FAIL2SCENE: TURNING NAVIGATION FAILURES INTO TRAINING ENVIRONMENTS

Anonymous authors
Paper under double-blind review

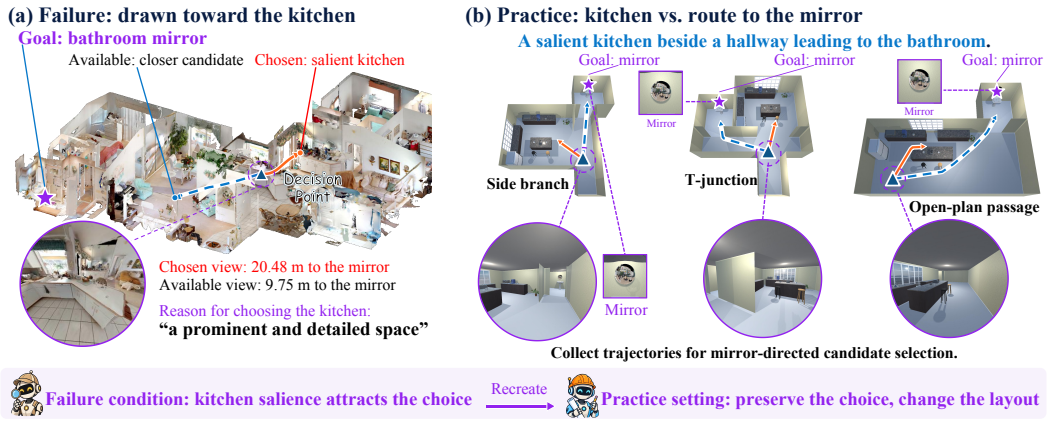


Figure 1: **Navigation failure and targeted practice.** (a) A navigator seeking a bathroom mirror chooses a salient kitchen over a closer available candidate. The source star marks the approximate bathroom region. (b) Illustrative practice layouts preserve the kitchen–route competition while varying spatial structure, supporting trajectory collection for candidate-selection training.

ABSTRACT

Improving embodied navigation requires training experience that targets the failure conditions behind incorrect navigation decisions. However, reproducing the objects and room types of a failed episode can omit the relationships among distracting cues, alternative routes, and the goal. We introduce **Fail2Scene**, a failure-driven curriculum framework that turns failed navigation decisions into new training environments. Its central idea is to preserve the visual and spatial relationships among competing choices while varying the surrounding layout. First, evidence-grounded failure attribution compares chosen and alternative views alongside the navigator's stated rationale to propose and verify explanations for failed decisions. Cross-scene consolidation identifies recurring failure conditions, which guide specifications of room connectivity and relevant visual cues. Second, mechanism-conditioned curriculum construction generates new layouts from these specifications. It then retains navigable scenes that reproduce the diagnosed failure conditions. Rendered trajectories provide candidate-selection examples for reinforcement learning, while repeated diagnosis updates subsequent scene construction as the navigator's failure patterns evolve. Trained in AI2-THOR and evaluated in Habitat, Fail2Scene achieves 70.50% success and 47.62% SPL on GOAT-Bench tasks in held-out buildings, improving success by 9.35 percentage points over a random curriculum with the same scene budget.

1 INTRODUCTION

Navigation agents built on vision-language models (VLMs) combine semantic priors with visual memory to explore unfamiliar environments (Liu et al., 2023a; Bai et al., 2025; Yang et al., 2025b; Yin et al., 2025). A policy's own failures reveal the decisions that subsequent learning should ad-

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.

dress, but an unsuccessful trajectory does not directly specify how to construct the training experience needed to improve those decisions.

Consider a navigator searching for a bathroom mirror that chooses a salient kitchen over a less conspicuous hallway toward the goal, as illustrated in Figure 1. Recreating the kitchen may preserve its appearance while removing the alternative route, in which case the new environment no longer presents the original decision problem. The difficulty therefore lies in recovering the relationship that attracted the wrong choice and preserving it as the surrounding layout changes.

