VERSE: VERIFIABLE SPATIAL SELF-EVOLUTION WITH GEOMETRIC PROCESS REWARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Spatial reasoning remains a fundamental bottleneck for vision–language models (VLMs), largely because models reason purely in 1D text tokens without external geometric anchors, whereas humans routinely sketch and mark up scenes. Equipping VLMs with agentic visual tools narrows this gap, yet training such tools traditionally demands prohibitively expensive human-annotated reasoning trajectories. While the self-evolution paradigm eliminates human annotation in code and mathematics, directly porting it to tool-assisted spatial reasoning encounters a critical failure mode: outcome-only rewards are completely blind to intermediate tool fidelity. Consequently, a policy can draw arbitrary annotations or cross physical obstacles, yet still be falsely reinforced via lucky textual guesses. To address this, we propose VERSE, an annotation-free framework that trains a VLM to actively propose, draw, and self-evolve within 3D environments. At the core of VERSE is the Agentic Verifier: by exploiting underlying 3D scene metadata and procedural topologies, it evaluates intermediate visual tool calls in closed form via computing deterministic grounding IoUs for bounding boxes and sequence-alignment fidelities for traced routes. This yields dense, engine-verified process rewards without relying on learned reward models or LLM judges. Extensive experiments across video, multi-view, and navigation domains demonstrate substantial improvements across scales: on Qwen2.5-VL-7B, VERSE elevates VSI-Bench accuracy from 33.68% to 40.52% and boosts MAZE performance from 33.50% to 90.80%, confirming that engine-verified process rewards provide the critical supervisory signal for multimodal self-improvement.

1 INTRODUCTION

Spatial reasoning is fundamental to human intelligence, underlying capabilities from tool manipulation and indoor navigation to scientific discovery. While large language models (LLMs) have achieved remarkable breakthroughs in complex symbolic and mathematical reasoning (Guo et al., 2025), contemporary vision–language models (VLMs) remain strikingly brittle when navigating, measuring, and reasoning in physical 3D space (Yang et al., 2025; Zheng et al., 2025). A prominent bottleneck lies in the reasoning medium: human spatial cognition routinely relies on *external visual scaffolding*, such as sketching intermediate marks, tracing prospective routes, and iteratively checking visual feedback. In contrast, standard VLMs reason strictly within a 1D tokenized text space, forcing rich 3D geometries into linguistic descriptions where metric precision and topological constraints are inevitably lost.

To bridge this cognitive gap, two promising paradigms have emerged, as conceptualized in Figure 1. The first equips VLMs with *agentic visual tools* (e.g., bounding-box mappers, path tracers) that interleave text generation with visual drawing actions (Hu et al., 2024; Li et al., 2025b; Wu et al., 2025). By externalizing geometric hypotheses directly onto the image, the agent gains inspectable visual states against which to self-correct. However, teaching models to use these tools effectively has traditionally been bottlenecked by scarce, expensive human-annotated trajectories. The second paradigm is *autonomous self-evolution*, which eliminates human supervision by enabling models to self-author problems, verify ground truth, and optimize via reinforcement learning (RL) (Zhao et al., 2025; Xia et al., 2026; Wang et al., 2026). In spatial domains, SpatialEvo (Li et al., 2026a) realizes this by employing a deterministic geometric environment (DGE) to synthesize verifiable scene-level answers directly from 3D metadata.

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.

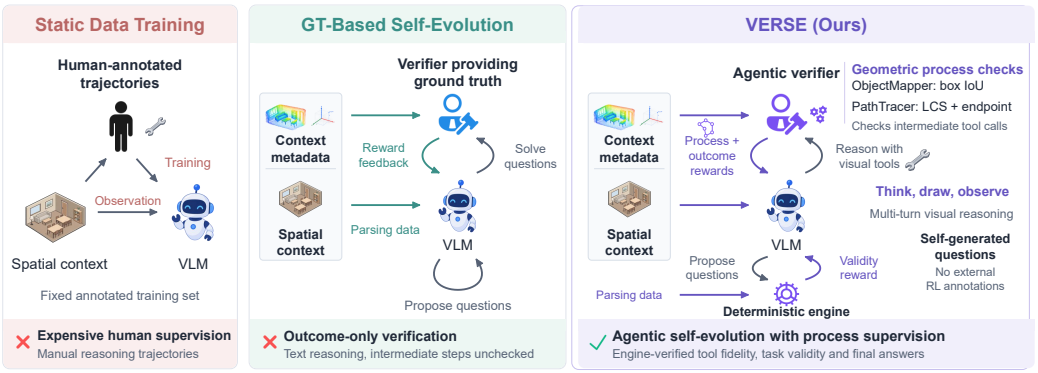


Figure 1: **Spatial reasoning training paradigms.** Left: static human-annotated trajectories. Middle: self-evolution with verified answers but unchecked intermediate reasoning. Right: VERSE verifies task outcomes and the geometric fidelity of visual tool actions.

However, simply coupling agentic visual tools with self-evolution does not yield faithful spatial reasoning. The root issue is that a standard verifier grades only the *final answer*. An outcome-only reward is structurally blind to the intermediate drawing process: a policy that mislocalizes an object or traces a route straight through obstacles, but happens to arrive at the correct answer through a lucky textual guess, receives the exact same maximum reinforcement as a model that grounds every step faithfully (Yu et al., 2026). Deprived of process supervision, the policy has no incentive to draw faithfully and is reinforced to treat visual tools as decorative tokens or ungrounded artifacts, leading to severe reward hacking and training collapse. What multimodal self-evolution fundamentally requires is a *verifiable process reward* that enforces the geometric fidelity of each intermediate tool execution.

In this work, we propose **VERSE** (**VER**ifiable **S**patial Self-**E**volution with Geometric Process Rewards), an agentic framework that allows VLMs to acquire grounded visual tool capabilities through autonomous self-play in 3D and maze environments. Operating on posed ScanNet scenes (Dai et al., 2017) and procedural mazes, VERSE implements a two-round proposer–solver loop governed by two complementary deterministic verifiers. Crucially, we introduce the **Agentic Verifier**: leveraging known 3D context metadata and grid topologies, it evaluates intermediate visual tool calls *in closed form*. Specifically, it computes dense *grounding fidelity* (bounding-box IoU against ground-truth objects) for object localization and *path fidelity* (longest common subsequence overlap and endpoint matching) for path navigation. Because this process signal is computed purely from deterministic geometric engines, it introduces zero learned reward-model drift and requires no human step annotations.

Our core contributions are summarized as follows:

- **Annotation-Free Spatial Self-Evolution:** We present VERSE, which unifies agentic visual tool usage with self-evolution over 3D contexts and eliminates external annotated trajectories from the compute-intensive RL stage, after supervised tool-use initialization.
- **Engine-Verified Geometric Process Rewards:** We design the *Agentic Verifier*, providing the first closed-form, deterministic process supervision for multimodal visual tool calls—instantiated via grounding fidelity for bounding boxes and path fidelity for trajectory tracing.
- **Superior Benchmark Gains Across Scales:** Across 3B and 7B scales, VERSE delivers strong benchmark gains: boosting Qwen2.5-VL-7B to 40.52% on VSI-Bench and from 33.50% to 90.80% on MAZE, while empirically showing that process verification discourages deceptive tool use and improves policy learning.

2 Related Work

Spatial Reasoning in Vision–Language Models. Spatial reasoning remains an enduring challenge for multimodal models across diverse formats, as rigorously benchmarked by Zheng et al.

(2025), including video spatial intelligence (Yang et al., 2025), single- and multi-view spatial relations (Zhang et al., 2025), and algorithmic maze navigation (Ivanitskiy et al., 2023). To bridge this gap, one major line of research scales up spatially grounded data: SpatialVLM synthesizes metric question–answer pairs from web imagery (Chen et al., 2024), GPT4Scene incorporates bird's-eye-view renderings with explicit spatial markers (Qi et al., 2026), SpaceR constructs verifiable spatial QA datasets via ScanNet scene geometry (Ouyang et al., 2025), and Smooth Operator shapes rewards for continuous spatial predictions (Jiao et al., 2026). VADAR uses a dynamically generated tool API for multi-step 3D spatial reasoning (Marsili et al., 2025). More recently, an emerging paradigm enables models to directly *act and sketch on visual inputs*: Visual Sketchpad employs sketching as an external visual scratchpad (Hu et al., 2024), and Visualization-of-Thought renders intermediate visual states to aid spatial deductions (Li et al., 2025b). Most related to our work, ViLaSR introduces an agentic drawing interface with reflective rejection sampling and reinforcement learning (Wu et al., 2025). ViLaSR (Wu et al., 2025) bootstraps drawing from teacher-generated reasoning trajectories and continues Stage-3 RL on a fixed annotated question set, while its RL reward checks answer correctness and format or tool executability rather than the geometric fidelity of intermediate drawings. In contrast, VERSE reuses supervised initialization but removes annotated data from Stage-3 RL: questions and answers are generated and certified online, and intermediate tool executions are evaluated geometrically.

The Self-Evolution Paradigm. Autonomous self-evolution enables models to transcend static human data by generating their own curricula and learning from automated verification. This paradigm has achieved remarkable success in formal domains such as mathematics and programming, where code interpreters provide unambiguous ground-truth verification (Zhao et al., 2025; Xia et al., 2026; Stojanovski et al., 2025). Recent efforts have begun extending self-evolution to multimodal domains: Vision-Zero elicits visual reasoning through gamified self-play (Wang et al., 2026), MM-Zero investigates label-free multimodal reinforcement learning (Li et al., 2026b), SpatialEvo synthesizes 3D spatial QA from deterministic geometric environments (Li et al., 2026a), and GeoX explores verifiable geospatial self-play (Ahn et al., 2026). SAGE uses geometric consistency as a training reward (Liu et al., 2026b), while Agent0-VL incorporates tools into self-evaluation and repair (Liu et al., 2026a) and NeSy-Spatial evolves reusable tool and geometry skills (Tian et al., 2026). Unlike these approaches, VERSE couples agentic visual drawing actions to a deterministic geometric environment and assigns closed-form rewards to intermediate tool executions during autonomous self-evolution.

