VLMs Walk the Scene: View Planning via Scene Self-Exploration

Kangrui Wang[1], Linjie Li[2], Zhengyuan Yang[3], Shiqi Chen[4], Zihan Wang[1], Li Fei-Fei[5], Jiajun Wu[5], Leonidas Guibas[5], Lijuan Wang[3], Manling Li[1]

[1]Northwestern University [2]University of Washington [3]Microsoft
[4]University of Oxford [5]Stanford University

kangrui.wang@northwestern.edu, manling.li@northwestern.edu

Homepage Code Models & Data

Can VLMs predict how each camera move changes the view, and plan many such moves ahead? We call this capability *view planning*, requiring (1) understanding how a single action transforms the view, and (2) composing many such transformations across multi-turn plans to identify a target view. We probe both abilities in our proposed VIEWSUITE, a 3D point-cloud environment on real ScanNet scenes. Across 13 frontier VLMs, a critical planning gap emerges: they possess basic view-action knowledge but fail to compose it across multi-turn plans, with the gap widening as viewpoint distance grows. To close this gap, we propose an iterative framework that alternates *self-exploration* with *view graph distillation*. The key insight is that all exploration trajectories, regardless of their outcome, collectively form a view graph that compactly captures how viewpoints connect across a scene. Distilling this graph into diverse supervised tasks reshapes the policy distribution and overcomes the sparse rewards that stall pure RL. This improves Qwen2.5-VL-7B from 2.5% to 47.8% on interactive view planning, surpassing GPT-5.4 Pro (18.5%) and Gemini 3.1 Pro (21.4%). Self-exploration emerges as a promising path toward VLMs that can actively reason and plan in 3D space.

1. Introduction

View search is fundamental to understanding the visual world: an agent must actively choose where to look and how each move changes its view. Prior view search is limited to 2D image regions (Wu and Xie, 2024; Wang et al., 2025d) or panoramic rotation (Yu et al., 2025). We take one step forward to *multi-step view planning in real 3D scenes*. Whether vision-language models (VLMs) (OpenAI, 2023; Gemini Team et al., 2023; Bai et al., 2025) can do this remains underexplored. Our setting differs in three key ways (Table 1): (1) *real 3D scenes* rather than synthetic graphics; (2) *pure 6-DoF viewpoint control*, different from physical affordance, embodied navigation, and 2D image cropping; and (3) *multi-turn view composition* rather than single-step decisions.

We *decompose* view planning into two coupled abilities: *understanding* how each action transforms the view, and *leveraging* that understanding for multi-turn planning. To probe both, we build VIEWSUITE, the first benchmark for view planning in real 3D scenes with full 6-DoF viewpoint control. VIEWSUITE is constructed on ~300 real ScanNet (Dai et al., 2017) indoor scenes,

Table 1 | Comparison with existing view reasoning benchmarks. *EmbodiedBench supports multiple action types.

Benchmark	Task	Scale	Real World	3D	Action Space	Multi-turn
ViewSpatial-Bench (Li et al., 2025a)	View-Centric QA	5.7K	✓		–	
VSI-Bench (Yang et al., 2025a)	Video QA	5K	✓	✓	–	
CameraBench (Lin et al., 2025)	Video QA	3K	✓		–	
MindCube (Wang et al., 2025b)	View-Centric QA	21K		✓	–	
V* (Wu and Xie, 2024)	Visual Search	–	✓		2-DoF	✓
ActiView (Wang et al., 2025d)	Visual Search	3K	✓		2-DoF	✓
H*Bench (Yu et al., 2025)	Visual Search	–	✓	✓	2-DoF	✓
HM3D-OVON (Yokoyama et al., 2024)	Embodied Agent	–	✓	✓	4-DoF	✓
EmbodiedBench (Yang et al., 2025b)	Embodied Agent	1.1K		✓	6-DoF*	✓
Theory of Space (Zhang et al., 2026)	Embodied QA	2.7K		✓	2-DoF	✓
VIEWSUITE (Ours)	Visual Search	165K	✓	✓	6-DoF	✓

yielding ~55K view pairs and ~165K task instances across three diagnostic tasks, with the IVP success threshold calibrated against human judgments via an alignment study (Appendix A.3): **Path-to-View** (P2V) predicts the resulting view from an action sequence; **View-to-Path** (V2P) infers the action sequence between two views; **Interactive View Planning** (IVP) plans view changes over multiple turns and submits a 6-DoF estimate of the target. P2V and V2P test *understanding* viewpoint transitions (single-turn); IVP tests *leveraging* that understanding through multi-turn view planning. Evaluating 13 frontier VLMs reveals a planning gap: the best models reach ~70% on short-horizon P2V/V2P but collapse to at most 21% on IVP, with most below 10%.

To close this gap, the natural first attempt is direct reinforcement learning. We find this surprisingly ineffective: with baseline success at only ~ 2.5%, direct PPO plateaus at 3.2%; switching to GRPO with reward-variance filtering only reaches 5.2%; and even iterating PPO with SFT on the small set of successful trajectories (Success-Only Bootstrapping) achieves only 6.2%. The breakthrough comes from a subtler observation: even *failed* trajectories encode valid view transitions, since moving from viewpoint A to B is meaningful supervision regardless of the original goal. Distilling this signal from raw, mostly-failed exploration is the central design challenge. We address it with *view graph distillation*: we condense trajectories into a structured graph, sample paths from it, reformulate them into supervised view-planning demonstrations, and alternate this with self-exploration. This combination, but neither stage alone, takes Interactive View Planning from 2.5% to 47.8%, surpassing GPT-5.4 Pro (18.5%) and Gemini 3.1 Pro (21.4%).

Our contributions are threefold:

- *Task formulation and benchmark*: we formulate view planning as two coupled abilities and build VIEWSUITE, a 3D point-cloud environment on real ScanNet scenes with three diagnostic tasks (Path-to-View, View-to-Path, Interactive View Planning).
- *Revealing the planning gap*: frontier VLMs achieve ~ 50–70% on short-horizon Path-to-View and View-to-Path but fall below 21% on Interactive View Planning.
- *Iterative training via self-exploration and view graph distillation*: our framework alternates self-exploration with view graph distillation, condensing exploration trajectories into a graph and reformulating them into supervised view-planning demonstrations. It improves a 7B VLM from 2.5% to 47.8% on Interactive View Planning, surpassing all frontier models.

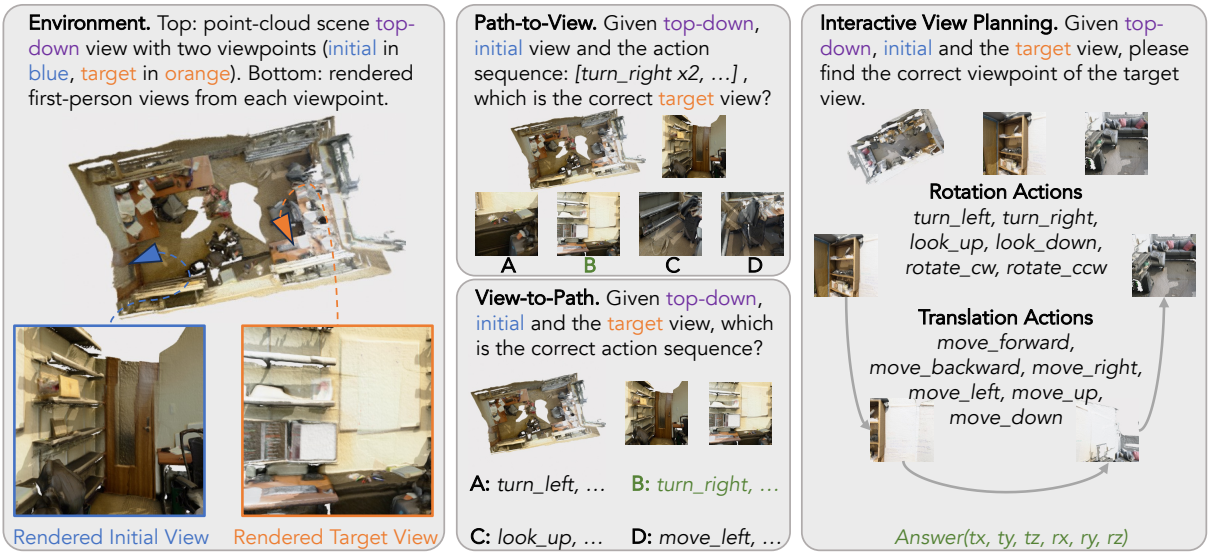


Figure 1 | Overview of VIEWSUITE. **Left**: Point cloud environment built on ScanNet with viewpoints (initial in blue, target in orange) and rendered first-person views. **Middle**: Path-to-View (P2V; predict the resulting view from an action sequence) and View-to-Path (V2P; infer the action sequence from two views), both single-turn. **Right**: Interactive View Planning (IVP), a multi-turn task where the agent plans view changes to match the target view and submits a viewpoint estimate.

Beyond these, we show that interactive view planning yields *transferable spatial priors*: under identical post-training, our model outperforms its base counterpart on related view-understanding tasks both within VIEWSUITE and on the external MindCube (Wang et al., 2025b) benchmark.

2. VIEWSUITE: Problem Formulation, Environment, and Benchmark

VIEWSUITE casts view planning as a multi-turn decision process: an agent issues 6-DoF viewpoint-altering actions in a 3D point-cloud environment built on real ScanNet (Dai et al., 2017) indoor scenes, and at termination submits a target viewpoint estimate scored by distance to the ground truth (formal MDP and reward in Section 4, Eq. 3).

2.1. Problem Formulation

We decompose view planning into two coupled abilities: (1) *understanding* viewpoint transitions, i.e., knowing how an action changes the view, and (2) *leveraging* that understanding for multi-turn view planning. We design three complementary tasks targeting these abilities.

Path-to-View (P2V). Given an initial view, a top-down reference, and an action sequence, the model predicts the resulting view from four options (multiple-choice), testing whether it can mentally simulate viewpoint transitions (Figure 6).

View-to-Path (V2P). Given initial and target views plus a top-down view, the model identifies which action sequence was executed, again from four options. P2V and V2P together test view-action understanding in both directions (Figure 7).

Interactive View Planning (IVP). Given an initial view, a target view, and a top-down reference, the agent issues multiple actions per turn, observes the resulting view and viewpoint, and within a fixed turn budget submits a 6-DoF estimate of where the target view was taken. Unlike the single-turn P2V and V2P, IVP requires the agent to plan a sequence of view changes over multiple turns to localize the target (Figure 8).

2.2. Viewpoint Control Interface

VIEWSUITE supports 6-DoF viewpoint control through 12 step-size-parameterized actions (Figure 1; full definitions in Appendix A.1). The interface is decoupled from the 3D backend (point clouds, meshes, or simulators) and uses Open3D (Zhou et al., 2018) for rendering. Step sizes are discretized so VLMs control the viewpoint without specifying precise motion parameters.

2.3. Data Collection and Evaluation

To construct the task data, we sample initial-target view pairs from ScanNet video frames. We use fixed step sizes s_t=0.5 m for viewpoint translation and s_r=30° for viewpoint rotation, so that each action can produce a visually distinguishable view change while maintaining fine-grained viewpoint control. We conduct scene-level and pair-level filtering then reformat each pair into P2V, V2P, and IVP instances (detailed in Appendix A.2). This yields ~ 55K view pairs across 286 ScanNet scenes.

Dataset splits. From the ~ 55K view pairs, we split pairs within each scene at 1:10 ratio to form VIEWSUITE-5K and VIEWSUITE-50K; all experiments in this paper use VIEWSUITE-5K, with the full set reserved for future scaling studies. Scenes are then partitioned into train/dev/test at 8:1:1 ratio, ensuring no scene overlap across splits. Each view pair yields three task instances (P2V, V2P, and IVP), giving ~ 15K instances in VIEWSUITE-5K, with the test set containing 530 pairs × 3 tasks = 1,590 instances. In total, VIEWSUITE provides ~ 165K task instances across both subsets.

