VLMs Control the View: Learning to Plan Camera Movements through Self-Exploration

Anonymous authors
Paper under double-blind review

Abstract

Can VLMs predict how camera movements change the view, and learn to choose those movements to find a target view? We study this capability as *view planning*, separating *tracking* a supplied path from *composing* one through interaction. We introduce ViewSuite, a benchmark with three tasks that pair supplied-path understanding and interactive camera control in real 3D scenes. Across 13 VLMs under our standard evaluation setting, the strongest models exceed 70% on short-distance tracking tasks but reach at most 21.3% on interactive view planning. Learning interactive view planning (IVP) is difficult when the base policy rarely succeeds: Qwen2.5-VL-7B begins at 2.5% success, and direct PPO and GRPO reach only 3.2% and 5.2% under the tested recipes. Our key observation is that every exploration trajectory, successful or not, records valid transitions between views. *View Graph Distillation* accumulates these transitions across episodes, merging nearby camera poses so that sampled paths can connect experience from different rollouts. Each path's endpoint becomes a new target, its recorded actions and intermediate views become a demonstration, and the same graph supplies auxiliary view-understanding supervision. Alternating this distillation with reinforcement learning improves Qwen2.5-VL-7B's IVP success from 2.5% to 47.7% on held-out scenes. The trained checkpoint is also a better starting point for related view-understanding tasks.

1 Introduction

To understand a 3D scene through interaction, an agent needs to recognize how camera movements change what it sees, choose where to look next, and connect the resulting observations to a target view. We study **View Planning**: choosing camera movements from visual feedback and identifying the pose corresponding to a target image. It couples two demands. *Tracking* asks how a supplied path changes the view, while *composing* asks the agent to construct a path by choosing actions and observing their outcomes. Intuitively, this is the difference between reading a path and writing one.

View planning is an instance of active perception (Bajcsy, 1988; Bajcsy et al., 2018), in which an agent chooses actions to obtain useful observations. Existing VLM benchmarks study this through searching within images (Wu & Xie, 2024; Wang et al., 2025e) and changing viewing direction in 360° panoramas (Yu et al., 2025). We study camera control in real 3D scenes, where the model can both translate and rotate, changing which surfaces are visible and how objects appear. The task shares target-image-conditioned action selection with image-goal navigation (Zhu et al., 2017). Here a general-purpose VLM receives an initial view, a target view, and a top-down reference, then chooses 6-DoF camera actions and reports the target pose. After each interaction, the environment supplies the resulting image and the current camera pose. The same VLM interprets the images, follows the action history, chooses movements, and produces the final estimate. This lets us study how pretrained multimodal capabilities can be developed into an interactive policy.

We build ViewSuite on 286 real ScanNet (Dai et al., 2017) indoor scenes, with approximately 55K view pairs and 165K task instances across three tasks. Path-to-View (P2V) and View-to-Path (V2P) supply the path, either in the question or among the answer choices, so the model must *read* how views and actions relate. Interactive View Planning (IVP) supplies no path: the model must *compose* one over multiple turns and submit a 6-DoF target-pose estimate.

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.

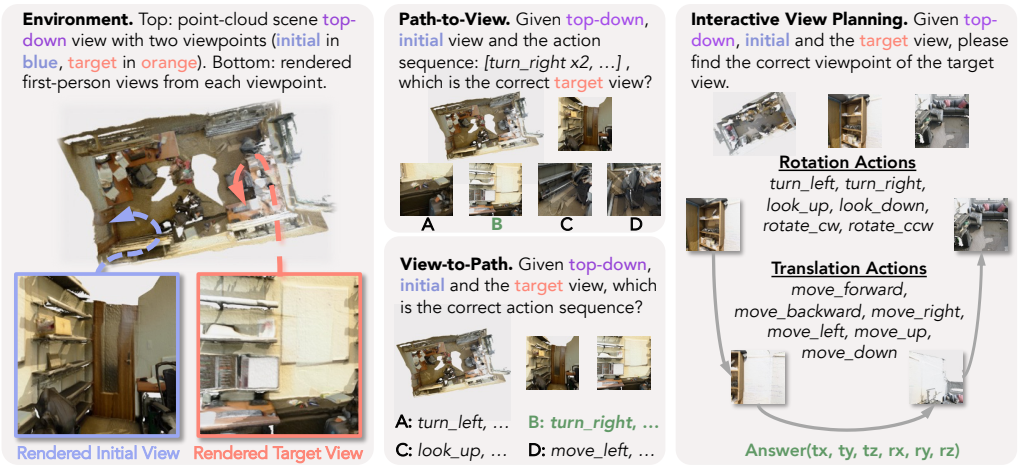

Figure 1: **The three VIEWSUITE tasks.** P2V selects the view produced by a supplied path, V2P selects the path connecting two views, and IVP chooses camera actions before reporting the target pose. Evaluating 13 VLMs under our standard evaluation setting reveals a sharp planning gap. The strongest models exceed 70% accuracy on short-distance P2V/V2P examples, but the best IVP result is 21.3%, most models remain below 10%, and every evaluated open-weight model remains below 5%. Viewpoint separation also affects the tasks differently: tracking degrades most with rotation, while IVP success falls most with position distance.

A natural way to learn interactive behavior is reinforcement learning (RL), yet the weak base policy makes reward difficult to reach. Qwen2.5-VL-7B (Bai et al., 2025a) begins at only 2.5% IVP success. Under the training recipes we test, direct PPO reaches 3.2%, direct GRPO with reward-variance filtering reaches 5.2%, and training only on successful trajectories reaches 6.2%. Our way past this bottleneck begins with a simple observation: every trajectory, successful or not, records valid transitions between views.

We introduce **View Graph Distillation** to turn those transitions into reusable supervision. A persistent graph stores the observed viewpoints and executed actions, merging nearby viewpoints so that paths can connect experience from different episodes. Each sampled path defines a new initial-target pair: its endpoint supplies the target image and pose, and the actions along the path supply a demonstration. The same graph also provides auxiliary tasks about differences between views. Distillation trains the VLM on these examples; RL then resumes on the original interactive goals, and the resulting exploration expands the graph for the next round. The supervision comes from the learner's movements and the environment's observations, poses, and geometry. This alternating procedure raises Qwen2.5-VL-7B from 2.5% to 47.7% on held-out scenes, with further gains on InternVL3-8B (Zhu et al., 2025) and in a separate AI2-THOR (Kolve et al., 2017) training run. The IVP-trained checkpoint also reaches higher accuracy after identical downstream fine-tuning.

Contributions. We make three contributions:

- **Benchmark construction.** We build VIEWSUITE with three tasks on shared view pairs, a common 6-DoF camera interface, and scene-disjoint data splits.
- **Systematic diagnosis of VLM view planning.** Evaluation of 13 VLMs measures how well they interpret supplied paths and construct paths through interaction. The results reveal a planning gap and show how rotation and position distance affect the two task types differently.
- **Learning from scene self-exploration.** We introduce View Graph Distillation to construct action demonstrations and view-understanding supervision from the VLM's own exploration. Alternating this supervision with RL improves interactive view planning on held-out scenes.

2 VIEWSUITE: TASKS, ENVIRONMENT, BENCHMARK

VIEWSUITE renders real ScanNet (Dai et al., 2017) indoor scenes and exposes a common camera-control interface across the three tasks illustrated in Figure 1. Table 1 compares its task design with related benchmarks. Appendix A.1 explains the comparison categories.

Table 1: **Benchmarks for view understanding and active perception.** VIEWSUITE pairs path understanding with interactive camera control in real scanned scenes.

Benchmark	Visual setting	Agent operations	Task
ViewSpatial-Bench (Li et al., 2025a)	Images	None	Spatial QA
VSI-Bench (Yang et al., 2025a)	Videos	None	Spatial QA
CameraBench (Lin et al., 2025)	Videos	None	Camera-motion QA
MindCube (Wang et al., 2025c)	Multi-view images	None	Spatial QA
V*Bench (Wu & Xie, 2024)	High-resolution images	Image-region search	Visual QA
ActiView (Wang et al., 2025e)	Images	Zoom and shift	Visual QA
H*Bench (Yu et al., 2025)	360° panoramas	Head rotation	Object/path search
HM3D-OVON (Yokoyama et al., 2024)	Scanned 3D scenes	Navigation	Object-goal navigation
EmbodiedBench (Yang et al., 2025b)	3D environments	Navigation/manipulation	Embodied tasks
Theory of Space (Zhang et al., 2026)	Simulated 3D scenes	Move, rotate, observe	Spatial-belief probing
VIEWSUITE (Ours)	Scanned 3D scenes	6-DoF camera control	P2V, V2P, IVP

2.1 TASK FORMULATION

We distinguish two demands. *Tracking* is recognizing how a supplied action sequence changes the view, while *composing* is constructing such a sequence through interaction. The three tasks measure these demands using the same initial-target view pairs.

Tracking supplied paths. In P2V, the model receives an initial view and a camera-action sequence, then selects the resulting view from four candidate images (Figure 6). In V2P, it receives the initial and target views, then selects the action sequence that connects them from the supplied candidates (Figure 7). Both tasks ask the model to interpret a path that is already given, either in the question or among the answer choices.

Composing a path through interaction. In IVP, the model receives the initial view, target image, and top-down reference, and chooses camera actions over multiple turns. After each turn, it receives the resulting image and current camera pose. It uses these observations to choose further actions and eventually submits a 6-DoF target-pose estimate (Figure 8). Moving to a view that matches the target and reporting the current camera pose is one valid way to complete the task.

2.2 ENVIRONMENT AND DATA

The interface provides 12 camera-movement primitives, covering translations along and rotations about the three local camera axes. Translation and rotation step sizes are 0.5 m and $30°$, respectively. Each interaction turn can contain up to four actions, followed by one observation of the resulting view and camera pose. The 10-turn budget includes the final answer. We use fixed action increments to give all three tasks a shared, reproducible camera-control interface. Discrete movement primitives also appear in AI2-THOR (Kolve et al., 2017) and Habitat navigation (Savva et al., 2019). Our default interface additionally uses rotation rounding during execution and renders observations with Open3D (Zhou et al., 2018). Full geometric definitions are given in Appendix A.2.

We construct examples from initial-target view pairs sampled from ScanNet video frames. After filtering scenes and pairs, we execute the generated action sequence and use its resulting pose as the target. Targets are therefore reachable using the benchmark's camera actions; Appendix A.3 describes how pairs are sampled and converted into action sequences. The constructed corpus contains approximately 55K view pairs across 286 scenes, corresponding to 165K task instances. Our main ScanNet experiments use the VIEWSUITE-5K subset, which contains 4,327 pairs and approximately 13K task instances; the full corpus is reserved for future scaling studies. Scene-disjoint train/dev/test splits prevent scene overlap, and the test set contains 530 view pairs, each used for all three tasks. Appendix A.1 gives the split sizes; the supplementary graph comparison uses the separate split described in Appendix C.10.

2.3 EVALUATION METRICS

P2V and V2P use multiple-choice accuracy. IVP success requires the submitted target pose to be within 0.5 m and $30°$ of the target, including the threshold boundaries. In a pilot study, these thresholds agree with human same-place judgments at $F_1{=}0.915$ (Appendix A.4). For analysis, a

Table 2: **Planning gap in frontier VLMs.** P2V/V2P accuracy and IVP success (%).

	P2V			V2P			IVP			
Model	Short	Long	All	Short	Long	All	Short	Long	All	Overall
Random Response	20.4	24.8	23.3	24.1	26.6	25.7	2.1	0.0	0.8	16.6
Proprietary Models										
GPT-5.4 Pro[1]	**70.6**	**43.7**	**53.2**	**72.7**	38.5	**50.6**	**34.8**	10.5	19.1	**41.0**
Gemini 3.1 Pro	63.1	41.1	48.9	52.9	**47.5**	49.4	28.9	**17.2**	**21.3**	39.9
GPT-5.4	57.8	42.6	47.9	59.9	37.6	45.5	33.2	7.6	16.6	36.7
Grok 4.20 Beta	61.0	38.2	46.2	44.4	44.6	44.5	17.1	2.9	7.9	32.9
GPT-5.1	59.9	35.3	44.0	51.9	33.5	40.0	11.8	3.2	6.2	30.1
Claude Opus 4.6	46.5	28.3	34.7	47.1	38.5	41.5	23.5	3.8	10.8	29.0
Gemini 3 Pro	50.3	30.9	37.7	44.4	35.6	38.7	13.9	6.7	9.2	28.5
Open-Weight Models										
Qwen3.5-397B	**57.8**	30.0	**39.8**	**43.9**	**31.2**	**35.7**	**12.3**	0.0	**4.3**	**26.6**
GLM-4.6V	36.0	23.3	27.8	31.6	29.5	30.2	9.1	**1.2**	4.0	20.7
Qwen2.5-VL-72B	28.3	29.2	28.9	35.8	30.0	32.1	2.1	0.6	1.1	20.7
Qwen3-VL-32B	26.7	27.7	27.4	40.6	28.9	33.0	4.3	0.0	1.5	20.6
Kimi K2.5	36.9	24.2	28.7	18.2	29.4	25.5	4.8	**1.2**	2.5	18.9
Qwen2.5-VL-7B	24.1	**32.4**	29.4	27.3	22.7	24.3	7.0	0.0	2.5	18.7

normalized viewpoint distance measured in action-step units defines SHORT ($d < 3$, 187 pairs) and LONG ($d \geq 3$, 343 pairs); Appendix B.6 gives the geometric definitions. Additional environments and the InternVL analysis use the split definitions stated in their respective tables.