Controllable simulation offers the means to vary indoor geometry and content (Deitke et al., 2022; Raistrick et al., 2024; Sun et al., 2025; Yang et al., 2025a; Pfaff et al., 2026), while generative simulation and adaptive curricula determine what experience an agent receives (Wang et al., 2024a;b; Dennis et al., 2020; Parker-Holder et al., 2022). Generated trajectories and sandbox learning further connect synthetic environments to policy optimization (Wang et al., 2025; Shen et al., 2026). For failure-driven adaptation, however, these capabilities leave a central question: which properties of a failed decision should new environments retain? Our insight is to recover reusable failure conditions from the chosen view, available alternatives, and stated rationale, so that execution-grounded explanation (Liu et al., 2023b; Duan et al., 2025; Gou et al., 2024) can guide the construction of subsequent practice.

Building on this insight, we introduce **Fail2Scene**, a framework that turns a navigator's current failures into an adaptive curriculum of generated environments. Its central idea is to reuse the visual and spatial relationships involved in failed choices across different scene layouts. First, *evidence-grounded failure attribution* uses distance comparisons within failed subtasks to select decisions with a closer available alternative. An examiner compares the candidate views with the task query and the navigator's stated rationale, jointly proposing an explanation and a scene specification. A verifier checks the explanation against recorded evidence and requests a revised pair when needed.

Second, *mechanism-conditioned curriculum construction* consolidates accepted pairs across source scenes into recurring failure conditions, termed *mechanisms*, together with scene specifications. Cross-scene support identifies which conditions recur across layouts, while the specifications describe room connectivity and salient visual cues for new training environments. A scene generator uses these specifications to produce new layouts. Generated scenes are retained when they are navigable and present the diagnosed conditions at sampled decision points.

To connect these environments to policy learning, we render views along goal-directed trajectories and construct task queries from terminal observations, detected objects, and spatial relationships. Reference views selected by trajectory position provide practice in continued exploration or choosing evidence for answering, with additional views from the available trajectory prefix forming the candidate set. The resulting examples train the navigator with reinforcement learning. Policy updates are evaluated on discovery tasks, and the retained navigator supplies the next round's failures. Repeating this process expands the training collection and updates the curriculum as the navigator's failure patterns change.

We evaluate this process on GOAT-Bench buildings (Khanna et al., 2024), using discovery buildings to guide adaptation and disjoint held-out buildings to test transfer to unseen layouts. Policy optimization uses generated AI2-THOR environments (Kolve et al., 2017), while diagnosis and navigation evaluation take place in Habitat (Szot et al., 2021). In summary, the key contributions of our work are:

- We formulate failure-conditioned environment construction around the competition among available navigation choices, identifying what practice should preserve when scene geometry changes.
- We develop an evidence-grounded attribution and curriculum loop that consolidates verified explanations across source scenes, constructs scenes from the resulting specifications, and converts rendered trajectories into candidate-selection training inputs.
- We demonstrate generalization to held-out GOAT-Bench buildings, achieving 70.50% SR and 47.62% SPL after five rounds and improving SR by 9.35 percentage points over a random curriculum with the same scene budget, with additional gains on A-EQA.

2 Related Work

Embodied Navigation. Semantic priors, visual memory, and unified goal representations enable agents to connect language or image queries with exploration decisions (Chaplot et al., 2020; Yu et al., 2023; Zhou et al., 2023; Yang et al., 2025b; Yin et al., 2025), while recent systems combine spatial reasoning with memory and policy learning (Deng et al., 2026; Feng et al., 2026). To improve these policies, existing approaches draw on video-instruction data (Han et al., 2025), large-scale reinforcement learning (Zeng et al., 2025), and self-refining trajectories (Wang et al., 2025). SAGE further distills physics-grounded sandbox experience into navigation policies (Shen et al., 2026). Fail2Scene uses the navigator's current failures to guide scene construction, adapting the training curriculum as the policy evolves.