Evaluation metrics. For P2V and V2P, we use accuracy (correct option selected). For IVP, we use *Success Rate*, defined based on viewpoint distance. We extract 6-DoF viewpoints from camera-to-world extrinsic matrices, decomposing each into a position vector $\mathbf{t} \in \mathbb{R}^3$ and a rotation matrix $R \in \mathrm{SO}(3)$. Translation distance and rotation distance between two viewpoints are

$$d_{\text{pos}} = \|\mathbf{t}_1 - \mathbf{t}_2\|_2, \quad d_{\text{rot}} = \arccos\left(\frac{\text{tr}(R_1^\top R_2) - 1}{2}\right). \tag{1}$$

The agent succeeds when $d_{\text{pos}} \le \beta_t \cdot s_t$ and $d_{\text{rot}} \le \beta_r \cdot s_r$ for threshold multipliers β_t, β_r. We calibrate β_t and β_r via a human alignment study: annotators judge planning success from rendered view pairs, and we select the combination that maximizes F_1 agreement with human judgments (full study in Appendix A.3). The best setting is β_t=1, β_r=1, i.e., one step size in each dimension (details in Table 8).

Unified view distance. We combine translation and rotation distance into a single difficulty score, the *unified view distance*:

$$d = \sqrt{(d_{\text{pos}}/s_t)^2 + (d_{\text{rot}}/s_r)^2}, \tag{2}$$

where normalizing by the step sizes makes each unit of d approximately correspond to one atomic action. Across the 530 test pairs, d ranges from 1.4 to 6.8 with mean 3.7 (Figure 5 in Appendix A). We split test pairs into SHORT ($d < 3$; 185 pairs) and LONG ($d \ge 3$; 345 pairs) subsets,

Table 2 | Main evaluation results on VIEWSUITE. Proprietary: GPT (OpenAI, 2023), Gemini (Gemini Team et al., 2023), Claude (Anthropic, 2024), Grok. Open-weight: Qwen (Bai et al., 2025), GLM (Zeng et al., 2024), Kimi. Accuracy (%) shown for Short / Long / Overall splits. Models sorted by Overall within each group; best per column in **bold**.

	Path-to-View (P2V)			View-to-Path (V2P)			View Planning (IVP)			
Model	Short	Long	All	Short	Long	All	Short	Long	All	Overall
Random Response	20.7	24.6	23.3	24.3	26.5	25.7	2.2	0.0	0.8	16.6
Proprietary Models										
GPT-5.4 Pro	**70.7**	**43.8**	**53.1**	**72.4**	39.0	**50.7**	32.6	11.0	18.5	**40.8**
Gemini 3.1 Pro	63.6	40.9	48.8	53.0	**47.7**	49.5	28.8	**17.4**	**21.4**	39.9
GPT-5.4	57.1	42.9	47.8	60.5	37.5	45.6	**33.7**	7.5	16.6	36.7
Grok 4.20 Beta	61.4	38.0	46.1	44.9	44.5	44.6	17.4	2.9	7.9	32.9
GPT-5.1	60.3	35.1	43.9	52.4	33.4	40.1	12.0	3.2	6.2	30.1
Claude Opus 4.6	46.7	28.4	34.8	47.6	38.4	41.6	23.9	3.8	10.8	29.0
Gemini 3 Pro	50.5	31.0	37.8	44.9	35.5	38.8	13.6	7.0	9.3	28.6
Open-Weight Models										
Qwen3.5-397B	**57.6**	30.1	**39.7**	**44.3**	**30.8**	**35.5**	**12.5**	0.0	**4.3**	**26.5**
GLM-4.6V	36.4	23.2	27.8	31.4	29.7	30.2	9.2	**1.2**	4.0	20.7
Qwen2.5-VL-72B	28.3	29.3	28.9	35.7	29.9	31.9	2.2	0.6	1.1	20.7
Qwen3-VL-32B	27.2	27.5	27.4	41.1	28.5	32.9	4.3	0.0	1.5	20.6
Kimi K2.5	35.9	24.6	28.5	18.4	29.4	25.5	4.9	**1.2**	2.5	18.8
Qwen2.5-VL-7B	23.9	**32.5**	29.5	27.0	22.7	24.2	7.1	0.0	2.5	18.7

and further decompose along the translation and rotation axes for finer-grained diagnosis in Section 5.3.

3. Frontier VLMs Show a View Planning Gap

3.1. Single-Turn Understanding, Multi-Turn Failure

We evaluate 13 frontier VLMs (7 proprietary, 6 open-weight) plus a random-response baseline, detailed in Table 2. The central finding is a planning gap: frontier VLMs understand local view transitions but cannot compose them into multi-turn plans toward a target view. On P2V and V2P, the best models achieve ~50% overall and over 70% on short-horizon samples (well above the 25% MCQ chance baseline), indicating non-trivial knowledge of view-action mappings that degrades on long-horizon samples requiring mental simulation of cumulative transformations. On IVP, performance drops sharply: the best model (Gemini 3.1 Pro) reaches only 21.4%, most models score below 10%, and on long-horizon samples most fall below 3%. This gap holds across both proprietary and open-weight models, with all open-weight models below 5% on IVP. Notably, GPT-5.4 Pro consistently outperforms GPT-5.4 across all tasks and splits, including IVP; given that GPT-5.4 Pro is widely believed to be a test-time scaled variant of GPT-5.4, this suggests that additional test-time computation can meaningfully improve spatial reasoning on our benchmark.

Table 3 | Effects of turn budget and rendering quality. Left three blocks: IVP accuracy (%) under increasing turn budgets with point-cloud rendering. Rightmost block: P2V / V2P / IVP All-split accuracy at budget=10 with higher-fidelity Gaussian Splat (GS) rendering.

	IVP, B = 10			IVP, B = 20			IVP, B = 30			GS, B = 10			
Model	Short	Long	All	Short	Long	All	Short	Long	All	P2V	V2P	IVP	Overall
Gemini 3.1 Pro	28.7	**17.4**	**21.3**	30.8	**18.8**	**23.0**	33.0	**17.9**	**23.2**	**55.3**	**49.4**	**23.2**	**42.6**
GPT-5.4	**33.5**	7.5	16.6	33.0	11.9	19.2	35.1	12.1	20.2	43.8	31.1	18.5	31.1
Grok 4.20 Beta	17.3	2.9	7.9	23.8	5.2	11.7	27.0	3.5	11.7	28.3	31.5	8.1	22.6
Claude Opus 4.6	23.8	3.8	10.8	**31.9**	10.1	17.7	**35.7**	11.0	19.6	35.3	41.3	12.3	29.6

3.2. What Bottlenecks Interactive View Planning?

Does turn budget affect IVP performance? A natural hypothesis is that models fail at IVP simply because 10 turns is insufficient. We test this by increasing the turn budget to 20 and 30 for four proprietary models (Table 3). All models improve from 10 to 20 turns, with Claude Opus 4.6 showing the largest gain (nearly doubling). However, gains from 20 to 30 turns are marginal or zero for most models. This diminishing return suggests that IVP performance is bottlenecked by planning ability rather than exploration horizon, as models exhaust their effective strategies well before the turn limit.

Does rendering quality affect model performance? A natural concern is that point-cloud rendering, with its sparse and noisy pixels, may itself bottleneck the agent. We re-render the test set with 3D Gaussian Splatting (Kerbl et al., 2023), a higher-fidelity neural renderer, using pretrained per-scene 3DGS reconstructions of the ScanNet scenes from SceneSplat-7K (Li et al., 2025b), and re-evaluate four proprietary models on all three tasks at budget= 10 (Table 3, rightmost block). The pattern across tasks is asymmetric: IVP improves consistently but only marginally (+0.2 to +1.9 points), whereas P2V and V2P show mixed and sometimes large changes—Gemini 3.1 Pro gains +6.5 on P2V, while GPT-5.4 and Grok 4.20 Beta lose −14.5 and −13.1 on V2P respectively (relative to their point-cloud rendering scores in Table 2). That a higher-fidelity renderer does not unlock single-turn performance, and yields only modest gains on IVP, indicates that the IVP bottleneck is not the visual fidelity of the rendered observation but the model's ability to compose view changes into a multi-turn plan.

Is rotation or translation the primary difficulty driver? Decomposing unified view distance into rotation and position axes (Figure 2) reveals contrasting difficulty drivers across task types. P2V/V2P degrade primarily with rotation distance (e.g., GPT-5.4 Pro loses ~ 25 points across rotation bins on P2V), since cumulative rotations are hard to mentally simulate. IVP reverses this: success collapses with position distance (~ 7× drop for GPT-5.4 Pro), as 3D translation requires spatial layout understanding and path planning beyond simple orientation control.

Sample-level factor analysis (Spearman ρ across 12 geometric, visual overlap, and directional factors defined in Appendix B.3; full results in Appendix B.2) further confirms the position-bottleneck for IVP and rotation-bottleneck for P2V/V2P. These analyses show that single-turn view-action understanding does not confer multi-turn view planning capability, motivating training specifically for IVP. Since no view-planning demonstrations exist for VLMs, the agent must learn through self-exploration of 3D scenes.

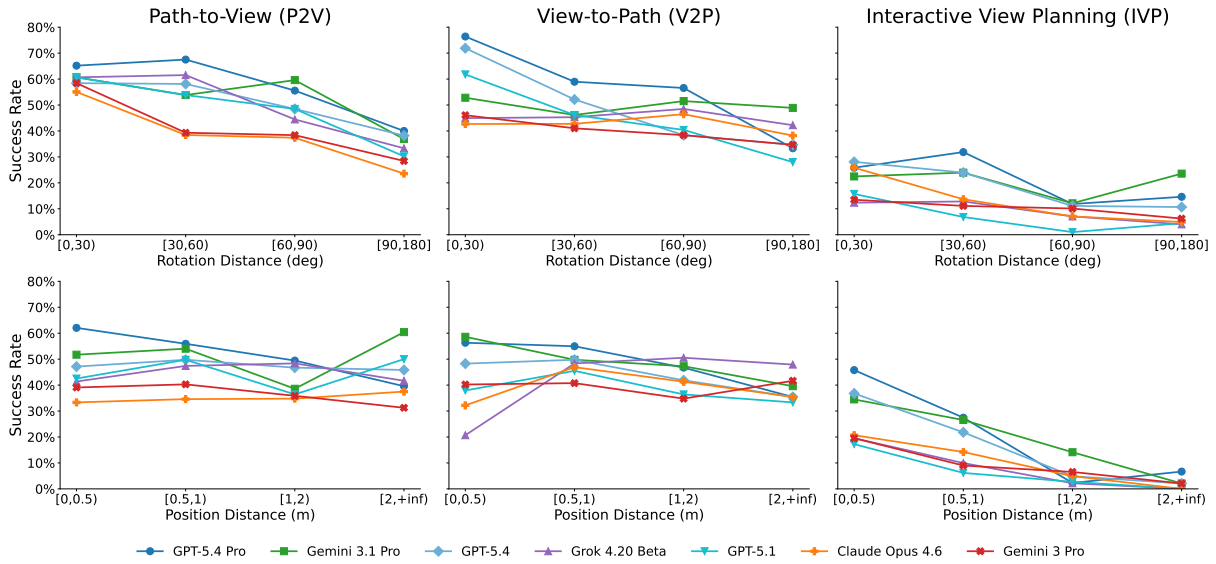


Figure 2 | Success rate vs. rotation distance (top) and position distance (bottom) for proprietary models across all three tasks.

4. Self-Exploration with View Graph Distillation

Section 3 reveals a planning gap: VLMs understand local view transitions but cannot compose them into multi-turn plans (~50–70% on P2V/V2P vs. ~1–21% on interactive view planning). Understanding how an action changes the view is fundamentally different from composing a sequence of actions whose accumulated observations let the agent localize a target view. We ask whether an agent can bridge this gap purely through self-exploration: interacting with 3D environments, learning from its own experience, and progressively improving without any external demonstration. Although our action set superficially resembles embodied navigation primitives, an IVP rollout is fundamentally a localization problem: actions move only the viewpoint (no body, no affordance) and form a planned trajectory of view manipulations; reward is granted for an accurate 6-DoF estimate of the target view, not for physically arriving at it. We therefore treat actions as evidence-gathering operators for a localization decision made within the turn budget. The challenge is that with no demonstrations and a naive policy succeeding only ~2.5% of the time, the agent must extract supervision from its own experience.

