILE MANIPULATION TO UNSEEN ENVIRONMENTS FROM A SINGLE VIDEO

**Anonymous authors**
Paper under double-blind review

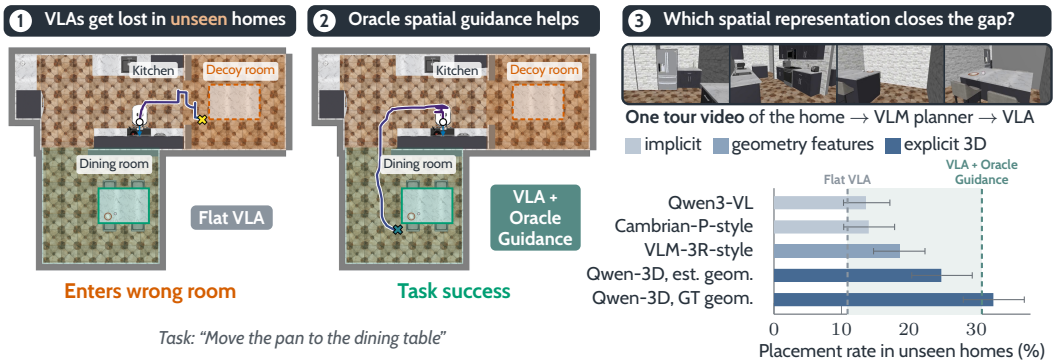



**Figure 1: Knowing where to go. (1)** In a difficult unseen home, a flat VLA ($\pi_{0.5}$) heads into a decoy room and never reaches the dining table. **(2)** Given oracle spatial guidance (ground-truth navigation goals), the same VLA goes straight to the dining room and completes the task. **(3)** We study which spatial representation lets a VLM planner recover this spatial guidance from a single tour video of the unseen home.

## ABSTRACT

Vision-language-action policies have achieved impressive mobile manipulation capabilities, but remain brittle in unfamiliar multi-room environments. We find that much of this brittleness stems from failures of spatial generalization: strong policies often resort to reactive exploration when task-relevant locations move outside familiar layouts. Oracle spatial guidance largely prevents this degradation. We study how spatial context from a prior environment walkthrough video should be represented to guide mobile manipulation policies under environment shift. We find that strong in-distribution spatial prediction can be a poor predictor of out-of-distribution transfer, while models that explicitly encode 3D structure generalize more robustly to new robot environments. Building on this, we develop a hierarchical policy that uses a single environment video to provide spatially grounded guidance to a VLA. We find that navigation performance of mobile manipulation policies is heavily bottlenecked by spatial understanding, and that our strongest policy more than doubles the placement rate of its flat counterpart (24.8% vs. 10.9%) in our hardest setup. For more information, see https://know-where-to-go.github.io/.

## 1 INTRODUCTION

Vision-language-action models (VLAs) have emerged as a promising approach for generalist robot control, combining large-scale vision-language pretraining with robot action data to acquire broad manipulation skills and generalize to new tasks, objects, and environments (Intelligence et al., 2025; Team, 2026; Team et al., 2025; Kim et al., 2024). Recent models have extended these capabilities beyond tabletop manipulation to mobile settings, where a single policy can jointly navigate and manipulate (Intelligence et al., 2025; Team et al., 2026; Intelligence et al., 2026; NVIDIA et al., 2025). However, deploying such policies in unfamiliar multi-room environments remains challenging. Unlike tabletop manipulation, mobile manipulation requires reasoning about task-relevant objects and locations that may lie far outside the robot's current field of view.



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We find that this spatial understanding remains a major weakness even for the strongest current policies. Under challenging layout shifts, state-of-the-art mobile manipulation policies frequently fall back on reactive exploration, repeatedly searching irrelevant parts of the environment before reaching task-relevant locations (§ 3). Providing ground-truth spatial guidance largely prevents this degradation without improving the underlying manipulation skills. Scaling manipulation experience alone does not appear to solve the problem of knowing where to go.

A natural way to overcome this limitation is to provide the policy with spatial context from a prior walkthrough video. Doing so, however, requires spatial representations that remain useful beyond the environments in which they were trained. Recent work has sought to improve spatial reasoning in VLMs through spatial supervision (Yang et al., 2025; 2026), geometric features (Fan et al., 2026; Qu et al., 2026; Zhang et al., 2026a; Huang et al., 2025; Wang et al., 2025a), and explicit 3D representations (Zhu et al., 2025; Lin et al., 2026; Liang et al., 2026; Hong et al., 2023). Yet these approaches are evaluated with different backbones, training data, and benchmarks, making it difficult to understand which design choices improve spatial generalization. Moreover, strong performance on spatial reasoning benchmarks (Yang et al., 2024; Zhang et al., 2026b; Jia et al., 2026; Mayer et al., 2025) does not necessarily imply that the learned representations will transfer to robot control in a new environment.

In this work, we study how spatial context from a prior video should be represented to reliably guide mobile manipulation under environment shift. Our contributions are:

- **A spatially grounded hierarchical VLA.** A high-level VLM conditions on a video tour of the environment and provides spatially grounded subtasks to an egocentric VLA, letting us vary the planner's spatial representation while keeping the low-level policy fixed (§ 4).
- **A controlled study of spatial VLMs.** Implicit and explicit geometric conditioning and supervision help our models fit our spatial midtraining data, but Qwen-3D (Lin et al., 2026), which conditions on an explicit 3D feature map, transfers most robustly to our target environment (§ 5).
- **Multi-room evaluation in unseen layouts.** On multi-room extensions of RoboCasa365 (Nasiriany et al., 2026) with previously unseen layouts, stronger spatial understanding generally leads to better mobile manipulation. In our hardest setting, our strongest policy achieves a $24.8\%$ placement rate, more than double the $10.9\%$ of its flat counterpart, and $32.5\%$ with ground-truth geometry (§ 6).

## 2 Related Work

### 2.1 Mobile Manipulation

Classical mobile manipulation and vision-and-language navigation (VLN) pipelines (Anderson et al., 2018; Krantz et al., 2020) typically rely on explicit spatial representations such as 3D metric maps (Huang et al., 2023), hierarchical scene graphs (Rana et al., 2023), or topological graphs (Chiang et al., 2024) which can be brittle and limit scalability in novel environments. Recent foundation model approaches to VLN (Cheng et al., 2025) often focus strictly on locomotion. In contrast, our approach eliminates the dependence on explicit maps. By directly processing a prior video tour, we jointly predict both navigation and manipulation actions to achieve end-to-end task completion in unseen multi-room environments.

For manipulation, hierarchical VLAs use intermediate language commands or visual plans to guide low-level action generation (Shi et al., 2025; Li et al., 2025; Team et al., 2025; Intelligence et al., 2025). These approaches improve manipulation performance and generalization to unseen homes, but their evaluations focus on single-room tasks. Extending to multi-room environments requires knowing where task-relevant objects are beyond the robot's immediate observations. Indeed, $\pi_{0.5}$ (Intelligence et al., 2025) identifies richer context and memory as necessary extensions for navigating between rooms and remembering where objects are stored. We provide spatial context through a prior video tour and compare representations for a foundation-model planner, while a learned egocentric policy executes navigation and manipulation.

Existing benchmarks like RoboCasa, RoboCasa365 (Nasiriany et al., 2024; 2026), and EBench (Gao et al., 2026) focus strictly on single-room manipulation. While Habitat 2.0 (Szot et al., 2021) and HomeRobot OVMM (Yenamandra et al., 2023) test multi-room settings, they predate modern VLAs and rely on exact object coordinates or active online search rather than prior spatial context.

Additionally, the 2025 BEHAVIOR Challenge (Li et al., 2024) features long-horizon multi-room tasks but evaluates policies exclusively within seen training environments. To our knowledge, no simulated benchmark evaluates VLA mobile manipulation in unseen multi-room settings. We extend RoboCasa365 with unseen layouts (§ 6.1) to probe this frontier.

### 2.2 Spatial Understanding in Vision-Language Models

Progress in VLM spatial understanding is difficult to assess because benchmarks mix distinct capabilities and can reward shortcuts. VSI-Bench (Yang et al., 2024) evaluates spatial understanding from indoor videos through tasks including object counting, room-size estimation, and relative-direction reasoning. However, Brown et al. (2025b) identify exploitable non-visual statistical shortcuts in VSI-Bench and other benchmarks, showing that benchmark gains need not reflect improved visual understanding. GST-Bench (Zhang et al., 2026b) highlights that many evaluations do not cleanly separate single-frame-solvable questions from those requiring globally consistent cross-view reasoning. These findings make it unclear whether benchmark gains reflect the spatial capabilities needed for robot planning. We study whether stronger spatial reasoning translates into better mobile manipulation.

Methods for improving VLM spatial understanding fall into three categories. *Implicit spatial learning* develops spatial representations within image- or video-based models: Cambrian-S (Yang et al., 2025) scales spatial question-answering, while Cambrian-P (Yang et al., 2026) adds auxiliary camera-pose prediction. *Geometric foundation-model augmentation* leverages geometric features or supervision from pretrained 3D models. VLM-3R (Fan et al., 2026) injects scene and camera-motion tokens, 3DRS (Huang et al., 2025) distills 3D-aware features into VLMs, and Loc3R-VLM (Qu et al., 2026) uses camera-pose priors to support layout reconstruction and viewpoint-aware reasoning. *Explicit 3D inputs* directly expose spatial structure to the VLM: 3D-LLM (Hong et al., 2023) consumes point-cloud features, LLaVA-3D (Zhu et al., 2025) and Qwen-3D (Lin et al., 2026) add 3D positional embeddings, and HiSpatial (Liang et al., 2026) incorporates auxiliary metric-scale point maps as inputs. These works continuously push the state of the art in common spatial understanding benchmarks, but their use of different backbones and training recipes makes it difficult to directly compare them. In this work, we present a study across several prominent methods to find which architectural factors are important.

## 3 Multi-Room Mobile Manipulation: Probing SoTA

We find that while state-of-the-art policies have impressive manipulation capabilities, their lack of global spatial knowledge leads them to spend their time wandering aimlessly, limiting success rates.

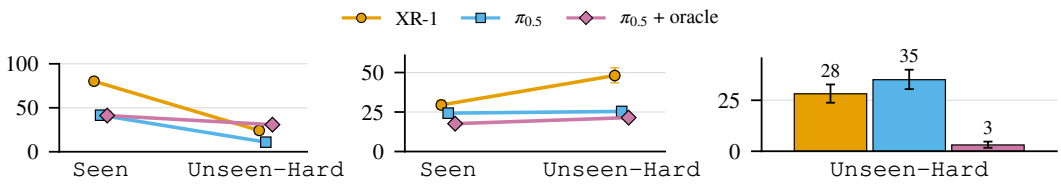
### 3.1 Problem Setting

We consider multitask mobile manipulation in environments unseen during robot-policy training. We assume access to a single offline dataset $\mathcal{D} = \{(\tau_i, V_i)\}_{i=1}^{N}$, where $\tau_i$ is a robot demonstration and $V_i$ is an associated *tour*: a prerecorded video of the environment showing its spatial layout and object locations. Demonstrations span multiple tasks and are collected in a finite set $\mathcal{E}_{\text{train}}$ of training environments. The tours are collected at the start of each episode ($t = 0$), before task execution.

At evaluation time, the policy must complete tasks in a set $\mathcal{E}_{\text{test}}$ of environments disjoint from those represented in the robot-training dataset (i.e., $\mathcal{E}_{\text{train}} \cap \mathcal{E}_{\text{test}} = \emptyset$). The policy receives an offline tour also at time $t = 0$ and the objective is to use it to guide task execution in a new environment.