Training Environment Synthesis. Procedural synthesis (Deitke et al., 2022; Raistrick et al., 2024), learned layouts (Tang et al., 2024), language-guided optimization (Çelen et al., 2025; Sun et al., 2025), and agentic construction (Yang et al., 2025a; Pfaff et al., 2026; Zhang et al., 2026) provide increasing control over indoor geometry and content. Digital cousins preserve geometric and semantic affordances for transfer (Dai et al., 2025), while TwinRL uses digital twins for policy improvement (Xu et al., 2026b). Generative simulation also constructs tasks and rewards (Wang et al., 2024a; Ma et al., 2024; Wang et al., 2024b), and adaptive curricula organize practice through regret, replay, or environment evolution (Dennis et al., 2020; Jiang et al., 2021; Parker-Holder et al., 2022). Fail2Scene conditions generation on visual and spatial relationships recovered from failed choices, preserving the competition among navigation alternatives across new layouts.

Failure Attribution. Execution traces support explanations of robot failures (Liu et al., 2023b; Duan et al., 2025), while multi-agent attribution identifies responsible agents and decisions (Zhang et al., 2025). Because unaided self-correction can fail (Huang et al., 2024), tool feedback provides evidence against which proposed explanations can be checked (Schick et al., 2023; Gou et al., 2024). Attribution has also been linked to capability-specific navigation tests (Chen et al., 2026) and to grasp selection and planning-data collection in manipulation (Xu et al., 2026a). Fail2Scene links attribution to scene construction by consolidating verified explanations across source scenes into reusable failure conditions and scene specifications.

3 Fail2Scene

As shown in Figure 2, Fail2Scene turns failed navigation decisions into an adaptive curriculum of generated environments. We first define the navigation decision interface and recorded traces (Section 3.1). We then select diagnostic decisions and verify their explanation–specification pairs against recorded evidence (Section 3.2). These verified pairs are consolidated across source scenes into recurring failure mechanisms that guide scene generation and validation (Section 3.3). Finally, rendered trajectories are converted into candidate-selection examples (Section 3.4), and policy optimization closes the adaptation loop (Section 3.5).

3.1 Navigation Decision Interface

At decision t in environment e, a vision-language policy π_θ receives $x_t = (q, o_t, \mathcal{F}_t, \mathcal{M}_t)$, comprising the query q, current observation o_t, frontier views $\mathcal{F}_t$, and memory views $\mathcal{M}_t$ accumulated within the episode. Image-goal queries include a target image, and memory views carry object labels. Each candidate $c = (I_c, p_c, \kappa_c)$ contains an image I_c, a navigable position p_c, and a type $\kappa_c \in \{\text{frontier}, \text{memory}\}$. The policy selects:

$$(a_t, z_t) \sim \pi_\theta(\cdot \mid x_t), \qquad a_t \in \mathcal{C}_t = \mathcal{F}_t \cup \mathcal{M}_t, \tag{1}$$

where z_t is a rationale for a frontier choice or an answer for a memory choice. A geometric planner executes the selected waypoint p_{a_t}.

An episode of T decisions produces the trace $\tau = (x_t, \mathcal{C}_t, a_t, z_t)_{t=1}^{T}$, together with subtask identifiers and outcomes. We collect traces containing failed subtasks as τ^-, preserving source identifiers for cross-scene attribution. Diagnostic tools additionally access simulator geometry and goal locations.