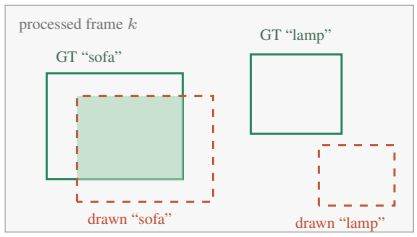
Verifiable and Process-Level Rewards. Reinforcement learning with verifiable outcome rewards has become the cornerstone of modern reasoning models, replacing noisy learned reward models with deterministic programmatic checks (Guo et al., 2025); in ungrounded domains, spurious agreement from majority voting is penalized by methods like RESTRAIN (Yu et al., 2026). To provide finer-grained guidance, process reward models (PRMs) supervise intermediate deduction steps (Lightman et al., 2024; Wang et al., 2024). However, existing PRMs rely on learned proxy models trained on step-level human annotations or costly Monte Carlo completions, leaving them susceptible to reward model drift and hallucination, especially in vision-language tasks. VERSE departs fundamentally from learned PRMs: because our visual tool calls represent concrete geometric primitives and the environment engine possesses full 3D ground truth, each drawing operation is verified deterministically in closed form via bounding-box intersection-over-union or maze path alignment, yielding dense, drift-free process supervision without human annotation.

3 METHODOLOGY

VERSE is an agentic self-evolution framework that trains vision–language models to interleave textual reasoning with visual drawing and self-improves via closed-form geometric process supervision. This section describes: (1) the agentic tool interface and interactive formulation; (2) the deterministic *Agentic Verifier*; and (3) the closed-loop training pipeline powered by dual-group GRPO.

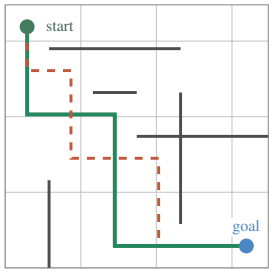
3.1 PROBLEM FORMULATION AND AGENTIC TOOL INTERFACE

Interactive Trajectory Formulation. Standard VLMs operate as passive observers, mapping a visual context $I = \{I_n\}_{n=1}^{N}$ ($N = 1$ for single images, $N > 1$ for video or multi-view sequences)



$\mathrm{IoU(sofa)} = 0.44,\ \mathrm{IoU(lamp)} = 0 \Rightarrow \Phi_{\mathrm{Mapper}} = 0.22$

(a) **Grounding fidelity (OBJECTMAPPER).** Mean per-label maximum IoU of drawn (dashed red) and ground-truth (solid green) boxes.



$\mathrm{LCS}(\hat{P}, P^\star)/|P^\star| = 0.45,\ \text{endpoint missed} \Rightarrow \Phi_{\mathrm{Tracer}} = 0.27$

(b) **Path fidelity (PATHTRACER).** Grid-snapped path overlap (LCS) and destination matching against the engine path.

Figure 2: **Agentic Verifier.** Green marks engine ground truth; red marks tool actions. Closed-form process rewards $\Phi \in [0, 1]$ score drawing fidelity without learned reward models or LLM judges.

and a language query Q directly to a textual output. In contrast, VERSE models spatial reasoning as an interactive, multi-turn Markovian process. At each reasoning step t, the policy π_θ generates an internal thought token sequence r_t followed by a set of executable visual tool calls e_t, and receives a re-rendered visual observation o_t as environment feedback. The full reasoning trajectory is formulated as:

$$R = \{(r_t, e_t, o_t)\}_{t=1}^{T}, \qquad e_t = \left\{(k_t^j,\, p_t^j,\, \ell_t^j)\right\}_{j=1}^{m_t}, \tag{1}$$

where each drawing primitive e_t specifies a target image index $k_t^j \in \{1, \ldots, N\}$, continuous geometric parameters p_t^j, and a semantic label ℓ_t^j. The trajectory terminates when the agent produces the terminal answer token $\hat{A}$. Environment observations o_t are dynamically appended to the conversation context and masked out during policy gradient backpropagation.

Visual Tool Primitives. Following the tool interface of ViLaSR (Wu et al., 2025), we provide the agent with two complementary, deterministic drawing tools:

- **OBJECTMAPPER:** Designed for discrete object localization. It accepts a list of image indices, 2D bounding boxes $p = [x_{\min}, y_{\min}, x_{\max}, y_{\max}]$, and category labels ℓ, rendering explicit colored bounding boxes and text tags onto the designated frame.
- **PATHTRACER:** Designed for topological and spatial navigation. It accepts start–end coordinate pairs $p = [x_{\text{start}}, y_{\text{start}}, x_{\text{end}}, y_{\text{end}}]$ with image indices and labels, drawing connected piecewise-linear trajectories.

Crucially, both renderers are non-parametric and deterministic. Coordinates are defined in absolute pixel space corresponding to the vision encoder's processed resolution, ensuring strict visual-geometric parity between what the model emits and what it observes in subsequent turns.

3.2 THE AGENTIC VERIFIER

The fundamental challenge in self-supervised spatial tool learning is that outcome-only verification is invariant to intermediate drawing fidelity: a policy can draw arbitrary boxes or path segments, yet still stumble upon the correct answer by chance. The core innovation of VERSE is the **Agentic Verifier** (Φ), which converts underlying geometric ground truth into dense, closed-form process rewards on intermediate tool calls $\{e_t\}$, completely decoupled from final answer correctness. We formulate two concrete instantiations, illustrated in Figure 2.

Grounding Fidelity (OBJECTMAPPER). For 3D scene tasks referencing physical objects, the deterministic geometric environment knows the exact 2D projection boxes for all ground-truth instances. Let $\mathcal{L}$ denote the set of query-relevant entity labels, $\mathrm{drawn}(\ell)$ the set of boxes drawn for label ℓ, and $\mathrm{GT}(\ell)$ the corresponding ground-truth bounding boxes across visible frames (using

case-insensitive substring matching). The grounding fidelity is defined as:

$$\Phi_{\text{Mapper}} = \frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} \max_{b \in \text{drawn}(\ell)} \max_{g \in \text{GT}(\ell)} \text{IoU}(b, g), \tag{2}$$

where $\text{IoU}(b, g) = \frac{\text{area}(b \cap g)}{\text{area}(b \cup g)}$, and the maximum over an empty set is defined as zero ($\max \emptyset = 0$) if a label is omitted. By computing the maximum IoU per label, Eq. equation 2 requires the model to faithfully localize each referenced entity at least once, penalizing ungrounded hallucinations while remaining robust to frame indexing conventions.

Path Fidelity (PATHTRACER). Path Tracer actions in both scene and maze tasks can be checked against the underlying geometry. For the grid-based navigation setting, the agent's continuous drawn coordinates are normalized by image margins and mapped to discrete grid cells. Deduplicating consecutive identical cells yields the executed trajectory $\hat{P} = (\hat{P}_1, \ldots, \hat{P}_{|\hat{P}|})$. Denoting the engine's canonical path as $P^\star = (P^\star_1, \ldots, P^\star_{|P^\star|})$, path fidelity balances structural route adherence with destination accuracy:

$$\Phi_{\text{Tracer}} = \lambda \frac{\text{LCS}(\hat{P}, P^\star)}{|P^\star|} + (1 - \lambda)\, \mathbf{1}\left[\hat{P}_{-1} = P^\star_{-1}\right], \qquad \lambda \in [0, 1], \tag{3}$$

where $\text{LCS}(\hat{P}, P^\star)$ computes the length of the longest common subsequence between executed and canonical paths, normalizing by path length $|P^\star|$ to reward topological correctness despite minor local deviations. The indicator $\mathbf{1}[\cdot]$ grants full credit only if the final cell $\hat{P}_{-1}$ hits the destination $P^\star_{-1}$. We fix $\lambda = 0.6$ throughout all experiments.

3.3 AUTONOMOUS SELF-EVOLUTION PIPELINE

As illustrated in Figure 3, VERSE follows a three-stage curriculum: cold-start tool syntax learning, reflective self-correction SFT, and dual-group reinforcement self-evolution.

Cold-Start and Reflective SFT. In the first stage, **Cold-start SFT** trains the base model on 34,745 teacher-generated, annotated tool-use trajectories to acquire tool syntax and JSON schemas, following ViLaSR (Wu et al., 2025). However, syntax alone does not induce interactive reasoning. To cultivate active visual verification, **Reflective SFT** fine-tunes the policy on 7,912 filtered annotated trajectories that meet three strict criteria: format validity, final-answer correctness, and the presence of an active re-drawing event (modifying geometric coordinates across successive turns upon visual feedback). These relatively inexpensive supervised stages initialize tool syntax and reflection; our annotation-efficiency claim applies specifically to the subsequent, substantially more compute-intensive RL stage, which consumes no annotated question–answer or reasoning trajectories. The resulting checkpoint serves as the initial policy and forms the frozen reference anchor π_{ref} for regularizing subsequent RL.

Context Sampling and Modality-Homogeneous Scheduling. During self-evolution, each step samples a 3D context C—either posed ScanNet (Dai et al., 2017) video frames with associated 3D bounding meshes or a procedurally generated maze—alongside a task type τ. A round-robin scheduler pins exactly one task type per gradient update. This design enforces batch homogeneity, preventing gradient interference caused by mixing disparate geometric modalities within a single step.