Interactive View Planning. We model this task as a finite-horizon decision process. At each turn t, the agent observes the rendered view o_t and the current 6-DoF viewpoint $p_t \in \mathrm{SE}(3)$, and selects an action $a_t \in \mathcal{A}$ from the 12-element action set. The environment deterministically updates the viewpoint, $p_{t+1} = T(p_t, a_t)$, and renders the next view. After at most T turns the agent submits a target estimate $\hat{p}^* \in \mathrm{SE}(3)$, scored by:

$$r(\hat{p}^*, p^*) = \mathbf{1}\left[d_{\mathrm{pos}}(\hat{p}^*, p^*) \le \beta_t s_t \ \wedge \ d_{\mathrm{rot}}(\hat{p}^*, p^*) \le \beta_r s_r\right] + 0.1\,\mathbf{1}_{\mathrm{format}}, \tag{3}$$

where p^* is the ground-truth target viewpoint and $\beta_t = \beta_r = 1$ are the human-calibrated thresholds (Section 2). A learned policy π_θ maps the rollout history to the next action and, upon termination, to the target estimate.

We observe, however, that every trajectory, whether or not it reaches its goal, traces valid view transitions through the scene. Aggregated together, these trajectories form a view graph:

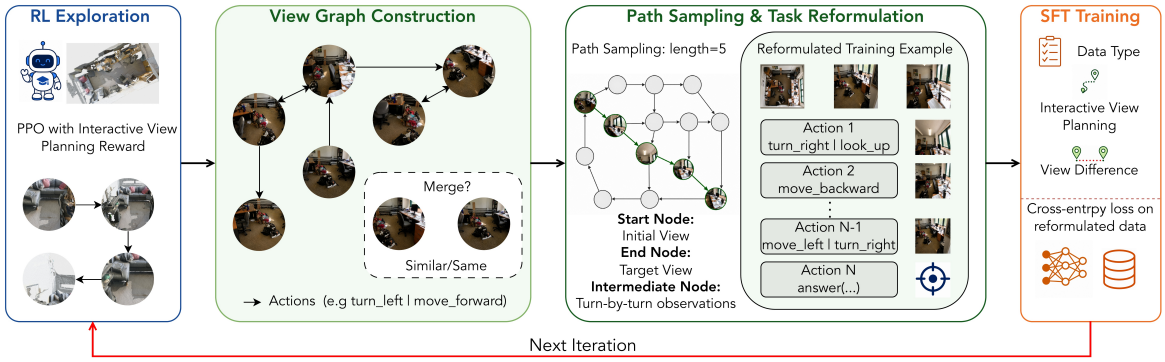


Figure 3 | Iterative training pipeline. **Left**: in the *self-exploration* stage, the agent actively explores VIEWSUITE environments under sparse outcome rewards; completed trajectories are continuously compressed into a view graph whose nodes are viewpoints and whose edges are actions. **Right**: in the *view graph distillation* stage, paths sampled from the graph are reformulated into multi-turn view-planning demonstrations and auxiliary supervision (view-difference estimation and multiple-choice). The distilled model initializes the self-exploration iteration, progressively bootstrapping the policy.

a compact representation of how viewpoints connect, in which connectivity discovered by one episode becomes reusable knowledge for any other. This mirrors how humans build spatial maps, where even a wrong turn teaches which rooms connect to which hallways. Motivated by this observation, we propose iterative training that alternates between self-exploration and view graph distillation, bootstrapping view planning capability from its own experience.

Each iteration consists of two stages (Figure 3). In the **self-exploration stage**, the agent actively interacts with VIEWSUITE environments and its trajectories are incrementally compressed into a view graph. In the **view graph distillation stage**, paths are sampled from this graph and reformulated into diverse view-planning demonstrations used to fine-tune the policy. The resulting model initializes the next self-exploration stage, enabling progressively stronger view planning.

Self-Exploration Stage with Sparse Rewards. The agent interacts with the VIEWSUITE environment using PPO (Schulman et al., 2017) with two reward components: an outcome reward of +1 if the agent's predicted target viewpoint is within the IVP success threshold ($d_{\text{pos}} \leq 0.5\,\text{m}$, $d_{\text{rot}} \leq 30°$, Section 2) and 0 otherwise, plus a format reward of +0.1 for producing a correctly structured response. A background process continuously converts completed trajectories into a view graph: each node represents a viewpoint with its rendered view, and each edge represents the actions taken between viewpoints. Nodes and edges are deduplicated via viewpoint-based similarity (Appendix C.4).

View Graph Distillation via Task Reformulation. In the distillation stage, we sample paths from the accumulated view graph and reformulate them into supervised training data. The key mechanism is *task reformulation*. For any path $P = (v_0, a_1, v_1, \ldots, a_K, v_K)$ in the graph, we define the operator

$$\mathcal{R}(P) = \big(o_{\text{init}} = v_0,\ o_{\text{target}} = v_K,\ (a_1, \ldots, a_K),\ \hat{p}^* = p_{v_K}\big), \tag{4}$$

which yields a valid IVP demonstration regardless of whether the original episode succeeded: the end node becomes the target, the start node becomes the initial view, and the action chain becomes the target action sequence. This is the lever that allows our framework to learn dense

supervision signals from agent-explored episodes regardless of their outcome (Algorithm 3 in Appendix C). We generate multiple supervision types from the same graph: (1) multi-turn view planning with task reformulation (primary task), (2) view difference estimation (predicting unified view distance between two views), and (3) view difference multiple-choice questions (details in Appendix C.5). The model is trained using standard cross-entropy loss with LLaMA-Factory (Zheng et al., 2024). The self-exploration stage is built on VAGEN (Wang et al., 2025a) and veRL (Sheng et al., 2024).

5. Self-Exploration Closes the Gap

5.1. Experimental Setup

We instantiate our framework on two base models: Qwen2.5-VL-7B-Instruct (Bai et al., 2025) as the primary base, and Qwen3-VL-8B-Instruct as a robustness check. In both cases, iterative training runs for four iterations on 3,419 VIEWSUITE training environments with up to 10 turns per episode (training/validation split in Appendix C.6). Each iteration alternates a self-exploration stage with 3 epochs of view graph distillation; the first three iterations use 60-step exploration stages for rapid bootstrapping, and the final iteration runs exploration to convergence. Full hyperparameters are provided in Appendix C.2 (RL) and Appendix C.3 (SFT).

Prompting Baselines. We include the untrained Qwen2.5-VL-7B-Instruct, GPT-5.4 Pro (OpenAI, 2023), and Gemini 3.1 Pro (Gemini Team et al., 2023) as zero-shot reference points.

Training Baselines. We compare against three RL methods, all trained from Qwen2.5-VL-7B-Instruct on the same environments with identical reward and action space:

- **Direct PPO.** PPO (Schulman et al., 2017) training from the base model without any distillation stage. This tests whether self-exploration alone can succeed given sufficient training steps.
- **Direct GRPO (filter).** GRPO (Shao et al., 2024) with n=4 rollouts per prompt and reward-variance-based filtering (Wang et al., 2025c). This tests whether an alternative RL algorithm with implicit best-of-n selection can bootstrap learning.
- **Success-Only Bootstrapping.** Iterates between PPO and SFT like our framework, but constructs SFT data by filtering successful RL trajectories (reward $>$ 0.5) rather than sampling from a view graph with task reformulation. This isolates the contribution of our framework's graph-based data generation from any trajectory, including failures.

Training Ablations. We evaluate three ablations of our framework on Qwen2.5-VL-7B-Instruct:

- **1 iter + RL** and **2 iter + RL.** Stop after fewer iterations to measure the contribution of the view graph distillation stage.
- **Random-graph.** Builds the view graph from a random action generator instead of model-collected trajectories, isolating the contribution of on-policy graph construction.

5.2. Closing the Gap on Interactive View Planning

As shown in Table 4, our framework improves Qwen2.5-VL-7B-Instruct from 2.5% to 47.8% on IVP, surpassing all frontier models; the same framework applied to Qwen3-VL-8B-Instruct reaches 32.5%, still well above every prompting and training baseline and within 11 points of

Table 4 | IVP success rates (%) across prompting and training baselines, ablations, and our methods.

Method	Short	Long	All
Prompting Baselines			
Qwen2.5-VL-7B-Instruct	7.1	0.0	2.5
GPT-5.4 Pro	32.6	11.0	18.5
Gemini 3.1 Pro	28.8	17.4	21.4
Training Baselines			
Direct PPO	7.0	1.2	3.2
Direct GRPO (filter)	10.8	2.2	5.2
Success-Only Bootstrapping	14.0	2.0	6.2
Training Ablations			
Random-graph	25.4	6.4	13.0
1 iter + RL	24.3	5.4	12.0
2 iter + RL	49.7	16.2	27.9
Our Methods			
Qwen2.5-VL-7B-Instruct	67.2	36.9	47.8
Qwen3-VL-8B-Instruct	56.8	19.4	32.5

the strongest frontier model (Gemini 3.1 Pro, 21.4%), indicating that the gains hold across pretrained backbones even though their absolute magnitude is backbone-dependent. All three training baselines remain below 7%: Direct PPO (3.2%) confirms that self-exploration alone cannot succeed when the base success rate is near zero; Direct GRPO with filtering (5.2%) shows that best-of-n selection helps only marginally; and Success-Only Bootstrapping (6.2%) underperforms our framework, highlighting the importance of view-graph construction and task reformulation that generate useful supervision from *any* trajectory rather than only successful ones. Iteration ablations show progressive improvement (12.0% $\rightarrow$ 27.9% $\rightarrow$ 47.8%) across 1, 2, and 3 iterations, while the Random-graph variant achieves only 13.0%, confirming that on-policy graph construction is critical. Graphs built from random-action trajectories cover state-space regions the model rarely visits during evaluation, so the resulting reformulated supervision transfers poorly. The ranking between methods is preserved under two evaluation-protocol relaxations—No-Snap (raw rotations executed as-is) and No-Submit (success the moment the pose enters the threshold), so the gain is not an artefact of rotation snapping or of the explicit submit step (Table 9, Appendix B.1).

5.3. What Has the Model Learned?

What exploration strategy does the trained model learn? We track 3D point cloud coverage across turns: **target intersection ratio** (fraction of target view vertices covered) and **scene coverage ratio** (fraction of all scene vertices observed); see Appendix D.1 for details.

Our trained model learns an effective exploration policy (Figure 4): scene coverage grows rapidly in early turns as the agent explores broadly, then plateaus as it locates the target direction; target intersection ratio accelerates in the middle turns as the agent moves toward the target, reaching $\sim 55\%$. This two-phase pattern (explore then approach) is absent in the base model and frontier models, which show flat or erratic target coverage throughout (full model comparison in Appendix D.1).

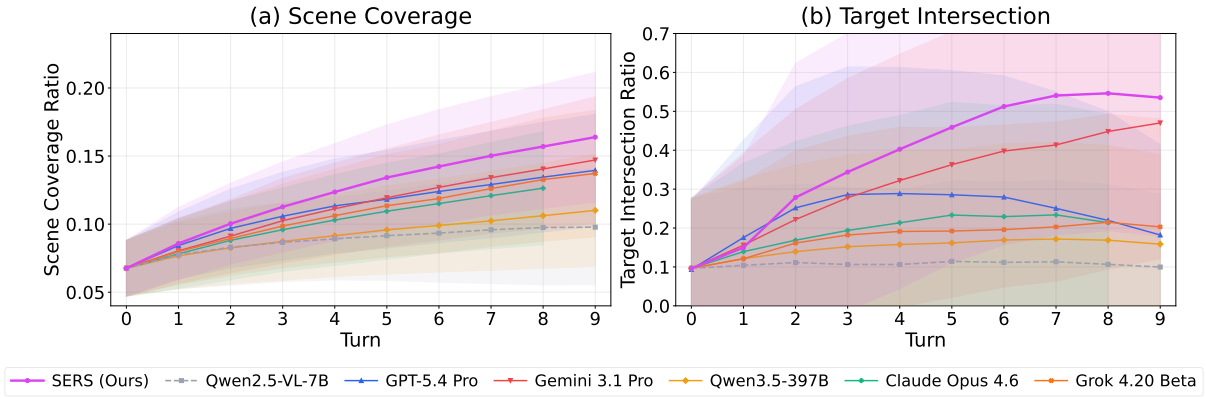


Figure 4 | Point cloud coverage across planning turns. (a) Scene coverage. (b) Target intersection.