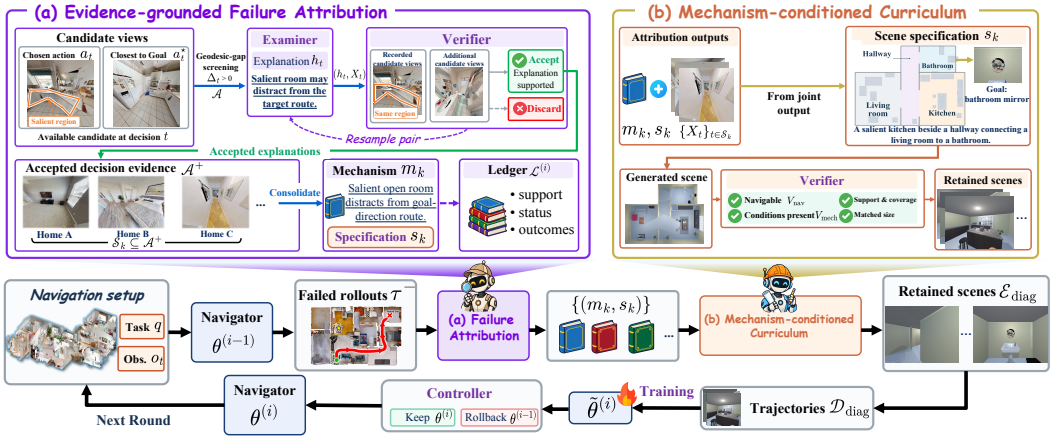


Figure 2: **Overview of Fail2Scene. Bottom:** Failed rollouts guide scene construction and trajectory collection for navigator training. The controller accepts updates that improve discovery performance and otherwise retains the previous policy. **Top:** (a) An examiner explains failed choices, and a verifier checks each explanation against the recorded decision evidence. Accepted explanations are grouped across scenes into recurring conditions and scene specifications. (b) Specifications guide scene generation. Checks of navigability, condition presence under a sampled decision configuration, and curriculum composition determine which scenes are retained. The ledger records source support and update outcomes.

3.2 Evidence-Grounded Failure Attribution

Selecting diagnostic decisions. To select decisions for inspection, we compare each chosen candidate in a failed subtask with the available candidate closest to the goal g:

$$a_t^\star \in \arg\min_{c \in \mathcal{C}_t} d_e(p_c, g), \qquad \Delta_t = d_e(p_{a_t}, g) - d_e(p_{a_t^\star}, g), \tag{2}$$

where $d_e(p, g)$ is the geodesic distance from position p to goal g. Eligible decisions have $\Delta_t > 0$ and at least two finite-distance candidates, including the selected one. We form the diagnostic set $\mathcal{A}$ by retaining the largest gaps, first within each failed subtask and then across the round, subject to per-subtask and per-round caps.

Explaining a failed choice. For each $t \in \mathcal{A}$, an examiner receives evidence X_t comprising the query, chosen and closest-candidate views, additional candidate views in recorded order, the navigator's rationale, and the distance comparison. It jointly proposes an explanation h_t of the visible conditions associated with the choice and a scene specification s_t describing a layout for practicing a similar decision. A verifier checks whether h_t is supported by X_t, accepting the pair (h_t, s_t) or requesting regeneration of both outputs. Decisions without an accepted pair within the attempt budget are discarded. Accepted decisions form $\mathcal{A}^+ \subseteq \mathcal{A}$.

3.3 Mechanism-Conditioned Curriculum Construction

Consolidating failure conditions. The examiner groups the current round's accepted explanation–specification pairs by recurring failure conditions. For each group, it jointly returns a mechanism m_k, a scene specification s_k, and supporting decisions $\mathcal{S}_k \subseteq \mathcal{A}^+$. We retain groups supported by at least $n_{\text{sc}} = 2$ distinct scenes, yielding K mechanisms:

$$\{(m_k, s_k, \mathcal{S}_k)\}_{k=1}^{K} \leftarrow \textsc{Consolidate}_{\mathcal{R}}\big(\{(h_t, s_t, t)\}_{t \in \mathcal{A}^+}\big). \tag{3}$$

Each specification describes room connectivity and relevant visual conditions. The construction constraints $\mathcal{R}$ bound specification length and guide the choice of connected room types and salient objects.

Generating and validating scenes. We pass each specification s_k directly to the scene generator Γ (Yang et al., 2024) and launch J generation jobs, producing scenes $e'_{k,j} \sim \Gamma(\cdot \mid s_k)$ for $j = 1, \ldots, J$.