Structured Proposer and Engine Certification. Given (C, τ), the policy acts as a Proposer, sampling candidate questions $\{q_i\} \sim \pi_\theta(\cdot \mid C, \tau)$. To prevent the proposer from collapsing into trivially answerable questions, the model operates in *structured mode*: it generates only symbolic semantic parameters (e.g., target object categories or goal coordinates), which are instantiated into linguistic queries Q_i via canonical task templates. The deterministic geometric environment (Li et al., 2026a) verifies the spatial validity of Q_i and computes the exact ground-truth answer: $(v_i, A_i) = \mathcal{V}(C, Q_i, \tau)$. For valid questions ($v_i > 0$), the proposer receives reward $r_i^Q = \lambda_f S_{\text{fmt}}(q_i) + (1 - \lambda_f) v_i S_{\text{obs}}(q_i)$. Here $S_{\text{obs}} \in \{1.0, 0.6, 0.3, 0.0\}$ is supplied by a lightweight external VLM, used as a text-only judge of the parsed observation given the task type and generated question; the deterministic validity term v_i gates this auxiliary quality score.

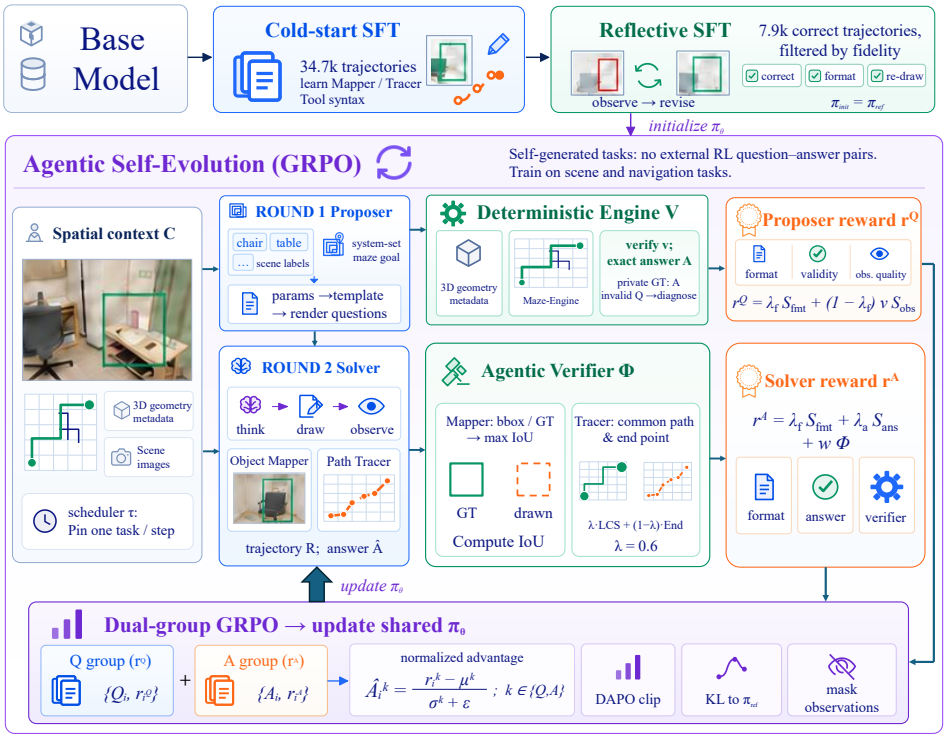


Figure 3: **VERSE pipeline.** Cold-start SFT learns tool syntax; reflective SFT learns to revise visual marks and yields π_{ref}. Self-evolution trains a proposer–solver loop with engine-verified outcome and process rewards.

Multi-Turn Solver with Agentic Process Rewards. For each validated task (Q_i, A_i), the model acts as a Solver, rolling out G_A interactive trajectories $R_i \sim \pi_\theta(\cdot \mid C, Q_i)$ through iterative 'think $\rightarrow$ draw $\rightarrow$ observe' loops. The solver trajectory is evaluated by:

$$r_i^A = \lambda_f\, S_{\text{fmt}}(R_i) + \lambda_a\, S_{\text{ans}}(A_i, \hat{A}_i) + w\, \Phi(R_i; C, Q_i), \qquad \lambda_f = 0.1, \quad \lambda_a = 0.9 - w, \tag{4}$$

where $S_{\text{fmt}} \in [0,1]$ checks multi-turn JSON schema compliance, $S_{\text{ans}} \in [0,1]$ computes exact match for discrete choices and mean relative accuracy for continuous spatial quantities (Yang et al., 2025), and $\Phi \in [0,1]$ is the deterministic process reward from Section 3.2. Setting the process weight $w = 0.2$ maintains a normalized reward scale ($\lambda_f + \lambda_a + w = 1$), directly rewarding geometrically faithful tool actions while preserving final task correctness as the dominant objective.

Dual-Group GRPO Optimization. Policy optimization is performed via Group Relative Policy Optimization (GRPO) (Shao et al., 2024). Because the proposer and solver address fundamentally different objectives with distinct reward distributions, we implement a *dual-group* formulation where advantages are normalized independently within each group $k \in \{Q, A\}$: $\hat{A}_i^k = (r_i^k - \mu^k)/(\sigma^k + \varepsilon)$. With importance ratio $\rho_{i,t}^k = \pi_\theta(y_{i,t}^k \mid x_i^k, y_{i,<t}^k)/\pi_{\text{old}}(y_{i,t}^k \mid x_i^k, y_{i,<t}^k)$, the joint self-evolution objective is:

$$\mathcal{L}_{\text{SE}}(\theta) = -\mathbb{E}_{C,\tau}\left[\sum_{k\in\{Q,A\}} \frac{1}{G^k} \sum_{i=1}^{G^k} \frac{1}{N_i^k} \sum_{t=1}^{N_i^k} \mathcal{C}\left(\rho_{i,t}^k,\, \hat{A}_i^k\right)\right] + \beta\, \text{KL}_{\text{lv}}(\pi_\theta \,\|\, \pi_{\text{ref}}), \tag{5}$$

where $\mathcal{C}$ denotes an asymmetric clipped surrogate (Yu et al., 2025) with clip boundaries $[\epsilon_l, \epsilon_h] = [0.2, 0.3]$ to encourage agentic exploration, and KL_{lv} is a low-variance KL penalty anchored to the reflective-SFT model π_{ref}. Gradient backpropagation masks out all environment observation tokens o_t, ensuring that parameter updates optimize solely for generative reasoning and grounded tool invocation.

Table 1: Main results (%) on VSI-Bench, SPAR-Bench, and MAZE. Best results are **bold**; second-best are underlined. Green values give the gain of each VERSE model over its untuned Qwen2.5-VL base.

Methods	Video									Multi-View				Image
	VSI-Bench									SPAR-Bench				MAZE
	Overall	Obj. Count	Abs. Dist.	Obj. Size	Room Size	Rel. Dist.	Rel. Dir.	Route Plan	Appr. Order	Overall	Low	Medium	High	
Qwen2.5-VL-3B (Bai et al., 2025)	29.02	36.31	20.30	17.22	28.02	34.65	43.39	27.84	24.43	28.62	24.07	27.16	32.90	26.70
Qwen2.5-VL-7B (Bai et al., 2025)	33.68	43.62	14.59	38.64	33.99	40.14	38.43	27.32	32.69	34.12	25.57	30.14	42.95	33.50
Kimi-VL-A3B-Instruct (16B) (Team et al., 2025)	34.14	32.25	29.15	47.74	**41.25**	35.35	37.29	29.38	20.71	33.75	27.74	30.03	40.33	49.30
LLaVA-OneVision-7B (Li et al., 2025a)	31.60	45.22	16.29	40.63	19.69	39.58	33.68	29.38	28.32	30.19	20.96	24.33	40.34	30.95
SpatialThinker-7B (Batra et al., 2025)	32.42	37.66	22.99	38.86	24.06	37.04	39.05	**30.93**	28.80	33.36	24.08	25.52	44.22	33.00
SpatialEvo-7B (Li et al., 2026a)	40.06	**53.96**	28.18	51.45	36.70	41.97	43.70	26.29	38.19	34.94	28.40	21.87	**45.11**	32.80
SpaceR-7B (Ouyang et al., 2025)	38.43	42.99	25.68	54.46	33.75	40.42	42.25	26.29	41.59	35.33	28.93	28.35	43.35	35.90
Ours														
Qwen2.5-VL-3B + VERSE	40.15 (+11.13)	49.84 (+13.53)	29.57 (+9.27)	52.82 (+35.60)	33.58 (+5.56)	35.77 (+1.12)	**44.21** (+0.82)	30.41 (+2.57)	**44.98** (+20.55)	33.51 (+4.89)	28.23 (+4.16)	28.50 (+1.34)	39.77 (+6.87)	87.50 (+60.80)
Qwen2.5-VL-7B + VERSE	**40.52** (+6.84)	45.10 (+1.48)	**30.14** (+15.55)	**54.56** (+15.92)	34.69 (+0.70)	**44.37** (+4.23)	**44.21** (+5.78)	28.87 (+1.55)	42.23 (+9.54)	**36.01** (+1.89)	**29.00** (+3.43)	**32.01** (+1.87)	43.55 (+0.60)	**90.80** (+57.30)

4 EXPERIMENT

4.1 EXPERIMENTAL SETUP

Models and Training Protocol. We evaluate VERSE with Qwen2.5-VL-3B and 7B. Both follow the three-stage curriculum in Section 3.3: cold-start SFT, reflective SFT, and dual-group GRPO over ScanNet (Dai et al., 2017) and procedural mazes (Ivanitskiy et al., 2023). Training contexts and hyperparameters appear in Table 4 and Appendix C.

Evaluation Benchmarks and Metrics. We cover three modalities: video spatial intelligence on **VSI-Bench** (Yang et al., 2025), multi-view relations on **SPAR-Bench** (Zhang et al., 2025), and image-based navigation on **MAZE** (Ivanitskiy et al., 2023). We report benchmark accuracy and the grounding- and path-fidelity metrics from Section 3.2.

4.2 MAIN RESULTS

Table 1 compares VERSE with open-weight multimodal foundations and specialized spatial baselines.