### 3.2 Motivating Experiment

Recent works have pushed the boundaries of mobile manipulation (Intelligence et al., 2025; Team et al., 2026; NVIDIA et al., 2025) and probing the capabilities of these models (Nasiriany et al., 2026). However, it is unclear how these models perform in unseen multi-room settings. Thus, we present a pedagogical experiment to understand how SOTA policy performance degrades in these settings, and whether global spatial information can help recover in-distribution performance.




**(a)** Placement rate (%) **(b)** Distance travelled (m) **(c)** Time in decoy room (%)

**Figure 2:** (a) Placement rate and (b) mean distance travelled by the mobile base per episode in `Seen` and `Unseen-Hard`; (c) fraction of each `Unseen-Hard` episode spent in the decoy room. XR-1 achieves strong in-distribution placement rates but in unseen environments it spends more time navigating into the wrong rooms, leading to a lower placement rate. $\pi_{0.5}$ achieves lower in-distribution placement rate but with a navigation oracle, it spends less time travelling into the irrelevant room, leading to stronger out-of-distribution placement rates.

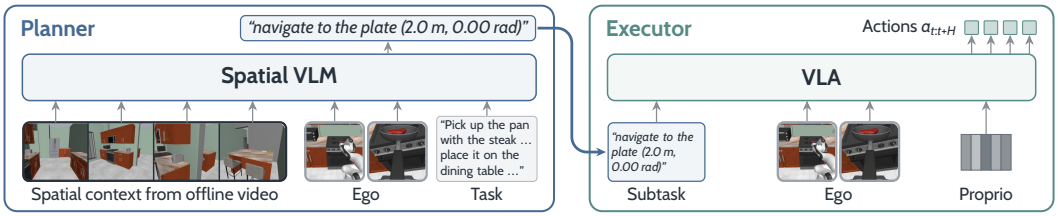



**Figure 3: Our spatially-aware planner-executor architecture.** Our spatial VLM planner is trained to extract spatial representations from an offline video of the environment. It predicts both task-level plans, such as manipulation subtasks, and spatial information, like metric navigation targets, completely end to end. The VLA actualizes the plan.

We experiment in our fork from RoboCasa365 described in detail in § 6.1, comparing results from the `Seen` and `Unseen-Hard` settings which probe in-distribution performance and performance under a large change in layout respectively. The `Unseen-Hard` setting moves task-relevant objects to a separate room and adds a decoy room with no task-relevant objects. We compare Xiaomi-Robotics-1 (Team et al., 2026), the current state of the art for RoboCasa365, with two variants of $\pi_{0.5}$ trained on our data mix (§ 6.2): a flat policy conditioning on the full high-level instruction and a hierarchical policy augmented with an oracle giving it ground-truth navigation goals. See App. D.1 for details.

In Fig. 2a we study change in placement rate (App. C.5) from seen to unseen settings. We find that while XR-1, which was pretrained on 100k hours of mobile manipulation data, achieves by far the best in-distribution placement rate, its performance drops $69.8\%$ under the distribution shift. $\pi_{0.5}$ sees a similar relative drop after starting from a lower in-distribution placement rate. In contrast, $\pi_{0.5}$ + oracle degrades far less, dropping only $25.2\%$ and outperforming XR-1. Next, we study the total distance travelled (Fig. 2b), finding that the policies travel similar distances for the seen layouts, but that XR-1 spends far more time navigating in the difficult unseen rooms. A possible explanation for this is that its extensive mobile manipulation pretraining instilled a strong prior for reactive exploration but this is insufficient for the difficult multi-room environment, leading it to wander aimlessly. We additionally study time spent in the decoy room (Fig. 2c), finding that the oracle policy is the only model that does not spend substantial time in it, substantiating the claim that even SOTA reactive policies are prone to getting lost in unseen environments. We visualize representative rollouts in App. E.1.

We conclude that even state-of-the-art policies trained on extensive mobile manipulation data are poorly equipped to handle unseen multi-room environments and that ground-truth spatial information helps address this. Consequently, we expect models with superior spatial awareness to improve mobile manipulation.

## 4 Building a Spatially Aware VLA

Most existing VLA literature is built around purely reactive policies, but our pipeline assumes access to offline data to help infer spatial context. Building on § 3.1, we detail our architecture.

While fully end-to-end policies demonstrate impressive capabilities, recent work has achieved the strongest results combining high-level planners with low-level executors, especially for long-horizon

tasks. Prior planner-executor interfaces include subtask instructions (Intelligence et al., 2025; Shi et al., 2025), code (Liang et al., 2022; Fu et al., 2025), and goal frames (Intelligence et al., 2026). We build on this by proposing that the spatial context should be handled by the planner while preserving an egocentric-only executor, as shown in Fig. 3. First, spatial context carries information about several rooms, making it heavy and slow to condition on. The planner need not replan as frequently as the executor, making it better equipped to handle the heavy context. Second, large datasets used for pretraining low-level executors like VLAs do not generally include much if any spatial context. Consequently, open-source pretrained policies cannot reliably condition on it and it would be difficult to train them to do so. Rather, we adopt a framework where the planner manages spatial context while passing spatially informed planning to the executor.

We opt for simplicity to isolate the effects of our design decisions. We instantiate the planner as a VLM trained to predict subtask instructions end-to-end and the executor as a VLA (we choose $\pi_{0.5}$ (Intelligence et al., 2025)) to predict actions based on these subtasks. To communicate task-relevant spatial information, the high-level policy predicts a *navigation hint*, the polar coordinate of the target object relative to the robot base, whenever it predicts a navigation subtask. We found this simple interface well-suited to steering the low-level policy and a tractable source of supervision for the planner.

## 5 What do Spatial VLMs Learn?

In § 3 we found spatial understanding to be an important bottleneck for unseen multi-room mobile manipulation. Thus, under our framework from § 4, we study *how* to impart spatial understanding in our VLM planner and *what* to expect from it.

As discussed in § 2.2, prior spatial VLM studies are confounded by differences in training data, budgets, and backbones. Furthermore, frontier spatial VLMs evaluate on tasks such as room size estimation and object counting (Yang et al., 2024), and it is unclear whether their spatial representations transfer to our setting.

We probe VLM spatial understanding by training two sets of models. (1) To disentangle the design choices underpinning spatial VLMs, we instantiate and train several prominent architectures under a shared training recipe. (2) To evaluate open-source spatial VLMs, we LoRA finetune existing models. Since our planner must predict polar navigation hints, we train and evaluate on this task.

### 5.1 Architecture Candidates

We use a Qwen3-VL-4B (Bai et al., 2025) backbone, fully finetune (unless otherwise noted) for 10k gradient steps (≈2 epochs) and train with the data mix described in § 5.2. Tour-conditioned models receive a 16-image tour and egocentric robot views, all at $256 \times 256$. For additional details, see App. B.3. We train the following architectures:

- *No tour* (**NT**), which conditions only on egocentric viewpoints and serves as a baseline demonstrating performance possible without conditioning on the offline tours. This is trained for one epoch only on the polar prediction data since predicting from purely egocentric images is not supported by VSI-590K (Yang et al., 2024).
- *Polar head* (**PH**), which adds a polar regression head predicting target locations.
- *Cambrian-S style* (**CS**) (Yang et al., 2025) involving no explicit spatial inputs or supervision and claiming to implicitly develop spatial understanding through VQA.
- *Cambrian-P style* (**CP**) (Yang et al., 2026) adding a camera pose prediction head to encourage implicit geometric understanding.
- *VLM-3R style* (**V3R**) (Fan et al., 2026) adding CUT3R (Wang et al., 2025c) representations encoding scene geometry and camera pose as an explicit input.
- *Qwen-3D style* (**Q3D**) (Lin et al., 2026) unprojecting the scene into a point cloud, replacing Qwen3-VL's position encodings with 3D RoPE, and LoRA (Hu et al., 2021) finetuning the attention layers.

Additionally, to ensure that we are fairly evaluating the capabilities of VLMs for which the authors conducted extensive full finetunes, we also LoRA finetune Cambrian-S and Cambrian-P on our bearing prediction data for 1k gradient steps (≈1 epoch). Since these models have already been

**Table 1:** Held-out bearing accuracy (%, within one of 16 angular bins) at the final checkpoint. Q3D uses ground-truth tour geometry. Explicit 3D inputs (Q3D) transfer best to RoboCasa despite the lowest in-distribution accuracy among tour-conditioned models.

| | | *Released 7B, LoRA-ft.* | | *Qwen3-VL-4B, midtrained* | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Cam.-S | Cam.-P | NT | PH | CS | CP | V3R | Q3D |
| MolmoSpaces | Houses | 51.6 | 59.9 | 33.0 | 70.3 | 70.3 | 70.5 | **72.0** | 46.7 |
| RoboCasa | Easy | 33.2 | 48.6 | 31.4 | 41.6 | 45.0 | 47.2 | 50.2 | **67.0** |
| | Hard | 27.0 | 41.3 | 25.0 | 37.1 | 39.3 | 43.1 | 40.1 | **56.5** |

**Table 2:** Bearing accuracy (%) at 10k steps with LoRA or full finetuning. Full finetuning makes Q3D overfit to MolmoSpaces, whereas LoRA makes CP underfit.

| | Houses | Easy | Hard |
|---|---|---|---|
| *CP* | | | |
| LoRA | 52.6 | 23.6 | 11.3 |
| Full | 70.5 | 47.2 | 43.1 |
| *Q3D* | | | |
| LoRA | 46.7 | **67.0** | **56.5** |
| Full | **71.9** | 46.6 | 39.6 |

**Table 3:** Bearing accuracy (%) when the target object is covered by a black square in every tour frame. Correct-tour accuracy is in Table 1. Accuracy barely changes, suggesting the models localize the target's room rather than the object.

| Model | Houses | Easy | Hard |
|---|---|---|---|
| Cambrian-S | 49.9 | 33.6 | 30.2 |
| Cambrian-P | 58.5 | 51.6 | 43.5 |
| CS | 71.8 | 44.1 | 36.6 |
| CP | 71.3 | 44.0 | 38.4 |
| V3R | 71.8 | 48.3 | 42.9 |
| Q3D | 46.3 | 62.5 | 54.1 |

trained for spatial prediction capabilities, the short LoRA run is simply intended to extract already-existent spatial understanding capabilities. We verify our CS and CP reimplementations in App. E.4.

### 5.2 Training and Evaluation Data

Motivated by SIMS-V (Brown et al., 2025a), we use ProcTHOR (Deitke et al., 2022) environments to generate spatial training data. Each sample pairs a short tour of two connected rooms with query viewpoints from robot egocentric cameras, and asks for the binned bearing of an object seen in the tour but not from the query views. We generate 9,120 tour videos across 4,205 houses, with 63,798 query poses and 750,324 bearing questions. Details are in App. B.1.

To avoid overfitting to our ProcTHOR data, we add VSI-590K (Yang et al., 2024) to our training mix, sampling between the two with equal probability. Each ProcTHOR tour comes with six questions about different targets, which the model answers independently from a single pass over the shared tour and query images, increasing supervision density without repeatedly encoding the heavy video context (App. E.5).

We evaluate on 2,000 bearing questions from 93 held-out MolmoSpaces houses, and on 500 and 496 questions generated in the `Unseen-Easy` and `Unseen-Hard` RoboCasa layouts of § 6.1, where the robot is in the kitchen and the target is in the dining room (App. B.4). Since we only train on MolmoSpaces data, RoboCasa is out-of-distribution.

### 5.3 Experiments

We now study which models best predict bearings in RoboCasa, and probe *how* they make their predictions.

Tab. 1 reports bearing accuracy across held-out MolmoSpaces houses and target-domain RoboCasa layouts. All models outperform the tour-free baseline, indicating that they extract meaningful signal from the tours. We find that CP outperforms both CS and PH, indicating that dense camera pose supervision is helpful for developing spatial understanding and that other continuous objectives, such as bearing prediction, do not lead to a similar boost. It performs similarly to V3R, indicating that both geometric supervision and geometric embedding inputs are potential avenues for improving VLMs.