3 THE PLANNING GAP IN FRONTIER VLMS

3.1 TRACKING SUPPLIED PATHS, COMPOSING INTERACTIVE PATHS

We evaluate 13 VLMs (7 proprietary and 6 open-weight) (OpenAI, 2026a;b; Google DeepMind, 2025; 2026; Anthropic, 2026; xAI, 2026; Bai et al., 2025a;b; Qwen Team, 2026; Hong et al., 2025; Kimi Team, 2026) without task-specific training, using ReAct-style prompts and *medium* reasoning effort when available. Task prompts and reasoning settings appear in Appendices A.6 and B.7. The central finding is a planning gap: frontier VLMs can track short supplied paths, but struggle when they must compose camera actions through interaction. On P2V and V2P, the best models achieve approximately 50% overall and exceed 70% on short-distance examples, well above the 25% multiple-choice chance rate. On IVP, the strongest model reaches only 21.3%, most models remain below 10%, and every evaluated open-weight model remains below 5%. Qwen2.5-VL-7B, the primary training backbone, obtains 29.4% on P2V, 24.3% on V2P, and only 2.5% on IVP. We use *planning gap* for this contrast between recognizing supplied paths and choosing actions through interaction. The multiple-choice and interactive tasks provide complementary measurements, each with its own answer format and success criterion.

3.2 WHAT MAKES INTERACTIVE VIEW PLANNING DIFFICULT?

Do additional turns, sharper rendering, or more reasoning help? Increasing the turn budget from 10 to 20 improves all four proprietary models we test, but gains from 20 to 30 turns are small or absent for most models. More turns help, but not enough to close the gap. Re-rendering the same scenes with higher-fidelity 3D Gaussian Splatting changes IVP by only $+0.2$ to $+1.9$ points. Increasing reasoning effort has a larger effect: GPT-5.4 rises from 4.2% without reasoning to 16.6% at medium and 19.6% at high effort, while Gemini 3.1 Pro rises from 13.2% to 21.3% and 26.9%. In these evaluations, sharper pixels help little; more thinking helps more. Full results appear in Appendices B.1 and B.7.

Does rotation or position drive difficulty? Figure 2 reveals contrasting difficulty profiles. P2V and V2P decline most across rotation-distance bins, consistent with the difficulty of tracking accumulated changes in orientation. IVP declines most with position distance: for GPT-5.4 Pro, success falls by approximately $7\times$ from the nearest to the farthest position bin. Reading a supplied path is

[1]GPT-5.4 Pro refuses 23 of the 530 IVP test instances under its violation policy. We count these refusals as failures, giving $101/530{=}19.1\%$ overall IVP success.

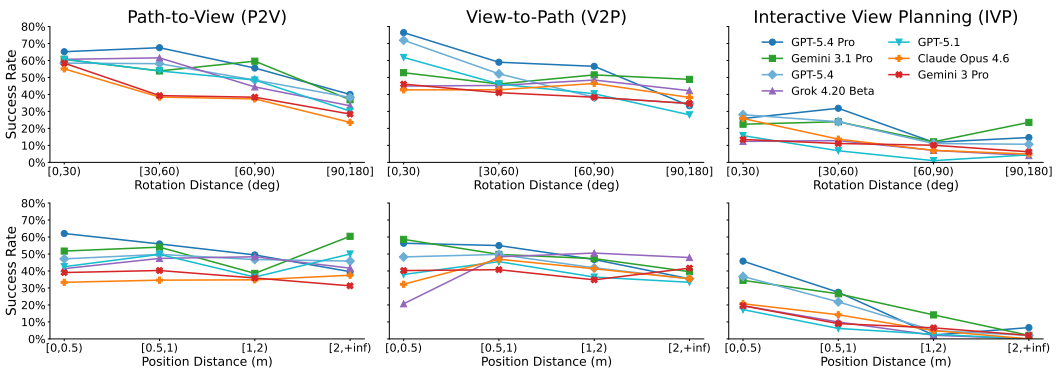

Figure 2: **Performance by rotation and position distance.** P2V/V2P decline most with rotation distance, while IVP declines most with position distance.

therefore most sensitive to rotation separation in these examples, while composing an interaction becomes especially difficult as the target moves farther away in 3D position. Appendix B.5 reports complementary sample-level correlations.

Most successes include an observation near the target pose. For Gemini 3.1 Pro and GPT-5.4 Pro, respectively, 96.5% and 94.1% of successful IVP episodes include an observed camera pose within $0.5\,\text{m}$ and 30° of the target before the final answer. The corresponding share for our trained Qwen2.5-VL-7B is 98.0%. This count includes the initial observation and the observation at the end of each turn, but not intermediate poses inside a multi-action turn. The pose logs show which views were available to the model, but do not establish how it produced the final answer. Appendix B.3 reports the counts for each model. The same planning gap appears on Habitat-GS, including outdoor plazas with metre-scale translation steps (Appendix F.2).

4 Self-Exploration with View Graph Distillation

We train a general-purpose VLM to choose camera actions and interpret the views they produce. Direct RL starts from a base policy with only 2.5% IVP success, making task-success rewards difficult to obtain. Each exploration episode records images, camera poses, and the actions connecting them, even when the original target is not reached. View Graph Distillation organizes this experience into camera-action demonstrations and auxiliary view-understanding tasks. We alternate learning from this supervision with RL on the original IVP goals. Appendix G explains the scope of the comparisons.

4.1 Learning from Self-Exploration

We accumulate exploration transitions in a persistent *view graph*. Each node stores a camera pose and its rendered image, and each directed edge stores the camera-action sequence executed between two observed viewpoints. When episodes reach nearby viewpoints, we merge those nodes and retain their incoming and outgoing action edges. The graph persists across rounds, so later distillation reuses both recent and earlier exploration. Appendix C.5 specifies node merging, image filtering, and graph accumulation.

We replay sampled graph paths using the environment's action model to check whether their recorded actions reach the stored target poses. Across the three collection rounds, 97.6%, 99.9%, and 99.0% of sampled paths end within the benchmark's position and rotation thresholds (Appendix C.5).

We alternate two stages (Figure 3). During self-exploration, the policy interacts with the original IVP environments and is updated with PPO using the task reward. Completed trajectories are added to the graph throughout these RL updates. During View Graph Distillation, we sample graph paths, turn them into supervised training examples, and fine-tune the policy on those examples. The resulting policy then resumes RL and contributes new exploration experience to the graph. After the alternating rounds, we continue PPO training without further distillation. The reward and full algorithm are specified in Appendices C.1 and C.2.

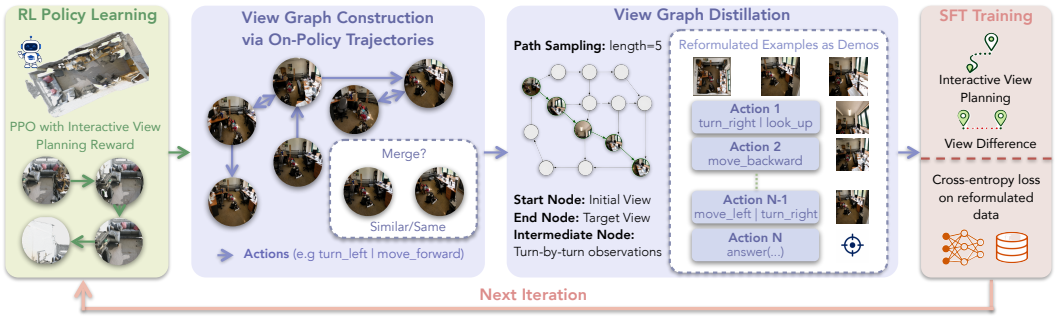

Figure 3: **Alternating self-exploration, graph distillation, and RL.** Exploration expands a persistent view graph; sampled paths become action and view-difference supervision before RL resumes.

View graph distillation via task reformulation. For a sampled path $P = (v_0, a_1, v_1, \ldots, a_K, v_K)$, task reformulation produces

$$\mathcal{R}(P) = (o_{\text{init}} = o_{v_0},\ o_{\text{target}} = o_{v_K},\ (a_1, \ldots, a_K),\ \hat{p}^* = p_{v_K})\,. \tag{1}$$

Here, o_v and p_v are the image and camera pose stored at node v, and each a_i is the recorded camera-action sequence on an edge. We call this *any-view-to-any-view* supervision: the endpoints of each sampled graph path define a new initial-target pair, regardless of the goals assigned during collection. Merging repeated viewpoints lets a sampled path use transitions from different episodes, such as a prefix from one rollout and a continuation from another. At each turn, the model receives the corresponding observation and is trained to predict the recorded actions; the endpoint pose provides supervision for its final pose report. From the same graph we also construct two auxiliary tasks that ask the model to estimate the distance between sampled views, either directly or as a multiple-choice answer. The graph therefore supports both action demonstrations and view-understanding supervision from the same exploration experience. The training examples come from the learner's interaction with the scene, without action demonstrations from a stronger model. Appendix C.6 gives the path-sampling settings and example construction for each format. At inference, the VLM selects actions from its interaction history without access to the training graph.

In a separate controlled study with Qwen3-VL-8B, we measure how graph-based sampling changes the training data constructed from the same exploration trajectories. We compare the subset of 1,024 demonstrations per method that use a view reached along the path as their target. Sampling shortest paths through the graph yields 147 examples with view distance $d \geq 6$, compared with 3 when paths are sampled within individual trajectories (Appendix C.10).

5 Training VLMs for Interactive View Planning

5.1 Experimental Setup

We train Qwen2.5-VL-7B-Instruct (Bai et al., 2025a) on 3,419 ViewSuite initial-target pairs, with at most 10 interaction turns per episode. Iterations 0 to 2 each use 60 RL steps followed by 3 epochs of View Graph Distillation. After these alternating rounds, a final RL stage continues training without further distillation. Implementation details appear in Appendix C: optimization settings and distillation checkpoint selection in Appendices C.3 and C.4, and training and validation sets in Appendix C.7.

The training recipes use the same base checkpoint, environments, reward, and action interface. **Direct PPO** tests learning from task rewards without distilled supervision. **Direct GRPO (filter)** also omits distillation and uses four rollouts per prompt with reward-variance filtering (Shao et al., 2024; Wang et al., 2025d), testing whether grouped rollouts make direct RL more effective. **Success-Only Distillation** alternates PPO and supervised training but retains only trajectories that complete their original targets. It tests whether those successful rollouts alone provide sufficient demonstrations. **Random-graph** constructs distillation data from random-action exploration to test the choice of collection policy. The iteration ablations stop the alternating procedure after fewer rounds to examine how performance changes with repeated collection and training. Checkpoints are selected using validation data. Table 3 reports single training runs. Appendix C.3 records the training schedules.

5.2 MAKING REWARD REACHABLE

The complete procedure raises Qwen2.5-VL-7B success from 2.5% to 47.7% on held-out scenes (253/530 episodes; Table 3). Under their reported schedules, Direct PPO, Direct GRPO, and Success-Only Distillation reach 3.2%, 5.2%, and 6.2%, respectively. The alternating procedure learns from paths collected even when their original goals are missed: their endpoints provide targets for supervised training before RL resumes on the original goals.

How exploration becomes training data. Using random-action exploration to build the graph yields 13.0% success, compared with 47.7% for policy-generated exploration. A plausible explanation is that policy-generated exploration supplies demonstrations more relevant to the learner's behavior. Each collection round refreshes this experience as the policy changes, following the learner-generated data principle used in DAgger (Ross et al., 2011) and on-policy distillation (Agarwal et al., 2024). Runs with one, two, and three rounds followed by RL reach 12.0%, 27.9%, and 47.7%. Across the first three collection rounds, the graph grows from 4,067 to 66,492 nodes (Appendix C.5). Its action mix also changes: the share of `move_forward` falls from 18.0% in round 0 to 7.3% in round 2, while backward, lateral, and vertical translations occupy larger shares (Figure 10).

Table 3: **Training a weak VLM policy.** IVP success on held-out scenes (%).

Method	Short	Long	All
Base and zero-shot references			
Qwen2.5-VL-7B	7.0	0.0	2.5
GPT-5.4 Pro	34.8	10.5	19.1
Gemini 3.1 Pro	28.9	17.2	21.3
Training Baselines			
Direct PPO	7.0	1.2	3.2
Direct GRPO (filter)	10.8	2.2	5.2
Success-only distillation	14.0	2.0	6.2
Training Ablations			
No-graph relabel + RL	64.7	29.2	41.7
Random-graph	25.7	6.1	13.0
1 iter + RL	24.3	5.4	12.0
2 iter + RL	49.7	16.0	27.9
Our Method			
Qwen2.5-VL-7B	66.8	37.3	47.7

The training framework also improves InternVL3-8B on VIEWSUITE and Qwen2.5-VL-7B when trained separately on AI2-THOR scenes (Table 4). The reported schedules differ across settings; implementation and evaluation details appear in Appendices C.11 and F.1, respectively.