Table 5 | Spatial prior transfer under identical GRPO post-training. P2V/V2P test transfer within view understanding; MindCube tests transfer to an external benchmark.

	P2V		V2P		MindCube	
Model	Init	Post	Init	Post	Init	Post
Base	32.1	45.1	29.2	44.8	33.0	56.3
Ours	25.7	**57.3**	31.6	**52.8**	33.1	**66.2**

We also analyze how training changes the model's attention mechanism (Appendix D.2): the trained model allocates higher image attention in early layers and transitions to text-space reasoning in deeper layers, with attention decreasing across turns as information accumulates.

Do the learned priors transfer to other view-related tasks? We ask whether the spatial priors acquired through interactive view planning transfer to other view-related tasks under further fine-tuning. We test this under identical GRPO post-training on (i) P2V and V2P from ViewSuite, which share scenes and action space with IVP but require different reasoning, and (ii) MindCube (Wang et al., 2025b), an external benchmark with no shared scenes, actions, or rendering pipeline. Details are provided in Appendix D.3. Despite comparable initial performance, our trained model outperforms the base by 8–12 points on P2V and V2P after post-training (Table 5), indicating that view-planning experience yields priors that go beyond the IVP task itself. On the external MindCube benchmark, our model gains ~ 10 points over the base, showing that the priors transfer to view-dependent spatial reasoning even outside our environment. It indicate that interactive view planning is not a narrow skill: it produces spatial priors that strengthen view understanding both within and beyond ViewSuite.

6. Related Work

View reasoning benchmarks. View-Centric QA benchmarks such as MindCube (Wang et al., 2025b) and ViewSpatial-Bench (Li et al., 2025a) test view-dependent reasoning from images but are non-interactive, as do broader static spatial QA benchmarks (Cheng et al., 2024; Chen et al., 2024; Ma et al., 2025). Video QA benchmarks (Yang et al., 2025a; Lin et al., 2025; Zhang et al., 2025) introduce temporal reasoning but still treat the model as a passive observer. Embodied QA (Zhang et al., 2026) and embodied agent benchmarks (Yang et al., 2025b; Yokoyama

et al., 2024) provide active interaction but optimize for *physical arrival* at semantic goals (objects, rooms), where success depends on affordance and traversability. VIEWSUITE instead targets *spatial localization through active view planning*: the agent plans view manipulations to gather visual evidence, then submits a 6-DoF estimate of where the target view was taken. Success requires localization accuracy, not physical arrival, isolating viewpoint reasoning from embodied navigation.

Visual search in VLMs. The benchmarks closest to ours probe view planning through visual search, an instance of active perception (Bajcsy, 1988; Aloimonos et al., 1988; Bajcsy et al., 2018): ActiView (Wang et al., 2025d) restricts the action space to zoom and shift within a 2D image; V* (Wu and Xie, 2024) performs LLM-guided visual search inside a single high-resolution image; H*Bench (Yu et al., 2025) studies head rotation over a 360° panorama. VIEWSUITE extends this thread to real 3D scenes with full 6-DoF viewpoint control, requiring multi-turn view planning to reach a target view (Table 1).

Agentic-RL and hindsight-replay. RLVR (Shao et al., 2024; DeepSeek-AI, 2025) demonstrate that outcome-supervised RL substantially improves LLM reasoning. Building on this, follow-up works (Zheng et al., 2025; Sheng et al., 2024; Wang et al., 2025c,a) extend RL training to agentic and multi-modal settings. In parallel, Hindsight Experience Replay (Andrychowicz et al., 2017; Zhang et al., 2023) relabels failed trajectories with achieved goals to generate denser reward signal. Unlike hindsight relabeling, which densifies reward by reassigning goals to failed episodes, we compress all exploration into a neat view-graph representation whose reusable spatial knowledge can be distilled into many SFT reformulations (view planning, view-difference estimation, forward/inverse dynamics). Distilling these reformulations reshapes the policy distribution so that subsequent RL rollouts reach high-reward trajectories more often, combining distribution sharpening (RL) with reshaping (SFT) to overcome sparse reward.

7. Conclusion and Limitations

We study *view planning*, the ability of an agent to compose viewpoint-altering actions into multi-turn plans, and reveal a planning gap in frontier VLMs: they understand local view transitions but cannot compose them into multi-turn plans toward a target view. We close this gap with an iterative framework alternating *self-exploration* and *view graph distillation*, improving a 7B model from 2.5% to 47.8% on interactive view planning, surpassing all frontier models, while building transferable spatial priors.

Limitations: we currently focus on static indoor scenes with a discrete 12-action interface, and our framework is validated on Qwen2.5-VL-7B and Qwen3-VL-8B; extending to outdoor or dynamic environments, continuous control, and larger model scales are natural next steps.

Acknowledgements

We acknowledge and disclose the use of AI tools (Claude Code and Codex) in code development and paper writing. We would also like to appreciate insightful discussions with Jiajun Liu, Baiqiao Yin, and Jihan Yang.

References

John Aloimonos, Isaac Weiss, and Amit Bandyopadhyay. Active vision. *International Journal of Computer Vision*, 1988.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, 2017.

Anthropic. The Claude model family, 2024. URL https://www.anthropic.com/claude.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025. URL https://arxiv.org/abs/2502.13923.

R. Bajcsy. Active perception. *Proceedings of the IEEE*, 1988.

Ruzena Bajcsy, Yiannis Aloimonos, and John K. Tsotsos. Revisiting active perception. *Autonomous Robots*, 42(2):177–196, 2018. URL https://arxiv.org/abs/1603.02729.

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Danny Driess, Pete Florence, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities, 2024. URL https://arxiv.org/abs/2401.12168.

An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. Spatialrgpt: Grounded spatial reasoning in vision-language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/f38cb4cf9a5eaa92b3cfa481832719c6-Abstract-Conference.html.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. URL https://doi.org/10.1109/CVPR.2017.261.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

Gemini Team, Rohan Anil, Sebastian Borgeaud, et al. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. URL https://arxiv.org/abs/2312.11805.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering, 2023. URL https://arxiv.org/abs/2308.04079.

Dingming Li, Hongxing Li, Zixuan Wang, Yuchen Yan, Hang Zhang, Siqi Chen, Guiyang Hou, Shengpei Jiang, Wenqiao Zhang, Yongliang Shen, Weiming Lu, and Yueting Zhuang. Viewspatial-bench: Evaluating multi-perspective spatial localization in vision-language models. *arXiv preprint arXiv:2505.21500*, 2025a. URL https://arxiv.org/abs/2505.21500.

Yue Li, Qi Ma, Runyi Yang, Huapeng Li, Mengjiao Ma, Bin Ren, Nikola Popovic, Nicu Sebe, Ender Konukoglu, Theo Gevers, et al. Scenesplat: Gaussian splatting-based scene understanding with vision-language pretraining. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025b. URL https://arxiv.org/abs/2503.18052.

Zhiqiu Lin, Siyuan Cen, Daniel Jiang, Jay Karhade, Hewei Wang, Chancharik Mitra, Tiffany Ling, Yuhan Huang, Sifan Liu, Mingyu Chen, Rushikesh Zawar, Xue Bai, Yilun Du, Chuang Gan, and Deva Ramanan. Towards understanding camera motions in any video. *arXiv preprint arXiv:2504.15376*, 2025. URL https://arxiv.org/abs/2504.15376.

Wufei Ma, Haoyu Chen, Guofeng Zhang, Yu-Cheng Chou, Jieneng Chen, Celso M de Melo, and Alan Yuille. 3dsrbench: A comprehensive 3d spatial reasoning benchmark, 2025. URL https://arxiv.org/abs/2412.07825.

OpenAI. GPT-4V(ision) system card, 2023. URL https://openai.com/research/gpt-4v-system-card.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*, 2024.

Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Chi Wan, Yiping Lu, Zhengyuan Yang, Lijuan Wang, Ranjay Krishna, Jiajun Wu, Li Fei-Fei, Yejin Choi, and Manling Li. Vagen: Reinforcing world model reasoning for multi-turn vlm agents. *arXiv preprint arXiv:2510.16907*, 2025a.

Qineng Wang, Baiqiao Yin, Pingyue Zhang, Jianshu Zhang, Kangrui Wang, Zihan Wang, Jieyu Zhang, Keshigeyan Chandrasegaran, Han Liu, Ranjay Krishna, Saining Xie, Manling Li, Jiajun Wu, and Li Fei-Fei. Spatial mental modeling from limited views. *arXiv preprint arXiv:2506.21458*, 2025b. URL https://arxiv.org/abs/2506.21458.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025c.

Ziyue Wang, Chi Chen, Fuwen Luo, Yurui Dong, Yuanchi Zhang, Yuzhuang Xu, Xiaolong Wang, Peng Li, and Yang Liu. Actiview: Evaluating active perception ability for multimodal large language models, 2025d. URL https://arxiv.org/abs/2410.04659.

Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal LLMs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. URL https://arxiv.org/abs/2312.14135.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025a. URL https://doi.org/10.1109/CVPR52734.2025.00994.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, Heng Ji, Huan Zhang, and Tong Zhang. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025b. URL https://proceedings.mlr.press/v267/yang25f.html.

Naoki Yokoyama, Ram Ramrakhya, Abhishek Das, Dhruv Batra, and Sehoon Ha. HM3D-OVON: A dataset and benchmark for open-vocabulary object goal navigation. *arXiv preprint arXiv:2409.14296*, 2024. URL https://arxiv.org/abs/2409.14296.

Heyang Yu, Yinan Han, Xiangyu Zhang, Baiqiao Yin, Bowen Chang, Xiangyu Han, Xinhao Liu, Jing Zhang, Marco Pavone, Chen Feng, Saining Xie, and Yiming Li. Thinking in 360°: Humanoid visual search in the wild, 2025. URL https://arxiv.org/abs/2511.20351.

Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. Chatglm: A family of large language models from GLM-130B to GLM-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024. URL https://arxiv.org/abs/2406.12793.

Pingyue Zhang, Zihan Huang, Yue Wang, Jieyu Zhang, Letian Xue, Zihan Wang, Qineng Wang, Keshigeyan Chandrasegaran, Ruohan Zhang, Yejin Choi, Ranjay Krishna, Jiajun Wu, Li Fei-Fei, and Manling Li. Theory of space: Can foundation models construct spatial beliefs through active exploration? In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2602.07055.

Tianjun Zhang, Fangchen Liu, Justin Wong, Pieter Abbeel, and Joseph E. Gonzalez. The wisdom of hindsight makes language models better instruction followers, 2023. URL https://arxiv.org/abs/2302.05206.

Ziang Zhang, Zehan Wang, Guanghao Zhang, Weilong Dai, Yan Xia, Ziang Yan, Minjie Hong, and Zhou Zhao. Dsi-bench: A benchmark for dynamic spatial intelligence, 2025. URL https://arxiv.org/abs/2510.18873.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Ma, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*, 2024.

Yaowei Zheng, Junting Lu, Shenzhi Wang, Zhangchi Feng, Dongdong Kuang, Yuwen Xiong, and Richong Zhang. Easyr1: An efficient, scalable, multi-modality rl training framework. https://github.com/hiyouga/EasyR1, 2025.

Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. URL http://arxiv.org/abs/1801.09847.

Appendix

Table of Contents

A. VIEWSUITE Details

A.1. Action Space Details

As described in Section 2.2, VIEWSUITE provides 12 camera actions (Figure 1a). Table 6 lists all actions with their geometric definitions. The camera coordinate frame follows the OpenCV convention: $+X$ is screen-right, $+Y$ is screen-down, and $+Z$ points into the scene (forward). The world coordinate frame uses the ScanNet convention with Z-up.

Translation actions. The six translation actions move the camera center along its local axes by $s_t = 0.5\,\text{m}$ per step. `move_forward` / `move_backward` translate along the camera's $+Z$ / $-Z$ axis; `move_left` / `move_right` translate along $-X$ / $+X$; `move_up` / `move_down` translate along the screen-up / screen-down direction ($-Y$ / $+Y$ under the OpenCV convention).