The most striking result is Q3D, which achieves the lowest score among tour-conditioned models for held-out MolmoSpaces houses and the highest score for our target RoboCasa scenes. This raises questions about whether other models are overfitting or it is underfitting. We therefore compare with a LoRA-finetuned CP model and a fully finetuned Q3D model in Tab. 2. Fully finetuned Q3D

**Table 4:** Bearing accuracy (%) with the full multi-room tour (*Full*), a tour of a single room that does not show the target (*1 room*), and a tour from a different scene (*Wrong*). Removing tour content generally hurts accuracy, so the models rely on the full tour.

| | Houses | | | Easy | | | Hard | | |
|---|---|---|---|---|---|---|---|---|---|
| Model | Full | 1 room | Wrong | Full | 1 room | Wrong | Full | 1 room | Wrong |
| Cambrian-S | 51.6 | 36.6 | 25.4 | 33.2 | 29.5 | 29.0 | 27.0 | 28.8 | 27.4 |
| Cambrian-P | 59.9 | 39.6 | 25.0 | 48.6 | 34.3 | 36.2 | 41.3 | 28.1 | 28.2 |
| CS | 70.3 | 41.0 | 24.1 | 45.0 | 21.7 | 27.0 | 39.3 | 22.4 | 24.4 |
| CP | 70.5 | 39.3 | 25.4 | 47.2 | 24.1 | 31.0 | 43.1 | 23.3 | 25.8 |
| V3R | 72.0 | 40.2 | 25.2 | 50.2 | 19.0 | 23.0 | 40.1 | 18.7 | 23.4 |
| Q3D | 46.7 | 26.4 | 19.1 | 67.0 | 31.3 | 24.3 | 56.5 | 22.9 | 23.7 |

does overfit, achieving stronger performance for the in-distribution houses and worse performance for RoboCasa data. However, LoRA finetuning CP leads to underfitting and even worse RoboCasa performance. This result does *not* eliminate the possibility that a correctly calibrated CP model could avoid overfitting, but does suggest that the LoRA confound does not immediately explain the difference in performance.

Finally, we compare to LoRA-finetuned versions of the author-released checkpoints of Cambrian-S and Cambrian-P. They perform worse on MolmoSpaces than their paired comparisons, as expected given only LoRA finetuning in this domain. More strikingly, they perform similarly to or worse than our variants despite their expansive midtraining and larger model size. Evidently, even state-of-the-art off-the-shelf implicit models lack spatial understanding we can readily tap into.

#### 5.3.1 Probing

While our spatial models outperform non-spatial baselines, it is not clear whether they are learning true multiview spatial understanding or exploiting shortcuts. This concern is justified—previous investigations have found that popular spatial benchmark tasks are prone to non-visual shortcuts (Brown et al., 2025b) and often solvable from a single frame (Zhang et al., 2026b).

In Tab. 4 we probe the models after restricting the tour to the robot's current room (*1 room*) and replacing the tour with that of another home (*Wrong*). Performance drops across the board, indicating that the model does use the content of the full tour. However, we also observe signal that it may learn some prior based on its location in the current room even if it never observes the target object. Notice that for the in-distribution MolmoSpaces houses, evaluating with the *1 room* tour is persistently better than evaluating with the mismatched tour or random chance. This is not the case for RoboCasa, where the one-room tour is often worse than the random tour, so this signal does not transfer.

Having concluded that the models use the entire tour, we probe whether they actually localize the targets or fit other patterns, by occluding the target with a black box in every tour frame (Tab. 3). Surprisingly, prediction accuracy is largely unchanged. Together with the single-room probe, this suggests the models localize the *room* containing the target rather than the target itself, likely using cues such as the unoccluded table it rests on. This suffices for navigating to the dining table in our downstream task, but localizing individual objects remains an open problem.

## 6 From Spatial Understanding to Mobile Manipulation

After establishing that the explicit 3D-based recipe achieves the strongest spatial understanding, we study whether this leads to stronger downstream mobile manipulation.

### 6.1 Evaluation Protocol

As no existing simulated benchmark evaluates VLA mobile manipulation policies in unseen multi-room layouts (§ 2.1), we contribute one building on RoboCasa365 (Nasiriany et al., 2026).

We select ten navigation-heavy RoboCasa365 tasks and execute them in unseen layouts where task-relevant objects are in a new room. We evaluate policies in three settings (examples in Fig. 5): `Seen`, the training kitchens, with all objects in the kitchen; `Unseen-Easy`, new layouts with the dining

**Table 5:** Placement rate (%) over 200 episodes (10 tasks × 20) per setting. The flat policy achieves strong `Seen` performance but it degrades quickly. Policies trained with implicit and explicit geometric conditioning and supervision maintain their performance best in the unseen settings.

| Method | `Seen` | `Unseen-Easy` | `Unseen-Hard` |
|---|---|---|---|
| Flat $\pi_{0.5}$ | **41.6** | 20.4 | 10.9 |
| Hier. $\pi_{0.5}$, no nav hint | 38.9 | 21.3 | 13.0 |
| *Hier. $\pi_{0.5}$ + nav hint from VLM planner* | | | |
| Stock Qwen3-VL-4B | 30.8 | 23.6 | 13.6 |
| CS | 33.9 | 18.9 | 13.0 |
| CP | 38.1 | 26.7 | 17.8 |
| V3R | 36.7 | 26.9 | 18.6 |
| Q3D, est. geometry | 37.0 | **27.6** | **24.8** |
| *Privileged: ground-truth tour geometry* | | | |
| Q3D, GT geometry | 36.3 | 34.4 | 32.5 |

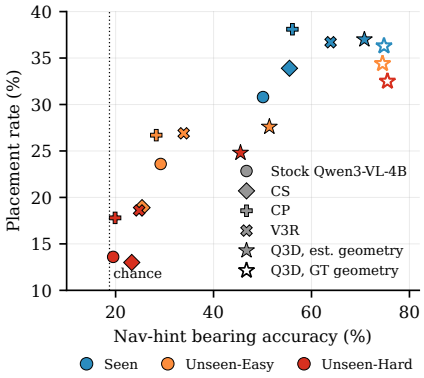



**Figure 4:** Navigation-hint bearing accuracy during rollouts (within one of 16 angular bins, over all VLM navigation queries) against placement rate from Table 5. Placement rate closely tracks hint accuracy (Pearson $r = 0.89$).

table moved to a separate room; and `Unseen-Hard`, which also adds a room holding a decoy table. The policy receives a tour video at $t = 0$ of the kitchen and dining room. Many of these tasks are long-horizon, requiring several successful $(\text{pick}, \text{navigate}, \text{place})$ sequences to achieve full task success and resulting in low and noisy success rates. To address this, we report the *placement rate*, the fraction of a task's placements completed, averaged over episodes (App. C.5). For more details refer to App. C.

### 6.2 Training Details

We train on the RoboCasa365 pretraining dataset (Nasiriany et al., 2026), which includes 100 demonstrations each across 300 tasks. All $\pi_{0.5}$ policies are trained on a mixture that samples 80% of action windows from these demonstrations and 20% from object-directed navigation demonstrations in MolmoSpaces (Kim et al., 2026). App. B.5 describes how the flat and hierarchical executors are derived from a shared midtraining run.

Every planner is a Qwen3-VL-4B model initialized from the corresponding checkpoint of § 5 and fine-tuned to generate the next subtask as text, on a mixture of 80% RoboCasa demonstrations with their tours and 20% MolmoSpaces bearing and distance questions rewritten as navigation subtasks. The target includes a binned polar hint (distance and bearing to the goal) for navigation subtasks. All planners share the same executor checkpoint during inference.

### 6.3 Experiments

We compare the following methods:

- **Flat** $\pi_{0.5}$, a baseline flat VLA.
- **Hier. $\pi_{0.5}$, no nav hint** receives no spatial context and predicts no navigation hint.
- **Hier. $\pi_{0.5}$ + nav hint**, our tour-conditioned planner–executor policy (§ 4). The planner predicts each subtask, including a polar navigation hint for navigation subtasks, and all variants share the same executor. We vary only the planner:
  - **Stock Qwen3-VL-4B**, fine-tuned for subtask prediction without spatial midtraining.
  - **CS**, **CP** and **V3R**, the midtrained models of § 5, which see the tour as an RGB video.
  - **Q3D**, which lifts the tour into a 3D point cloud using its depth and camera poses. We estimate these from RGB (*est. geometry*, App. D.2) and also report a privileged variant that receives the simulator's ground-truth values (*GT geometry*).

Tab. 5 reports placement rates. Whereas the flat policy maintains only 26% of its `Seen` performance in `Unseen-Hard`, the spatially aware policies are much more robust, with RGB-only Q3D retaining 67%. Neither a non-tour-conditioned hierarchical policy nor a tour-conditioned hierarchical policy lacking spatial midtraining leads to meaningful improvement over the flat policy. We find that geometric supervision and inputs are helpful, with both CP and V3R outperforming CS. Finally,

Q3D maintains $90\%$ of its `Seen` performance on `Unseen-Hard` with ground-truth geometry, but only $67\%$ when the geometry comes from our SLAM pipeline built on a state-of-the-art geometry foundation model (App. D.2), a gap of 7.7 points. This suggests that for explicit-3D planners, reconstructing accurate geometry from RGB tours, rather than using it, is now the main bottleneck.

To understand where these differences come from, Fig. 4 plots each planner's navigation-hint accuracy during rollouts against its placement rate. Placement rate closely tracks hint accuracy (Pearson $r = 0.89$ over all 18 planner–setting pairs), indicating that the quality of the planner's spatial estimates drives the differences in Tab. 5. The planners that see the tour only as RGB video lose most of their hint accuracy as the setting gets harder: in `Unseen-Hard` their accuracy is 19–$25\%$, close to the $18.8\%$ expected by chance, and their placement rates (13–$19\%$) are correspondingly close to that of the hierarchical policy without hints ($13.0\%$). Inaccurate hints can also hurt: stock Qwen3-VL-4B predicts the correct bearing only half the time even in `Seen`, and places fewer objects there than the no-hint policy ($30.8\%$ vs. $38.9\%$). In contrast, Q3D with ground-truth geometry keeps its hint accuracy near $75\%$ in all three settings, so its placement rate changes little between them. With estimated geometry, Q3D remains well above chance in the unseen settings ($51\%$ and $46\%$), but its hint accuracy and placement rate both fall short of ground truth by similar margins (App. E.3).

## 7 Discussion and Conclusion

Our results point to a gap between the capabilities that current mobile manipulation policies acquire from large-scale robot data and those required to operate in unfamiliar environments. VLAs can learn strong local navigation and manipulation behaviors, yet these skills do not by themselves provide persistent knowledge of where task-relevant locations lie across a multi-room scene. This distinction becomes apparent under layout shift: policies that are strong in familiar environments revert to reactive exploration when familiar spatial relationships change, while providing the same low-level policy with ground-truth spatial guidance largely avoids this failure. These results suggest that scaling mobile manipulation data alone may not be sufficient for generalization to new environments; policies also need spatial representations that remain useful when the layout changes.

Our spatial VLM study further shows that strong spatial prediction within the training distribution does not necessarily imply strong transfer to new robot environments. Several models achieve similar or better performance on held-out environments from the spatial midtraining distribution, yet their relative performance changes substantially when evaluated in RoboCasa. In our experiments, representations that explicitly encode 3D structure transfer most robustly, and these differences carry over to downstream control. In our hardest unseen setting, our policy more than doubles the placement rate of a flat $\pi_{0.5}$ policy trained on the same robot data ($24.8\%$ vs. $10.9\%$). This suggests that spatial benchmarks should be evaluated not only by how well models fit the distribution on which spatial capabilities are trained, but also by whether those capabilities remain useful when transferred to new environments and downstream behavior.