Table 4: **Training gains across backbones and environments on IVP success (%)**

Environment	Backbone	Training schedule	Base	Trained	Best zero-shot
VIEWSUITE	Qwen2.5-VL-7B	3×60 RL steps then to convergence	2.5	**47.7**	21.3 (Gemini 3.1 Pro)
VIEWSUITE	InternVL3-8B	400-step RL stage	0.6	20.2	21.3 (Gemini 3.1 Pro)
AI2-THOR	Qwen2.5-VL-7B	3 iterations, no final RL stage	5.6	**61.1**	36.1 (GPT-5.4)

What is distilled? We compare supervision formats after one collection-and-distillation round in a reduced-budget experiment. Appendix C.8 gives the validation protocol and training settings. Training only on interactive planning reduces P2V accuracy from 30.0% to 14.0% and V2P accuracy from 27.1% to 11.1%. The auxiliary tasks ask the model to estimate the difference between two views, either directly or by choosing among candidate answers. Adding both formats brings P2V accuracy to 28.0% and V2P accuracy to 29.0% (Table 5). These tasks help retain single-turn view understanding during distillation. After the final RL stage, P2V and V2P accuracy under the main test protocol is 6.0% and 20.9%, respectively. The final model often responds to the multiple-choice prompts in the IVP action format (Appendix C.8).

Table 5: **Distilled-data ablations.** Validation accuracy and IVP success (%).

Supervision	P2V	V2P	IVP
Base	30.0	27.1	0.0
Planning only	14.0	11.1	3.0
+ Difference estimate	27.2	28.8	**9.0**
+ Difference MCQ	27.0	23.5	8.0
+ Both	**28.0**	**29.0**	8.0

Graph-based supervision versus trajectory-level relabeling. We compare the two ways of constructing supervision through three alternating rounds and the final RL stage. The trajectory-level baseline uses all collected trajectories, including unsuccessful ones, and treats a reached viewpoint as the new goal for distillation. Both variants use the same number of demonstrations and the same two auxiliary supervision types. On the 530 test pairs, trajectory-level relabeling reaches 41.7%, compared with 47.7% using the graph. The difference is larger on Long targets (29.2% versus

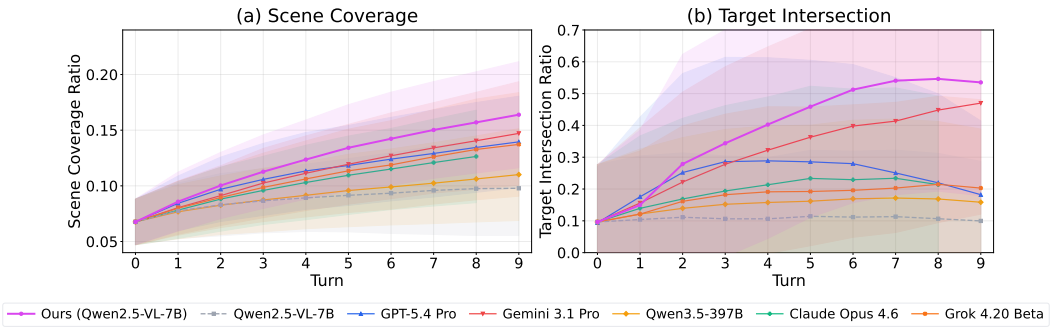

Figure 4: **Scene coverage and target overlap.** Lines average the cumulative fractions of scene points and target-view points observed, using episodes with an observation at each turn. Shading indicates one standard deviation.

37.3%) than on Short targets (64.7% versus 66.8%). Paths through the graph can connect transitions from different episodes, providing demonstrations between more distant views. Appendix C.9 gives the comparison settings, and Appendix C.10 measures how graph-based sampling changes the distribution of training paths.

Execution-protocol sensitivity of the trained policy. No-Snap removes rotation rounding at test time while leaving the success criterion unchanged. In the original paired runs, our model's success falls from 47.8% to 19.6%. Appendix B.4 provides the execution rules and full comparison.

No-Submit keeps the default action dynamics and changes what is scored: an episode succeeds if an observed camera pose falls within the target thresholds, whereas the default metric evaluates the pose estimate submitted by the model. Our model reaches 57.5% on this measure, compared with 32.8% for Gemini 3.1 Pro and 26.0% for GPT-5.4. Of the targets it reaches, 18.7% are not followed by a correct submission, compared with 37.4% for Gemini. Thus, the trained policy reaches the target region more often and converts a larger share of reached targets into correct answers (Table 14; evaluation details in Appendix B.4).

Under the default submitted-pose metric, DINOv2 similarity search with two-step lookahead reaches 9.1% IVP success, while MASt3R relative-pose estimation using the benchmark images reaches 8.1%. Appendix G.2 also reports image-similarity search and relative-pose estimation as reference methods, with their observation and input conditions specified separately.

5.3 What Behavior Changes?

Scene coverage and target overlap. We track which 3D scene points have become visible over the course of an episode. Scene coverage is the fraction of all scene points observed so far; target overlap is the fraction of points visible in the target image that have also been observed. In Figure 4, the trained policy's scene coverage rises quickly during early turns, while target overlap grows faster during the middle turns. Each average includes the episodes with an observation at that turn, so later turns contain only episodes that continue interacting. Appendix D.1 gives the metric definitions, aggregation procedure, and turn-usage analysis.

What remains difficult? Among the trained policy's 277 failed episodes, none is an immediate pose submission without exploration, and only 0.4% fail to produce a valid answer within the turn budget. Most failures involve inaccurate position estimates: 40.4% have a position error alone and 49.1% have both position and rotation errors (Table 6). Success falls from 71.1% for targets at view distance $d < 2$ to 28.7% for targets at $d \geq 5$, using the same geometric distance as the main evaluation. Appendices E.2 and E.3 report the model-specific interaction failures and further trajectory statistics.

Table 6: **IVP failure outcomes.** Automatic hierarchical categories over failed episodes (%).

Failure category	Gemini	Grok	Ours
No valid answer	1.0	31.4	0.4
Immediate answer	0.2	24.0	0.0
Orientation flip	10.8	4.1	7.6
Position + rotation	50.6	21.5	49.1
Position only	35.3	17.2	40.4
Rotation only	2.2	1.8	2.5

5.4 Beyond the Primary Training Setting

Downstream initialization. We examine whether experience from interactive camera control provides a useful starting point for further view-understanding training. P2V and V2P ask the model to relate camera actions to images, while MindCube asks the model to track object transformations across viewpoints. We start downstream GRPO training from two checkpoints: the base Qwen2.5-VL-7B and our IVP-trained model. For each task, both use the same downstream data and hyperparameters, and train for 401 steps. On MindCube (Wang et al., 2025c), they begin at nearly the same accuracy, but training from the IVP checkpoint reaches 66.2%, compared with 56.3% from the base model. The IVP checkpoint also reaches higher final accuracy on P2V and V2P (Table 7), making it a better initialization under these downstream training settings. Appendix D.2 provides the evaluation protocols, training configurations, and reward functions.

Table 7: **Downstream initialization.** Accuracy before and after identical GRPO post-training (%).

	P2V		V2P		MindCube	
Model	Pre	Post	Pre	Post	Pre	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**

6 Related Work

View reasoning and active visual search. MindCube (Wang et al., 2025c), ViewSpatial-Bench (Li et al., 2025a), and VSI-Bench (Yang et al., 2025a) evaluate spatial reasoning from supplied observations, while ScanQA (Azuma et al., 2022) and SQA3D (Ma et al., 2023) address question answering in 3D scenes. These tasks provide the scene as input; IVP also requires choosing camera actions to gather new views. Cambrian-P (Yang et al., 2026) uses camera-pose supervision to improve video understanding and estimate camera poses from observed frames. Visual-search agents select information through cropping, zooming, and viewpoint changes (Wu & Xie, 2024; Wang et al., 2025e; Zheng et al., 2025; Wang et al., 2025a), while E3VS-Bench (Sakamoto et al., 2026) studies viewpoint-dependent active perception in Gaussian-splatting scenes. ViewSuite asks questions about supplied action sequences and evaluates multi-turn camera control on the same view pairs.

Navigation and training-time scene graphs. Image-goal navigation conditions action selection on a target image (Zhu et al., 2017); IVP shares this interface while exposing 6-DoF camera control and requiring an explicit target-pose report. Topological navigation methods such as SPTM (Savinov et al., 2018) and SoRB (Eysenbach et al., 2019) organize experience into graphs that guide search. View Graph Distillation uses a graph during training: paths through the graph become action demonstrations and view-understanding examples for the VLM. At test time, the trained policy chooses actions from images and interaction history without graph lookup.

Learning VLM policies from interaction. Outcome-supervised RL and self-training reuse model-generated experience for policy improvement (Shao et al., 2024; DeepSeek-AI, 2025; Zelikman et al., 2022; Singh et al., 2024; Oh et al., 2018; Wang et al., 2025d;b). Using an observed outcome as a training goal is related to Hindsight Experience Replay (Andrychowicz et al., 2017), which reuses collected experience with goals achieved during that experience. Visual Hindsight Self-Imitation Learning (Kim et al., 2023) uses achieved goals to construct demonstrations for visual navigation. Like on-policy distillation (Agarwal et al., 2024), collection follows the learner's evolving policy. In our setting, recorded camera actions and the poses reached in the environment provide the supervision. The graph retains these observations across collection rounds and turns paths through them into demonstrations for the learner.

7 Conclusion and Limitations

ViewSuite benchmarks supplied-path understanding and interactive camera control on the same view pairs, revealing a planning gap in current VLMs. View Graph Distillation turns a weak policy's scene exploration into reusable supervision between RL stages, raising Qwen2.5-VL-7B from 2.5% to 47.7% IVP success. The same recipe improves InternVL3-8B and a Qwen2.5-VL-7B model trained separately on AI2-THOR.

Limitations. We train in static indoor scenes with discrete camera actions. Dynamic scenes and continuous control remain future work.

Ethics Statement

ViewSuite is constructed from ScanNet (Dai et al., 2017), a publicly released dataset of real indoor scans, and introduces no new recordings of people or private spaces. The threshold-calibration study uses human judgments of rendered view pairs, as described in Appendix A.4. All evaluated models are either publicly available open-weight checkpoints or are accessed through their providers' public APIs, and are used within their stated terms of service. We see no direct harmful application of view planning beyond the general dual-use considerations that apply to embodied and active-perception systems.

Reproducibility Statement

Benchmark construction, task definitions, and data statistics for ViewSuite are described in Section 2 and detailed in Appendix A. Evaluation settings and prompts are provided in Appendices A and B. Training hyperparameters and the self-exploration, graph-construction, and distillation procedures are described in Appendix C. Code, model checkpoints, and datasets will be released publicly; links are withheld here to preserve anonymity during review.

AI Use Statement

We used generative AI tools to assist with writing code and with drafting and polishing the text of this paper. All AI-assisted code was reviewed, executed, and tested by the authors, and all AI-assisted text was reviewed and edited by the authors. We also use a VLM judge to screen rendered Habitat-GS views for recognisability, as described in Appendix F.2. We take responsibility for the final content of this paper, including all text, claims, and artifacts produced with the aid of generative AI.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2306.13649.

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. Claude opus 4.6 system card, 2026. URL https://www.anthropic.com/claude-opus-4-6-system-card.

Daichi Azuma, Taiki Miyanishi, Shuhei Kurita, and Motoaki Kawanabe. Scanqa: 3d question answering for spatial scene understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. URL https://arxiv.org/abs/2112.10482.

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*, 2025a. URL https://arxiv.org/abs/2502.13923.

Shuai Bai et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025b. URL https://arxiv.org/abs/2511.21631.

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.

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.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/5c48ff18e0a47baaf81d8b8ea51eec92-Abstract.html.

Google DeepMind. Gemini 3 pro model card, 2025. URL https://deepmind.google/models/model-cards/gemini-3-pro/.

Google DeepMind. Gemini 3.1 pro model card, 2026. URL https://deepmind.google/models/model-cards/gemini-3-1-pro/.

Wenyi Hong et al. GLM-4.5V and GLM-4.1V-Thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *arXiv preprint arXiv:2507.01006*, 2025. URL https://arxiv.org/abs/2507.01006.

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.

Kibeom Kim, Kisung Shin, Min Woo Lee, Moonhoon Lee, Minsu Lee, and Byoung-Tak Zhang. Visual hindsight self-imitation learning for interactive navigation, 2023. URL https://arxiv.org/abs/2312.03446.

Kimi Team. Kimi K2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026. URL https://arxiv.org/abs/2602.02276.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, Aniruddha Kembhavi, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An interactive 3d environment for visual AI. *arXiv preprint arXiv:1712.05474*, 2017. URL https://arxiv.org/abs/1712.05474.

Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European Conference on Computer Vision (ECCV)*, 2024.

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.

Xiaojian Ma, Silong Yong, Zilong Zheng, Qing Li, Yitao Liang, Song-Chun Zhu, and Siyuan Huang. Sqa3d: Situated question answering in 3d scenes. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2210.07474.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018. URL https://arxiv.org/abs/1806.05635.

OpenAI. GPT-5 system card. *arXiv preprint arXiv:2601.03267*, 2026a. URL https://arxiv.org/abs/2601.03267.

OpenAI. GPT-5.4 thinking system card, 2026b. URL https://deploymentsafety.openai.com/gpt-5-4-thinking.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Qwen Team. Qwen3.5: Towards native multimodal agents, 2026. URL https://qwen.ai/blog?id=qwen3.5.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011. URL https://arxiv.org/abs/1011.0686.

Koya Sakamoto, Taiki Miyanishi, Daichi Azuma, Shuhei Kurita, Shu Morikuni, Naoya Chiba, Motoaki Kawanabe, Yusuke Iwasawa, and Yutaka Matsuo. E3vs-bench: A benchmark for viewpoint-dependent active perception in 3d gaussian splatting scenes. *arXiv preprint arXiv:2604.17969*, 2026. URL https://arxiv.org/abs/2604.17969.