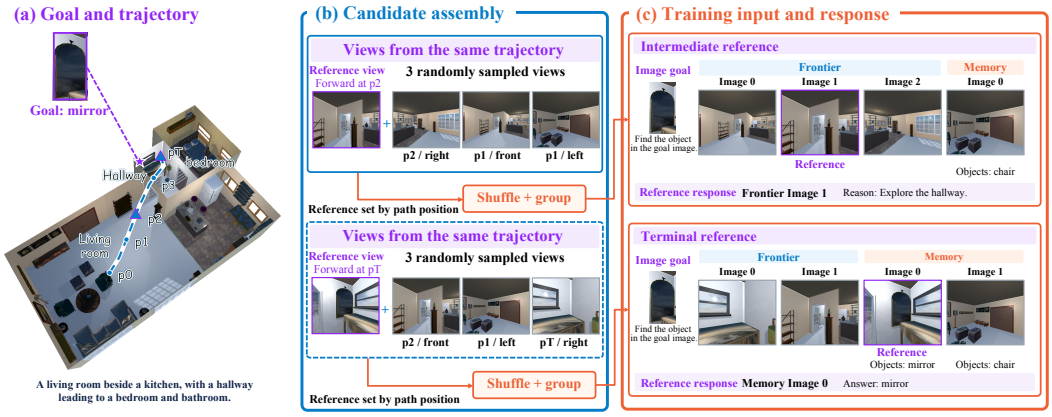


Figure 3: **Constructing candidate-selection training examples.** (a) Forward and side views are rendered along a trajectory to the target. (b) Each input combines a reference view with three views sampled at or before its trajectory position, shuffled within frontier and memory groups. (c) Intermediate references specify frontier choices with rationales, whereas terminal references specify memory choices with answers. Purple borders mark reference views.

For each generated scene e', we sample a decision configuration U comprising a goal, a decision position, and candidates with their rendered views. The scene is accepted when:

$$V(e', m_k; U) = V_{\text{nav}}(e')\, V_{\text{mech}}(e', m_k; U) = 1, \tag{4}$$

where the two binary checks assess navigability and mechanism presence: V_{nav} checks reachable space, connectivity, and floor alignment, while V_{mech} evaluates whether the rendered candidates exhibit m_k using scene geometry and goal distances.

Accepted scenes form the current round's collection $\mathcal{E}_{\text{diag}}$, with each scene linked to its mechanism and supporting decisions. The ledger $\mathcal{L}^{(i)}$ records source support, scene-generation records, and policy-update outcomes accumulated through curriculum attempt i.

3.4 Candidate-Selection Training Examples

We collect trajectories $\mathcal{D}_{\text{diag}}$ in the retained scenes $\mathcal{E}_{\text{diag}}$ and construct candidate-selection examples from their rendered views, following the procedure illustrated in Figure 3. In each AI2-THOR scene (Kolve et al., 2017), we select a target and a terminal pose p_T from which it is visible, then use A* to plan a path from a distant reachable start p_0 to p_T. Forward and side views are rendered along the path, with intermediate forward views facing the next waypoint and the terminal forward view facing the target. Trajectories are screened for endpoint visibility, path length, and detected-object counts, followed by reward-based difficulty filtering.

Each example contains a query, an optional image goal, and four candidate views. The reference is a noninitial intermediate forward view for frontier selection or the terminal forward view for memory selection. At intermediate nodes, the candidate set is constrained by observation time: it includes only views available at the current decision and excludes observations from later nodes, including the unreached terminal node. Three additional views are sampled from the available trajectory prefix. Candidates are shuffled, grouped as frontier or memory by position and viewing direction, and indexed within each group. The reference response specifies the candidate type and index, followed by a frontier rationale or memory answer.

3.5 Policy Optimization

We train the navigator on accumulated generated examples with Group Relative Policy Optimization (GRPO) (Shao et al., 2024). The reward combines agreement with the reference candidate, answer agreement for memory selections, and output-format validity, as defined in Appendix D.