At the same time, our probing experiments show that current spatial models remain imperfect. Although the models make use of information from the multi-room tour, masking the target object has surprisingly little effect on bearing prediction, suggesting that they often localize the relevant room or surrounding landmarks rather than the target itself. This is sufficient for many of the navigation decisions in our benchmark, but more general household manipulation will require precise object-level spatial reasoning. Our evaluation is also limited to controlled simulated environments and assumes access to a single tour before task execution. Real homes are dynamic, and future systems will likely need spatial representations that can be updated online as objects move and the environment changes. Finally, the gap between estimated and ground-truth geometry leaves substantial room for improving how spatial structure is recovered directly from video. More broadly, our results suggest that progress toward generalist mobile manipulation will require treating spatial generalization as a first-class problem rather than expecting it to emerge automatically from larger robot datasets or stronger manipulation policies. A robot that can manipulate almost anything it sees is only useful across a home if it can also reason about what it cannot currently see. As VLAs become increasingly capable at deciding how to act, robust deployment in new environments will increasingly depend on whether they also know where to go.

## AI USE STATEMENT

In this work, we used generative AI tools to help propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, clean and reformat datasets, support qualitative and thematic data analysis, and help interpret results. We have not used generative AI tools for translation. Developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, and assisting in the writing of proofs are not applicable to this work. Additionally, we used generative AI tools to create or modify scientific figures or images, suggest experimental parameters, create or edit software code, create artifacts, draft parts of a research paper, summarize or analyze existing literature, discover research topics or identify gaps, brainstorm, source and search for information, edit a research paper to improve readability, and identify relevant literature. We have reviewed all AI-assisted work. AI was helpful for some writing brainstorming, but the majority of main paper content is human-written. We manually reviewed every word in the main paper and appendix. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REPRODUCIBILITY STATEMENT

We will open source evaluation code and data. We provide extensive details in the appendix to help with reproducibility.

## REFERENCES

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Ellis Brown, Arijit Ray, Ranjay Krishna, Ross Girshick, Rob Fergus, and Saining Xie. Sims-v: Simulated instruction-tuning for spatial video understanding, 2025a. URL https://arxiv.org/abs/2511.04668.

Ellis Brown, Jihan Yang, Shusheng Yang, Rob Fergus, and Saining Xie. Benchmark designers should "train on the test set" to expose exploitable non-visual shortcuts. *arXiv preprint arXiv:2511.04655*, 2025b.

An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Zaitian Gongye, Xueyan Zou, Jan Kautz, Erdem Bıyık, Hongxu Yin, Sifei Liu, and Xiaolong Wang. Navila: Legged robot vision-language-action model for navigation. In *RSS*, 2025.

Hao-Tien Lewis Chiang, Zhuo Xu, Zipeng Fu, Mithun George Jacob, Tingnan Zhang, Tsang-Wei Edward Lee, Wenhao Yu, Connor Schenck, David Rendleman, Dhruv Shah, et al. Mobility vla: Multimodal instruction navigation with long-context vlms and topological graphs. In *CoRL*, 2024.

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-scale embodied AI using procedural generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Zhiwen Fan, Jian Zhang, Renjie Li, Junge Zhang, Runjin Chen, Hezhen Hu, Kevin Wang, Huaizhi Qu, Shijie Zhou, Dilin Wang, Zhicheng Yan, Hongyu Xu, Justin Theiss, Tianlong Chen, Jiachen Li, Zhengzhong Tu, Zhangyang Wang, and Rakesh Ranjan. Vlm-3r: Vision-language models augmented with instruction-aligned 3d reconstruction, 2026. URL https://arxiv.org/abs/2505.20279.

Max Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Fei-Fei Li, Guanya Shi, Jiajun Wu, Shankar Sastry, Yuke Zhu, Ken Goldberg, and Jim Fan. CaP-X: A framework for benchmarking and improving coding agents for robot manipulation. *arXiv preprint arXiv:2603.22435*, 2025. URL https://arxiv.org/abs/2603.22435.

Ning Gao, Jinliang Zheng, Xing Gao, Haoxiang Ma, Hanqing Wang, Yukai Wang, Jiantong Chen, Zanxin Chen, Shujie Zhang, Mingda Jia, et al. Ebench: Elemental diagnosis of generalist mobile manipulation policies. *arXiv preprint arXiv:2606.18239*, 2026.

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, 2023. URL https://arxiv.org/abs/2307.12981.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. In *ICRA*, 2023.

Xiaohu Huang, Jingjing Wu, Qunyi Xie, and Kai Han. 3drs: Mllms need 3d-aware representation supervision for scene understanding, 2025. URL https://arxiv.org/abs/2506.01946.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, Vedant Choudhary, Foster Collins, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Maitrayee Dhaka, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachlan Groom, Haroun Habeeb, Hunter Hancock, Karol Hausman, Gashon Hussein, Victor Hwang, Brian Ichter, Connor Jacobsen, Szymon Jakubczak, Rowan Jen, Tim Jones, Gregg Kammerer, Ben Katz, Liyiming Ke, Mairbek Khadikov, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Brendon LeCount, Sergey Levine, Xinyu Li, Adrian Li-Bell, Vladislav Lialin, Zhonglin Liang, Wallace Lim, Yao Lu, Enyu Luo, Vishnu Mano, Nandan Marwaha, Aikys Mongush, Liam Murphy, Suraj Nair, Tyler Patterson, Karl Pertsch, Allen Z. Ren, Gavin Schelske, Charvi Sharma, Baifeng Shi, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Jiaming Tang, Jimmy Tanner, Shalom Tekeste, Marcel Torne, Kyle Vedder, Quan Vuong, Anna Walling, Haohuan Wang, Jason Wang, XuDong Wang, Chris Whalen, Samuel Whitmore, Blake Williams, Charles Xu, Sukwon Yoo, Lili Yu, Wuming Zhang, Zhuoyang Zhang, and Ury Zhilinsky. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities, 2026. URL https://arxiv.org/abs/2604.15483.

Mengdi Jia, Zekun Qi, Shaochen Zhang, Wenyao Zhang, Xinqiang Yu, Jiawei He, He Wang, and Li Yi. Omnispatial: Towards comprehensive spatial reasoning benchmark for vision language models, 2026. URL https://arxiv.org/abs/2506.03135.

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.

Yejin Kim, Wilbert Pumacay, Omar Rayyan, Max Argus, Winson Han, Eli VanderBilt, Jordi Salvador, Abhay Deshpande, Rose Hendrix, Snehal Jauhri, Shuo Liu, Nur Muhammad Mahi Shafiullah, Maya Guru, Arjun Guru, Ainaz Eftekhar, Karen Farley, Donovan Clay, Jiafei Duan, Piper Wolters, Alvaro Herrasti, Ying-Chun Lee, Georgia Chalvatzaki, Yuchen Cui, Ali Farhadi, Dieter Fox, and Ranjay Krishna. Molmospaces: A large-scale open ecosystem for robot navigation and manipulation, 2026.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *ECCV*, 2020.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez, Hang Yin, Michael Lingelbach, Minjune Hwang, Ayano Hiranaka, Sujay Garlanka, Arman Aydin, Sharon Lee, Jiankai Sun, Mona Anvari, Manasi Sharma, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Yunzhu Li, Silvio Savarese, Hyowon Gweon, C. Karen Liu, Jiajun Wu, and Li Fei-Fei. Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation, 2024. URL https://arxiv.org/abs/2403.09227.

Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Raymond Yu, Caelan Reed Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, and Ankit Goyal. Hamster: Hierarchical action models for open-world robot manipulation. In *ICLR*, 2025.

Huizhi Liang, Yichao Shen, Yu Deng, Sicheng Xu, Zhiyuan Feng, Tong Zhang, Yaobo Liang, and Jiaolong Yang. Hispatial: Taming hierarchical 3d spatial understanding in vision-language models, 2026. URL https://arxiv.org/abs/2603.25411.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *arXiv preprint arXiv:2209.07753*, 2022.

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, 2025. URL https://arxiv.org/abs/2511.10647.

Lucy Lin, Ayush Jain, Yifan Liu, and Katerina Fragkiadaki. Qwen-3d: A generalist 3d vision-language model for spatial understanding. In *European Conference on Computer Vision (ECCV)*, 2026.

Julius Mayer, Mohamad Ballout, Serwan Jassim, Farbod Nosrat Nezami, and Elia Bruni. ivispar – an interactive visual-spatial reasoning benchmark for vlms, 2025. URL https://arxiv.org/abs/2502.03214.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations (ICLR)*, 2026.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025. URL https://arxiv.org/abs/2503.14734.

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. URL https://arxiv.org/abs/2603.18002.

Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. Sayplan: Grounding large language models using 3D scene graphs for scalable robot task planning. In *CoRL*, 2023.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi robot: Open-ended instruction following with hierarchical vision-language-action models, 2025. URL https://arxiv.org/abs/2502.19417.

Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. In *NeurIPS*, 2021.

Gemini Robotics Team, Abbas Abdolmaleki, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Ashwin Balakrishna, Nathan Batchelor, Alex Bewley, Jeff Bingham, Michael Bloesch, Konstantinos Bousmalis, Philemon Brakel, Anthony Brohan, Thomas Buschmann, Arunkumar Byravan, Serkan Cabi, Ken Caluwaerts, Federico Casarini, Christine Chan, Oscar Chang, London Chappellet-Volpini, Jose Enrique Chen, Xi Chen, Hao-Tien Lewis Chiang, Krzysztof Choromanski, Adrian Collister, David B. D'Ambrosio, Sudeep Dasari, Todor Davchev, Meet Kirankumar Dave, Coline Devin, Norman Di Palo, Tianli Ding, Carl Doersch, Adil Dostmohamed, Yilun Du, Debidatta Dwibedi, Sathish Thoppay Egambaram, Michael Elabd,

Tom Erez, Xiaolin Fang, Claudio Fantacci, Cody Fong, Erik Frey, Chuyuan Fu, Ruiqi Gao, Marissa Giustina, Keerthana Gopalakrishnan, Laura Graesser, Oliver Groth, Agrim Gupta, Roland Hafner, Steven Hansen, Leonard Hasenclever, Sam Haves, Nicolas Heess, Brandon Hernaez, Alex Hofer, Jasmine Hsu, Lu Huang, Sandy H. Huang, Atil Iscen, Mithun George Jacob, Deepali Jain, Sally Jesmonth, Abhishek Jindal, Ryan Julian, Dmitry Kalashnikov, M. Emre Karagozler, Stefani Karp, Matija Kecman, J. Chase Kew, Donnie Kim, Frank Kim, Junkyung Kim, Thomas Kipf, Sean Kirmani, Ksenia Konyushkova, Li Yang Ku, Yuheng Kuang, Thomas Lampe, Antoine Laurens, Tuan Anh Le, Isabel Leal, Alex X. Lee, Tsang-Wei Edward Lee, Guy Lever, Jacky Liang, Li-Heng Lin, Fangchen Liu, Shangbang Long, Caden Lu, Sharath Maddineni, Anirudha Majumdar, Kevis-Kokitsi Maninis, Andrew Marmon, Sergio Martinez, Assaf Hurwitz Michaely, Niko Milonopoulos, Joss Moore, Robert Moreno, Michael Neunert, Francesco Nori, Joy Ortiz, Kenneth Oslund, Carolina Parada, Emilio Parisotto, Amaris Paryag, Acorn Pooley, Thomas Power, Alessio Quaglino, Haroon Qureshi, Rajkumar Vasudeva Raju, Helen Ran, Dushyant Rao, Kanishka Rao, Isaac Reid, David Rendleman, Krista Reymann, Miguel Rivas, Francesco Romano, Yulia Rubanova, Peter Pastor Sampedro, Pannag R Sanketi, Dhruv Shah, Mohit Sharma, Kathryn Shea, Mohit Shridhar, Charles Shu, Vikas Sindhwani, Sumeet Singh, Radu Soricut, Rachel Sterneck, Ian Storz, Razvan Surdulescu, Jie Tan, Jonathan Tompson, Saran Tunyasuvunakool, Jake Varley, Grace Vesom, Giulia Vezzani, Maria Bauza Villalonga, Oriol Vinyals, René Wagner, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Chengda Wu, Markus Wulfmeier, Fei Xia, Ted Xiao, Annie Xie, Jinyu Xie, Peng Xu, Sichun Xu, Ying Xu, Zhuo Xu, Jimmy Yan, Sherry Yang, Skye Yang, Yuxiang Yang, Hiu Hong Yu, Wenhao Yu, Wentao Yuan, Yuan Yuan, Jingwei Zhang, Tingnan Zhang, Zhiyuan Zhang, Allan Zhou, Guangyao Zhou, and Yuxiang Zhou. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer, 2025. URL https://arxiv.org/abs/2510.03342.