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1803.00653.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied AI research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019. URL https://arxiv.org/abs/1904.01201.

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.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, et al. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research (TMLR)*, 2024. URL https://arxiv.org/abs/2312.06585.

Haozhe Wang, Alex Su, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel reasoner: Incentivizing pixel-space reasoning with curiosity-driven reinforcement learning. *arXiv preprint arXiv:2505.15966*, 2025a. URL https://arxiv.org/abs/2505.15966.

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*, 2025b.

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*, 2025c. 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*, 2025d.

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, 2025e. 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.

xAI. Grok 4.20 model documentation, 2026. URL https://docs.x.ai/developers/models/grok-4.20.

Ziyuan Xia, Jingyi Xu, Chong Cui, Yuanhong Yu, Jiazhao Zhang, Qingsong Yan, Tao Ni, Junbo Chen, Xiaowei Zhou, Hujun Bao, Ruizhen Hu, and Sida Peng. Habitat-GS: A high-fidelity navigation simulator with dynamic gaussian splatting, 2026. URL https://arxiv.org/abs/2604.12626.

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.

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.

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.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2203.14465.

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.

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.

Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. Deepeyes: Incentivizing thinking with images via reinforcement learning. *arXiv preprint arXiv:2505.14362*, 2025. URL https://arxiv.org/abs/2505.14362.

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.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. InternVL3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025. URL https://arxiv.org/abs/2504.10479.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017. URL https://prior.allenai.org/projects/target-driven-visual-navigation.

A VIEWSUITE DETAILS

A.1 BENCHMARK CONTEXT

Table 1 in the main paper compares the visual settings, observation-gathering operations, and tasks of selected benchmarks. ViewSpatial-Bench, VSI-Bench, CameraBench, and MindCube ask questions about supplied images or videos. The active-perception benchmarks let an agent gather additional observations through image-region selection, changes in viewing direction, or movement in an environment. The operation column describes these observation-gathering actions; "None" means the benchmark supplies the visual observations for answering.

VIEWSUITE constructs P2V, V2P, and IVP from the same initial-target view pairs. P2V and V2P ask the model to interpret supplied camera paths, while IVP lets it execute camera actions and report the target pose. All three use the same scene geometry and camera-action definitions. The construction pipeline produces approximately 55K view pairs and 165K task instances across 286 ScanNet scenes. The main experiments use the scene-disjoint VIEWSUITE-5K subset, with 4,327 view pairs and approximately 13K task instances.

The subset contains 244 scenes from the released pool. Its train/dev/test splits contain 193 / 22 / 29 scenes and 3,419 / 378 / 530 view pairs, respectively, with no scene shared between splits. Each view pair supplies one instance of each task. The supplementary graph comparison in Appendix C.10 uses its own split.

A.2 ACTION SPACE DETAILS

As described in Section 2.2, VIEWSUITE provides 12 camera actions (Figure 1a). Table 8 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 $s_t = 0.5$ 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); their names describe the direction of camera rotation about its viewing axis.

Discrete snapping. In discrete mode, after each rotation action the camera-to-world rotation matrix is decomposed into extrinsic XYZ Euler angles (SciPy's `xyz`), each angle is snapped to the nearest multiple of s_r, and the rotation matrix is recomposed. The resulting Euler angles lie on the discrete rotation grid. The same Euler-angle convention is used for pose readouts and submitted answers; it reproduces the rotation errors recorded by the environment.

Actions and observations within a turn. The agent can issue up to four actions in one turn. The environment executes them in order and returns the image and pose after the final action; intermediate poses do not produce observations. The 10-turn budget includes the answering turn, allowing up to nine action turns and 36 atomic actions.

A.3 DATA SAMPLING AND FILTERING PIPELINE

Table 9 lists the sampling and filtering 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.

Table 8: Detailed action definitions. All rotations are about the camera center in local coordinates.

Action	Type	Axis	Step size
`move_forward`	Translation	Camera $+Z$	$0.5\,\text{m}$
`move_backward`	Translation	Camera $-Z$	$0.5\,\text{m}$
`move_left`	Translation	Camera $-X$	$0.5\,\text{m}$
`move_right`	Translation	Camera $+X$	$0.5\,\text{m}$
`move_up`	Translation	Screen up ($-Y$)	$0.5\,\text{m}$
`move_down`	Translation	Screen down ($+Y$)	$0.5\,\text{m}$
`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°

Target snapping. For each sampled pair, we snap the target onto the discrete pose lattice anchored at the initial pose. The six axes are processed in a fixed order (yaw, pitch, roll, forward, right, up). For each axis, we try step counts up to the per-axis maximum and choose the count that minimizes the unified view distance d to the sampled target before proceeding to the next axis. The resulting pose replaces the sampled target in all three tasks, so every target is reachable using the camera actions. We discard pairs whose snapped target coincides with the initial pose or lies strictly inside its success threshold. We also discard pairs whose target is farther from the initial pose than the per-axis maxima allow.

Distractor generation. For the multiple-choice P2V and V2P tasks, we create $K{=}3$ distractors per pair by perturbing the action sequence of length ℓ that reaches the snapped target. 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 any existing option is below 0.02.

Scene-level filtering. In addition to the per-pair filters above (viewpoint identity, distance from the initial pose, and per-axis limits), we apply scene-level quality filtering based on the top-down reference view. We first use a VLM 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.4 Success Threshold Calibration

We calibrate the success thresholds of Section 2.3 through a pilot human alignment study. The study uses 100 test-split rollouts, with 50 each from Gemini 3.1 Pro and GPT-5.4 Pro, sampled to balance rollouts that pass and fail the $0.5\,\text{m}$ / 30° criterion at approximately half each. We express candidate thresholds as multipliers β_t and β_r of the translation and rotation step sizes. 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 an expert annotator. The annotator judges whether the two views depict the same place (*match*) or not; the labels contain 45 matches and 55 non-matches. We then sweep (β_t, β_r) over translation thresholds $\{0.25, 0.5, 0.75, 1.0\}\,\text{m}$ and rotation thresholds $\{30^\circ, 60^\circ, 90^\circ\}$, treating the threshold-based success indicator at each setting as a binary classifier of the human label. Table 10 reports precision, recall, F_1, and accuracy across the resulting 4×3 grid. The combination $0.5\,\text{m}$ and 30°, equivalent to $(\beta_t, \beta_r) = (1, 1)$ given the discrete step sizes $s_t = 0.5\,\text{m}$ and $s_r = 30^\circ$, achieves the highest F_1 (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: with $\beta_t = 2$ ($1\,\text{m}$), precision drops to 0.72 at 30° and below 0.6 at 90°. Threshold selection and the reported agreement use the same 100 examples.

Table 9: 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)	$[0, 1]$
Target snapping & filtering	
Axis order	Rotation axes first
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

Table 10: Success threshold calibration on IVP rollouts. Each row evaluates the threshold-based success indicator at one (position, rotation) threshold pair against the human label. The 0.5 m / 30° setting achieves the best agreement (F_1=0.915) and is adopted throughout the paper.

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

A.5 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, measured in the normalized translation and rotation units defined in Appendix B.6. We split the test set at $d = 3$ into a SHORT subset (187 pairs, $d < 3$) and a LONG subset (343 pairs, $d \geq 3$); these subsets are used in the difficulty-stratified analysis of Section 3. The cutoff groups examples by geometric separation between their initial and target viewpoints. Distances are computed from the camera-to-world matrices. No pair starts strictly inside the success threshold, but 12 pairs lie exactly on its inclusive boundary. Submitting the initial pose therefore succeeds on $12/530 = 2.3\%$ of the test set. These pairs are retained for consistency with the released benchmark.

A.6 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.

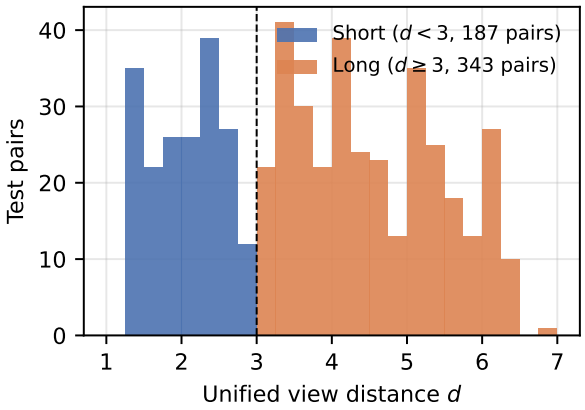

Figure 5: Distribution of unified view distance d across 530 test pairs. The threshold $d = 3$ separates SHORT (187 pairs) and LONG (343 pairs) subsets.

> **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 (incorrect), 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` × 5, `move_left`]
> C. [`turn_right` × 2, `move_forward`, `move_left` × 5, `move_up`]
> D. [`turn_left` × 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.

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 camera pose estimation task.
>
> GOAL: Predict the target view absolute camera pose (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 pose should be as close as possible to the target pose.
>
> 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)`.



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



(c) Option A (d) Option B



(e) Option C (f) Option D

Figure 6: Path-to-View (P2V) example: given the initial view (a), the top-down view (b), and an action sequence, the model picks the resulting view among options (c)-(f). Action: [`turn_right` $\times$ 5]. GPT-5.4 Pro selects C (incorrect).



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

Figure 7: View-to-Path (V2P) example: given the initial (a), top-down (b), and target (c) views, the model picks which of four action sequences connects the initial view to the target view. GPT-5.4 Pro selects B (correct): [`turn_left` $\times$ 5, `move_left`].

> **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 $\leq$ 0.5 m, rotation error $\leq 30°$.

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



(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: Interactive View Planning (IVP) example. Our trained Qwen2.5-VL-7B plans view changes from the initial view (b) to localize the target view (a) in 6 turns. Final pose error: 0.061 m / 0°. Success.

B Extended Evaluation Results

B.1 Interaction Budget and Rendering Diagnostics

We increase the IVP turn budget from 10 to 20 and 30 for four proprietary models (Table 11). All four improve from 10 to 20 turns, with Claude Opus 4.6 showing the largest relative gain. Improvements from 20 to 30 turns are smaller or absent for most models. More interaction helps, but the gains largely saturate before closing the planning gap.

We also re-render the test set using 3D Gaussian Splatting (Kerbl et al., 2023), with pretrained per-scene reconstructions from SceneSplat-7K (Li et al., 2025b), and evaluate the same four models at a 10-turn budget. IVP improves by 0.2 to 1.9 points. Single-turn results change more and in both directions: Gemini 3.1 Pro gains 6.4 points on P2V, while GPT-5.4 and Grok 4.20 Beta lose 14.4 and 13.0 points on V2P relative to the default rendering.

The 10-turn default columns reproduce Table 2, using the matrix-based Short/Long split (187 / 343 pairs). The 20-turn, 30-turn, and GS results come from separate runs in the original logs, which used the earlier 185 / 345 split. Their All values are reported over the episodes completed in each run.

B.2 Performance by Viewpoint Separation

Figure 2 groups examples by rotation and position distance. P2V and V2P show stronger degradation across rotation-distance bins, while IVP success varies more strongly with position distance. The sample-level correlations in Appendix B.5 show the same contrast at the example level.

Figure 2 is presented with the main-text diagnostic analysis in Section 3.2.

Table 11: **Sensitivity to turn budget and rendering quality.** Left three blocks: IVP success rate (%) at turn budgets of B=10, 20, and 30 with point-cloud rendering. Rightmost block: P2V / V2P / IVP All-split results at B=10 with higher-fidelity Gaussian Splatting (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	28.9	**17.2**	**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	**33.2**	7.6	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	17.1	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	23.5	3.8	10.8	**31.9**	10.1	17.7	**35.7**	11.0	19.6	35.3	41.3	12.3	29.6

B.3 Target-Proximate Observations in Successful Episodes

For each of the 13 zero-shot VLMs and our trained Qwen2.5-VL-7B, we examine its successful IVP episodes and check whether any observed camera pose lies within 0.5 m and 30° of the target pose before the final answer. This includes the initial observation, so an episode can satisfy the condition without moving the camera. We also include observations at the end of each turn, but exclude intermediate poses within a multi-action turn because the model receives no image there. Table 12 reports the counts; percentages use each model's successful episodes as the denominator.

Target-proximate observations precede 96.5% of Gemini 3.1 Pro's successes, 94.1% of GPT-5.4 Pro's, and 98.0% of the trained model's. The statistic records which views were available before the answer. Pose proximity is a geometric proxy for visual access: target content may be visible from farther away or occluded at a nearby pose. Determining whether a model copied an observed pose or inferred the target pose by another process would require additional evidence beyond these pose logs.

Table 12: **Target-proximate observations in successful episodes.** Counts and percentages using observations at initialization and at the end of each turn.