Rotation actions. The six rotation actions rotate the camera about its center by $s_r = 30^\circ$ per step. `turn_left` / `turn_right` apply yaw (rotation about the camera's local Y axis); `look_up` / `look_down` apply pitch (rotation about the local X axis); `rotate_ccw` / `rotate_cw` apply roll (rotation about the local Z axis), producing on-screen counter-clockwise / clockwise content rotation.

Discrete snapping. In discrete mode, after each rotation action the camera-to-world rotation matrix is decomposed into intrinsic XYZ Euler angles, each angle is snapped to the nearest multiple of s_r, and the rotation matrix is recomposed. This ensures that camera orientations remain on a regular grid, making action sequences exactly invertible.

Table 6 | Detailed action definitions. All rotations are about the camera center in local coordinates.

Action	Type	Axis	Step size
`move_forward`	Translation	Camera $+Z$	0.5 m
`move_backward`	Translation	Camera $-Z$	0.5 m
`move_left`	Translation	Camera $-X$	0.5 m
`move_right`	Translation	Camera $+X$	0.5 m
`move_up`	Translation	Screen up ($-Y$)	0.5 m
`move_down`	Translation	Screen down ($+Y$)	0.5 m
`turn_left`	Rotation	Yaw (local Y, $-$)	30°
`turn_right`	Rotation	Yaw (local Y, $+$)	30°
`look_up`	Rotation	Pitch (local X, $+$)	30°
`look_down`	Rotation	Pitch (local X, $-$)	30°
`rotate_ccw`	Rotation	Roll (local Z, $-$)	30°
`rotate_cw`	Rotation	Roll (local Z, $+$)	30°

A.2. Data Sampling and Filtering Pipeline

Algorithm 1 gives the pseudocode; Table 7 lists all hyperparameters. Below we describe the four main stages.

Frame sampling. The temporal gap $\delta = f_{\text{tgt}} - f_{\text{init}}$ between initial and target frames is drawn from a mixture over three ranges: $\delta \in [50, 99]$ with weight 0.3, $\delta \in [100, 300]$ with weight 0.5, and the remaining frame indices uniformly with weight 0.2. The heavier weight on larger gaps means most pairs involve substantial viewpoint changes, though some nearby pairs are included as well.

Action planning. Given a sampled pair, we plan an action sequence from the initial to the target viewpoint using a greedy, rotation-first strategy (Algorithm 2). The six axes are processed in a fixed order (yaw, pitch, roll, forward, right, up); for each axis we try all step counts up to a maximum and pick the one that most reduces viewpoint error, then commit those steps before moving on. Because the planning is deterministic and single-pass, it consistently produces short sequences. Pairs whose sequence length falls outside $[2, 10]$ are discarded.

Distractor generation. For the multiple-choice P2V and V2P tasks, we create K=3 distractors per pair by perturbing the ground-truth action sequence. At $\lceil 0.3 \cdot \ell \rceil$ randomly chosen positions we apply one of three operations (replace with prob. 0.6, remove 0.2, insert 0.2); replacements favor the same motion category with prob. 0.7. Each perturbed sequence is executed and rendered, and we reject any distractor whose mean pixel difference from every existing option is below 0.02.

Scene-level filtering. In addition to the per-pair filters above (viewpoint identity, sequence length), we apply scene-level quality filtering based on the top-down reference view. We first use a vision LLM to classify each scene's top-down view as *good* (clear room layout visible from above, floor plan discernible) or *bad* (mostly occluded by ceiling or other geometry, layout not discernible), using 12 few-shot examples (6 good, 6 bad). The automated labels are then manually verified. All view pairs from scenes classified as bad are removed from the dataset. This filtering step removes scenes where the top-down view provides little useful spatial context, which would make the benchmark tasks ill-defined.

A.3. Success Threshold Calibration

We calibrate the threshold multipliers β_t and β_r in the success criterion of Section 2 via a small human alignment study. For each rollout we render the agent's submitted answer viewpoint with the same renderer and intrinsics used at evaluation time, and present it side by side with the ground-truth target view to expert annotators who judge whether the two views depict the same place (*match*) or not. We then sweep (β_t, β_r) over translation thresholds {0.25, 0.5, 0.75, 1.0} m and rotation thresholds {30°, 60°, 90°}, treating the threshold-based success indicator at each setting as a binary classifier of the human label. Table 8 reports precision, recall, F1, and accuracy across the resulting 4×3 grid. The combination 0.5 m and 30°, equivalent to $(\beta_t, \beta_r) = (1, 1)$ given the discrete step sizes $s_t = 0.5$ m and $s_r = 30°$, achieves the highest F1 (0.915) and accuracy (0.920) and is therefore adopted as the default success criterion throughout the paper. Loosening either threshold further keeps recall essentially saturated but degrades precision sharply: with $\beta_t = 2$ (1 m), precision drops to 0.72 at 30° and below 0.6 at 60°, indicating that the human annotators consider many such viewpoints visibly different despite their proximity in pose space.

Algorithm 1 VIEWSUITE data construction pipeline

Require: Point cloud $\mathcal{P}$; video frames with viewpoints $\{(f_i, P_i)\}_{i=1}^{N}$; delta distribution $\mathcal{D}$; length bounds $[\ell_{\min}, \ell_{\max}]$; num. distractors K; pixel threshold τ

$V_{\text{top}} \leftarrow \text{RENDERTOPDOWN}(\mathcal{P})$
for $n = 1, \ldots, N_{\text{pairs}}$ **do**
  Sample $\delta \sim \mathcal{D}$; sample f_{init}; $f_{\text{tgt}} \leftarrow f_{\text{init}} + \delta$
  $P_{\text{init}}, P_{\text{tgt}} \leftarrow$ viewpoints at $f_{\text{init}}, f_{\text{tgt}}$
  if $P_{\text{init}} \approx P_{\text{tgt}}$ **then skip**
  end if
  $\mathbf{a} \leftarrow \text{PLANACTIONS}(P_{\text{init}}, P_{\text{tgt}})$ ▷ Alg. 2
  if $|\mathbf{a}| \notin [\ell_{\min}, \ell_{\max}]$ **then skip**
  end if
  $P^*_{\text{tgt}} \leftarrow \text{EXECUTE}(P_{\text{init}}, \mathbf{a})$ ▷ Snap to discrete grid
  $V_{\text{init}} \leftarrow \text{RENDER}(\mathcal{P}, P_{\text{init}})$; $V_{\text{tgt}} \leftarrow \text{RENDER}(\mathcal{P}, P^*_{\text{tgt}})$
  $O \leftarrow \{(\mathbf{a}, V_{\text{tgt}})\}$ ▷ Options: GT first
  for $k = 1, \ldots, K$ **do** ▷ Generate distractors
    repeat
      $\hat{\mathbf{a}} \leftarrow \text{PERTURB}(\mathbf{a})$ ▷ Replace / remove / insert ops
      $\hat{V} \leftarrow \text{RENDER}\big(\mathcal{P}, \text{EXECUTE}(P_{\text{init}}, \hat{\mathbf{a}})\big)$
    until $\forall\, (\_, V') \in O :\ \text{PIXDIFF}(\hat{V}, V') > \tau$
    $O \leftarrow O \cup \{(\hat{\mathbf{a}}, \hat{V})\}$
  end for
  Emit P2V, V2P, IVP instances from $(V_{\text{init}}, V_{\text{top}}, O)$
end for

A.4. View Distance Distribution

Figure 5 shows the empirical distribution of the unified view distance d (Section 2) across the 530 test pairs. The distribution spans roughly 1.4 to 6.8 with mean 3.7, indicating that most pairs require several atomic actions to traverse rather than trivial single-step adjustments. We split the test set at $d = 3$ into a SHORT subset (185 pairs, $d < 3$) and a LONG subset (345 pairs, $d \geq 3$); these subsets are used in the difficulty-stratified analysis of Section 5.3. The threshold $d = 3$ corresponds to about three atomic-action units of separation between initial and target viewpoints, which empirically produces a clean visual divide between trajectories that can typically be solved with a few correct actions and those that demand sustained planning.

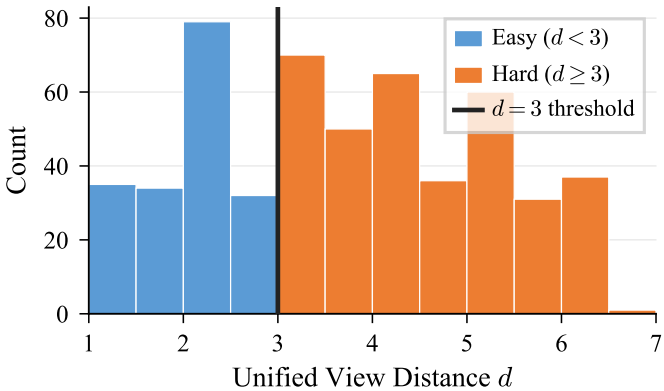


Figure 5 | Distribution of unified view distance d across 530 test pairs. The threshold $d = 3$ separates SHORT (185 pairs) and LONG (345 pairs) subsets.

Algorithm 2 Greedy rotation-first action planning

Require: Initial viewpoint P_{init}; target viewpoint P_{tgt}; axis order $\mathcal{A}$ = [yaw, pitch, roll, fwd, right, up]; max steps per axis $k_{\max}$
Ensure: Action sequence **a** such that $\textsc{Execute}(P_{\text{init}}, \mathbf{a}) \approx P_{\text{tgt}}$

$P_{\text{cur}} \leftarrow P_{\text{init}}$; $\mathbf{a} \leftarrow ()$
for each axis $(a^+, a^-) \in \mathcal{A}$ **do**
  $e_0 \leftarrow \textsc{PoseError}(P_{\text{cur}}, P_{\text{tgt}})$
  $k^* \leftarrow \underset{k \in [-k_{\max}, k_{\max}]}{\arg\min} \; \textsc{PoseError}\big(\textsc{Execute}(P_{\text{cur}}, k),\ P_{\text{tgt}}\big) + 0.01|k|$
  if $\textsc{PoseError}(\textsc{Execute}(P_{\text{cur}}, k^*), P_{\text{tgt}}) < e_0$ **then**
    Append $|k^*|$ copies of (a^+ if $k^* > 0$ else a^-) to **a**
    $P_{\text{cur}} \leftarrow \textsc{Execute}(P_{\text{cur}}, k^*)$
  end if
end for
return a

A.5. Task Examples

We show one example for each of the three tasks, including the system prompt and user prompt given to the model. Images are rendered from ScanNet point clouds. Placeholder `[image]` tokens mark where images are inserted in the multimodal input.

Path-to-View (P2V). Figure 6 shows a P2V instance. The full prompt is given below.

> **System:** You are a spatial reasoning agent. You are given a question and a set of images. You need to answer the question based on the images. You can think first, which is optional, then answer, respond in this format: `<think>...</think><action>answer(x)</action>` where x is A, B, C, or D.
>
> **User:** Given the initial view `[image]` and a top-down reference `[image]`, after you execute the following action sequence (translation step = 0.5 m; rotation step = 30.0 degrees per step): `[turn_right, turn_right, turn_right, turn_right, turn_right]`, which of the following images corresponds to the result? A. `[image]` B. `[image]` C. `[image]` D. `[image]`

GPT-5.4 Pro selects option C but the correct answer is a different option, illustrating that even strong models struggle with large cumulative rotations.

View-to-Path (V2P). Figure 7 shows a V2P instance. The full prompt is given below.

> **System:** You are a spatial reasoning agent. You are given a question and a set of images. You need to answer the question based on the images. You can think first, which is optional, then answer, respond in this format: `<think>...</think><action>answer(x)</action>` where x is A, B, C, or D.
>
> **User:** Given the initial view `[image]` and a top-down reference `[image]`, which action sequence will reach the target view `[image]`? (Action semantics: translation step = 0.5 m; rotation step = 30.0 degrees per step.) A. `[look_up, move_forward, move_left]` B. `[turn_left` $\times 5$, `move_left]` C. `[turn_right` $\times 2$, `move_forward, move_left` $\times 5$, `move_up]` D. `[turn_left` $\times 2$]

GPT-5.4 Pro correctly selects B, reasoning that the target view is behind the initial direction with reversed wall orientation, consistent with a large left rotation plus a lateral shift.