Generalist Team. Gen-1.5: Embodied foundation models are one-shot learners. *Generalist AI Blog*, 2026. https://generalistai.com/blog/gen-1.5.

Xiaomi Robotics Team, Jun Guo, Piaopiao Jin, Jason Li, Peiyan Li, Yingyan Li, Futeng Liu, Wanli Peng, Optimus Qin, Yifei Su, et al. Xiaomi-robotics-1: Scaling vision-language-action models with over 100k hours of real-world trajectories. *arXiv preprint arXiv:2607.15330*, 2026.

Haochen Wang, Yucheng Zhao, Tiancai Wang, Haoqiang Fan, Xiangyu Zhang, and Zhaoxiang Zhang. Ross3d: Reconstructive visual instruction tuning with 3d-awareness, 2025a. URL https://arxiv.org/abs/2504.01901.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *CVPR*, 2025b.

Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A. Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. In *CVPR*, 2025c.

Jihan Yang, Shusheng Yang, Anjali W. Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in Space: How Multimodal Large Language Models See, Remember and Recall Spaces. *arXiv preprint arXiv:2412.14171*, 2024.

Jihan Yang, Zifan Zhao, Xichen Pan, Shusheng Yang, Junyi Zhang, Bingyi Kang, Hu Xu, Shang-Wen Li, and Saining Xie. Cambrian-p: Pose-grounded video understanding, 2026. URL https://arxiv.org/abs/2605.22819.

Shusheng Yang, Jihan Yang, Pinzhi Huang, Ellis Brown, Zihao Yang, Yue Yu, Shengbang Tong, Zihan Zheng, Yifan Xu, Muhan Wang, Daohan Lu, Rob Fergus, Yann LeCun, Li Fei-Fei, and Saining Xie. Cambrian-s: Towards spatial supersensing in video, 2025. URL https://arxiv.org/abs/2511.04670.

Sriram Yenamandra, Arun Ramachandran, Karmesh Yadav, Austin Wang, Mukul Khanna, Theophile Gervet, Tsung-Yen Yang, Vidhi Jain, Alexander William Clegg, John Turner, et al. Homerobot: Open-vocabulary mobile manipulation. In *CoRL*, 2023.

Jian Zhang, Shijie Zhou, Bangya Liu, Achuta Kadambi, and Zhiwen Fan. Spatialstack: Layered geometry-language fusion for 3d vlm spatial reasoning, 2026a. URL https://arxiv.org/abs/2603.27437.

Qifeng Zhang, Kaixiang Huang, Heng Dong, Huang Fang, Junting Chen, Junjie Zhu, Yonghang Chen, Zhiyu Zhang, and Wei Li. Gst-bench: Can vlms develop global spatial awareness from video? *arXiv preprint arXiv:2608.05747*, 2026b. URL https://arxiv.org/abs/2608.05747.

Chenming Zhu, Tai Wang, Wenwei Zhang, Jiangmiao Pang, and Xihui Liu. Llava-3d: A simple yet effective pathway to empowering lmms with 3d-awareness, 2025. URL https://arxiv.org/abs/2409.18125.

# A Architecture and Implementation Details

We implement the CS, CP, V3R and Q3D mechanisms on Qwen3-VL-4B-Instruct and train them on our bearing task. All keep Qwen3-VL's vision encoder, merger and DeepStack. Each of the 16 tour views is repeated once so that Qwen3-VL's two-frame temporal patches give one group per view. The six questions of an example share one encoded visual prefix, and each answer attends only to the prefix and itself. The decoder is causal except in Q3D. We score the decoded answer text; the polar and pose heads are used only in training.

**Polar head.** A two-layer MLP reads the last prompt token's hidden states from layers 12, 24 and 36 and regresses radius (smooth L1) and bearing direction (cosine loss), with weight 0.1.

**CS.** The model is trained with next-token prediction only, with no geometric inputs or auxiliary losses (Yang et al., 2025).

**CP.** Following Cambrian-P (Yang et al., 2026), we insert one learned camera token after each view. A randomly initialized VGGT-style head (Wang et al., 2025b) reads the final-layer hidden states at these tokens and predicts each view's translation, quaternion and field of view. The head has width 512, 8 attention heads, 4 transformer blocks and 4 refinement iterations, and its output branch is $D \to D/2 \to D/2 \to 9$. Each refinement detaches the previous estimate where it conditions the next update but not in the residual. Pose targets are the 16 tour views and both query views, relative to one of the query cameras, and come from MolmoSpaces only. The camera loss is L1 with weight 0.2. The VSI-Bench CP model (App. E.4) also attends bidirectionally among visual tokens.

**V3R.** Following VLM-3R (Fan et al., 2026), a frozen CUT3R (Wang et al., 2025c) processes each view at $432 \times 432$ and supplies camera and patch tokens. After Qwen3-VL's merger, a new cross-attention module (8 heads, width 512) lets each view's visual tokens attend to that view's CUT3R tokens and adds the result with a 0.1-scaled residual. The language model is fully finetuned.

**Q3D.** Following Qwen3D (Lin et al., 2026), the visual tokens of all posed views are unprojected with depth into the robot base frame, mean-pooled into 5 cm voxels (at most 4,096) and encoded with metric 3D RoPE. Depth and poses come from the simulator, so Q3D alone receives ground-truth geometry in Table 1. We add a learned robot token at the origin. Scene tokens attend bidirectionally to each other but not to the question, so the six questions share one scene encoding. The LoRA adapters (rank 8) cover the attention, MLP and output layers.

**Released Cambrian-S/P.** We keep the released 7B models' native preprocessing and resolution and freeze everything except rank-16 decoder LoRA, including Cambrian-P's camera head, which receives no pose loss. Each example uses 16 images: the two query views and 14 tour frames.

# B Training Details

## B.1 Spatial Training Data

This appendix describes the MolmoSpaces data used to train bearing prediction in § 5. Each sample pairs a short exploration video (a *tour*) of two connected rooms with the robot's current camera views (the *query*), and asks for the direction to an object that was seen in the tour but is not visible from the query. Labels are computed from simulator ground truth.

**Relation to GST-Bench.** Our pipeline closely follows the automatic generation pipeline of GST-Bench (Zhang et al., 2026b). As in GST-Bench, each question pairs an egocentric exploration video with a query view sampled away from the video's camera path, targets are never visible from the query view so answering requires integrating information across the tour, and labels are exact angles computed from simulator geometry. We make four changes. We generate data in MolmoSpaces instead of OmniGibson with BEHAVIOR-1K. Each tour covers two rooms connected by a doorway. The query is rendered from the robot's own shoulder cameras, with bearings measured relative to the robot base. Finally, of the GST-Bench question types we use only the egocentric bearing of a target, rounded to one of 16 bins.

**Scenes and tours.** We use procedurally generated ProcTHOR houses (Deitke et al., 2022) in MolmoSpaces (Kim et al., 2026). For each house we select pairs of adjacent rooms connected by a doorway and render one tour per pair. A tour is a camera path that starts deep in one room, passes

through the doorway and ends in the other. Its viewpoints are chosen greedily so that together they see most of the objects and walls in both rooms, and each tour has at most 48 frames.

**Queries and targets.** For each tour we place the Mobile Franka robot at several random collision-free poses inside the two toured rooms and render its two shoulder cameras. The targets of a query are up to six objects in the toured rooms that appear in the tour but in neither query image, so the model can only locate them using the tour. Targets may be in the robot's room or in the adjacent room.

**Labels.** The bearing of a target is its direction in the horizontal plane relative to the robot base's forward axis, with $0^\circ$ straight ahead and positive angles to the left. We round it to the nearest of 16 directions spaced $22.5^\circ$ apart, and the question refers to the target by its object category. A prediction counts as correct if it is within one bin of the label, so chance is $3/16$.

**Training inputs.** The model receives 16 tour frames and both query images at $256 \times 256$. We hold out 2% of tours, with all their queries, for validation.

### B.2 Training Mixture

All Qwen3-VL-4B recipes in § 5 use the same mixture of MolmoSpaces and VSI-590K (Yang et al., 2025). The original Cambrian-S and Cambrian-P models are LoRA-finetuned on MolmoSpaces only.

**Sampling.** Each training example is drawn from MolmoSpaces or VSI-590K with equal probability. A MolmoSpaces example is one tour and query with six questions about different targets. The six questions share the visual input but are answered independently, so no answer can see another. The MolmoSpaces data has 9,120 tours in 4,205 houses from ProcTHOR's training split, with 63,798 query poses and 750,324 questions; after holding out 2% of tours and dropping queries with fewer than six targets, 119,028 examples (714,168 answers) remain for training. A VSI-590K example is a single question; after holding out 2%, 576,857 remain. One pass over the mixture therefore visits each MolmoSpaces example about 2.9 times and about 60% of the VSI-590K examples.

**Questions and answers.** MolmoSpaces questions ask for both the distance and the bearing of the target, and refer to it either by its category or by a red box drawn around it in the tour. The answer is the text `dI, aJ`, where $I = \lfloor r/0.5 + 0.5 \rfloor$ for distance $r$ in metres, with no upper limit, and $J$ is the nearest of the 16 bearing bins. VSI-590K questions and answers are used unchanged.

**Visual inputs.** For MolmoSpaces we sample 16 distinct tour frames with a random start and stride and add the robot's two shoulder images. For VSI-590K we sample 16 frames uniformly from each clip; single images and short clips are kept. All images are resized to $256 \times 256$, and no top-down maps are used.

**Pose supervision.** For recipes with a camera-pose head, MolmoSpaces supplies the poses of the 16 tour frames and the two query views, expressed relative to one of the query cameras, which alternates between the left and right shoulder. The pose loss has weight 0.2. Examples whose camera metadata is incomplete are used for question answering only. VSI-590K provides no pose targets.

**Loss balance.** The language loss is the mean over answer tokens in each batch. Because a MolmoSpaces example contributes six answers and a VSI-590K example one, MolmoSpaces supplies about six of every seven answers and, since its answers are also longer, about 93% of supervised answer tokens. Equal sampling is therefore not equal loss weighting.

**VSI-590K sources.** Tab. 6 lists the question counts per source before the 2% hold-out.

### B.3 Training Settings

All Qwen3-VL-4B recipes start from Qwen3-VL-4B-Instruct, train on the mixture above with a global batch of 128 and use AdamW. Tab. 7 lists the settings that differ between recipes. Unless noted, the learning rate warms up over 100 steps and then follows a cosine schedule over 50k steps; we report the 10k-step checkpoint, at which the learning rate is still about 90% of its peak. The language loss is next-token prediction on the answer text.