Model	Successes	Proximate observation	No such observation
GPT-5.4 Pro	101	95 (94.1%)	6 (5.9%)
Gemini 3.1 Pro	113	109 (96.5%)	4 (3.5%)
GPT-5.4	88	79 (89.8%)	9 (10.2%)
Grok 4.20 Beta	42	34 (81.0%)	8 (19.0%)
GPT-5.1	33	22 (66.7%)	11 (33.3%)
Claude Opus 4.6	57	53 (93.0%)	4 (7.0%)
Gemini 3 Pro	49	43 (87.8%)	6 (12.2%)
Qwen3.5-397B	23	15 (65.2%)	8 (34.8%)
GLM-4.6V	21	19 (90.5%)	2 (9.5%)
Qwen2.5-VL-72B	6	5 (83.3%)	1 (16.7%)
Qwen3-VL-32B	8	5 (62.5%)	3 (37.5%)
Kimi K2.5	13	10 (76.9%)	3 (23.1%)
Qwen2.5-VL-7B	13	7 (53.8%)	6 (46.2%)
Ours (Qwen2.5-VL-7B)	253	248 (98.0%)	5 (2.0%)

B.4 Evaluation-Protocol Ablations on IVP

The default IVP results in Tables 2 and 3 use the discrete camera interface and require an explicit pose submission. We evaluate two changes to this protocol for the trained Qwen2.5-VL-7B model and two proprietary reference models, Gemini 3.1 Pro and GPT-5.4. No-Snap changes the execution dynamics at evaluation time, while No-Submit retains the default dynamics and changes the success criterion. These experiments examine execution sensitivity and target-region reaching separately.

- **No-Snap.** Our policy is trained with the benchmark's default discrete camera interface. During execution, this interface rounds the camera's extrinsic Euler angles to the nearest multiples of 30° after each action (Appendix A.2). No-Snap removes this rounding at test time and keeps the pose

obtained from the rotation, so actions execute differently from those seen during training. The submitted-pose success criterion is unchanged.

- **No-Submit.** Success is credited when an observed camera pose enters the target's 0.5 m / 30° acceptance region, without requiring the agent to submit an answer. We re-score the main-evaluation rollouts, using the initial observation and observations at the end of turns; no new rollouts are collected. This is a goal-reaching measure, rather than a measure of the submitted pose estimate.

Table 13 retains the original paired Default/No-Snap evaluation, with its original 185 / 345 Short/Long split and logged rates. In those runs, Qwen2.5-VL-7B falls from 47.8% to 19.6% overall, while Gemini 3.1 Pro falls from 21.3% to 15.7%. The aggregate advantage therefore narrows from 26.5 to 3.9 points. On the Long split, Gemini scores 9.6%, slightly above the trained Qwen2.5 model's 9.3%. All three evaluated models lose accuracy under No-Snap. The trained Qwen2.5 policy shows a larger drop than either zero-shot reference model under this change in execution rules. The original default score is kept with this comparison; the main training table uses the recomputed $253/530 = 47.7\%$ result.

Table 14 reports No-Submit separately using the main rollouts and the current 187 / 343 split. Qwen2.5-VL-7B reaches 57.5%, while Gemini 3.1 Pro and GPT-5.4 reach 32.8% and 26.0%. The trained policy therefore also achieves stronger goal-reaching outcomes under the default camera interface when success is measured from observed poses. Among the episodes that reach the target region, $57/305 = 18.7\%$ for our model and $65/174 = 37.4\%$ for Gemini do not end in a correct submission. These conditional proportions differ from simply subtracting the two success rates because some correct submissions occur without a target-proximate observation.

Coverage note for GPT-5.4 Pro. GPT-5.4 Pro refuses 23 of the 530 IVP test episodes, leaving 101 successes. We count refusals as failures in the primary comparison, yielding $101/530 = 19.1\%$ overall, with $65/187 = 34.8\%$ on Short and $36/343 = 10.5\%$ on Long targets. Conditional on its 507 completed episodes, the same 101 successes give 19.9%. Its No-Submit score likewise uses the full test set: $144/530 = 27.2\%$, rather than $144/507 = 28.4\%$ conditional on completed episodes. GPT-5.4 Pro completes all 530 P2V and V2P examples. GLM-4.6V and Random Response each complete 529 instances per task; their main-table rates retain the completed-instance denominators from the released records.

Random-response baseline. For P2V and V2P, Random Response selects uniformly from the four options. For IVP, it issues random actions for nine turns and then submits a randomly selected previously visited pose. Thus, its IVP score measures random exploration followed by a pose submission, rather than an unconstrained random guess in pose space.

Table 13: **Sensitivity to rotation snapping.** Original paired Default/No-Snap results (%), using the original 185 / 345 Short/Long split.

	Default (original runs)			No-Snap		
Method	Short	Long	All	Short	Long	All
Gemini 3.1 Pro	28.6	17.4	21.3	27.0	9.6	15.7
GPT-5.4	33.5	7.5	16.6	27.6	5.2	13.0
Ours (Qwen2.5-VL-7B)	67.2	36.9	47.8	38.9	9.3	19.6

Table 14: **Submitted-pose and goal-reaching success.** Default and No-Submit rates (%) from the same main-evaluation rollouts, using 187 Short and 343 Long pairs.

	Default			No-Submit		
Method	Short	Long	All	Short	Long	All
GPT-5.4 Pro	34.8	10.5	19.1	52.9	13.1	27.2
Gemini 3.1 Pro	28.9	17.2	21.3	49.2	23.9	32.8
GPT-5.4	33.2	7.6	16.6	49.2	13.4	26.0
Ours (Qwen2.5-VL-7B)	66.8	37.3	47.7	79.7	45.5	57.5

B.5 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.6). 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 has the strongest positive association with success (ρ approximately $+0.19$ to $+0.30$), with same-facing camera pairs more often answered correctly. For IVP, position distance has a strong negative association with success (ρ as low as -0.42 for GPT-5.4 Pro), consistent with the distance profiles in Appendix B.2. Visual overlap factors show mild positive correlations with P2V/V2P accuracy. Grok 4.20 Beta differs from the other evaluated models on V2P, showing near-zero or slightly positive correlations with distance factors.

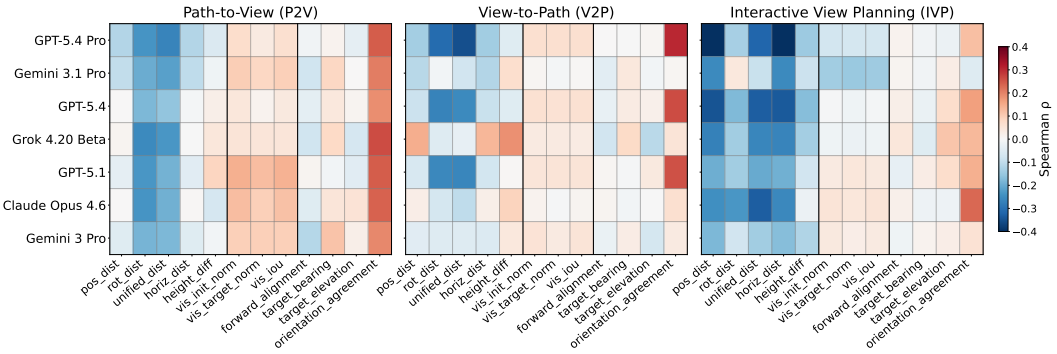

Figure 9: Spearman ρ between per-model success (rows) and sample-level factors (columns), grouped into geometric distance, visual overlap, and directional geometry. Position distance is the strongest predictor of IVP success; orientation agreement is the strongest for P2V/V2P.

B.6 Success Factor Definitions

We define the 12 sample-level factors used in Figure 9. All factors are computed from the initial 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). The factor-analysis code uses the OpenGL convention with local $-Z$ forward. It is related to the OpenCV convention of Appendix A.2 by a fixed axis flip applied consistently to every pose.

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.
- `unified_dist`: $\sqrt{(d_{\text{pos}}/s_t)^2 + (d_{\text{rot}}/s_r)^2}$ (steps), where $s_t = 0.5\,\text{m}$ and $s_r = 30^\circ$ 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.

Table 15: **Reasoning helps, but scaling reasoning effort does not close the planning gap.** Success rates (%) of four proprietary models on the 530-instance VIEWSUITE test set under three reasoning-effort settings (S: Short, $d<3$; L: Long, $d\geq 3$). *Medium* is the default used in Table 2. Overall is the mean of the three All columns.

		Path-to-View (P2V)			View-to-Path (V2P)			Interactive View Planning (IVP)			
Model	Effort	S	L	All	S	L	All	S	L	All	Overall
GPT-5.4	nothink	49.7	24.9	33.6	35.7	27.2	30.2	9.7	1.2	4.2	22.6
	medium	57.8	42.6	47.9	59.9	37.6	45.5	33.2	7.6	16.6	36.7
	high	65.9	44.3	51.9	68.6	31.9	44.7	36.6	8.7	19.6	38.8
Gemini 3.1 Pro	nothink	55.7	38.8	44.7	48.6	44.9	46.2	22.7	8.1	13.2	34.7
	medium	63.1	41.1	48.9	52.9	47.5	49.4	28.9	17.2	21.3	39.9
	high	64.3	45.8	52.3	54.6	44.1	47.7	31.6	24.1	26.9	42.3
Claude Opus 4.6	nothink	44.9	25.5	32.3	24.9	18.0	20.4	18.9	5.2	10.0	20.9
	medium	46.5	28.3	34.7	47.1	38.5	41.5	23.5	3.8	10.8	29.0
	high	49.2	29.9	36.6	51.9	35.1	40.9	18.4	2.6	8.1	28.6
Grok 4.20 Beta	nothink	37.3	24.1	28.7	41.6	27.0	32.1	10.3	0.0	3.6	21.4
	medium	61.0	38.2	46.2	44.4	44.6	44.5	17.1	2.9	7.9	32.9
	high	61.1	33.0	42.8	38.9	36.5	37.4	19.1	1.8	7.9	29.3

- `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 the initial to the target position.

- `forward_alignment`: $\hat{\mathbf{f}}_{\text{init}} \cdot \hat{\mathbf{d}}$. Ranges from $+1$ (target ahead) to -1 (target behind).
- `target_bearing`: $\arccos(\text{clip}(\texttt{forward_alignment}, -1, 1))$ (degrees). Angle between init forward direction and displacement to target.
- `target_elevation`: $\text{atan2}(\Delta z, \|\Delta_{xy}\|)$ (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.

B.7 EFFECT OF REASONING EFFORT

All evaluations in the main paper use ReAct-style prompt templates with reasoning enabled by default; for proprietary reasoning models, the reasoning effort is set to *medium*. To characterize sensitivity to test-time reasoning settings, we re-evaluate four proprietary models on the full 530-instance VIEWSUITE test set under three settings: *nothink* (reasoning disabled), *medium* (the default used in Table 2), and *high*. Table 15 reports success rates on all three tasks.

The medium rows reproduce the current main evaluation with 187 Short and 343 Long pairs. The nothink and high rows are separate runs from the original logs, which used the earlier 185 / 345 split; All and Overall are reported over the episodes completed in each run.

Reasoning settings affect the measured performance. For GPT-5.4, IVP success rises from 4.2% with reasoning disabled to 16.6% at medium effort and 19.6% at high effort. Gemini 3.1 Pro improves from 13.2% to 21.3% and 26.9% across the same settings. Further increases are not uniformly beneficial: Claude Opus 4.6 and Grok 4.20 Beta have lower Overall scores at high than at medium effort. More reasoning helps the strongest models, but no tested setting solves IVP.

B.8 TRAINED MODEL UNDER A RENDERING SHIFT

Appendix B.1 reports that re-rendering with 3D Gaussian Splatting (3DGS) produces small IVP changes for the evaluated zero-shot proprietary models. We also evaluate the trained Qwen2.5-VL-7B checkpoint on the same 530 held-out ScanNet test instances under the two renderers (Table 16).

IVP falls from 47.7% under the default renderer to 36.0% under 3DGS, a decline of 11.7 points. The checkpoint keeps much of its improvement over the base policy, while remaining sensitive to the change in rendered appearance.

Table 16: **The trained checkpoint under a rendering shift.** IVP success rates (%) on 530 VIEWSUITE test instances under the default renderer and 3D Gaussian Splatting (3DGS).

Task	Default rendering	3DGS	Δ (Default $-$ 3DGS)
IVP	47.7	36.0	$+11.7$

C ITERATIVE TRAINING IMPLEMENTATION DETAILS

C.1 INTERACTIVE TASK AND REINFORCEMENT LEARNING

At turn t, the VLM receives a rendered view o_t and current camera pose $p_t \in \mathrm{SE}(3)$, together with the target view, top-down reference, and interaction history. It chooses a sequence of up to four actions from the 12 camera-movement primitives. The environment applies those actions in order and returns one observation and camera pose at the end of the turn. It does not return intermediate observations between actions in the same turn. The 10-turn budget includes the final answering turn, allowing at most nine action turns and 36 atomic actions before the model submits a target-pose estimate $\hat{p}^*$. For an atomic action a, the environment updates the camera deterministically as $p' = T(p, a)$. The training reward is

$$r(\hat{p}^*, p^*) = \mathbf{1}[d_{\mathrm{pos}}(\hat{p}^*, p^*) \le 0.5\,\mathrm{m} \ \wedge \ d_{\mathrm{rot}}(\hat{p}^*, p^*) \le 30^\circ] + 0.1\,\mathbf{1}_{\mathrm{format}}, \tag{2}$$

where the first term evaluates task success and the second rewards a correctly structured response. PPO (Schulman et al., 2017) updates the policy during self-exploration. The task-success term evaluates the original target; graph distillation uses additional targets constructed from the recorded exploration.

The submitted position is expressed in meters. The submitted rotation uses extrinsic XYZ Euler angles in degrees, corresponding to SciPy's lowercase `xyz` convention. Position and rotation errors are computed from the resulting camera-to-world pose, using the same geometric definitions as the benchmark evaluation.

C.2 ALGORITHM

Algorithm 1 summarizes the alternating stage of our framework. Each iteration appends new self-exploration trajectories to a persistent view graph, samples paths from it, and reformulates them via Eq. 1 into supervised view-planning demonstrations. After these iterations, we continue PPO training from the resulting policy without further graph construction or distillation.