Overall Performance and Subtask Gains. Both scales improve across all three benchmarks. VERSE-7B reaches 40.52% on VSI-Bench (+6.84 over base), 36.01% on SPAR-Bench (+1.89), and 90.80% on MAZE (+57.30); VERSE-3B reaches 40.15% (+11.13), 33.51% (+4.89), and 87.50% (+60.80), respectively. VSI-Bench gains are largest for absolute distance (14.59% to 30.14%) and object size (38.64% to 54.56%), while room-size estimation changes only from 33.99% to 34.69%, consistent with the tools' localized geometric affordances.

4.3 ABLATION STUDY

To dissect the architectural components of VERSE, Table 2 presents systematic ablations on Qwen2.5-VL-7B across all VSI-Bench categories.

Component Ablations. Removing reinforcement self-evolution (Row 1) lowers overall accuracy from 40.52% to 34.93% (-5.59 points). The largest category drops occur in object-size estimation (54.56% to 46.47%) and route planning (28.87% to 23.79%), showing that the improvement is not explained by supervised tool-use initialization alone. Disabling proposer-side engine verification (Row 2) yields 38.61% (-1.91 points), consistent with the cost of admitting invalid or ambiguous generated tasks into training. Setting $w = 0$ (outcome-only RL, Row 3) yields 38.89% (-1.63 points). Although this aggregate decrease is smaller, outcome rewards cannot distinguish faithful intermediate drawings from lucky final answers; the process-level analyses below and in Appendix B examine this distinction directly.

Table 2: **VSI-Bench ablations on Qwen2.5-VL-7B (%).** Identical checkpoint schedules; best results in bold, second-best underlined.

Ablation Configuration	VSI-Bench (Video)								
	Overall	Obj. Count	Abs. Dist.	Obj. Size	Room Size	Rel. Dist.	Rel. Dir.	Route Plan	Appr. Order
Qwen2.5-VL-7B (Untuned Base)	33.68	43.62	14.59	38.64	33.99	40.14	38.43	27.32	32.69
VERSE (Full Framework, $w = 0.2$)	**40.52**	**45.10**	**30.14**	**54.56**	**34.69**	**44.37**	**44.21**	**28.87**	**42.23**
(1) w/o Self-Evolution (Reflective SFT only)	34.93	43.71	22.83	46.47	30.58	39.21	38.67	23.79	34.18
(2) w/o Ground-Truth Engine (Unverified Proposer)	38.61	<u>44.71</u>	27.54	51.83	<u>33.08</u>	<u>42.49</u>	42.06	26.02	<u>41.14</u>
(3) w/o Process Rewards (Outcome-Only RL, $w = 0$)	<u>38.89</u>	44.59	<u>28.19</u>	<u>52.46</u>	32.94	42.36	<u>42.73</u>	<u>26.84</u>	40.97

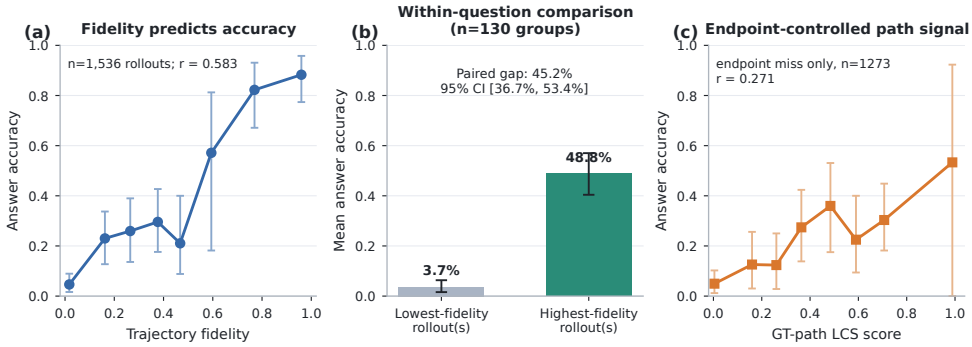


Figure 4: **Path-fidelity validation** ($n = 1,536$ **navigation rollouts).** **(a)** Fidelity correlates with accuracy ($r = 0.583$). **(b)** Paired questions show a 45.2-point gap ($n = 130$, $p < 0.001$). **(c)** Correlation persists for endpoint misses ($n = 1,273$, $r = 0.271$).

4.4 Analysis

Process Reward Mechanism Verification. Outcome-only rewards permit correct answers from ungrounded drawings, whereas the Agentic Verifier explicitly scores the executed geometric process. Across 1,536 navigation rollouts (Figure 4), path fidelity predicts task accuracy ($r = 0.583$, 95% CI $[0.496, 0.665]$): accuracy rises from approximately 5% in the lowest-fidelity bin to above 85% in the highest. To control for question difficulty, we additionally compare rollouts within 130 question-matched groups; the highest-fidelity rollouts outperform the lowest-fidelity ones by 45.2 percentage points ($p < 0.001$). Finally, among 1,273 rollouts that miss the destination, path overlap remains positively correlated with success ($r = 0.271$). Thus, the signal reflects intermediate topological progress rather than merely reproducing the endpoint indicator.

Training Dynamics. We trace the 7B self-evolution run for 200 steps. Figure 5 tracks two complementary aspects of the closed loop: whether the proposer increasingly authors valid questions and the solver follows the required response protocol, and whether the solver's visual actions become geometrically faithful.

Question validity and protocol compliance improve. Figure 5(a) reports, for non-overlapping 25-step intervals, the fraction of proposer questions certified by the deterministic engine and the solver's schema-compliance reward. The valid-question rate increases from 0.58 in the first interval to 0.71 in the last. Format reward rises in parallel from 0.177 to 0.239, indicating that training improves both task construction and adherence to the agentic interaction format rather than only one reward component.

Self-evolution improves grounding fidelity. Reward trends alone do not establish faithful visual-tool use. We therefore compare the solver's drawn boxes against engine ground-truth geometry independently of final-answer correctness (Figure 5(b)). Mean grounding fidelity increases from 0.318 in the first 20% of scene updates to 0.411 in the last 20%. Hit Rate@0.5 rises from 0.364

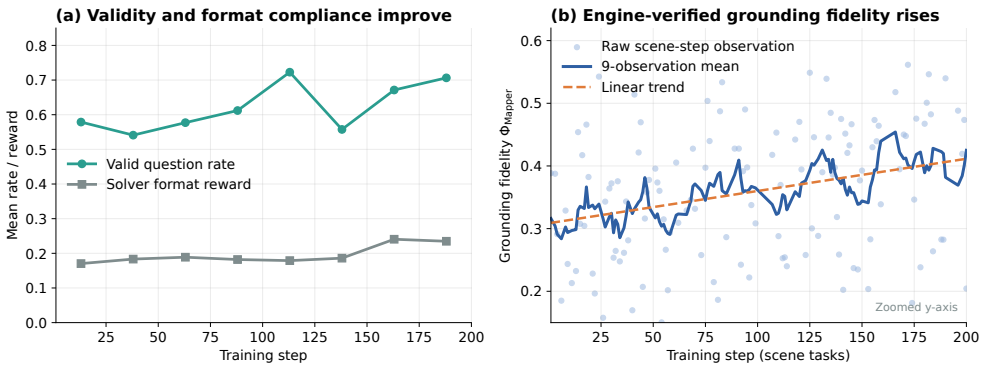


Figure 5: **Training dynamics of VERSE-7B (200 steps). (a)** Proposer validity and solver format reward in 25-step bins. **(b)** Grounding fidelity over 129 scene-task updates (points), with a centered 9-update mean (solid) and linear trend (dashed).

Table 3: **Grounding behavior of VERSE-7B.** First versus last 20% of 129 scene-task updates. Hit Rate@0.5 is the fraction of relevant objects with best drawn-box IoU ≥ 0.5; Coverage is the fraction with any drawn box.

Self-evolution stage	Mean Max-IoU ↑	Hit Rate@0.5 ↑	Coverage (%) ↑
First 20% of updates	0.318	0.364	60.9
Last 20% of updates	**0.411**	**0.467**	**68.1**
Relative change	+29.2%	+28.3%	+11.8%

Table 4: **Self-evolution context pool.** Unlabeled scene and maze contexts used across the 200-step run.

Context source	Contexts	Updates
ScanNet scenes	1,196	150
Procedural mazes	268	50
Total	1,464	200

to 0.467, while object coverage increases from 0.609 to 0.681 (Table 3). Individual updates remain noisy because they span different scenes and tasks, but both the smoothed trajectory and the linear trend improve over self-evolution.

Together, these diagnostics separate three behaviors that final benchmark accuracy conflates: generating answerable tasks, respecting the tool-use protocol, and grounding visual actions in scene geometry. Their concurrent improvement supports the intended mechanism of verified self-evolution, while the residual variance in Figure 5(b) indicates that geometric fidelity is informative but not sufficient by itself for a correct answer.

Consistency across task families. The two process analyses test distinct forms of faithfulness: topological path overlap for maze navigation and 3D box grounding for scene reasoning. In both cases, fidelity is evaluated separately from final-answer correctness. Together with the outcome-only ablation, their aligned trends indicate that process rewards provide supervision that cannot be recovered from the terminal answer reward alone.

5 Conclusion

We introduced VERSE, a spatial self-evolution framework combining structured question generation, deterministic ground truth, agentic visual tools, and engine-verified process rewards. It improves both 3B and 7B models across VSI-Bench, SPAR-Bench, and MAZE, demonstrating gains in video, multi-view, and navigation settings. The verifier makes intermediate grounding measurable and supplies faithful tool-execution supervision without labeled question–answer pairs or annotated RL trajectories. Training dynamics also show that concurrent gains in question validity, protocol compliance, and box grounding, while the maze analysis links higher path adherence to success. Together, the results show that verified process supervision teaches faithful tool use rather than merely exposing models to an interface and relying on tool access alone to produce grounded behavior.