Table 7 | Data pipeline hyperparameters.

Hyperparameter	Value
Frame sampling	
$\delta \in [50, 99]$ weight / $[100, 300]$ weight / complement	0.3 / 0.5 / 0.2
Frame span (fraction of video)	[0, 1]
Action planning & filtering	
Axis order	Rot-first
Sequence length bounds $[\ell_{\min}, \ell_{\max}]$	[2, 10]
Max steps per axis (rotation / translation)	12 / 10
Distractor generation	
Num. distractors K	3
Perturb ratio $\lceil r \cdot \ell \rceil$	r=0.3
Op probs (replace / remove / insert)	0.6 / 0.2 / 0.2
Same-category replacement prob	0.7
Pixel uniqueness threshold τ	0.02
Max attempts per distractor	20
Limits	
Sampling attempts per pair	20
Timeout per pair	30 s

Interactive View Planning (IVP). Figure 8 shows an IVP instance solved by our trained Qwen2.5-VL-7B. The system prompt is given below (abridged; full action list omitted for space).

> **System:** You are solving an interactive view-planning viewpoint estimation task.
>
> GOAL: Predict the target view absolute viewpoint (camera-to-world, c2w) as a 6-DoF vector: [tx, ty, tz, rx, ry, rz]. You may explore the 3D scene using camera-control actions, then submit a final answer. Your predicted viewpoint should be as close as possible to the target viewpoint.
>
> TURN LIMIT: You must complete the task within 10 turns, including the final answer.
>
> OUTPUT FORMAT: `<think>...</think><action>action_1|action_2|...</action>`. The final response must contain exactly one `answer(tx, ty, tz, rx, ry, rz)`.
>
> **User:** You're in scene `scene0474_00`. Please study the target view `[image]`, the initial view `[image]`, and the top-down view `[image]`. You start from the initial view. Move toward the target view using actions. Initial view camera 6-DoF: [tx=4.07, ty=3.28, tz=1.66, rx=−90°, ry=0°, rz=−120°]. Success thresholds: position error ≤ 0.5 m, rotation error ≤ 30°.

Over 6 turns, the agent executes: `turn_right` (step 1), `turn_right` × 2 (step 2), `turn_right`, `look_down` (step 3), `move_left` (step 4), `move_forward` (step 5), then submits a viewpoint estimate (step 6). The final viewpoint error is 0.061 m position and 0° rotation, well within the success threshold.

B. Extended Evaluation Results

B.1. Evaluation-Protocol Ablations on IVP

The IVP results in Table 2 and Table 4 follow our default evaluation protocol: at each turn the agent's rotation actions are rounded (*snapped*) to integer multiples of the discrete step size s_r=30°, and an episode succeeds only if the agent issues an explicit `submit` action while its pose

Table 8 | Success threshold calibration on IVP rollouts. Each row evaluates the threshold-based success indicator at one (position, rotation) threshold pair against the human label.

Position thr.	Rotation thr.	Precision	Recall	F1	Accuracy
0.25 m	30°	**1.000**	0.600	0.750	0.820
0.25 m	60°	0.931	0.600	0.730	0.800
0.25 m	90°	0.931	0.600	0.730	0.800
0.50 m	30°	0.878	0.956	**0.915**	**0.920**
0.50 m	60°	0.843	0.956	0.896	0.900
0.50 m	90°	0.843	0.956	0.896	0.900
0.75 m	30°	0.811	0.956	0.878	0.880
0.75 m	60°	0.694	0.956	0.804	0.790
0.75 m	90°	0.683	0.956	0.796	0.780
1.00 m	30°	0.721	**0.978**	0.830	0.820
1.00 m	60°	0.611	**0.978**	0.752	0.710
1.00 m	90°	0.587	**0.978**	0.733	0.680

lies within the unified-distance threshold of the target. To check that our findings do not hinge on these two protocol details, we re-evaluate our fully-trained models alongside two strong proprietary baselines (Gemini 3.1 Pro and GPT-5.4) under two relaxations:

- **No-Snap.** Per-step rotations are no longer rounded to step-size multiples; the raw rotation magnitudes emitted by the agent are executed as-is. This isolates whether the planning gains depend on the discrete action grid.
- **No-Submit.** The agent does not need to explicitly submit. An episode is counted as successful at the first turn its pose enters the unified-distance threshold of the target, similar to a pure "reach the goal" criterion used in standard navigation benchmarks.

Table 9 compares the default protocol against the two ablations on the Short / Long splits defined in Section 2. The ordering between models is preserved under all three protocols: our trained models continue to outperform Gemini 3.1 Pro and GPT-5.4 by a wide margin (e.g., 19.6 vs. 15.7 on No-Snap, 60.2 vs. 31.5 on No-Submit). Relative to the default protocol, No-Snap *lowers* overall success for every model—without rounding, per-step rotation residuals accumulate over the 10-turn horizon and the agent drifts off the on-grid pose distribution from which targets are drawn—whereas No-Submit *raises* it, since the success criterion no longer requires the agent to commit to a final answer and credits any turn at which its pose enters the unified-distance threshold. Across all three protocols the Qwen2.5-VL-7B backbone outperforms Qwen3-VL-8B (47.8 vs. 32.5 on Default, 19.6 vs. 18.5 on No-Snap, 60.2 vs. 48.3 on No-Submit), reinforcing that our framework's gains transfer to both backbones, with Qwen2.5-VL-7B being the stronger starting point on this benchmark.

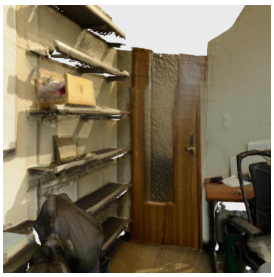
B.2. Sample-Level Factor Analysis

We compute Spearman ρ between 12 sample-level factors and per-model binary success across all three tasks (Figure 9; factor definitions in Appendix B.3). Factors span geometric distance, visual overlap (from pointcloud coverage), and directional geometry.

Across all tasks, distance factors show consistent negative correlations: farther view pairs are harder. For P2V and V2P, orientation agreement is the strongest positive predictor ($\rho \approx$ +0.19–+0.30), confirming that same-facing camera pairs are easier to reason about. For IVP, posi-

(a) Initial view

(b) Top-down view



(c) Option A

(d) Option B

(e) Option C

(f) Option D

Figure 6 | P2V example. Action: [`turn_right` × 5]. GPT-5.4 Pro selects C (incorrect).

tion distance dominates (ρ up to −0.42 for GPT-5.4 Pro), consistent with the position-bottleneck finding in Section 5.3. Visual overlap factors show mild positive correlations for P2V/V2P, indicating that shared visual content helps single-turn prediction. One notable outlier is Grok 4.20 Beta on V2P, which shows near-zero or slightly positive correlations with distance factors, suggesting a qualitatively different (and possibly less spatially grounded) reasoning strategy.

B.3. Success Factor Definitions

We define the 12 sample-level factors used in Figure 9. All factors are computed from the init and target camera-to-world extrinsics (4×4 matrices) shared across the 530 test view pairs. We write position $\mathbf{t} = C_{[:3,3]} \in \mathbb{R}^3$ (the translation column), rotation $R = C_{[:3,:3]} \in \mathbb{R}^{3\times3}$ (the rotation submatrix), and camera forward direction $\mathbf{f} = -R_{[:,2]} \in \mathbb{R}^3$ (negative z-axis of the camera frame, transformed to world coordinates).

Group A: Geometric Distance.

- `pos_dist` (d_{pos}): $\|\mathbf{t}_{\text{init}} - \mathbf{t}_{\text{target}}\|_2$ (meters). Euclidean distance between camera positions.
- `rot_dist` (d_{rot}): $\arccos\big(\text{clip}\big(\frac{\text{tr}(R_{\text{init}}^\top R_{\text{target}})-1}{2}, -1, 1\big)\big)$ (degrees). Geodesic angle between orientations.

(a) Initial view (b) Top-down view (c) Target view

Figure 7 | V2P example. GPT-5.4 Pro selects B (correct): [`turn_left` × 5, `move_left`].

Table 9 | IVP success rates (%) under the default protocol and two evaluation-protocol ablations. *Default*: per-step rotations are snapped to integer multiples of s_r=30°, and success requires an explicit `submit` within the unified-distance threshold; numbers are reproduced from Table 2 and Table 4. *No-Snap*: rotations are not snapped; raw rotation magnitudes are executed as-is. *No-Submit*: no explicit `submit` required; an episode is counted as successful as soon as its pose falls within the unified-distance threshold of the target. "Ours" denotes our fully-trained models (Section 5.2).

	Default			No-Snap			No-Submit		
Method	Short	Long	All	Short	Long	All	Short	Long	All
Gemini 3.1 Pro	28.8	17.4	21.4	27.0	9.6	15.7	49.7	21.7	31.5
GPT-5.4	33.7	7.5	16.6	27.6	5.2	13.0	57.3	17.4	31.3
Ours (Qwen2.5-VL-7B)	67.2	36.9	47.8	38.9	9.3	19.6	79.5	49.9	60.2
Ours (Qwen3-VL-8B)	56.8	19.4	32.5	41.6	6.1	18.5	80.5	31.0	48.3

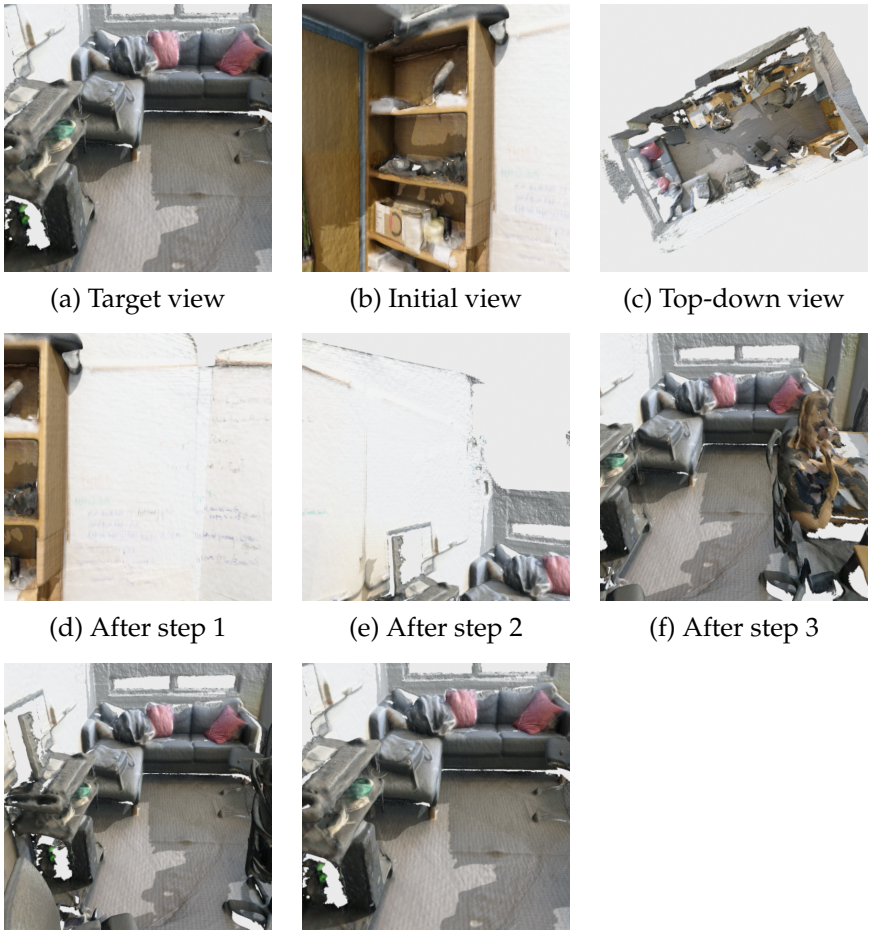
- `unified_dist`: $\sqrt{(d_{\text{pos}}/s_t)^2 + (d_{\text{rot}}/s_r)^2}$ (steps), where $s_t = 0.5\,\text{m}$ and $s_r = 30°$ are the discrete step sizes in ViewSuite. Equivalent to the unified view distance d defined in Section 2.
- `horiz_dist`: $\|\mathbf{t}^{xy}_{\text{init}} - \mathbf{t}^{xy}_{\text{target}}\|_2$ (meters). Horizontal distance, ignoring vertical displacement.
- `height_diff`: $|\mathbf{t}^{z}_{\text{init}} - \mathbf{t}^{z}_{\text{target}}|$ (meters). Absolute vertical difference.