Algorithm 1 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. 2
    Append trajectories: $G_{k+1} \leftarrow G_k \cup \mathrm{traj}(\pi_{\theta_k})$
  View graph distillation stage:
    Sample paths $\{P_i\} \subset G_{k+1}$
    Reformulate: $\mathcal{D}_{k+1} \leftarrow \{\mathcal{R}(P_i)\}$ via Eq. 1
    Fine-tune via SFT: $\theta_{k+1} \leftarrow \arg\min_\theta \mathcal{L}_{\mathrm{SFT}}(\theta; \mathcal{D}_{k+1})$
end for
return π_{θ_K}

C.3 RL HYPERPARAMETERS

Table 17 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.

Iteration-specific overrides. Iterations 0-2 use 60 RL training steps each, followed by a distillation stage. The final stage continues RL training to convergence without further distillation.

Table 17: 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

Comparison schedules. Success-Only Bootstrapping, Random-graph, and trajectory-level relabeling use the same three-round exploration and distillation schedule as the full method, followed by the final RL stage. Their training data are constructed from successful trajectories, random-action trajectories, and individual relabeled trajectories, respectively. The one- and two-round ablations complete fewer exploration and distillation rounds before entering the final RL stage. They therefore test the number of alternating rounds rather than a fixed total training budget. Checkpoint selection uses the IVP validation subset described in Appendix C.7.

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.4 SFT Hyperparameters

Table 18 lists the SFT hyperparameters used in our framework. Each of iterations 0-2 includes three epochs of distillation. The final RL stage has no SFT stage.

The validation loss in Table 18 is computed on a 20% holdout from the constructed distillation examples and selects the checkpoint within an SFT stage. This is separate from IVP checkpoint selection, which evaluates interactive success on the fixed dev subset described in Appendix C.7.

C.5 View Graph Construction

During the three alternating rounds, a background process runs concurrently with RL training and incrementally merges completed trajectories into the view graph. The final RL stage does not construct or distill a graph.

Table 18: 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
Model selection	
Validation split	20%
Eval strategy	Per epoch
Best model metric	Eval loss

Node and edge representation. Each node stores a 6-DoF viewpoint (position and 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 $0.25\,\mathrm{m}$ *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.

Checking the recorded paths by replay. Merging nearby viewpoints can make a path's recorded pose sequence differ from the poses obtained by executing its actions. We check this by replaying the saved graph edges and sampled paths with the environment's deterministic action model. This action model reproduces all $1{,}049$ recorded rollout transitions used to validate it. We then execute each edge's action sequence from its stored source pose and compare the result with its stored target pose. We also replay paths of three to five edges sampled by random walk, matching the path lengths used for distillation.

Table 19 reports both exact-pose agreement and agreement within the task's success threshold. Here, exact agreement means a position error below $1\,\mathrm{mm}$ and a rotation error below $0.5°$. Across the three iterations, 97.6%, 99.9%, and 99.0% of sampled paths end within $0.5\,\mathrm{m}$ and $30°$ of their recorded target. The check measures how often the sampled action sequence reaches the target used to construct its demonstration.

Table 19: **Replay validity of graph edges and sampled paths.** Agreement (%) between the executed endpoint and the stored target pose, with sampled-path counts.

Check	Iter. 0	Iter. 1	Iter. 2
Edges: exact pose	97.6	99.9	99.3
Edges: within success threshold	98.7	100.0	99.7
Sampled paths	4,085	13,075	12,875
Paths: exact pose	96.7	99.8	98.0
Paths: within success threshold	97.6	99.9	99.0

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 policy explores different regions.

Table 20 shows the graph growth across iterations. The graph grows by an order of magnitude from iteration 0 to iteration 1 as the policy explores more effectively, then grows incrementally in iteration 2. A scene enters the graph only when at least one trajectory passes the image-quality filters. In iteration 0, seven of the 193 training scenes have no usable node, leaving 186 scenes represented in the graph.

Table 20: View graph growth through iterations 0-2. The graph grows by an order of magnitude from iteration 0 to iteration 1, then incrementally.

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

Figure 10 shows the action composition of the accumulated view graphs, using cached action frequencies computed from the saved graph edges. We split each edge's recorded action sequence into individual action strings, count each occurrence, and normalize by the total number of action records in that round's graph. The share of `move_forward` falls from 18.0% in round 0 to 7.3% in round 2. Left and right turns together rise from 27.9% to 33.3%, while looking up and down falls from 19.2% to 11.7%. Backward, lateral, and vertical translations all occupy larger shares of the graph in round 2 than in round 0.

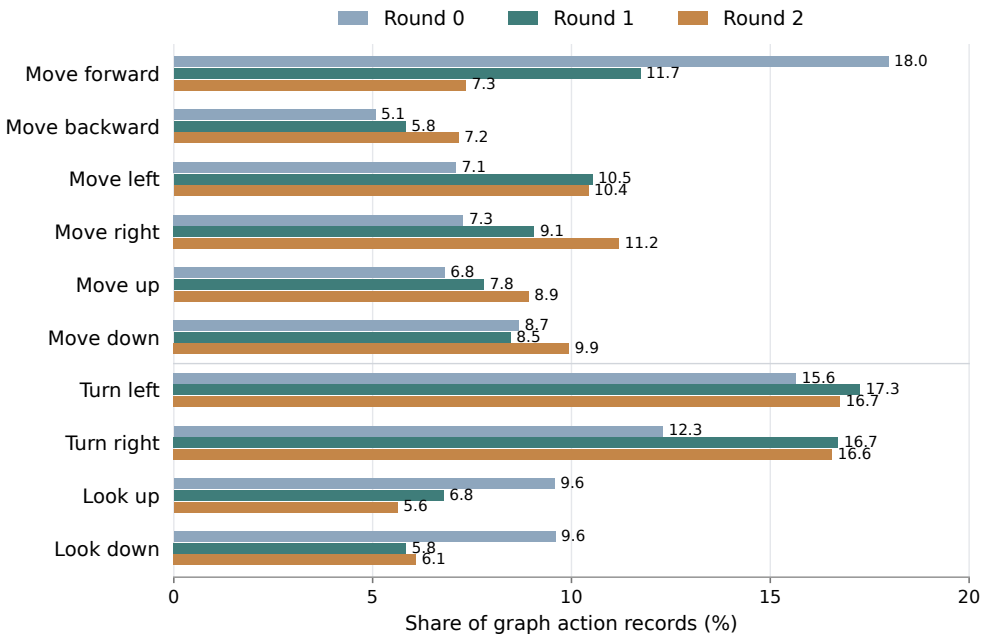

Figure 10: **Action composition of the accumulated view graphs.** Each action string stored on a graph edge contributes one count; percentages are normalized within each saved round.

C.6 Task Reformulation Details

We generate three supervision types from paths sampled from the view graph. Table 21 summarizes the sampling parameters for each task type. We train on these examples with a cross-entropy objective using LLaMA-Factory (Zheng et al., 2024), and run the RL stages with VAGEN (Wang et al., 2025b) and veRL (Sheng et al., 2024).

Table 21: 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 recorded camera-action sequence 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 the unified view distance between them. This auxiliary task supplies supervision about spatial distance between views in addition to the action predictions used in the primary 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 alongside the action sequences used in the primary task. Appendix C.8 evaluates how including these auxiliary tasks affects accuracy on the three evaluation tasks.

C.7 Training and Validation Environments

Training. We use the $3{,}419$ ViewSuite training view pairs for RL, each defining one IVP environment (a scene with an initial and a target view). Each episode runs for up to 10 turns at 512×512 image resolution, rendered via Open3d.

Validation. The dev split contains 378 view pairs, providing 378 validation instances each for P2V and V2P. For IVP checkpoint selection, we use a fixed 100-pair subset because each evaluation requires an interactive rollout. The separate supervision-type ablation below uses its own validation subsets.

C.8 Ablation of Supervision Types in View Graph Distillation

Section 4.1 distills three supervision types from the same view graph: multi-turn view planning (the primary task), view-difference estimation, and its multiple-choice variant (Appendix C.6). The two auxiliary tasks come from the same graph at no extra collection cost. We compare their contribution after one round of self-exploration and distillation on Qwen2.5-VL-7B (Table 22). This experiment uses a longer distillation stage than an individual round of the main recipe, but has no final RL stage. Evaluation uses validation subsets rather than the test set, so the table compares supervision choices within this experiment.

After training on the primary task alone, P2V/V2P accuracy falls from $30.0/27.1$ (untrained) to $14.0/11.1$. Adding either auxiliary signal substantially reduces this loss in single-turn accuracy, while IVP success increases from 3.0% to 8-9%. Using both auxiliary signals yields P2V/V2P accuracy of $28.0/29.0$, close to the untrained model's performance. Auxiliary view-understanding supervision helps retain single-turn accuracy when exploration data are distilled into the VLM policy.

The clearest effect is improved retention of P2V/V2P accuracy relative to primary-task-only training. This result concerns the checkpoint after distillation. After the final RL stage, the full model scores 6.0% on P2V and 20.9% on V2P under the main evaluation protocol, compared with 29.4% and 24.3% for the base model. In these final-checkpoint evaluations, the model often produces IVP-style action responses instead of the requested multiple-choice answer.

Table 22: **Auxiliary supervision preserves single-turn accuracy.** Validation results (%) after one round of self-exploration and distillation on Qwen2.5-VL-7B. *Main*: multi-turn view planning; *Est.*: view-difference estimation; *MCQ*: its multiple-choice variant.

Supervision	P2V	V2P	IVP
Untrained base (reference)	30.0	27.1	0.0
Main only	14.0	11.1	3.0
Main + Est.	27.2	28.8	**9.0**
Main + MCQ	27.0	23.5	8.0
Main + Est. + MCQ (default)	**28.0**	**29.0**	8.0

C.9 Effect of Graph-Based Supervision Construction

What is compared? An episode that misses its original goal can still provide a demonstration for a viewpoint it did reach. We compare two ways of turning this exploration into supervision: relabeling trajectories individually, and constructing training examples from a persistent view graph. In the trajectory-level variant, we relabel *all* collected trajectories, including unsuccessful ones, using the reached viewpoint as the new target. The recorded observations and actions become demonstrations for the VLM, without constructing a graph. This use of achieved goals is related to hindsight relabeling (Andrychowicz et al., 2017). Both variants therefore reuse failures; the difference is how their experience is organized into training examples.

How the graph changes the training examples. The graph variant merges nearby viewpoints and retains the action edges entering and leaving each merged node (Appendix C.5). When two episodes meet at such a node, a sampled path can follow a segment from one episode and continue along an edge discovered in the other. Its endpoints define a new initial-target pair, while its intermediate observations and recorded actions form the demonstration. The graph is accumulated across rounds, so later sampling can reuse earlier exploration as well. From the same structure, we construct view-difference estimation and multiple-choice tasks alongside the action demonstrations. Appendix C.6 specifies these three supervision types and their path-sampling settings.

Experimental settings. Both variants start from Qwen2.5-VL-7B and use three rounds of self-exploration and distillation followed by the final RL stage. The reported comparison matches the exploration schedule, the number of demonstrations per round, the distillation settings, and the final RL stage. Both variants also include the same two auxiliary supervision types. The no-graph variant constructs their view pairs from within individual trajectories, while the graph variant samples pairs from the accumulated graph. Each method is trained once, with checkpoint selection on IVP validation and evaluation on the 530 held-out test pairs. Short and Long use the main view-distance split of 187 and 343 pairs.

Results and interpretation. Trajectory-level relabeling reaches 41.7% success, compared with 47.7% for graph-based supervision (Table 23). The difference is 2.1 percentage points on Short targets and 8.1 points on Long targets. Relabeling individual trajectories therefore already produces a much stronger policy than direct RL in these runs. The graph adds a further improvement, concentrated on more distant targets. The next experiment examines how cross-trajectory path construction changes the demonstrations available for distillation.

Table 23: **Graph construction versus trajectory-level relabeling.** IVP success (%) on the held-out test set after three alternating rounds and final RL.

Method	Short	Long	All
Trajectory-level relabeling	64.7	29.2	41.7
View graph (ours)	**66.8**	**37.3**	**47.7**

C.10 How Graph Paths Change the Distillation Data

A controlled comparison of path construction. We compare two ways of sampling demonstration paths from the same exploration data. This supplementary experiment uses Qwen3-VL-8B-Instruct and an updated implementation with a harder re-split of ViewSuite. Its mean view distance is 5.0,

compared with 3.7 in the main test set. The evaluation contains 560 held-out prompts, with eight sampled responses per prompt. A response succeeds only when its position error is strictly below 0.5 m and its rotation error is strictly below 30°. We report pass@8, the percentage of prompts for which at least one of the eight responses succeeds. The main experiment instead uses inclusive thresholds and single-rollout success, so we report this comparison separately.

Shared exploration and training settings. Both variants use the same pool of 113K episodes collected over three exploration rounds in 96 training scenes. Each constructs 1,536 demonstrations and uses the same SFT recipe and starting checkpoint. Of these demonstrations, 1,024 use a reached view as the new target. The other 512 retain the episode's original target, truncate the trajectory within two steps of it, and use the original target pose as the final answer. The variants differ in how they select a demonstration's path. Trajectory-level relabeling samples two turns $i < j$ within one trajectory and uses the segment between them. Graph-based construction selects a shortest path between viewpoints in the scene graph. Such a path can continue through edges collected in different episodes.