AI USE STATEMENT

Generative AI tools assisted with code implementation and debugging, experiment execution, analysis and interpretation of experimental results, and language editing of this manuscript. The authors reviewed AI-assisted code, checked experimental outputs and analyses, and revised the final text. The authors take responsibility for the implementation, reported results, claims, and conclusions in this submission.

ETHICS STATEMENT

Our experiments use existing indoor-scene data and spatial-reasoning benchmarks, together with procedurally generated mazes. The observed benchmark gains do not establish that the trained model is safe or reliable for deployment in physical systems. Such use would require additional task-specific robustness and safety evaluation, especially in unseen environments. Use and redistribution of the underlying datasets and pretrained models should respect their applicable licenses, terms, and privacy requirements.

REPRODUCIBILITY STATEMENT

Sections 3 and 4 specify the method, datasets, evaluation protocols, metrics, and the self-evolution context pool. Appendix A gives the matched ViLaSR comparison, and Appendix C reports the supervised-data sizes, split audit, hardware, training schedule, optimization settings, and reward weights. Section 4.4 documents training diagnostics; Appendices D and F document the tool-access control and representative instructions; Appendix E provides complete step-by-step inference cases. Together, these descriptions support reproduction of the reported experiments.

REFERENCES

Kyeongjin Ahn, Seungeon Lee, Krishna P. Gummadi, and Meeyoung Cha. GeoX: Mastering geospatial reasoning through self-play and verifiable rewards. *arXiv preprint arXiv:2605.20006*, 2026.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Hunar Batra, Haoqin Tu, Hardy Chen, Yuanze Lin, Cihang Xie, and Ronald Clark. SpatialThinker: Reinforcing scene graph-grounded spatial reasoning via dense rewards. *arXiv preprint arXiv:2511.07403*, 2025.

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. SpatialVLM: Endowing vision-language models with spatial reasoning capabilities. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 14455–14465, 2024.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 5828–5839, 2017.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi

Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z.

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Ranjay Krishna. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2024.

Michael Igorevich Ivanitskiy, Rusheb Shah, Alex F. Spies, Tilman Räuker, Dan Valentine, Can Rager, Lucia Quirke, Chris Mathwin, Guillaume Corlouer, Cecilia Diniz Behn, and Samy Wu Fung. A configurable library for generating and manipulating maze datasets. *arXiv preprint arXiv:2309.10498*, 2023.

Siwen Jiao, Tianxiong Lv, Kangan Qian, Chenxu Zhao, Xiuyuan Zhu, Tianlun Li, Xiaolong Cheng, Jinyu Li, Zhihao Liao, and Yang Cai. Smooth operator: Smooth verifiable reward activates spatial reasoning ability of vision-language model. *arXiv preprint arXiv:2601.07695*, 2026.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. LLaVA-OneVision: Easy visual task transfer. *Trans. Mach. Learn. Res. (TMLR)*, 2025a.

Chengzu Li, Wenshan Wu, Huanyu Zhang, Yan Xia, Shaoguang Mao, Li Dong, Ivan Vulić, and Furu Wei. Imagine while reasoning in space: Multimodal visualization-of-thought. In *Proc. Int. Conf. Mach. Learn. (ICML)*, 2025b.

Dinging Li, Yingxiu Zhao, Xinrui Cheng, Kangheng Lin, Hongbo Peng, Hongxing Li, Zixuan Wang, Yuhong Dai, Haodong Li, Jia Wang, Yukang Shi, Liang Zhao, Jianjian Sun, Zheng Ge, Xiangyu Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. SpatialEvo: Self-evolving spatial intelligence via deterministic geometric environments. *arXiv preprint arXiv:2604.14144*, 2026a.

Zongxia Li, Hongyang Du, Chengsong Huang, Xiyang Wu, Lantao Yu, Yicheng He, Jing Xie, Xiaomin Wu, Zhichao Liu, Jiarui Zhang, and Fuxiao Liu. MM-Zero: Self-evolving multi-model vision language models from zero data. *arXiv preprint arXiv:2603.09206*, 2026b.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2024.

Jiaqi Liu, Kaiwen Xiong, Peng Xia, Yiyang Zhou, Haonian Ji, Lu Feng, Siwei Han, Mingyu Ding, and Huaxiu Yao. Agent0-VL: Exploring self-evolving agent for tool-integrated vision-language reasoning. In *Proc. Int. Conf. Mach. Learn. (ICML)*, 2026a.

Junming Liu, Yuqi Li, Yifei Sun, Maonan Wang, Piotr Koniusz, Yirong Chen, and Ding Wang. Self-evolving spatial reasoning in vision language models via geometric logic consistency. *arXiv preprint arXiv:2605.18162*, 2026b.

Damiano Marsili, Rohun Agrawal, Yisong Yue, and Georgia Gkioxari. Visual agentic AI for spatial reasoning with a dynamic API. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 19446–19455, 2025.

Kun Ouyang, Yuanxin Liu, Haoning Wu, Yi Liu, Hao Zhou, Jie Zhou, Fandong Meng, and Xu Sun. SpaceR: Reinforcing MLLMs in video spatial reasoning. *arXiv preprint arXiv:2504.01805*, 2025.

Zhangyang Qi, Zhixiong Zhang, Ye Fang, Jiaqi Wang, and Hengshuang Zhao. GPT4Scene: Understand 3D scenes from videos with vision-language models. In *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2026.

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. *arXiv preprint arXiv:2402.03300*, 2024.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning gym: Reasoning environments for reinforcement learning with verifiable rewards. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2025.

Kimi Team, Angang Du, Bohong Yin, Bowei Xing, Bowen Qu, Bowen Wang, Cheng Chen, Chenlin Zhang, Chenzhuang Du, Chu Wei, Congcong Wang, Dehao Zhang, Dikang Du, Dongliang Wang, Enming Yuan, Enzhe Lu, Fang Li, Flood Sung, Guangda Wei, Guokun Lai, Han Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haoning Wu, Haotian Yao, Haoyu Lu, Heng Wang, Hongcheng Gao, Huabin Zheng, Jiaming Li, Jianlin Su, Jianzhou Wang, Jiaqi Deng, Jiezhong Qiu, Jin Xie, Jinhong Wang, Jingyuan Liu, Junjie Yan, Kun Ouyang, Liang Chen, Lin Sui, Longhui Yu, Mengfan Dong, Mengnan Dong, Nuo Xu, Pengyu Cheng, Qizheng Gu, Runjie Zhou, Shaowei Liu, Sihan Cao, Tao Yu, Tianhui Song, Tongtong Bai, Wei Song, Weiran He, Weixiao Huang, Weixin Xu, Xiaokun Yuan, Xingcheng Yao, Xingzhe Wu, Xinhao Li, Xinxing Zu, Xinyu Zhou, Xinyuan Wang, Y. Charles, Yan Zhong, Yang Li, Yangyang Hu, Yanru Chen, Yejie Wang, Yibo Liu, Yibo Miao, Yidao Qin, Yimin Chen, Yiping Bao, Yiqin Wang, Yongsheng Kang, Yuanxin Liu, Yuhao Dong, Yulun Du, Yuxin Wu, Yuzhi Wang, Yuzi Yan, Zaida Zhou, Zhaowei Li, Zhejun Jiang, Zheng Zhang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Zijia Zhao, Ziwei Chen, and Zongyu Lin. Kimi-VL technical report. *arXiv preprint arXiv:2504.07491*, 2025.

Shi-Yu Tian, Zhuo-Xia Wang, Xuan-Yi Zhu, Zhi Zhou, Xinwei Yang, Kun-Yang Yu, Ming Yang, Yang Chen, and Yu-Feng Li. Self-evolving neuro-symbolic skills for tool-augmented spatial reasoning. *arXiv preprint arXiv:2608.07955*, 2026.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proc. 62nd Annu. Meeting Assoc. Comput. Linguistics (ACL)*, pp. 9426–9439, 2024.

Qinsi Wang, Bo Liu, Tianyi Zhou, Jing Shi, Yueqian Lin, Yiran Chen, Hai Helen Li, Kun Wan, and Wentian Zhao. Vision-Zero: Scalable VLM self-evolution via multi-agent self-play. In *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2026.

Junfei Wu, Jian Guan, Kaituo Feng, Qiang Liu, Shu Wu, Liang Wang, Wei Wu, and Tieniu Tan. Reinforcing spatial reasoning in vision-language models with interwoven thinking and visual drawing. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2025.

Peng Xia, Kaide Zeng, Jiaqi Liu, Can Qin, Fang Wu, Yiyang Zhou, Caiming Xiong, and Huaxiu Yao. Agent0: Unleashing self-evolving agents from zero data via tool-integrated reasoning. In *Proc. Conf. Lang. Model. (COLM)*, 2026.

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 *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 10632–10643, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, juncai liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou,

Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2025.

Zhaoning Yu, Zhaolun Su, Leitian Tao, Haozhu Wang, Aashu Singh, Hanchao Yu, Jianyu Wang, Hongyang Gao, Weizhe Yuan, Jason E. Weston, Ping Yu, and Jing Xu. RESTRAIN: From spurious votes to signals — self-training RL with self-penalization. In *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2026.

Jiahui Zhang, Yurui Chen, Yanpeng Zhou, Yueming Xu, Ze Huang, Jilin Mei, Junhui Chen, Yu-Jie Yuan, Xinyue Cai, Guowei Huang, Xingyue Quan, Hang Xu, and Li Zhang. From flatland to space: Teaching vision-language models to perceive and reason in 3D. In *Adv. Neural Inf. Process. Syst. (NeurIPS), Datasets and Benchmarks Track*, 2025.

Andrew Zhao, Yiran Wu, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2025.