Let $\theta^{(0)}$ denote the initial navigator parameters and $\Theta^{(i)}$ the set of checkpoints evaluated during update attempt i. We select the candidate checkpoint by maximizing the mean response reward S_{gen}

on generated validation examples from environments excluded from training:

$$\widetilde{\theta}^{(i)} \in \arg\max_{\theta \in \Theta^{(i)}} S_{\text{gen}}(\theta). \tag{5}$$

The controller evaluates the candidate on discovery tasks $\mathcal{V}$, which also supply failure traces. The navigation score and policy update are:

$$S_{\mathcal{V}}(\theta) = \tfrac{1}{2}\big[\text{SR}_{\mathcal{V}}(\theta) + \text{SPL}_{\mathcal{V}}(\theta)\big],$$
$$\theta^{(i)} = \begin{cases} \widetilde{\theta}^{(i)}, & S_{\mathcal{V}}(\widetilde{\theta}^{(i)}) > S_{\mathcal{V}}(\theta^{(i-1)}), \\ \theta^{(i-1)}, & \text{otherwise.} \end{cases} \tag{6}$$

Both metrics use percentage scales and the 1.0 m distance success criterion. The controller records the update outcome in $\mathcal{L}^{(i)}$. The retained policy $\pi_{\theta^{(i)}}$ supplies the next round's diagnostic traces, and newly generated scenes extend the accumulated training collection. We allow up to $I_{\max}$ update attempts, with construction-agent parameters fixed throughout, as specified in Appendix F.

4 Experiments

4.1 Experimental Setup

Evaluation tasks. We evaluate GOAT-Bench in Habitat (Khanna et al., 2024; Szot et al., 2021) with 18 discovery and 18 held-out buildings, each split containing 139 subtasks. Discovery and held-out denote evaluation splits: discovery tasks provide failure evidence and determine cross-round update acceptance, while disjoint held-out buildings measure generalization independently of adaptation. Episodes comprise five to ten category, language, or image goals, with visual memory retained across subtasks (Yang et al., 2025b). Full-set navigation additionally includes discovery buildings. A-EQA is evaluated on 184 and 557 questions (Majumdar et al., 2024; Yang et al., 2025b).

Comparison protocol. Fail2Scene (D) constructs a diagnostic curriculum from failure evidence, whereas Random (R) writes scene specifications without it. Both train a Qwen3-VL-2B navigator (Bai et al., 2025) on generated AI2-THOR scenes (Yang et al., 2024; Kolve et al., 2017), sharing the specification interface, generator, construction constraints, and scene budget, and select checkpoints on disjoint generated validation scenes within each round. From the shared initialization π_0, round i yields policies π_{Di}^{GOAT} and π_{Ri}^{GOAT} for GOAT, or π_{Di}^{EQA} and π_{Ri}^{EQA} for separately adapted A-EQA. Here, D/R identify training-scene construction, independently of the discovery/held-out evaluation split.

Training inputs. The nominal 1k, 2k, and 4k budgets count candidate-selection examples of the form shown in Figure 3, with Random training, failure replay, local SAGE, and Fail2Scene all using 4k in the main comparison.

Metrics. Distance SR measures arrival within 1.0 m of a valid goal viewpoint, while SPL weights success by reference-to-traveled path length (Anderson et al., 2018). Snapshot SR/SPL instead require selecting a goal-containing memory view. For A-EQA, we report LLM-match, its path-weighted counterpart, and unanswered rate. We report metrics on percentage scales and express gains as absolute differences.

4.2 GOAT and A-EQA Results

GOAT-Bench. After five rounds, Fail2Scene reaches 70.50% SR and 47.62% SPL on the 139 held-out subtasks, compared with 61.15% and 43.16% for random training, as reported in Table 1 (a). Failure replay slightly improves SR over random generation but lowers SPL. Fail2Scene improves both, showing a benefit from constructing new environments around the failed choices. The improvement also appears when success requires choosing a target-containing memory view: Snapshot SR/SPL rise from 46.04%/27.83% under random training to 51.08%/33.51%. The separate full-set evaluation in Appendix G, which includes discovery buildings, reaches 68.35% distance SR and 46.10% SPL.

Table 1: **Main results.** GOAT-Bench distance SR/SPL and A-EQA match/path-weighted scores (%). *Yang et al. (2025b). †Shen et al. (2026). A-EQA evaluation uses GPT-4 for 3D-Mem and Qwen3-235B-A22B for SAGE. Bold compares our policies within each set. Qwen3 denotes Qwen3-VL.