Group B: Visual Overlap. Computed from GPU-rendered pointcloud coverage: for each viewpoint, we determine which mesh vertices are visible via depth rendering, yielding vertex sets V_{init} and V_{target}.

- `vis_init_norm`: $|V_{\text{init}} \cap V_{\text{target}}| \,/\, |V_{\text{init}}|$. Fraction of init-visible vertices also visible from target.
- `vis_target_norm`: $|V_{\text{init}} \cap V_{\text{target}}| \,/\, |V_{\text{target}}|$. Fraction of target-visible vertices already visible from init.
- `vis_iou`: $|V_{\text{init}} \cap V_{\text{target}}| \,/\, |V_{\text{init}} \cup V_{\text{target}}|$. Intersection-over-union of visible vertex sets.

Group C: Directional Geometry. Let $\hat{\mathbf{d}}$ denote the unit displacement vector from init to target position.

- `forward_alignment`: $\hat{\mathbf{f}}_{\text{init}} \cdot \hat{\mathbf{d}}$. Ranges from +1 (target ahead) to −1 (target behind).
- `target_bearing`: arccos(clip(`forward_alignment`, −1, 1)) (degrees). Angle between init



(a) Target view (b) Initial view (c) Top-down view

(d) After step 1 (e) After step 2 (f) After step 3

(g) After step 4 (h) After step 5 (final)

Figure 8 | IVP example. Our trained Qwen2.5-VL-7B plans view changes from (b) to match (a) in 6 turns. Final viewpoint error: 0.061 m / 0°. Success.

forward direction and displacement to target.

- `target_elevation`: atan2(Δz, $\|\Delta_{xy}\|$) (degrees). Vertical angle from init to target.
- `orientation_agreement`: $\hat{\mathbf{f}}_{\text{init}} \cdot \hat{\mathbf{f}}_{\text{target}}$. Cosine between camera forward directions. +1 = same facing, −1 = opposite.

C. Iterative Training Implementation Details

C.1. Algorithm

Algorithm 3 summarizes our iterative framework, alternating self-exploration with view graph distillation. Each iteration appends new trajectories to a persistent view graph, samples paths from it, and reformulates them via Eq. 4 into supervised view-planning demonstrations.

C.2. RL Hyperparameters

Table 10 lists the RL training hyperparameters used across all iterations of our framework and RL baselines. All methods use the same PPO configuration unless otherwise noted.

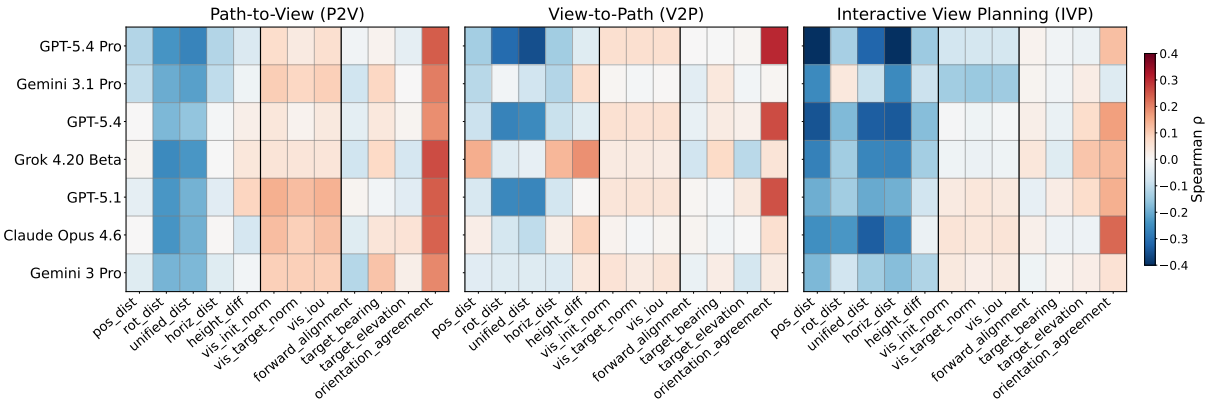


Figure 9 | Spearman ρ between sample-level factors (rows) and per-model success (columns). Factors grouped into geometric distance, visual overlap, and directional geometry.

Algorithm 3 Self-Exploration with View Graph Distillation

Require: initial policy π_{θ_0}, environments $\mathcal{E}$, iterations K

$G_0 \leftarrow \emptyset$ ▷ empty view graph

for $k = 0, 1, \ldots, K-1$ **do**

  Self-exploration stage:

    Run PPO updates of π_{θ_k} on $\mathcal{E}$ with reward Eq. 3

    Append trajectories: $G_{k+1} \leftarrow G_k \cup \text{traj}(\pi_{\theta_k})$

  View graph distillation stage:

    Sample paths $\{P_i\} \subset G_{k+1}$

    Reformulate: $\mathcal{D}_{k+1} \leftarrow \{\mathcal{R}(P_i)\}$ via Eq. 4

    Fine-tune via SFT: $\theta_{k+1} \leftarrow \arg\min_\theta \mathcal{L}_{\text{SFT}}(\theta; \mathcal{D}_{k+1})$

end for

return π_{θ_K}

Iteration-specific overrides. Iterations 0–2 use 60 RL training steps each for rapid bootstrapping. The final iteration (iter 3) is trained to convergence.

Direct GRPO baseline. The Direct GRPO (filter) baseline uses identical infrastructure but replaces GAE with the GRPO advantage estimator (Shao et al., 2024), sets n=4 rollouts per prompt for filtering, and trains for 1,000 steps.

C.3. SFT Hyperparameters

Table 11 lists the SFT hyperparameters used in our framework.

C.4. View Graph Construction

During RL exploration, a background process incrementally builds a view graph from completed trajectories. The graph builder runs concurrently with RL training and merges new trajectories into the graph.

Table 10 | RL training hyperparameters for our framework and baselines.

Hyperparameter	Value
Algorithm	
Advantage estimator	GAE
Actor	
Learning rate	1×10^{-6}
Mini batch size	128
Micro batch size per GPU	2
FSDP param offload	True
FSDP optimizer offload	True
Gradient checkpointing	True
Critic	
Learning rate	1×10^{-5}
Micro batch size per GPU	2
FSDP param offload	True
FSDP optimizer offload	True
Critic warmup steps	0
Rollout	
Engine	SGLang (async)
Max batched tokens	32,768
GPU memory utilization	0.6
Tensor parallel size	1
Data	
Max prompt length	4,000
Max response length	10,000
Train batch size	128
Infrastructure	
GPUs per node	8
Nodes	1

Node and edge representation. Each node stores a 6-DoF viewpoint (position + rotation) and its rendered view at 512×512 resolution. Each directed edge stores the sequence of camera actions taken between two viewpoints. Before adding a node, we apply image quality filters: frames with void fraction > 0.7 (indicating the camera is looking outside the point cloud) or pixel standard deviation < 10.0 (indicating a near-uniform, uninformative view) are discarded.

Deduplication. Nodes are deduplicated by viewpoint similarity: a new node is merged with an existing node if their position distance is below 0.25 m *and* rotation distance is below 15°. When two nodes are merged, all edges incident to the new node are redirected to the existing node. Edges are then deduplicated by (source, target, action sequence) identity. This deduplication prevents the graph from growing unboundedly as the same regions of the scene are revisited across episodes.

Cross-iteration accumulation. The graph is persisted to disk and accumulated across all self-exploration iterations. Later distillation stages sample from the full exploration history, not just the most recent iteration. This means that spatial knowledge discovered in early iterations (when the policy is weak) remains available for training in later iterations, even if the improved

Table 11 | SFT training hyperparameters for our framework.

Hyperparameter	Value
Training	
Learning rate	1×10^{-5}
Weight decay	0.01
LR scheduler	Cosine
Warmup ratio	0.1
Per-device batch size	2
Gradient accumulation steps	2
Cutoff length	16,384
Precision	BF16
Flash attention	FA2
Distributed strategy	DeepSpeed ZeRO-2
Epochs	
Iterations 0–2	3
Iteration 3 (final)	4
Model selection	
Validation split	20%
Eval strategy	Per epoch
Best model metric	Eval loss

policy explores different regions.

Table 12 shows the graph growth across iterations. The graph grows by an order of magnitude from iteration 0 to iteration 1 as the bootstrapped policy explores more effectively, then grows incrementally in iteration 2. Figure 10 shows how the action distribution shifts across iterations. In iteration 0, `move_forward` dominates (18.0%), reflecting the base policy's tendency to move straight ahead. By iteration 2, rotations (`turn_left`, `turn_right`) become the most frequent actions (~33% combined), and translations become more balanced across all six directions, indicating that the trained policy has learned more diverse exploration strategies.

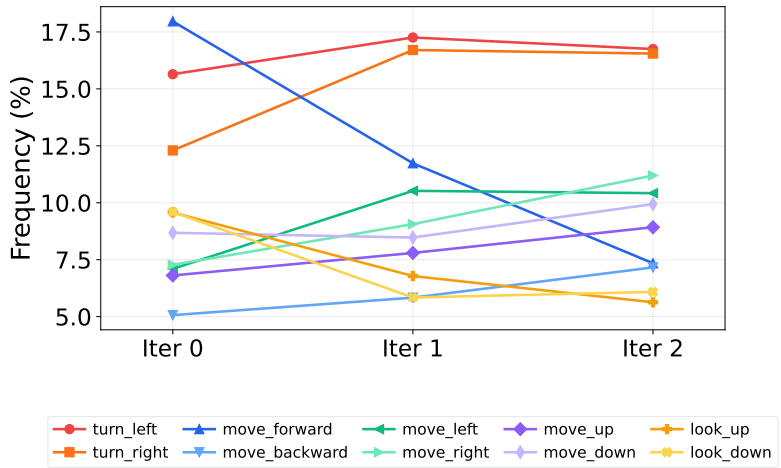


Figure 10 | Action frequency distribution across iterations. The base policy (iter 0) favors `move_forward`; later iterations shift toward rotations and more balanced translations.

Table 12 | View graph growth across iterations (iterations 0–2; the final iteration uses RL only without graph construction).

Iteration	Scenes	Nodes	Edges	Avg Nodes/Scene	Avg Actions/Edge
0	186	4,067	2,875	21.9	1.6
1	193	61,862	62,445	320.5	1.5
2	193	66,492	65,577	344.5	1.7

C.5. Task Reformulation Details

We generate three types of SFT tasks from paths sampled from the view graph. Table 13 summarizes the sampling parameters for each task type.

Table 13 | Task reformulation sampling parameters.

Task Type	Path Length	Samples/Scene	Balanced
Multi-turn view planning	3–5	20	No
View difference estimation	2–5	15	Yes
View difference MCQ	2–5	15	Yes

Multi-turn view planning (primary task). For a sampled path of length ℓ ($3 \leq \ell \leq 5$ edges), the end node is designated as the target view and the start node as the initial view. The intermediate nodes provide turn-by-turn observations. The model is trained to predict the correct camera action at each turn, given the current view, the target view, and the planning history. We oversample each path 10 times with different random seeds to increase diversity. This task directly trains the IVP capability.

View difference estimation. Given two views sampled from nodes at path distance ℓ ($2 \leq \ell \leq 5$), the model predicts unified view distance between them. This auxiliary task encourages the model to develop a sense of spatial distance between views, complementing the view-planning task. Balanced sampling ensures equal representation across path lengths.

View difference MCQ. Same setup as view difference estimation, but presented as a multiple-choice question with four options. This provides an alternative answer format, preventing the model from overfitting to a single task format during SFT.