Xu Zheng, Zihao Dongfang, Lutao Jiang, Boyuan Zheng, Yulong Guo, Zhenquan Zhang, Giuliano Albanese, Runyi Yang, Mengjiao Ma, Zixin Zhang, Chenfei Liao, Dingcheng Zhen, Yuanhuiyi Lyu, Yuqian Fu, Bin Ren, Linfeng Zhang, Danda Pani Paudel, Nicu Sebe, Luc Van Gool, and Xuming Hu. Multimodal spatial reasoning in the large model era: A survey and benchmarks. *arXiv preprint arXiv:2510.25760*, 2025.

A Matched 200-Step Comparison with ViLaSR

ViLaSR (Wu et al., 2025) is the closest baseline because we adopt its visual tool interface and supervised initialization. We therefore rerun its Stage-3 RL with the same Qwen2.5-VL-7B initialization, 200-step GRPO configuration, decoding policy, agentic evaluation harness, and benchmark scoring used for VERSE. Table 5 reports the resulting matched comparison.

Table 5: **Matched 200-step Stage-3 comparison (%).** Mean is the unweighted average of the three benchmark scores. "RL inputs" counts the fixed annotated instances for ViLaSR and the unlabeled contexts from which VERSE generates and verifies questions online.

Method	Stage-3 RL source	RL inputs	VSI-Bench	SPAR-Bench	MAZE	Mean
ViLaSR-7B	Annotated QA instances	38,610	**41.91**	35.44	**91.50**	**56.28**
VERSE-7B	Unlabeled contexts	1,464	40.52	**36.01**	90.80	55.78

We do not claim that VERSE outperforms ViLaSR in this matched setting: ViLaSR is 0.50 points higher on the three-benchmark mean, 1.39 points higher on VSI-Bench, and 0.70 points higher on MAZE; VERSE is 0.57 points higher on SPAR-Bench. The two methods are nevertheless comparable in aggregate. The distinction is supervision and data efficiency. ViLaSR continues RL on a fixed set of 38,610 externally annotated questions and answers, whereas VERSE uses 1,464 unlabeled scene and maze contexts and creates, certifies, and solves new tasks online. This comparison concerns Stage 3 only: both methods still rely on the supervised initialization described in Appendix C.

B Higher Grounding Fidelity Predicts Correct Answers

Grounding fidelity is behaviorally meaningful only if it tracks task success rather than merely rewarding well-formed drawings. Across 129 scene-task updates, mean max-IoU grounding fidelity and Round-2 answer accuracy have a moderate positive Pearson correlation ($r = 0.497$). Thus, updates with higher fidelity are more likely to answer correctly, although the residual variation shows that grounding alone does not determine success. Because task mix and question difficulty vary across updates, this association should not be read as a causal estimate of the process reward. The complementary rollout-level analysis of path fidelity on navigation tasks is reported in Section 4.4.

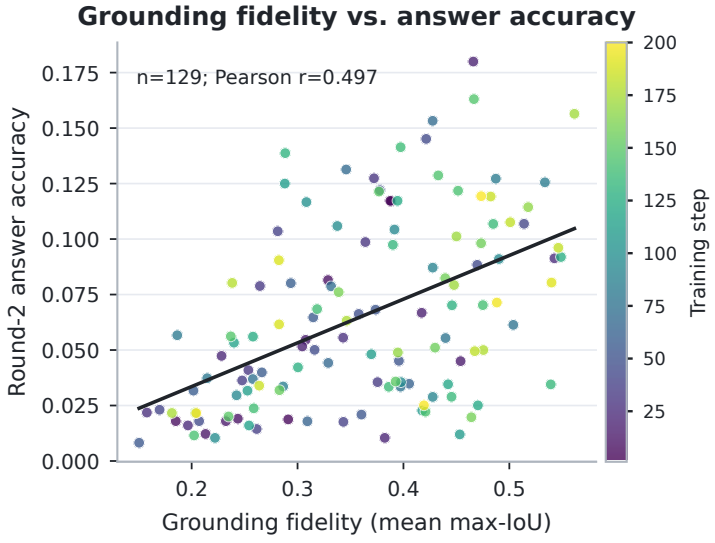


Figure 6: **Higher grounding fidelity is associated with higher answer accuracy.** Each point is one scene-task update, color-coded by training step. The black line is an ordinary least-squares fit over all 129 observations. Grounding fidelity and Round-2 answer accuracy have Pearson correlation $r = 0.497$; the association is positive but leaves substantial variation unexplained.

C Training Implementation Details

Hardware and wall-clock cost. All experiments were run on a single node with $8\times$ NVIDIA A800-SXM4 (80 GB) GPUs, dual Intel Xeon Platinum 8336C CPUs (128 threads), and 1.8 TB of host memory. Each training stage (cold-start SFT, reflective SFT, and self-evolution) uses 4 GPUs; the remaining GPUs are used for concurrent evaluation. Table 6 reports the measured wall-clock cost of every stage for both model scales. During self-evolution, one optimization step (proposer rollout, engine certification, G^A multi-turn solver rollouts, reward computation, and two GRPO updates) takes 21.0 minutes on average for the 7B model and 18.9 minutes for the 3B model. We trace the cost of the first 100 self-evolution steps, which amounts to roughly 35 and 32 GPU-node hours, respectively.

Table 6: **Measured training cost of each stage of VERSE on 4×A800-80GB.** Steps are optimizer updates. The self-evolution rows report the traced cost of the first 100 steps of each run.

Stage	Model	Steps	Wall-clock	Time / step
Cold-start SFT	Qwen2.5-VL-3B	3,258	22.8 h	25.2 s
Reflective SFT	Qwen2.5-VL-3B	744	5.6 h	27.1 s
Self-evolution (GRPO)	Qwen2.5-VL-3B	100 (traced)	31.5 h	18.9 min
Cold-start SFT	Qwen2.5-VL-7B	3,258	23.6 h	26.1 s
Reflective SFT	Qwen2.5-VL-7B	744	8.4 h	40.6 s
Self-evolution (GRPO)	Qwen2.5-VL-7B	100 (traced)	35.0 h	21.0 min

Supervised data. Both SFT stages use image-cleaned, annotated ViLaSR tool-use trajectories (Wu et al., 2025) (Table 7); reflective trajectories also satisfy the filtering criteria in Section 3.3. Accordingly, the complete three-stage pipeline is not annotation-free. We use that term only for Stage-3 self-evolution, which introduces no annotated RL questions, answers, or tool-use traces. The supervised initialization is cheaper than the long-horizon interactive RL stage: for the 7B model, the two SFT stages take 32.0 hours in total, while the first 100 traced self-evolution steps take 35.0 hours under the same four-GPU allocation.

Table 7: **Supervised data sizes.** Number of tool-use trajectories after cleaning and filtering.

Source	Cold-start SFT	Reflective SFT
SR-91k	11,352	3,000
Spatial VQA	10,000	1,000
Maze	7,298	2,055
GPT4Scene	6,095	1,857
Total	34,745	7,912

Stage-3 train/evaluation split audit. The self-evolution file contains 1,464 unlabeled contexts: 1,196 ScanNet contexts from 332 scans (161 physical rooms) and 268 procedural mazes. All 332 ScanNet scans belong to the official training split. The ScanNet portion of VSI-Bench uses 88 official validation scans, giving zero overlap at both scan and physical-room level; its ARKitScenes and ScanNet++ portions come from different source datasets. Training mazes were produced independently with our generator (seed 20260724), while evaluation uses ViLaSR's released test split; file-level checks find no shared instances, and the 500 evaluation mazes of grid size 3×3 are unseen during self-evolution. SPAR-Bench does not release scene identifiers, so we additionally traced its split provenance and performed an exhaustive 64-bit perceptual-hash comparison between 7,566 ScanNet-source evaluation images and 5,260 local training frames. No pair has Hamming distance 0–3, the range containing exact or re-encoded duplicates; all nearest candidates at distances 4–8 were manually verified as unrelated rooms. These checks find no evaluation-instance reuse in the Stage-3 context pool.

Hyperparameters. Table 8 lists the hyperparameters of all three stages; the same values are used for the 3B and 7B models. Both SFT stages perform full-parameter fine-tuning of the language model with the vision tower and the multimodal projector frozen, using DeepSpeed ZeRO-3 (no CPU offload) in bf16. Self-evolution is implemented on top of the EasyR1/verl GRPO trainer with FSDP full sharding, parameter and optimizer CPU offloading, gradient checkpointing, and vLLM rollouts (tensor-parallel size 2). The vision tower remains frozen throughout. The process weight $w = 0.2$ is applied to the Agentic Verifier term of the task at hand: Φ_{Mapper} (grounding fidelity) on scene tasks and Φ_{Tracer} (path fidelity) on maze tasks, with the same $\lambda_f = 0.1$ and $\lambda_a = 0.7$ in both cases.

Table 8: **Hyperparameters of VERSE.** Effective batch sizes are per optimizer update on 4 GPUs.

Cold-start SFT and Reflective SFT (identical settings)	
Optimizer / schedule	AdamW, cosine decay, peak LR 1×10^{-5}, warm-up ratio 0.1
Epochs	3
Effective batch size	32 (per-device 1 $\times$ grad. accumulation 8 $\times$ 4 GPUs)
Sequence / image budget	cutoff 18,432 tokens; image resolution $\leq$ 1,003,520 px (1280$\times$28$\times$28)
Trainable parameters	language model only; vision tower and projector frozen
Self-evolution (dual-group GRPO)	
Initialization / KL anchor	Reflective-SFT checkpoint π_{ref}
Contexts per step	16
Proposer group size G^Q	8 candidate questions per context
Solver group size G^A	8 multi-turn rollouts per unique question
Tool budget per trajectory	scene tasks: up to 6 calls; maze: no explicit call cap; $\leq$ 1,024 tokens per turn
Prompt / response / model length	6,144 / 8,192 / 32,768 tokens
Image resolution	[100,352, 200,704] px per frame
Sampling	temperature 1.0, top-p 1.0
Optimizer	AdamW, constant LR 1×10^{-6}, $\beta = (0.9, 0.999)$, weight decay 0.01
Gradient clipping	global norm 1.0
PPO epochs / loss aggregation	1 / token-level mean
Clipping	asymmetric $[\epsilon_l, \epsilon_h] = [0.2, 0.3]$, dual clip 3.0
KL regularization	low-variance KL to π_{ref}, coefficient 0.01
Solver reward weights	$\lambda_f = 0.1$, $\lambda_a = 0.7$, process weight $w = 0.2$ (Φ_{Mapper} on scene tasks, Φ_{Tracer} on maze tasks)
Proposer reward	$\lambda_f = 0.1$ format $+ 0.9 \cdot v_i \, S_{\text{obs}}$; lightweight external VLM judge
Tracer mixing parameter	$\lambda = 0.6$ (path overlap) / 0.4 (destination hit)

The six-call limit applies only to scene questions. Maze trajectories are not truncated by a tool-call counter and may therefore exceed six calls (including the 9.89-call average reported in Table 10); they remain bounded by the per-turn token limit, total model context, and generation termination.