The resulting demonstrations. Among the 1,024 demonstrations with a reached-view target, trajectory-level sampling produces 34 medium-distance and 3 long-distance examples. Graph-based sampling produces 338 and 147, respectively. The corresponding demonstration paths average 1.3 and 2.8 turns. Here, Short, Medium, and Long refer to view distances below 3.5, from 3.5 to below 6, and at least 6. These bands differ from the main benchmark's Short/Long split.

Results from two initializations. Starting from the same checkpoint after exploration training, graph-based demonstrations yield 12.0% pass@8, compared with 9.1% for trajectory-level demonstrations (Table 24). The largest difference occurs in the Medium band. Starting from the untrained base model instead, graph-based demonstrations yield 4.8%, compared with 5.6% for trajectory-level demonstrations. The relative benefit of the longer graph paths thus differs between the two starting checkpoints. These results connect the path-construction choice to both the distribution of training targets and the policy from which distillation starts.

Table 24: **Path construction and initialization.** Pass@8 (%) in the supplementary controlled comparison.

Initialization	Demonstration paths	Short	Medium	Long	All
After exploration	Within trajectories	20.3	8.1	3.2	9.1
After exploration	Through graph	21.6	13.9	4.1	12.0
Untrained base	Within trajectories		Not reported		5.6
Untrained base	Through graph		Not reported		4.8

Reasoning text in the demonstrations. We also compared demonstrations with a fixed reasoning template, a model-written rewrite, and an exact geometric description of the relative pose. The reported comparison found no change in the outcome across these three forms.

Evaluation records. Distance metadata are available for 559 of the 560 prompts: 134 Short, 208 Medium, and 217 Long. One prompt is excluded from the distance-band breakdown because its distance was not recorded. Individual runs also omit a few episodes whose outputs were not saved, so the reported band denominators can differ slightly across runs. In particular, the trajectory-level run initialized from the base model reports 5.6% over 557 prompts, because three prompts lack saved outputs. The table retains the aggregate rates reported for each run.

C.11 Backbone Generalization: InternVL3-8B

We apply the same training recipe to InternVL3-8B (Zhu et al., 2025), a distinct model family, and evaluate IVP on the same 530 ViewSuite test instances (Table 25). The reported training settings include a 400-step RL stage. Short/Long in this table follows ground-truth path length (≤ 5 vs. ≥ 6 actions).

The recipe also improves InternVL3-8B from 0.57% to 20.19% IVP success in this run, with short-path success increasing from 0.82% to 35.10%.

Table 25: **The recipe also improves InternVL3-8B.** IVP success (%) before and after training on the 530 VIEWSUITE test instances. Short/Long split by ground-truth path length (≤ 5 vs. ≥ 6 actions).

Split	Base	Trained	Δ
Short (≤ 5 actions)	0.82	35.10	+34.3
Long (≥ 6 actions)	0.35	7.37	+7.0
All	0.57	20.19	+19.6

D EXTENDED ANALYSIS

D.1 POINT CLOUD COVERAGE: FULL MODEL COMPARISON

Figure 4 in the main paper compares the trained policy with selected reference models, and Figure 11 reports all 14 models shown in its legend: the 13 VLMs in Table 2, including the base Qwen2.5-VL-7B, plus our trained model. The trained policy's average target intersection ratio rises throughout the interaction, while several proprietary models plateau or decline after turns 5 to 7. Open-weight models generally show lower coverage.

Methodology. We analyze available rollout trajectories from the 530 VIEWSUITE test pairs. GPT-5.4 Pro contributes 507 trajectories because its 23 refused episodes produce no trajectory, and GLM-4.6V contributes 529; the other models contribute all 530. 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. Each mean is computed over the trajectories with observations at that turn, so later turns include only episodes that continued interacting.

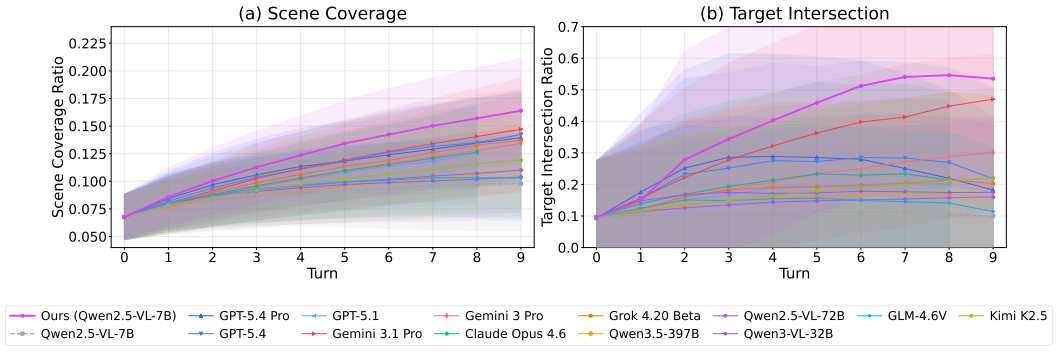

Figure 11: Point cloud coverage on IVP across 14 models. (a) Scene coverage ratio (fraction of all scene vertices observed). (b) Target intersection ratio (fraction of target view vertices covered). Lines show per-turn means over trajectories with observations at that turn; shaded regions show ± 1 standard deviation.

Turn distribution and success rate. Figure 12(a) shows the distribution of turns used per rollout. The base Qwen2.5-VL-7B-Instruct terminates 328 of its 530 rollouts after a single turn, whereas GPT-5.4 Pro never terminates after one turn and uses 7.4 turns on average. A turn can contain several camera actions, so turn count alone does not establish that no action was taken. Our trained model and Gemini 3.1 Pro concentrate at 10 turns (using all available turns). Figure 12(b,c) shows

successful rollouts and success rate by turn count. Our trained model has higher success rates among episodes that terminate earlier. Table 12 conditions only on successful episodes, so frequent one-turn termination overall can coexist with target-proximate visits among successful episodes.

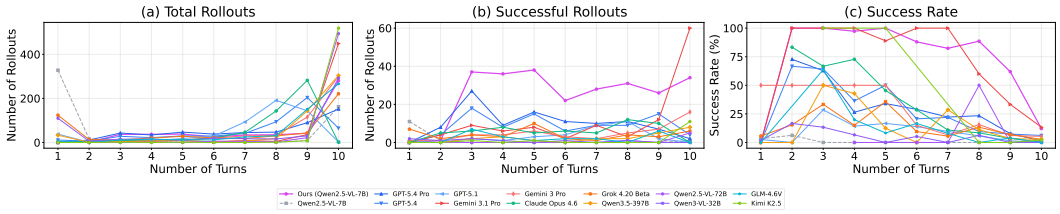

Figure 12: Turn usage on IVP. (a) Rollouts by turn count: the base model stops after a single turn on most rollouts, while our trained model and Gemini 3.1 Pro use the full 10-turn budget; GPT-5.4 Pro uses 7.4 turns on average and never stops after one turn. (b) Successful rollouts and (c) success rate by turn count: our trained model succeeds at high rates on short episodes.

D.2 Downstream Fine-Tuning from the IVP-Trained Checkpoint

We post-train both the IVP-trained checkpoint and the base Qwen2.5-VL-7B-Instruct using GRPO with identical downstream hyperparameters on each task. Table 7 in the main paper reports accuracy before and after this training. P2V and V2P are measured on the ViewSuite development split with the post-training harness (377 of 378 instances completed; one did not), while MindCube is measured on its own test split with the same harness. These initialization scores are therefore not directly comparable to the ViewSuite test-set scores in Table 2. The IVP-trained checkpoint starts lower on P2V, slightly higher on V2P, and at a similar level on MindCube, then reaches higher final accuracy on all three tasks after the same 401 post-training steps. These single-run results support the narrower claim that it is a useful initialization under the reported post-training setup; they do not by themselves isolate a transferable spatial mechanism.

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., 2025c): 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}, and KL penalty disabled ($\lambda_{\text{KL}} = 0$). Each initialization is post-trained for the same 401 steps on the same task data, using 8 GPUs with FSDP and gradient checkpointing. Table 26 summarizes per-task differences.

Table 26: 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

Reward functions. All tasks use a weighted sum of binary format and answer rewards. The model must produce a valid response in the format `...<action>...</action>`. 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:

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

Table 27: **P2V failure categories.** Percentage of inspected P2V failures per category for each frontier model. Coherent geometric reasoning grounded onto the wrong image dominates for every model.

Failure category	GPT-5.4	GPT-5.4 Pro	Gemini 3.1 Pro	Claude Opus 4.6	Grok 4.20 Beta
Coherent geometry, wrong grounding	94.3	99.2	88.5	92.9	93.3
Direction/sign error	0.0	0.0	1.6	1.6	3.3
Magnitude/count error	0.0	0.0	4.7	0.0	1.7
Semantic-only matching	0.0	0.0	0.0	0.0	1.7
Low-confidence guess	5.0	0.0	0.0	0.0	0.0
Format/parse error	0.7	0.8	5.2	5.5	0.0

The downstream stages use the same number of steps and the same data. The initial checkpoints have different prior training histories, so this controls downstream training conditions rather than total training cost.

E Failure Analysis

We complement the aggregate analyses of Sections 3.2 and 5 with a failure analysis over 7,900+ released rollouts from five frontier models (GPT-5.4, GPT-5.4 Pro, Gemini 3.1 Pro, Claude Opus 4.6, and Grok 4.20 Beta) across all three tasks, together with our trained Qwen2.5-VL-7B on IVP. We compute trajectory- and outcome-level metrics automatically over all rollouts, and manually inspect a stratified sample of several hundred model-generated reasoning traces to categorize single-turn failures. The manual categories describe the content of the generated explanations and their relation to the selected answers. We report these proportions as exploratory descriptions of the inspected sample.

E.1 Single-Turn Failures on P2V and V2P

For the single-turn tasks we assign each inspected failure to one of six mutually exclusive categories based on its reasoning trace:

- **Coherent geometry, wrong grounding.** The stated transformation explanation is internally consistent, but the selected image is incorrect.
- **Direction/sign error.** The generated explanation reverses the direction of a rotation or translation.
- **Magnitude/count error.** The generated explanation contains an incorrect step count or transformation magnitude.
- **Semantic-only matching.** The explanation compares candidates by object semantics without stating a geometric argument.
- **Low-confidence guess.** The trace explicitly expresses uncertainty and guesses.
- **Format/parse error.** The response cannot be parsed into a valid answer.

Tables 27 and 28 report the distribution for each model. The most frequent category on both tasks, for every model, is *coherent geometry, wrong grounding* (74.8-99.2% of inspected failures). Explicit direction or step-count errors in the traces are rare, with the exception of GPT-5.4's 21.6% magnitude/count errors on V2P, and semantic-only matching appears mainly for Claude Opus 4.6 on V2P (18.0%). An apparently coherent transformation explanation is often accompanied by an incorrect visual answer.

E.2 Multi-Turn Failures on IVP

For IVP we use a hierarchical, mutually exclusive taxonomy computed automatically from each episode's actions and final pose estimate. We first identify episodes with **no valid answer** (the agent never commits a pose estimate within the turn budget) and an **immediate answer** (the agent submits an estimate immediately, without exploring). Among the remaining episodes we separate **orientation flips** (rotation error $\geq 150°$), and categorize the rest by which component of the $0.5\,\text{m}$ / $30°$ success threshold is violated: **position and rotation off**, **position off only**, or **rotation off only**. Table 29 reports the distribution over failed episodes for the five frontier models and our trained model.

Table 28: **V2P failure categories.** Percentage of inspected V2P failures per category for each frontier model. Grounding errors again dominate; GPT-5.4 additionally shows step-count errors and Claude Opus 4.6 semantic-only matching.

Failure category	GPT-5.4	GPT-5.4 Pro	Gemini 3.1 Pro	Claude Opus 4.6	Grok 4.20 Beta
Coherent geometry, wrong grounding	74.8	94.2	86.9	80.1	90.0
Direction/sign error	1.7	0.0	0.0	0.0	6.7
Magnitude/count error	21.6	5.0	0.0	0.0	0.0
Semantic-only matching	0.0	0.0	1.6	18.0	3.3
Low-confidence guess	1.7	0.0	6.3	0.0	0.0
Format/parse error	0.3	0.8	5.2	1.9	0.0

Table 29: **IVP failure categories.** Percentage of each model's failed episodes in hierarchical, mutually exclusive categories. Classification rules are given in the text; the trained-model column contains 277 failed episodes.

Failure category	GPT-5.4	GPT-5.4 Pro	Gemini 3.1 Pro	Claude Opus 4.6	Grok 4.20 Beta	Ours
No valid answer	0.0	6.1	1.0	0.0	31.4	0.4
Immediate answer	2.5	0.0	0.2	0.0	24.0	0.0
Orientation flip ($\geq 150°$)	7.2	9.1	10.8	15.4	4.1	7.6
Position and rotation off	49.3	45.2	50.6	55.8	21.5	49.1
Position off only	37.8	37.3	35.3	25.2	17.2	40.4
Rotation off only	3.2	2.3	2.2	3.6	1.8	2.5

For GPT-5.4 Pro, the 23 refused episodes are counted as failures with no valid answer, giving 429 failed episodes out of 530 test pairs. This matches the refusal handling in the main success-rate evaluation.