Additional reformulations (not used in main experiments). The view graph is a task-agnostic representation, and the three tasks above are only the subset we use for training. The same graph naturally admits further reformulations. *Inverse dynamics*: given two views sampled as graph nodes, the model predicts the action sequence labeling the connecting edges. *Forward dynamics*: given an initial view and an action sequence, the model selects the resulting view from several candidate images. We include these to show that distilling structured spatial knowledge from the graph is not tied to goal-conditioned relabeling; studying their effect on training is left to future work.

C.6. Training and Validation Environments

Training. We use 3,419 ScanNet scenes for RL training. Each episode runs for up to 10 turns at 512×512 image resolution, rendered via a client-based point cloud renderer.

Validation. The validation set consists of 100 ScanNet scenes for IVP (10 turns) and 378 scenes each for P2V and V2P (single-turn, with extended response).

D. Extended Analysis

D.1. Point Cloud Coverage: Full Model Comparison

Figure 11 extends the coverage analysis from Section 5.3 to all 15 evaluated models. The pattern is consistent: our trained model is the only model that achieves sustained, monotonic growth in target intersection ratio across turns. Frontier proprietary models (GPT-5.4 Pro, Gemini 3.1 Pro, Gemini 3 Pro) show moderate initial increases but plateau or decline after turn 5–7, suggesting they explore broadly without maintaining target-directed trajectories. Open-weight models (Qwen3.5-397B, Qwen2.5-VL-72B, Qwen3-VL-32B) generally track below the proprietary models in both metrics.

Methodology. For each model, we collect 530 rollout trajectories (585 for GPT-5.4 Pro) on the VIEWSUITE test set. At each turn, we render the agent's viewpoint against the scene's 3D point cloud and compute the set of visible vertices using depth-buffered rendering. We track two metrics:

- **Scene coverage ratio**: $|\bigcup_{t=0}^{T} V_t| \,/\, |V_{\text{total}}|$, where V_t is the set of vertices visible at turn t and V_{total} is the full scene point cloud.
- **Target intersection ratio**: $|\bigcup_{t=0}^{T} V_t \cap V_{\text{target}}| \,/\, |V_{\text{target}}|$, where V_{target} is the set of vertices visible from the ground-truth target viewpoint.

All models are evaluated for up to 10 turns (turn 0 is the initial view). Some models produce fewer turns due to early stopping; turns with fewer than 1% of the maximum trajectory count are excluded. Lines show per-turn means; shaded regions indicate ±1 standard deviation.

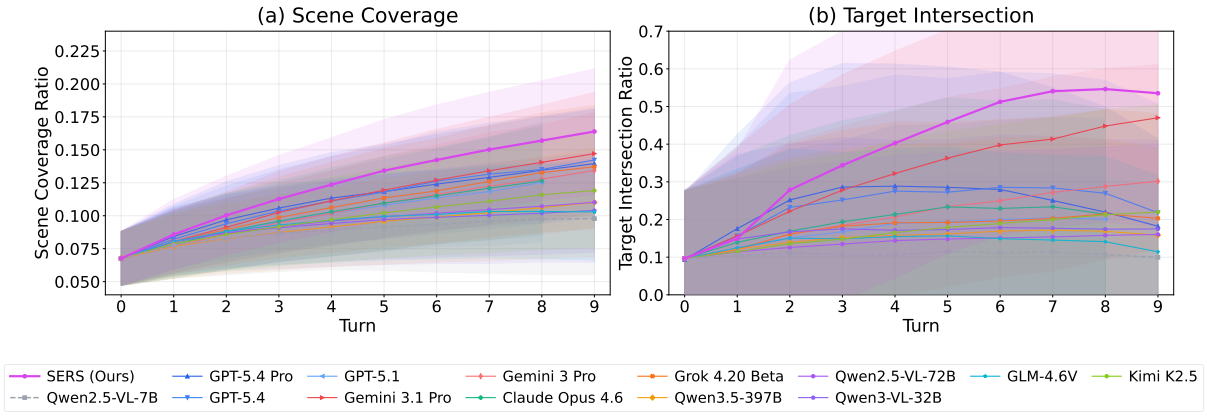


Figure 11 | Point cloud coverage across all 15 models. (a) Scene coverage ratio. (b) Target intersection ratio.

Turn distribution and success rate. Figure 12(a) shows the distribution of turns used per rollout. The base Qwen2.5-VL-7B-Instruct and GPT-5.4 Pro terminate most rollouts after a single turn (no planning action taken). Our trained model and Gemini 3.1 Pro concentrate at 10 turns (using all available turns). Figure 12(b,c) show successful rollouts and success rate by turn count. Our trained model achieves high success rates at low turn counts (efficient view planning on easy cases) with decreasing rates at higher turn counts (harder episodes).

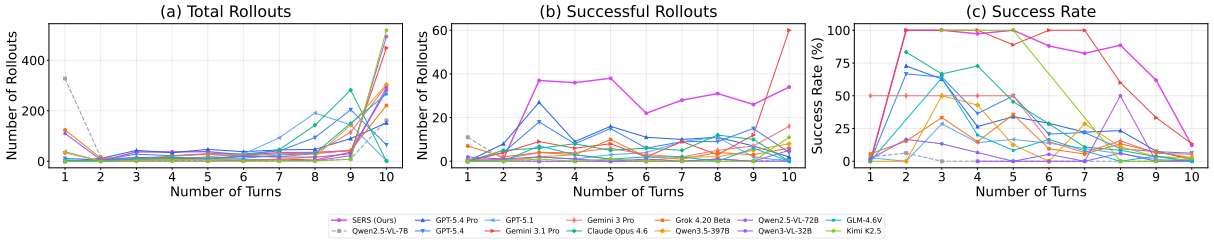


Figure 12 | (a) Total rollouts by turn count. (b) Successful rollouts. (c) Success rate.

D.2. Attention Analysis: Methodology and Full Results

We measure the **image attention fraction** (fraction of response-token attention directed toward image tokens) across uniformly spaced layers on the same 530 trajectories for both models. Figure 13 reveals two patterns. **Layer-wise**: our trained model allocates higher image attention in early layers (L0–L4), suggesting stronger visual grounding; in deeper layers (L8+), the base model maintains high image attention (peaking at L12–L16) while our trained model drops substantially, suggesting that it internalizes visual information early and transitions to text-space reasoning in later layers. **Turn-wise**: our trained model's image attention decreases monotonically across turns, consistent with progressive information accumulation, while the base model remains flat.

Setup. We run both our trained model and the base Qwen2.5-VL-7B on the same 530 trajectories. For each trajectory, we perform a forward pass and extract head-averaged attention matrices from selected layers. We identify image token positions via the processor's token-to-image mapping and compute the **image attention fraction**: for each response token at position q, we measure $\sum_{k \in \mathcal{I}} \alpha_{q,k} \,/\, \sum_k \alpha_{q,k}$, where $\mathcal{I}$ is the set of image token indices and $\alpha_{q,k}$ is the attention weight. We report this fraction averaged across response tokens within each turn. Trajectories with fewer than 3 images or no response turns are excluded. Error bars indicate ±1 standard deviation across trajectories.

Implementation. Qwen2.5-VL-7B has 28 transformer layers. To extract attention weights, we set target layers to "eager" attention mode (which produces explicit attention matrices) while other layers use the default SDPA implementation. Extracting all 28 layers simultaneously exceeds single-GPU memory, so we extract in 4 groups of 7 uniformly spaced layers: $\{0, 4, 8, 12, 16, 20, 24\}$, $\{1, 5, 9, 13, 17, 21, 25\}$, $\{2, 6, 10, 14, 18, 22, 26\}$, and $\{3, 7, 11, 15, 19, 23, 27\}$, each requiring a separate forward pass. Both models use identical eager configurations for each group to ensure fair comparison. The main text figure uses one group; the full 28-layer figure below combines all four groups.

Full layer-wise results. Figure 13 shows the image attention fraction for all 28 layers: our trained model shows elevated image attention in early layers relative to the base model, with a crossover in mid-layers where the base model's image attention substantially exceeds ours. Our trained model's monotonic turn-wise decrease is visible across nearly all layers, while the base model remains flat.

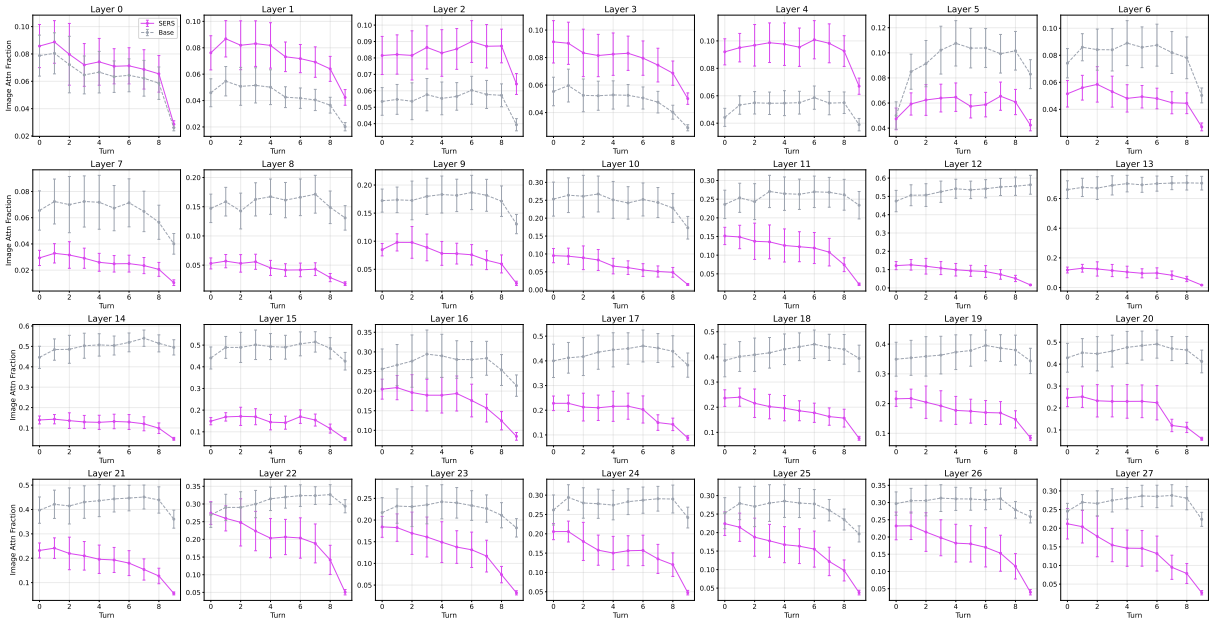


Figure 13 | Image attention fraction across all 28 layers. Both models evaluated on the same 530 trajectories.

D.3. Spatial Prior Transfer: Post-Training Details

For spatial prior transfer experiments (Section 5.3), we post-train both our trained model and the base Qwen2.5-VL-7B-Instruct using GRPO with identical hyperparameters on each task. We evaluate on three tasks:

- **P2V** and **V2P**: view-action understanding tasks from VIEWSUITE, requiring understanding of how the viewpoint changes under actions. These are trained jointly from the same data splits.
- **MindCube** (Wang et al., 2025b): mental rotation and spatial simulation, requiring the model to track object transformations across viewpoints.

Training hyperparameters. All tasks use GRPO with n=8 rollouts per prompt, actor learning rate 1×10^{-6}, critic learning rate 1×10^{-5}, and KL penalty disabled ($\lambda_{\text{KL}} = 0$). Training runs for 401 steps on 8 GPUs with FSDP and gradient checkpointing. Table 14 summarizes per-task differences.

Reward functions. All tasks use a binary reward composed of a format reward and an answer reward. The model must produce a valid response in the format `<think>...</think><action>answer(x)`. For P2V, V2P, and MindCube, the answer reward checks whether the predicted option letter matches the ground truth (case-insensitive first-character match). The reward weights are:

Table 14 | Per-task hyperparameters for downstream transfer post-training. All other settings are shared (see text).

	P2V / V2P	MindCube
Train batch size	64	32
Max prompt length	4,000	3,000
Max response length	2,000	4,000

- P2V / V2P: $r = 0.1 \cdot r_{\text{format}} + 0.9 \cdot r_{\text{answer}}$
- MindCube: $r = 0.2 \cdot r_{\text{format}} + 0.8 \cdot r_{\text{answer}}$

Both models are trained for the same number of steps on the same data to ensure a fair comparison of the spatial priors each model brings.