D TOOL ACCESS WITHOUT TRAINING DOES NOT HELP THE BASE MODEL

A natural question is whether the gains of VERSE come simply from *giving* the model visual tools rather than from *teaching* it to use them. We therefore evaluate the untuned Qwen2.5-VL-7B under three protocols on the same benchmarks: (i) **direct** single-turn answering, as in Table 1; (ii) **tools, untrained**: the base model is placed in exactly the agentic harness used for VERSE (identical system prompt with the OBJECTMAPPER/PATHTRACER specifications, the same multi-turn `think`→`draw`→`observe` loop, rendering, frame budget, and decoding settings), but with no tool-use training of any kind; and (iii) **VERSE** after our full pipeline.

Table 9: **Giving the untuned base model the tool interface hurts; learning to use it helps.** Accuracy (%) of Qwen2.5-VL-7B under the direct protocol, under the agentic tool harness without any tool-use training, and after VERSE. The VERSE row is identical to Table 1.

Qwen2.5-VL-7B protocol	VSI-Bench									SPAR-Bench				MAZE
	Overall	Obj. Count	Abs. Dist.	Obj. Size	Room Size	Rel. Dist.	Rel. Dir.	Route Plan	Appr. Order	Overall	Low	Medium	High	
Direct (no tools)	33.68	43.62	14.59	38.64	33.99	40.14	38.43	27.32	32.69	34.12	25.57	30.14	42.95	33.50
Tools, untrained	17.00	16.46	4.06	9.24	3.13	29.01	36.05	11.34	26.70	25.94	15.47	22.87	36.27	29.00
Tools + VERSE (ours)	**40.52**	**45.10**	**30.14**	**54.56**	**34.69**	**44.37**	**44.21**	**28.87**	**42.23**	**36.01**	**29.00**	**32.01**	**43.55**	**90.80**

Table 10: **Tool-use behavior of the untuned base model versus VERSE under the identical agentic harness.** "No tool call" is the fraction of trajectories that never invoke OBJECTMAPPER or PATHTRACER; "No final answer" is the fraction that exhaust the turn budget without emitting an `<answer>` tag. VERSE statistics are from the step-100 checkpoint.

Qwen2.5-VL-7B	No tool call (%) ↓			Tool calls / trajectory			No final answer (%) ↓		
	VSI	SPAR	MAZE	VSI	SPAR	MAZE	VSI	SPAR	MAZE
Tools, untrained	41.3	59.0	31.8	2.51	1.78	3.74	18.7	11.2	4.6
Tools + VERSE	**0.0**	**0.0**	**0.0**	4.68	2.83	9.89	**0.7**	**0.3**	**0.2**

Tool access alone is harmful. Table 9 shows that merely exposing the base model to the tool interface *halves* its VSI-Bench accuracy (33.68% → 17.00%) and also degrades SPAR-Bench (34.12% → 25.94%) and MAZE (33.50% → 29.00%). The collapse is most severe on exactly the metric subtasks where grounded drawing should help the most: absolute distance drops from 14.59% to 4.06%, object size from 38.64% to 9.24%, and room size from 33.99% to 3.13%. Table 10 explains why. The untuned model does not know when or how to use the tools: it skips them entirely in 32–59% of trajectories, and when it does draw, the boxes and paths are ungrounded placeholders that it then over-trusts ("the visualization indicates that the final destination is D") rather than checks. Between 5% and 19% of its trajectories exhaust the turn budget without ever committing to an answer. In other words, a capable VLM equipped with visual tools *out of the box* neither uses them reliably nor benefits from them.

VERSE turns the same tools into a gain. After the VERSE pipeline the same architecture invokes tools in every trajectory, makes more and purposeful calls (e.g. 9.9 PATHTRACER steps per maze, one per action), almost always reaches a final answer, and improves over the untrained-tools setting by $+23.5$ points on VSI-Bench, $+10.1$ on SPAR-Bench, and $+61.8$ on MAZE—surpassing the direct baseline on every benchmark and every VSI-Bench subtask. The benefit of visual tools is therefore not a property of the interface but of the learned, engine-verified habit of grounding intermediate reasoning in it.

E QUALIFIED INFERENCE CASES

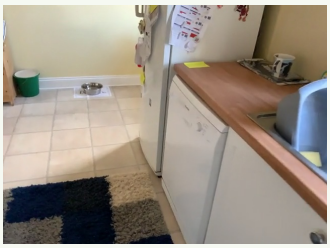
We present three correct spatial-reasoning examples. Each case shows the question, the complete model response at every step, the associated tool output, and the final answer against the ground truth. Together, these examples show how the model uses visual tools as an external workspace for perception and reasoning.

E.1 OBJECT SIZE ESTIMATION

Dishwasher size estimation **2 tool calls + final answer**

Question and input

What is the length of the longest dimension (length, width, or height) of the dishwasher, measured in centimeters?



Referenced input image

Turn 1 — Object Mapper

<think> To determine the length of the longest dimension of the dishwasher, I need to identify the dishwasher in the images and measure its dimensions. The dishwasher is likely to be in image 11, as it shows a kitchen area with a dishwasher visible. </think>

Action: Object Mapper

Tool arguments (JSON)

```
[
{"index": 11,
"bbox_2d": [311, 135, 450, 475],
"label": "dishwasher"}
]
```



Tool output 1

Dishwasher size estimation (continued)

Turn 2 — Path Tracer

<think> The bounding box around the dishwasher is correct. Now, I need to measure the longest dimension of the dishwasher. The height appears to be the longest dimension. I will use the Path Tracer tool to measure the height. </think>

Action: Path Tracer

Tool arguments (JSON)

```
[
{"index": 11,
"start_point_2d": [360, 140],
"end_point_2d": [360, 470],
"label": "height of dishwasher"}
]
```



Tool output 2

Turn 3 — Final response

<think> The path tracer has correctly marked the height of the dishwasher. Now, I need to estimate the length of this path in centimeters. Given the scale of the image and the typical dimensions of a dishwasher, the height appears to be around 85 cm. </think>

Action: Answer

<answer>85 </answer>

Model answer: 85 cm. **Ground truth: 89 cm.**

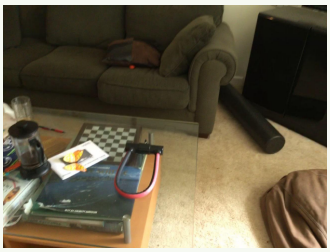
This prediction receives a full evaluation score of 1.0; the continuous-valued answer differs from the recorded ground truth by 4 cm.

E.2 Observer-Centric Spatial Relation

Pillow relative to the observer **2 tool calls + final answer**

Question and input

From the main viewing angle, what is the spatial position of object pillow (bbox)? Calculate or judge based on the 3D center points of these objects. The observer's perspective is captured by setting the first image as the primary view.
The options describe the spatial relationship between object and observer in terms of left-right (left, right, or empty if indistinguishable), above-below (above, below, or empty if indistinguishable), and front-behind (front, behind, or empty if indistinguishable).
Select the correct response from the given choices.
A. left, below, front
B. right, below, behind
C. , below, behind
D. , below, front
Your answer can only include one of options A, B, C or D.



Primary input view

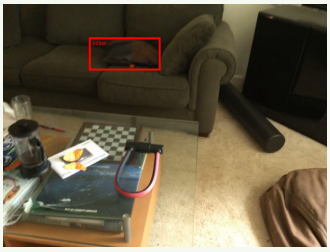
Turn 1 — Object Mapper

<think> To determine the spatial position of the pillow, I need to first identify its location in the image. The pillow is on the couch, so I'll use the Object Mapper tool to map its position relative to the couch and the observer's perspective. </think>

Action: Object Mapper

Tool arguments (JSON)


```
[
{"index": 1,
"bbox_2d": [178, 69, 326, 137],
"label": "pillow"}
]
```

Tool output 1

Pillow relative to the observer (continued)

Turn 2 — Object Mapper



Tool output 2

<think> The visualization confirms the pillow's location on the couch. Now, I need to determine its spatial relationship to the observer's perspective. The pillow is on the left side of the couch, which would be to the left of the observer's perspective. </think>

Action: Object Mapper

Tool arguments (JSON)

```
[
{"index": 1,
"bbox_2d": [0, 0, 484, 278],
"label": "couch"}
]
```

Turn 3 — Final response

<think> The visualization confirms the couch's location and the pillow's position on it. The pillow is on the left side of the couch, which would be to the left of the observer's perspective. Now, I need to determine the spatial relationship in the front-behind dimension. The pillow is closer to the observer than the couch, so it is in the front. </think>

Action: Answer

<answer>A </answer>

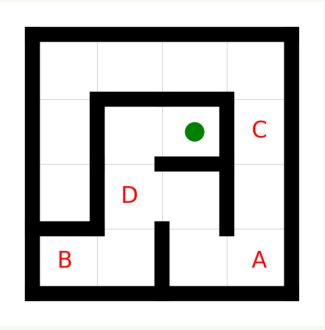
Model answer: A (left, below, front). **Ground truth: A (left, below, front).**

E.3 STEP-BY-STEP MAZE NAVIGATION

Maze navigation **4 tool calls + final answer**

Question and input

Task: Maze Navigation Simulation
Determine the final destination (A, B, C or D) from the starting point (green point) following the action sequence.
The definitions of the actions are as below.
* Go up/left/down/right: move one grid space in the absolute up/left/down/right direction.
Full Action Sequence: Go left. Go down. Go down. Go left.