**Recipe details.** In the CP recipe, only the attention layers and the pose head train for the first 1k steps, after which the rest of the language model and the vision encoder join. Its pose head predicts, for each of the 16 tour frames and two query views, the camera translation, rotation (as a quaternion)

**Table 6:** VSI-590K questions per source, before holding out 2% for validation.

| Source | Questions |
|---|---|
| Hypersim | 176,774 |
| ScanNet++ v2 | 138,701 |
| ScanNet | 92,145 |
| ADT | 60,207 |
| ARKitScenes | 57,760 |
| YouTube room tours | 20,100 |
| ProcTHOR | 20,092 |
| Robotics | 19,645 |
| S3DIS | 5,187 |
| Total | 590,611 |

**Table 7:** Training settings per recipe. LR: peak learning rates for the language model / vision encoder / new modules. WD: weight decay.

| Recipe | Trained | LR | Extra losses | WD |
|---|---|---|---|---|
| No tour | Full | 5e−6 / 1e−6 / – | – | 0 |
| Polar head | Full | 5e−6 / 1e−6 / 1e−4 | Polar regression (0.1) | 0 |
| CS | Full | 5e−6 / 1e−6 / – | – | 0 |
| CP | Full | 5e−6 / 1e−6 / 1e−4 | Camera pose (0.2) | 0 |
| CP, LoRA | LoRA (rank 16) | 5e−6 / frozen / 1e−4 | Camera pose (0.2) | 0 |
| V3R | Full | 5e−6 / 1e−6 / 1e−4 | – | 0 |
| Q3D (LoRA) | LoRA (rank 8) | 1e−4 / frozen / 1e−4 | – | 0.05 |
| Q3D, full | Full | 5e−6 / 1e−6 / 1e−4 | – | 0.05 |

and field of view relative to one of the query cameras, with an L1 loss. The polar head reads decoder features at one third, two thirds and the full depth and regresses the target's distance and bearing; the text answer is still trained alongside it. The V3R recipe keeps CUT3R (Wang et al., 2025c) frozen and trains a cross-attention module that adds its features to Qwen3-VL's visual tokens. Q3D maps each visual token to a metric 3D point using depth and camera poses, pools visual tokens into 5 cm voxels, and replaces the position encoding with 3D RoPE; a learned token marks the robot's position. Its LoRA variant trains rank-8 adapters and the robot token with a 2k-step warmup and a constant learning rate. Its full-finetune variant trains only the attention layers and the robot token for the first 5k steps, then the rest of the model, again at a constant learning rate after warmup. The CP LoRA variant trains rank-16 adapters on the attention and MLP layers and the full pose head.

**No tour.** The no-tour baseline sees only the two query images, trains on MolmoSpaces alone and uses a 10k-step cosine schedule. Its held-out loss was lowest at 1k steps and rose afterwards, so we report the 1k-step checkpoint.

**Original Cambrian models.** We LoRA-finetune the released Cambrian-S and Cambrian-P 7B models on MolmoSpaces alone for 1k steps, training rank-16 adapters in the language model at a learning rate of $2 \times 10^{-4}$ with everything else frozen. Each example uses 16 images in total, the query views followed by tour frames, at each model's native resolution.

### B.4 Evaluation Data

Each question comes with a tour of up to 48 frames and the robot's two shoulder images; models sample 16 tour frames as in training. All evaluation questions refer to the target by category, and we score bearing only. A prediction is correct if its bearing bin is within one bin of the label (chance $3/16$); answers that do not parse are excluded, which affects at most 5% of answers for any model.

**MolmoSpaces houses.** We generate evaluation data with the same pipeline in 100 houses from ProcTHOR's validation split, which share no houses with the training data; 93 of them yield questions. They contain 6,174 bearing questions, 48% of which have the target in the room the robot is not in, and we evaluate a fixed set of 2,000.

**RoboCasa layouts.** We generate questions in the `Unseen-Easy` and `Unseen-Hard` layouts of § 6.1, for the same ten dining tasks. For each episode we render a tour of the kitchen and dining room, place the robot at random poses in the kitchen and ask for the bearing of task objects on the

dining table that are not visible from those poses. The question states that the target is in the adjoining dining room. `Unseen-Easy` has 1,020 bearing questions from 125 episodes across layouts 61–70, and we evaluate a fixed subset of 500. `Unseen-Hard` has 985 questions from 124 episodes across layouts 82, 84, 86, 88 and 90, and we evaluate a fixed subset of 496.

### B.5 Mobile Manipulation Training

**Training procedure.** All $\pi_{0.5}$ policies come from a single midtraining run of 100k steps. Its examples are conditioned on the task and the current subtask, and navigation examples sample between having a binned polar hint and having none (see below). The executor for the hierarchical policies with a nav hint needs no further training; we use the 85k-step checkpoint of this run. The other two policies continue from the 100k-step checkpoint for another 30k steps on the same mixture. The flat $\pi_{0.5}$ is trained without subtask conditioning, on the task alone. The executor of the no-hint hierarchical baseline is trained on subtasks without nav hints.

**Executor data.** The mixture weights are defined over sampled action windows, not over trajectories or hours of data.

- **RoboCasa (80%).** Full human demonstrations from the RoboCasa365 pretraining tasks, including both their navigation and manipulation segments. Four tasks are excluded (TurnOffSimmered-SauceHeat, PreheatPot, RinseCuttingBoard and ScalePortioning), as are trajectories flagged by an automatic quality filter. Within RoboCasa, a task $t$ is sampled with probability proportional to $N_t^{0.4}$, where $N_t$ is its number of valid action windows; an episode is then sampled uniformly, and a window uniformly within it. This reduces the dominance of large tasks without making all tasks equally likely.
- **MolmoSpaces (20%).** Object-directed navigation demonstrations from two collections, one of which biases initial placements toward targets that are occluded from the start. The two collections are pooled into a single navigation dataset. They are not restricted to cross-room navigation.

A loader scan of the same data sources found about 31k RoboCasa episodes across 296 task folders (27.1M valid windows) and about 48k MolmoSpaces episodes (8.4M valid windows).

**Executor inputs and targets.** Each training example is a chunk of 20 actions at 20 Hz. Windows overlap, with anchors 50 ms apart. RoboCasa examples use two external cameras and a wrist camera. MolmoSpaces provides two shoulder cameras, and its missing wrist camera is replaced by a masked placeholder. RoboCasa language is `Task: {task}. Subtask: {subtask}`, using the demonstration's subtask annotation; MolmoSpaces language is the navigation command. On navigation windows from both sources, half of the examples (chosen deterministically by sample index) append a polar hint binned in the same way as the planner's output, and the rest have no hint. Non-navigation windows keep ordinary task and subtask conditioning. This describes midtraining; the flat and no-hint fine-tunes change only the conditioning, as described above.

**Planner fine-tuning.** Each tour-conditioned planner starts from its spatial checkpoint of § 5: 10k steps, except 8k for CS, or Qwen3-VL-4B-Instruct for the stock planner. It is then fully fine-tuned to predict the next subtask. Training samples 80% of examples from RoboCasa human demonstrations, paired with a tour of the scene, and 20% from MolmoSpaces, where the direction and distance questions about a target are merged into one navigation subtask. The input is the tour, the current left-shoulder, right-shoulder and wrist images, and the prompt `Predict the next subtask. Task: {instruction}.` The target is the annotated subtask to complete from the current observation, as text. For navigation subtasks it includes the goal's distance, binned to one of $\{0.5, 1, 1.5, 2, 3, 4, 6\}$ m, and its bearing in $22.5°$ steps, e.g. `navigate to the plate (3.0 m, 0.39 rad)`. The tour is 16 distinct $256 \times 256$ views in chronological order. The loss is next-token cross-entropy on the answer tokens only; there are no pose, polar-regression or distillation losses, and any spatial heads kept from the upstream checkpoint are frozen and unused.

**Planner optimization.** All planners train for 25k updates with a global batch of 16 (400k examples), peak learning rate $10^{-5}$, no weight decay and gradient clipping at norm 1.0. The learning rate warms up over 100 updates and then follows a cosine schedule with a 100k-update horizon, so it is still about $8.5 \times 10^{-6}$ at 25k. Optimizer, scheduler and data order start fresh. The Q3D planner receives the same 16 tour views as posed RGB-D images, which are lifted to 3D points in the robot base frame (5 cm voxel pooling, at most 4,096 points). All of its weights train, including its adapters, vision

| Task | Example instruction | Placements |
|---|---|---|
| AlignSilverware | Take the fork and place it on the left side of the plate. Then take the spoon and place it on the right side of the plate. | 2 |
| PackFoodByTemp | Place the cold items from the fridge in one tupperware on the dining counter. Place the warm items in the stove area in another tupperware. | 4 |
| MatchCupAndDrink | There is a wine glass and glass cup on the dining counter. Place the wine bottle next to the wine glass and the juice bottle next to the glass cup. | 2 |
| PrepareStoringLeftovers | Retrieve the tupperware and aluminum foil from the cabinet and place them on the dining counter near the plate. | 2 |
| PrepareDrinkStation | There is a tray on the dining counter. Grab the cup, mug, and pitcher and place them on the tray. | 3 |
| SetupWineGlasses | Retrieve the wine glasses from the counter and place one next to each plate. | 2 |
| DeliverBrewedCoffee | Pick up the brewed coffee from the coffee machine and place it next to the breakfast plate on the dining counter. | 1 |
| HotDogSetup | Move the hot dog bun to the plate on the dining table and place the condiment bottle next to that plate. Then take the sausage from the fridge and place it on the plate. | 3 |
| DistributeChicken | Take the chicken drumsticks from the pan and place one on each plate on the dining counter. | 2 |
| ServeSteak | Pick up the pan with the steak in it and place it on the dining table. Then place the steak on the plate. | 2 |

**Table 8:** Benchmark tasks with example (shortened) instructions. *Placements* is the number of independent placement goals $P_\tau$ used for the placement rate (App. C.5).

encoder and base anchor. The V3R planner keeps its CUT3R encoder frozen and trains its vision encoder, decoder and fusion layers.

**No-tour planner.** The no-hint hierarchical baseline (§ 6.3) and the oracle policy (§ 3.2) use a planner without tour conditioning. It is Qwen3-VL-4B-Instruct fine-tuned on RoboCasa alone to predict the next subtask, without navigation hints, from the current images and the task. We use its 25k-update checkpoint, trained with the optimization settings above.

# C Benchmark Details

This appendix describes the benchmark from § 6.1: tasks (App. C.1), layouts (App. C.2), tours (App. C.3), episodes (App. C.4) and metrics (App. C.5).

## C.1 Tasks

We use the ten RoboCasa365 composite tasks in Tab. 8. Each has a kitchen phase, in which the robot manipulates objects at a kitchen fixture, followed by a delivery to the dining table, where a plate, tray or other receptacle is placed at reset. Tasks, objects and success criteria are unchanged from RoboCasa365; only the position of the dining table changes between settings.

## C.2 Layouts

`Seen` uses the RoboCasa training kitchens that contain a dining counter, usually an island or peninsula inside the kitchen. For each of `Unseen-Easy` and `Unseen-Hard` we generate 10 new layouts by editing training kitchens (Fig. 5). In `Unseen-Easy`, the dining table is moved into a new dining room connected to the kitchen by a doorway, and any kitchen island is removed so the dining-room table is the only dining counter. The kitchen fixtures are therefore familiar and the novelty is the floor plan. `Unseen-Hard` adds to each `Unseen-Easy` layout a third room, reached through its own doorway from the kitchen and holding a bare decoy table that copies the dining table. The decoy has normal visual and collision geometry but is excluded from RoboCasa's task fixtures, so it is never used for object placement or as a task goal. We checked every task reset in these layouts for connected, reachable floor in all rooms and for no task objects in the decoy room. All settings use RoboCasa's training object and visual-style splits.