Two patterns emerge. First, *position error is common among failed estimates*. For models with low premature-termination rates, 81% to 87% of failures involve a position error, whereas rotation-only errors account for only 2% to 4%; the median position error is about 1 m, twice the 0.5 m threshold. This is consistent with the association between position distance and IVP success in Appendix B.2. Second, the interactive setting exposes *model-specific behaviors* that static QA cannot reveal: Grok 4.20 Beta frequently either never commits (31.4% of failures) or answers immediately (24.0%); Claude Opus 4.6 often reverses its own actions (in 90% of its failed episodes); and inspected Gemini 3.1 Pro trajectories include extensive exploration followed by pose estimates with apparent coordinate-frame sign errors.

E.3 Residual Failures of the Trained Model

Among the trained model's 277 failed episodes, none submits an answer immediately without exploring, and 0.4% ends without a valid pose estimate (Table 29). The residual failures concentrate on more distant targets (Table 30): success decreases across five unified-view-distance bins, from 71.1% for $d < 2$ to 28.7% for $d \geq 5$.

On long, multi-translation targets, inspected trajectories often contain repeated rotations followed by insufficient translation toward the target. Position is involved in 89.5% of residual failures (median position error 0.96 m), although many misses are near the threshold. The median angular error rounds to 30.0°; 38 of the 277 failed episodes have computed angular errors between 30° and 30.01°, just above the inclusive 30° threshold, and are scored as rotation failures under the same rule used for every model. In addition, 44% of failures end within 1 m and 60° of the target. Failed episodes use 9.8 of 10 turns on average, and 53% revisit at least one previously observed pose. These are descriptive properties of the trained model's residual failures, not evidence that training caused a particular internal planning strategy.

Table 30: **Residual failures concentrate on distant targets.** IVP success (%) of the trained Qwen2.5-VL-7B on the 530 VIEWSUITE test pairs, binned by unified view distance d between the initial and target poses.

View distance d	n	Success
$d < 2$	83	71.1
$2 \leq d < 3$	104	63.5
$3 \leq d < 4$	115	46.1
$4 \leq d < 5$	99	38.4
$d \geq 5$	129	28.7

F GENERALIZATION TO NEW ENVIRONMENTS: AI2-THOR AND HABITAT-GS

VIEWSUITE is built on real ScanNet scans. We extend the study to two further environments with complementary roles: AI2-THOR (synthetic, indoor) tests whether the training recipe improves a VLM when reproduced on a new scene corpus, while Habitat-GS (photorealistic Gaussian-splatting reconstructions, indoor and outdoor) tests whether the planning gap also appears in a broader evaluation setting. The AI2-THOR experiment uses three iterative training rounds on that environment's own training scenes, without the additional final RL stage used for the primary VIEWSUITE run. Habitat-GS evaluates frontier models and an untrained base model, with no model trained on Habitat-GS.

F.1 AI2-THOR

We reproduce the full data-collection, evaluation, and training pipeline on AI2-THOR (iTHOR) (Kolve et al., 2017), an environment that differs from VIEWSUITE along several axes: *synthetic* rather than real-scanned scenes, *mesh* rather than point-cloud rendering, and a fully disjoint scene corpus. This experiment evaluates the applicability of the training recipe beyond ScanNet, with a separate model trained on AI2-THOR and evaluated on its held-out scenes.

Environment and splits. We use 120 iTHOR scenes balanced across four room types (30 each: kitchen, living room, bedroom, bathroom), and build scene-disjoint train/dev/test splits of 84/18/18 scenes. Running the VIEWSUITE sampling and filtering pipeline (Appendix A.3) yields $\sim$1,636 instances per task; the IVP test set contains 252 view pairs with an average ground-truth path length of 4.25 actions. Under the action-length split used in Table 31, 38 pairs are Short ($\leq$ 2 actions) and 214 are Long ($>$ 2 actions). The action space, turn budget, success threshold, and evaluation protocol are identical to VIEWSUITE.

The planning gap replicates. Table 31 evaluates four frontier models zero-shot. As on VIEWSUITE, they achieve substantially higher success on the single-turn tasks, including up to 77.0% overall on P2V and 94.7% on Short V2P, than on multi-turn IVP, where success decreases further on long-horizon instances. GPT-5.4 is the strongest model here at 36.1% IVP; the other three fall between 9.9% and 17.1%. On AI2-THOR, the frontier VLMs again perform better at interpreting supplied paths than at choosing camera actions through interaction.

The recipe also works on AI2-THOR. We train Qwen2.5-VL-7B with three iterative self-exploration and view graph distillation rounds on the 84 training scenes, omitting the additional final RL stage used in the primary VIEWSUITE run. The untrained base obtains 5.6% IVP success, compared with 6.2% for random navigation. After three iterations, the separately trained model reaches 61.1% IVP success (Short 73.7% / Long 58.9%), compared with 36.1% for the strongest evaluated zero-shot model on the same held-out AI2-THOR test set.

F.2 HABITAT-GS: PHOTOREALISTIC GAUSSIAN-SPLATTING SCENES, INDOOR AND OUTDOOR

Both VIEWSUITE and AI2-THOR are indoor. Habitat-GS (Xia et al., 2026) is a Habitat-Sim variant that renders 3D Gaussian-splatting (3DGS) reconstructions, and its scene corpus is the only

Table 31: **The planning gap and training gains recur on AI2-THOR.** Success rates (%) on 252 held-out test pairs (S/L split by ground-truth path length, ≤ 2 / > 2 actions, 38 / 214 pairs). Frontier models are evaluated zero-shot; the trained Qwen2.5-VL-7B uses three iterative rounds on AI2-THOR training scenes without the additional final RL stage used in the primary VIEWSUITE run. Overall is the mean of the three All columns. Best All per task in **bold**.

	Path-to-View (P2V)			View-to-Path (V2P)			Interactive View Planning (IVP)			
Model	S	L	All	S	L	All	S	L	All	Overall
Frontier models (zero-shot)										
GPT-5.4	65.8	75.2	73.8	94.7	65.0	69.4	47.4	34.1	36.1	59.8
Gemini 3.1 Pro	76.3	77.1	**77.0**	89.5	65.4	69.0	21.1	15.9	16.7	54.2
Grok 4.20 Beta	73.7	72.0	72.2	63.2	73.4	**71.8**	15.8	8.9	9.9	51.3
Claude Opus 4.6	39.5	57.9	55.2	65.8	55.1	56.7	28.9	15.0	17.1	43.0
Qwen2.5-VL-7B										
Random navigation	-	-	-	-	-	-	10.5	5.4	6.2	-
Untrained base	-	-	-	-	-	-	13.2	4.2	5.6	-
Ours (3 iterations)	-	-	-	-	-	-	73.7	58.9	**61.1**	-

one of our three worlds that is not exclusively indoor: 129 scenes in two families, InteriorGS interiors (`interior_*`) and sceneNN scenes, many of which are outdoor plazas and streets. Scene scale spans an order of magnitude (interiors have a $\sim$23 m navigable diagonal, sceneNN scenes a $\sim$90 m median and up to 538 m), so a fixed 0.5 m step would cross a room yet do nothing in a plaza; we therefore derive the translation step per scene from the navigable-area diagonal, clamped to $[0.25, 8]$ m, and state it in the prompt. Rotation and turn semantics follow Habitat (yaw-only body with a pitching sensor, no roll). Because a Gaussian reconstruction is only faithful near its training views, every sampled view is screened by a VLM judge for recognisability (3,096 generated samples, 1,874 kept); the released test split has 198 instances per task over 16 held-out scenes (9 interior, 7 sceneNN), with ground-truth path lengths of 3-10 actions and per-scene steps of 0.37-4.31 m. Turn budget, success threshold, and evaluation protocol are otherwise identical to VIEWSUITE.

The planning gap persists in a broader scene corpus. Table 32 evaluates the same four frontier models zero-shot. Single-turn performance is substantially higher (54.5-82.8% on P2V and 59.6-75.3% on V2P) than IVP success: the strongest model, GPT-5.4, reaches 29.8% on IVP, and the others fall between 12.6% and 20.2%; every model has lower IVP success on long paths. A descriptive split by scene family gives InteriorGS versus sceneNN IVP success of 28.7/31.3% for GPT-5.4, 18.3/14.5% for Gemini 3.1 Pro, 12.2/13.3% for Grok 4.20 Beta, and 18.3/22.9% for Claude Opus 4.6. Because sceneNN is a dataset family rather than an outdoor-only subset, these numbers are not a controlled indoor-versus-outdoor comparison. The aggregate results show that the planning gap also appears in a corpus that includes outdoor environments and larger movement scales.

As reference points, the untrained Qwen2.5-VL-7B reaches 39.4% on P2V and 27.3% on V2P but only 0.5% on IVP, compared with 1.0% for a random-navigation policy.

G SCOPE OF THE CAPABILITY AND LEARNING COMPARISONS

G.1 VIEW PLANNING AS A VLM CAPABILITY AND LEARNING PROBLEM

We study how a general-purpose VLM learns to connect images, camera movements, and pose estimates through interaction. The same model reads the target and current views, chooses actions, interprets their outcomes, and submits a pose estimate. VIEWSUITE evaluates its understanding of supplied paths and its ability to construct paths on the same view pairs. The learning experiments then ask how the VLM's own exploration can supply supervision when it rarely reaches the original task's reward.

IVP scores the reported target pose. Approaching a matching view and reporting the environment-provided camera pose is one valid strategy; estimating the target pose from other observations is another. The metric evaluates the final answer, while the trajectory analyses describe the observations and actions preceding it.

Table 32: **Capability evaluation on Habitat-GS, a corpus including outdoor scenes.** Success rates (%) on the released Habitat-GS test split (198 instances per task, 16 scenes). Random navigation explores for the full budget and submits a randomly chosen visited pose. IVP S/L is split by ground-truth path length (≤ 5 / ≥ 6 actions, 64 / 134 pairs; the split has no paths shorter than 3). Frontier models are evaluated zero-shot with medium reasoning effort; the untrained Qwen2.5-VL-7B base uses the same prompts without a reasoning-effort setting. No model is trained on Habitat-GS. Overall is the mean of the three All columns. Best All per task in **bold**.

	Path-to-View (P2V)			View-to-Path (V2P)			Interactive View Planning (IVP)			
Model	S	L	All	S	L	All	S	L	All	Overall
Frontier models (zero-shot)										
GPT-5.4	93.8	77.6	**82.8**	82.8	67.2	72.2	39.1	25.4	**29.8**	61.6
Gemini 3.1 Pro	75.0	73.9	74.2	79.7	63.4	68.7	18.8	15.7	16.7	53.2
Grok 4.20 Beta	67.2	76.1	73.2	70.3	77.6	**75.3**	14.1	11.9	12.6	53.7
Claude Opus 4.6	59.4	52.2	54.5	76.6	51.5	59.6	21.9	19.4	20.2	44.8
Reference points										
Random navigation	-	-	-	-	-	-	1.6	0.7	1.0	-
Qwen2.5-VL-7B (untrained)	46.9	35.8	39.4	39.1	21.6	27.3	0.0	0.7	0.5	22.4

G.2 Scope of the Method Comparisons

The zero-shot evaluations characterize the capabilities of existing VLMs under the shared interface. The training comparisons study different ways to improve the same base VLM: direct RL, distillation from successful rollouts, and supervision constructed from a broader collection of exploration trajectories. Their training schedules are reported in Appendix C.

We also evaluate two vision-only approaches as reference points for task difficulty: image-similarity search and relative-pose estimation (Table 33). These experiments use the same 530 test pairs, discrete camera actions, and submitted-pose success thresholds as the main evaluation. The available observations differ from the VLM interface, as described below. The learning experiments remain focused on improving general-purpose VLMs, with the view graph used to construct training data and the learned VLM acting from its interaction history at test time.

Image-similarity search. At each step, we render candidate successor views and score their cosine similarity to the target image using DINOv2 (Oquab et al., 2024). One-step lookahead evaluates the 12 single-action successors; two-step lookahead evaluates 144 two-action sequences. The policy moves to the best candidate while similarity improves, then submits the environment-returned pose. This search observes candidate views before choosing an action, whereas the VLM observes the view after executing its chosen actions. Success is 5.8% with one-step lookahead and 9.1% with two-step lookahead. One-step search stops after 2.8 moves on average and succeeds mainly on Short targets (15.5%, compared with 0.6% on Long targets).

Relative-pose estimation. MASt3R (Leroy et al., 2024) estimates the target camera relative to the initial camera from the two images. We combine this estimate with the known initial pose, snap the resulting target pose to the action grid, and execute the movement open loop. With the benchmark images, success is 8.1%; using only the predicted rotation gives 7.0%. The median rotation error before snapping is 83°.

We separately evaluate MASt3R with renderer-consistent images of both the initial and target poses. This condition reaches 27.2%, compared with 8.1% using the benchmark images. It uses an oracle target image: generating that image requires the target pose, which is unavailable to the agent. We therefore report it as a separate input condition in the table.

Table 33: **Vision-only reference methods.** IVP success (%) on the main test set. Observation access and input conditions are specified in Appendix G.2.

Method	Short	Long	All
Submit the initial pose	6.4	0.0	2.3
Random Response	2.1	0.0	0.8
DINOv2 search, one-step lookahead	15.5	0.6	5.8
DINOv2 search, two-step lookahead	19.8	3.2	9.1
MASt3R, benchmark images	19.3	2.0	8.1
Predicted rotation only	16.6	1.7	7.0
Qwen2.5-VL-7B (base)	7.0	0.0	2.5
Gemini 3.1 Pro	28.9	17.2	21.3
Ours (Qwen2.5-VL-7B)	66.8	37.3	47.7
Oracle input condition			
MASt3R, renderer-consistent images	58.8	9.9	27.2