Input image 1

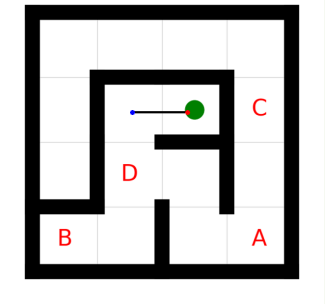
Turn 1 — Path Tracer

<think> To determine the final destination, I need to trace the path step by step according to the given action sequence. Starting from the green point, I will move left, then down, then down again, and finally left. I will use the Path Tracer tool to visualize each step and ensure the path is correct. </think>

Action: Path Tracer

Tool arguments (JSON)

```
[
{"index": 1,
"start_point_2d": [340, 240],
"end_point_2d": [240, 240],
"label": "Go left"}
]
```



Tool output 1 — accumulated path

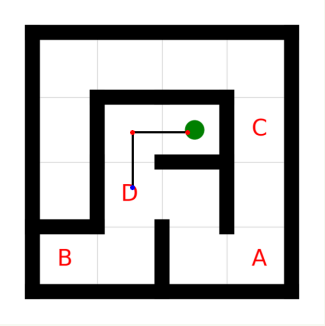
Turn 2 — Path Tracer

<think> The first step of going left is correctly visualized. Now, I need to move down from the current position. I will use the Path Tracer tool to draw the path from the end of the previous step down to the next point. </think>

Action: Path Tracer

Tool arguments (JSON)

```
[
{"index": 2,
"start_point_2d": [240, 240],
"end_point_2d": [240, 340],
"label": "Go down"}
]
```



Tool output 2 — accumulated path

Maze navigation (continued)

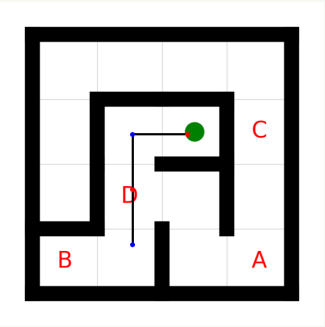
Turn 3 — Path Tracer

<think> The path has been correctly drawn for the first two steps: going left and then going down. Now, I need to continue the path by going down one more grid space according to the action sequence. I will use the Path Tracer tool to draw the path from the end of the previous step down to the next point. </think>

Action: Path Tracer

Tool arguments (JSON)

```
[
{"index": 3,
"start_point_2d": [240, 340],
"end_point_2d": [240, 440],
"label": "Go down"}
]
```



Tool output 3 — accumulated path

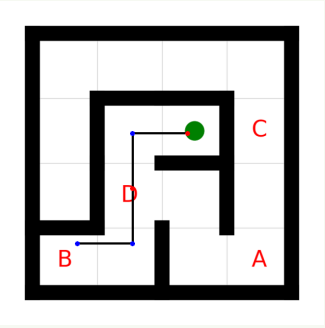
Turn 4 — Path Tracer

<think> The path has been correctly drawn for the first three steps: going left, going down, and then going down again. Now, I need to complete the final step of the action sequence by going left one grid space. I will use the Path Tracer tool to draw the path from the end of the previous step to the next point. </think>

Action: Path Tracer

Tool arguments (JSON)

```
[
{"index": 4,
"start_point_2d": [240, 440],
"end_point_2d": [140, 440],
"label": "Go left"}
]
```



Tool output 4 — accumulated path

Turn 5 — Final response

<think> The path has been correctly drawn for all the steps in the action sequence: going left, going down, going down again, and finally going left. The final destination is marked by the end of the path, which is at point B. Therefore, the final destination is B. </think>

Action: Answer

<answer>B </answer>

Model answer: B. **Ground truth: B.**

Takeaway. Across the three examples, the model first uses visual tools to perceive and ground the relevant objects or state, then uses the resulting marks and traces to carry out the spatial inference. Object mapping supports object identification and spatial relations, while path tracing supports geometric measurement and multi-step navigation. The final responses show that the model can use these tools for both perception and spatial reasoning.

F SYSTEM PROMPT

Below are the complete Scene Proposer, Maze Proposer, and Solver prompt templates used in Stage-3 self-evolution. Runtime task content and image dimensions are represented by explicit named placeholders; no instruction lines are omitted.

Scene Proposer system prompt

```
# SYSTEM ROLE:
You are an expert 3D scene analyst. You are assigned ONE question type and you
will choose object labels for it.
# ASSIGNED TASK TYPE:
[task display name] ([task type])
# TASK GOAL:
[task-specific goal inserted at runtime]
# AVAILABLE SCENE OBJECT LABELS:
[task-specific scene object pools inserted at runtime]
# WHICH LABELS YOU MAY USE:
[task-specific candidate guidance inserted at runtime]
# PARAM SCHEMA YOU MUST OUTPUT:
[task-specific parameter schema inserted at runtime]
# YOUR JOB:
1. Write one grounded whole-scene observation describing the spatial layout.
2. Pick object labels from the pools above that satisfy the task requirements,
and emit them in the structured <question_params> block exactly as the schema
shows.
3. Copy every label EXACTLY from the provided pools (same spelling, same
singular/plural form). Do not invent labels, paraphrase them, pluralize
differently, or add modifiers such as "left", "on the table", or "partially
visible".
4. For tasks needing Unique objects, pick only labels marked (Unique).
5. Do NOT write the final question text -- the system will render it from your
params. You only provide the params.
# TASK REQUIREMENTS:
[task-specific requirements inserted at runtime]
# EXAMPLE:
[task-specific observation style inserted at runtime]
<observation>[example grounded observation]</observation>
<question_params>[example JSON object]</question_params>
# HARD CONSTRAINTS:
- Output exactly one <observation> and one <question_params> block.
- Use only labels that appear verbatim in the provided pools.
- If a required label is missing from the pool, pick a different valid combination
rather than inventing a label.
- Do not include the answer or any <question> text.
# OUTPUT FORMAT:
<observation>[grounded scene observation]</observation>
<question_params>[JSON object matching the schema]</question_params>
```

System Prompt (continued)

Maze Proposer system prompt

```
# SYSTEM ROLE:
You are an expert maze navigation question designer.
Assigned task type: Maze Navigation (maze_navigation).
# TASK GOAL:
Design one maze navigation question for the maze image shown. You only need to
pick ONE destination letter (A, B, C, or D) as the intended endpoint. The system
will automatically use the verified shortest action sequence to that destination
-- you do NOT need to write the action sequence yourself.
# AVAILABLE MAZE CONTEXT (deterministic ground facts about this maze):
[maze context inserted at runtime]
# YOUR JOB:
1. Write one grounded observation of the maze image: describe the grid layout,
where the green start point is, where the red destination letters are, and the
wall structure.
2. Pick ONE destination letter (A, B, C, or D) as the intended endpoint. Choose a
destination whose route you find interesting or non-trivial.
3. Write exactly one question in EXACTLY this style (only the letter differs):
Which destination should the solver reach? <answer>X</answer>
where X is your chosen letter (A, B, C, or D).
4. Do NOT write any action sequence yourself -- the system will fill it in from
the verified shortest path.
# HARD CONSTRAINTS:
- The answer must be exactly one of A, B, C, D.
- Output exactly one observation and one question.
- Do NOT reveal the answer letter in the observation.
# OUTPUT FORMAT:
<observation>[grounded maze observation]</observation>
<question>Which destination should the solver reach?
<answer>X</answer></question>
```

System Prompt (continued)

Solver system prompt

```
### Guidance:
You are a spatial reasoning assistant with access to two powerful visualization
tools.
Your task is to break down complex spatial problems and iteratively refine your
solution through visualization feedback.
### Available tools:
You can use the following two tools to visualize. After each tool usage, you must
wait for and analyze the visualization feedback before proceeding.
1. **Object Mapper**
- Purpose: Identifies and maps key items in the space
- Input format: JSON
[{"index": i, "bbox_2d": [x1, y1, x2, y2], "label": "object name/description"}]
- Output: Generates bounding boxes for visual inspection of the i-th image
2. **Path Tracer**
- Purpose: Plots movement or connections between points
- Input format: JSON
[{"index": i, "start_point_2d": [x1, y1], "end_point_2d": [x2, y2], "label":
"trace_description"}]
- Output: Generates visual paths for verification of the i-th image
### Required Output Format:
For each reasoning step, you must structure your response as follows:
<think> [Your detailed reasoning process] </think> Action: [Object Mapper/Path
Tracer]
[JSON format coordinates]
After your reasoning and iteratively refine your solution through visualization
feedback, you should arrive at a final answer and structure your response as
follows:
<think> [Your detailed reasoning process] </think> Action: Answer
<answer> [Your final answer] </answer>
### Please NOTE the following reasoning techniques:
1. Initial Analysis
- Break down the spatial problem
- Plan your approach
2. Iterative Reasoning for Each Step
- Choose appropriate tool
- Provide absolute coordinates in JSON format (The top-left corner of the image is
(0, 0) and the bottom-right corner is ({width}, {height}))
- Observe the visualization output
- Reflect on the visualization:
* Is the placement/path accurate?
* Does it align with your reasoning?
* What adjustments are needed?
- Backtrack and Adjust:
* If errors found, backtrack to previous step to modify actions or decisions as
needed
```