### C.3 TOURS

Each tour is rendered from the episode's frozen start state with the robot hidden; no physics is stepped and nothing in the scene moves. A room-balanced coverage planner spreads candidate camera positions over each room's navigable floor. It greedily selects position–heading pairs that cover weighted object, fixture and boundary targets in each room, adds a required view that frames the dining table, and connects the views with navigable paths under a cap on per-frame heading change. Tours have 48 frames at 8 fps, split evenly between the kitchen and dining room.

Tours are task-informed. They are generated with privileged scene information, deliberately visit the dining room and frame its table, and skip the decoy room, whose geometry stays in the scene. The policy receives only the RGB frames. Fig. 6 shows one tour per setting.

### C.4 EPISODES

Every episode has a saved simulator state, including compiled model and runtime state, captured at the start of evaluation. The tour is rendered from a fresh reconstruction of this state, and evaluation restores the same state, so all methods start from identical scenes and see identical tours. Episodes are balanced over each task's supported layouts. Resets in which the task is already partly or fully solved are rejected using task predicates, never policy rollouts. We evaluate for 20 episodes per task, for a total of 200 episodes, for all settings.

### C.5 METRICS

Let $\mathcal{E}$ be the set of evaluated episodes and $\tau_e$ the task of episode $e$. Each task $\tau$ defines $P_\tau \geq 1$ independent placement goals (Tab. 8), each satisfied when its object comes to rest at its target. Let $c_e \in \{0, \ldots, P_{\tau_e}\}$ be the number of distinct placement goals satisfied at any step of episode $e$; if the episode succeeds, $c_e = P_{\tau_e}$. We report the success rate and the placement rate,

$$\mathrm{SR} = \frac{1}{|\mathcal{E}|} \sum_{e \in \mathcal{E}} \mathbf{1}[\text{episode } e \text{ succeeds}], \qquad \mathrm{PR} = \frac{1}{|\mathcal{E}|} \sum_{e \in \mathcal{E}} \frac{c_e}{P_{\tau_e}}, \tag{1}$$

where success is RoboCasa's task success. PR weights every episode equally regardless of how many placements its task has: completing one of four placements in PackFoodByTemp contributes $0.25$. Because every task has the same number of episodes, PR is also the mean of the per-task placement rates. § 3.2 and § 6.3 report PR. In `Unseen-Hard` we additionally log which rooms the robot enters.

## D EXPERIMENT DETAILS

### D.1 MOTIVATING EXPERIMENT DETAILS

**Policies.** § 3.2 compares three policies. All of them act through the same left-head, right-head and wrist cameras.

- **XR-1.** The released RoboCasa365 checkpoint of Xiaomi-Robotics-1 (Team et al., 2026), evaluated without fine-tuning. We use the authors' reference inference settings: 5 denoising steps, 4 history frames per camera and 16-action chunks. Actions are in RoboCasa365's native end-effector and base action space.
- $\pi_{0.5}$**.** The flat policy of App. B.5: the midtraining checkpoint at 100k steps, fine-tuned for 30k steps on the task instruction alone.
- $\pi_{0.5}$ **+ oracle.** A hierarchical policy whose planner is the no-tour planner of App. B.5 and whose executor is the 85k-step midtraining checkpoint. The planner predicts each subtask from the current images and the task. Whenever it predicts a navigation subtask, the evaluator computes the ground-truth distance and bearing from the robot to the named target, bins them as in training, and appends them to the subtask before passing it to the executor.

The oracle supplies only the geometry of the goal. It does not choose the target: the planner can still name the wrong object or repeat a finished subtask. If the named target cannot be resolved in the scene, the subtask is passed on without a hint.




**Figure 5:** How the unseen layouts are built, shown for four of the 10 layout families. Each row starts from a training kitchen (left), moves the dining table into a new room (middle), and adds a decoy room (right).

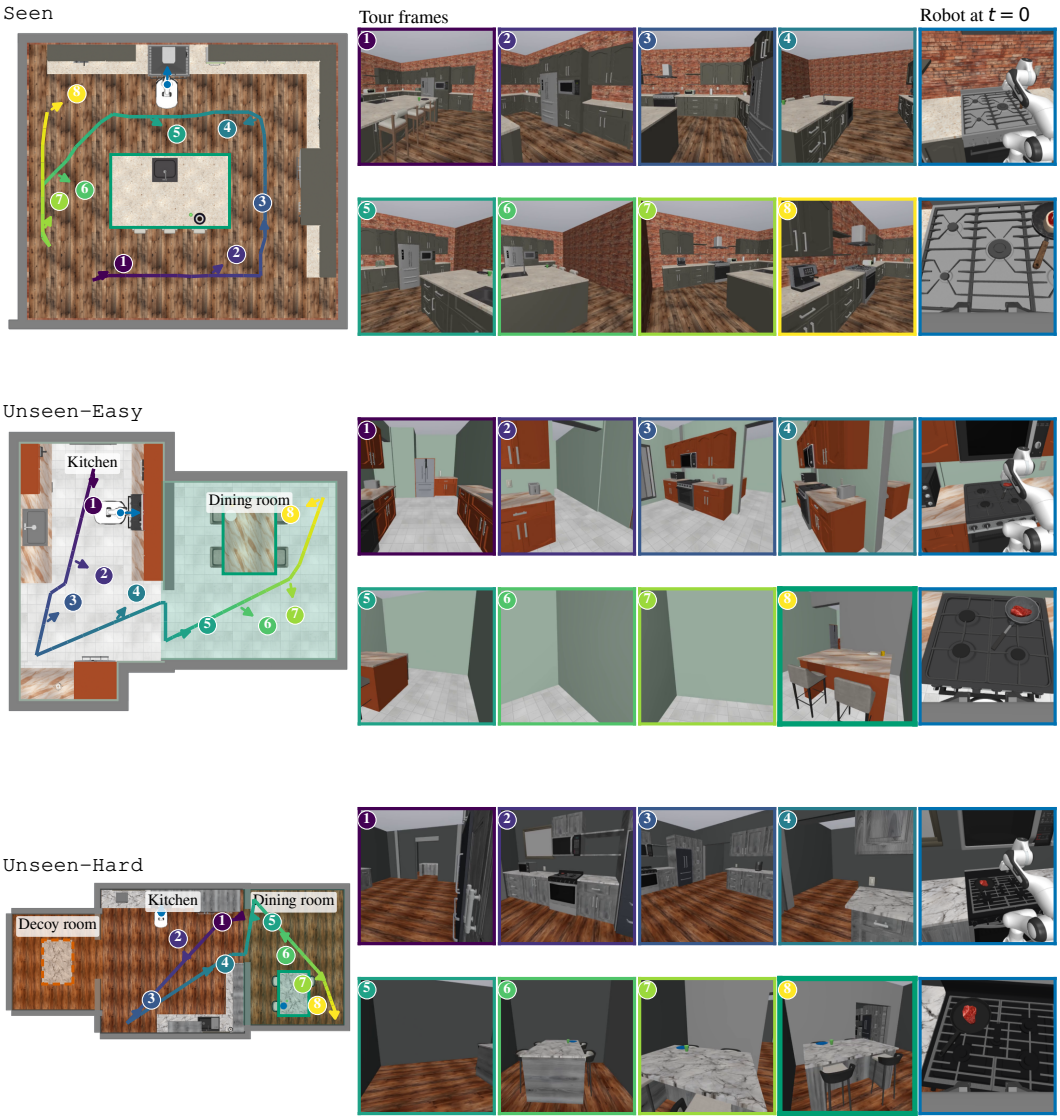



**Figure 6:** One episode per setting. **Left**: start state with the robot (blue) and the tour camera path coloured by time. **Middle**: tour frames numbered to match the path; the required dining-table view is outlined in green. **Right**: the robot's head and wrist cameras at $t$=0.

**Protocol and metrics.** Each policy is evaluated on the 200 `Seen` and 200 `Unseen-Hard` episodes of § 6.1, starting from the same saved initial states (App. C.4), with each task's native step limit. We report the placement rate (App. C.5). Distance travelled is the length of the mobile base's planar path over the recorded states of an episode, averaged over episodes. Time in the decoy room is the fraction of recorded states in which the base lies inside the decoy room's floor, averaged over `Unseen-Hard` episodes (App. E.1).

### D.2 Estimated Tour Geometry for Q3D

In the "GT geometry" row of Table 5, Q3D receives the simulator's tour depth and camera poses, which no other planner gets. The "est. geometry" row instead estimates them from RGB with Depth Anything 3 (DA3) (Lin et al., 2025). At the first observation, DA3 registers the tour views to the robot's cameras. During the episode, a SLAM-style module relocalizes the robot's camera views against a fixed DA3 reconstruction of the whole tour and corrects this registration. The robot's odometry is taken from the simulator. On held-out episodes this brings most tour-camera estimates to within 5° and 0.25 m of ground truth by the end of the episode. The remaining gap to ground-truth geometry in the unseen settings mostly comes from errors in the DA3 reconstruction itself, which relocalization cannot fix.

## E Additional Results

### E.1 Trajectory Visualizations

Fig. 7 shows where XR-1 and $\pi_{0.5}$ + oracle drive the mobile base in `Unseen-Hard` episodes of the motivating experiment (§ 3.2).

**Paths.** Every evaluation episode records the simulator state after each control step (20 Hz), including the planar pose of the mobile base. Each path in Fig. 7 is the sequence of recorded base positions, drawn from the start of the episode to its end at task success or the task's step limit. Path colour encodes the fraction of the step limit elapsed, so that paths of tasks with different horizons are comparable; an unsuccessful episode ends at the lightest colour.

**Overheads.** Each episode starts from a saved simulator state that is identical for all policies (App. C.4). We restore this state and render it with an orthographic top-down camera, removing wall geometry above 1.5 m for display only. Room floors, the dining table and the decoy table are taken from the layout and drawn in world coordinates, so they align with the paths. Layouts that are taller than they are wide are rotated by 90° to fit the page.

**Per-panel statistics.** Below each panel we report the number of placement goals completed (App. C.5) and the percentage of recorded states in which the base lies inside the decoy room's floor. The latter is the per-episode quantity averaged in Fig. 2c.

**Episode selection.** To avoid choosing episodes by hand, we sample 12 of the 200 `Unseen-Hard` episodes uniformly without replacement; both policies are shown on every sampled episode. A single sample of 12 can differ substantially from the full evaluation, so we draw one sample for each of the seeds $0, \ldots, 99$ and keep the one closest to the full set of episodes. For a sample $S$ we measure its distance to the full set $\mathcal{E}$ as

$$d(S) = \sum_{\pi \in \{\text{XR-1},\, \pi_{0.5}+\text{oracle}\}} \Big( |\overline{\mathrm{PR}}_\pi(S) - \overline{\mathrm{PR}}_\pi(\mathcal{E})| + |\overline{\mathrm{D}}_\pi(S) - \overline{\mathrm{D}}_\pi(\mathcal{E})| \Big), \tag{2}$$

where $\overline{\mathrm{PR}}_\pi$ and $\overline{\mathrm{D}}_\pi$ are policy $\pi$'s mean placement rate and mean fraction of time in the decoy room over the given episodes. The selected sample (seed 64) has $d = 0.057$, against a median of $0.24$ over the 100 seeds. Its means are close to those of the full evaluation for both policies: XR-1 has a placement rate of 22% (vs. 24%), spends 26% (vs. 28%) of its time in the decoy room and travels 47 m (vs. 48 m); $\pi_{0.5}$ + oracle has 31% (vs. 31%), 2% (vs. 3%) and 19 m (vs. 21 m). Distance travelled was not used for selection.

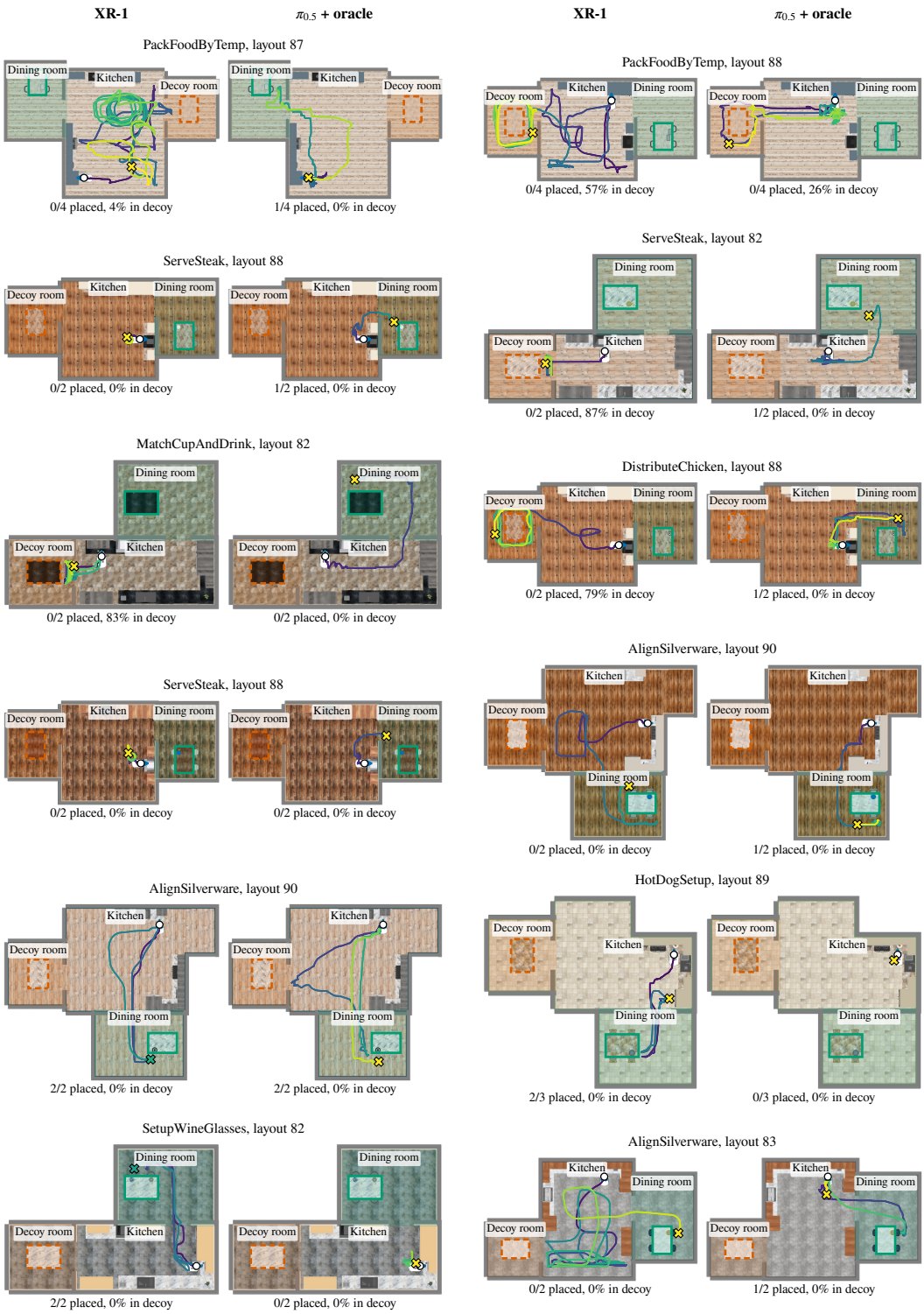



**Figure 7:** Base paths of XR-1 and $\pi_{0.5}$ + oracle in 12 `Unseen-Hard` episodes sampled uniformly at random and selected to be representative of the full evaluation (App. E.1). Both policies start each episode from the same saved state. Paths are coloured by the fraction of the task's time limit elapsed, from dark (start) to light (time limit); the circle marks the start and the cross the final position. The dining table is outlined in solid green and the decoy table in dashed orange. Below each panel are the placements completed and the percentage of the episode spent in the decoy room. Layouts taller than they are wide are rotated by 90°.

**Table 9:** Room choice in `Unseen-Hard` (200 episodes per row). *Left kitchen*: episodes in which the base enters another room. *Correct first room*: of those, the fraction whose first room is the dining room rather than the decoy room; choosing at random gives 50%. *Entered decoy*: episodes in which the base enters the decoy room at any point. Only Q3D planners pick the dining room well above chance; with ground-truth geometry they match the oracle.

| Method | Left kitchen (%) | Correct first room (%) ↑ | Entered decoy (%) ↓ |
|---|---|---|---|
| Flat $\pi_{0.5}$ | 83.0 | 41.6 | 63.0 |
| Hier. $\pi_{0.5}$, no nav hint | 81.0 | 46.3 | 58.0 |
| *Hier. $\pi_{0.5}$ + nav hint from VLM planner* | | | |
| Stock Qwen3-VL-4B | 83.5 | 41.3 | 57.5 |
| CS | 86.5 | 42.2 | 58.0 |
| CP | 88.5 | 45.8 | 55.0 |
| V3R | 90.0 | 51.1 | 52.5 |
| Q3D, est. geometry | 77.5 | **73.5** | **25.5** |
| *Privileged: ground-truth geometry* | | | |
| Q3D, GT geometry | 85.5 | 85.4 | 16.5 |
| Hier. $\pi_{0.5}$ + GT nav hint (oracle) | 84.5 | 84.0 | 15.5 |

**Table 10:** Q3D planner with different sources of tour geometry. Hint accuracy: navigation-hint bearing within one of 16 bins of ground truth during rollouts. Correct first room: as in Table 9. Placement rate follows hint accuracy, and SLAM refinement recovers part of the gap to ground-truth geometry.

| | Hint accuracy (%) | | | Placement rate (%) | | | Correct first |
|---|---|---|---|---|---|---|---|
| Tour geometry | `Seen` | `Easy` | `Hard` | `Seen` | `Easy` | `Hard` | room, `Hard` (%) |
| Ground truth | 74.8 | 74.5 | 75.5 | 36.3 | 34.4 | 32.5 | 85.4 |
| DA3 + SLAM | 70.8 | 51.4 | 45.5 | 37.0 | 27.6 | 24.8 | 73.5 |
| DA3, first frame only | 66.6 | 40.9 | 39.3 | 34.3 | 20.7 | 21.0 | 78.7 |

### E.2 Room Choice in `Unseen-Hard`

Table 9 reports which room each policy enters first in `Unseen-Hard`, where the kitchen connects to both the dining room and a decoy room. Every policy without explicit 3D input picks the dining room first in 41–51% of the episodes in which it leaves the kitchen, which is no better than choosing at random. Adding a hierarchy without spatial information does not change this, and neither do the 2D spatial planners. The Q3D planner picks the dining room first in 74% of these episodes with estimated geometry and 85% with ground-truth geometry. The latter matches the oracle that receives ground-truth navigation hints (84%). It also halves or better the fraction of episodes that ever enter the decoy room.

### E.3 Source of Tour Geometry

Table 10 varies only the source of Q3D's tour geometry. With ground-truth depth and poses, hint accuracy stays near 75% in all three settings. With Depth Anything 3 estimates from the first frame only, it falls to about 40% in the unseen settings, and placement rate falls with it. Refining the estimate online with SLAM relocalization (App. D.2) recovers part of this gap: hint accuracy rises by 6–11 points and placement rate by 4–7 points in the unseen settings. Across all three sources, placement rate follows hint accuracy, so the remaining gap to ground-truth geometry comes mainly from the accuracy of the estimated geometry. The first-frame-only variant picks the correct first room slightly more often than SLAM (79% vs. 74%), but leaves the kitchen in only 68% of `Unseen-Hard` episodes, against 78% with SLAM.

### E.4 VSI-Bench Validation of Reimplemented Recipes

To verify that our reimplementations of the implicit recipes in § 5.1 are faithful, we evaluate them on VSI-Bench (Yang et al., 2024), where the original methods report results. We train CS and CP variants of Qwen3-VL-4B on VSI-590K alone and evaluate zero-shot on all 5,130 questions with 64 frames per video. Our CS recipe improves the stock model from 58.4 to 71.1, exceeding the

**Table 11:** VSI-Bench results. Our models start from Qwen3-VL-4B and are trained on VSI-590K only; they are evaluated with 64 frames at 64 visual tokens per frame. Published results are from Yang et al. (2026); † is Cambrian-S-7B finetuned only on VSI-590K, and "64 fr." rows are its frame-matched ablation (overall only). Numerical categories use mean relative accuracy, multiple-choice categories use accuracy, and Avg. is the equal-weight mean of the eight categories. Our CS and CP reimplementations are competitive with the published Cambrian models.

| | | | Numerical | | | | Multiple choice | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Model | Steps | Avg. | Count | Abs. dist. | Obj. size | Room size | Rel. dist. | Rel. dir. | Route | Appr. order |
| *Published, 7B* | | | | | | | | | | |
| VLM-3R | – | 60.9 | 70.2 | 49.4 | 69.2 | 67.1 | 65.4 | 80.5 | 45.4 | 40.1 |
| Cambrian-S | – | 67.5 | 73.2 | 50.5 | 74.9 | 72.2 | 71.1 | 76.2 | 41.8 | 80.1 |
| Cambrian-S† | – | 69.2 | 73.6 | 53.7 | 75.2 | 74.7 | 71.5 | 82.0 | 38.7 | 84.3 |
| Cambrian-P | – | 73.7 | 74.9 | 60.1 | 76.0 | 76.9 | 74.8 | 89.5 | 52.6 | 85.0 |
| Cambrian-P w/o pose, 64 fr. | – | 70.3 | – | – | – | – | – | – | – | – |
| Cambrian-P, 64 fr. | – | 73.1 | – | – | – | – | – | – | – | – |
| *Ours, Qwen3-VL-4B* | | | | | | | | | | |
| Stock | – | 58.4 | 68.6 | 50.5 | 75.6 | 61.2 | 58.2 | 50.0 | 34.0 | 69.4 |
| CS | 3k | 70.8 | 73.5 | 55.1 | 76.1 | 72.3 | 74.1 | 85.8 | 44.8 | 84.6 |
| CS | 7k | 71.1 | 71.3 | 59.5 | 74.6 | 74.7 | 72.7 | 85.0 | 45.9 | 85.0 |
| CP | 3k | 71.7 | 70.2 | 58.2 | 76.1 | 72.9 | 71.0 | 92.0 | 48.5 | 85.1 |

**Table 12:** CP model trained with six or one labels per tour. Bearing accuracy (%, within one of 16 bins) at 10k steps on MolmoSpaces houses, `Unseen-Easy` (Easy) and `Unseen-Hard` (Hard). Six labels per tour improve accuracy on every split, especially RoboCasa.

| # Labels | Houses | Easy | Hard |
|---|---|---|---|
| 1 | 66.5 | 36.0 | 35.9 |
| 6 | **70.5** | **47.2** | **43.1** |

published Cambrian-S-7B trained on the same data (69.2). Adding pose supervision raises accuracy to 71.7, within 1.4 points of the published Cambrian-P at the same frame budget (Table 11).

### E.5 Multiple Labels per Tour

Since processing long video context is so expensive, we study the effects of including **multiple labels per context**, increasing supervision density, in Tab. 12. We train the CP model with 1 and 6 labels per spatial context under a fixed gradient step budget and find that this leads to a modest improvement across the board.