(a) GOAT-Bench

Method	Backbone	SR ↑	SPL ↑
External baselines: 278 subtasks			
Explore-EQA*	GPT-4o	55.0	37.9
3D-Mem*	GPT-4o	69.1	48.9
3D-Mem†	Qwen3-2B	46.4	20.3
SAGE†	Qwen3-2B	56.69	38.90
SAGE†	Qwen3-4B	64.81	44.89
Controlled comparison: 139 held-out subtasks			
Initial π_0^{GOAT}	Qwen3-2B	40.29	23.14
Existing scenes	Qwen3-2B	56.83	41.05
Failure replay	Qwen3-2B	61.87	41.32
SAGE (local)	Qwen3-2B	56.12	39.20
Random π_{R5}^{GOAT}	Qwen3-2B	61.15	43.16
Fail2Scene π_{D5}^{GOAT}	Qwen3-2B	**70.50**	**47.62**
Full-set evaluation (includes discovery)			
3D-Mem*	GPT-4o	62.9	44.7
SAGE†	Qwen3-2B	53.3	36.6
Fail2Scene π_{D5}^{GOAT}	Qwen3-2B	68.35	46.10

(b) A-EQA

Method	Backbone	Match ↑	Wtd. ↑
External baselines: 184 questions			
Explore-EQA*	GPT-4o	46.9	23.4
3D-Mem*	GPT-4o	52.6	42.0
3D-Mem†	Qwen3-2B	44.3	19.4
SAGE†	Qwen3-2B	53.21	37.07
SAGE†	Qwen3-4B	60.16	47.21
Controlled comparison: 184 questions			
Initial	Qwen3-2B	44.12	27.51
Random π_{R3}^{EQA}	Qwen3-2B	59.14	36.25
Fail2Scene π_{D3}^{EQA}	Qwen3-2B	**64.50**	**48.12**
Full set: 557 questions			
3D-Mem*	GPT-4o	53.3	38.0
SAGE†	Qwen3-2B	54.1	36.2
Initial	Qwen3-2B	40.16	22.52
Random π_{R3}^{EQA}	Qwen3-2B	59.69	40.18
Fail2Scene π_{D3}^{EQA}	Qwen3-2B	**62.24**	**42.33**

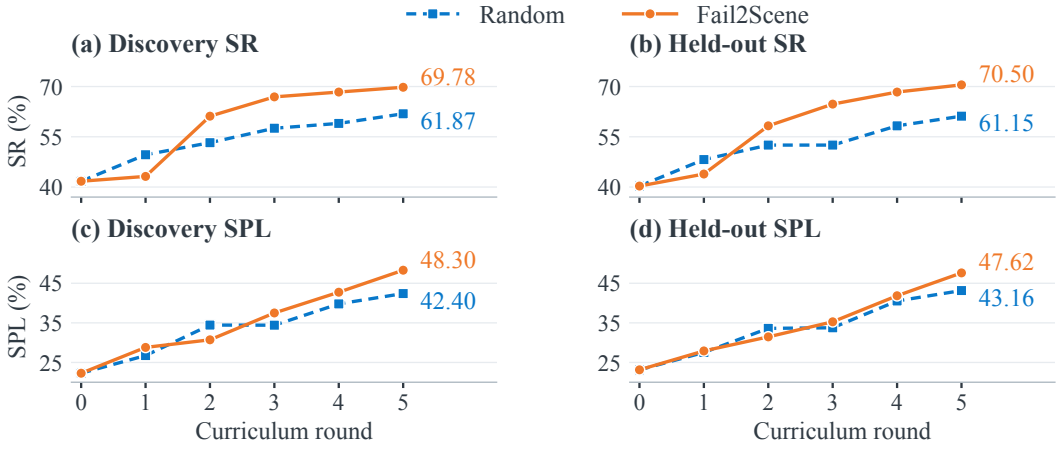


Figure 4: **GOAT performance across curriculum rounds.** Distance SR (top) and SPL (bottom) on discovery (left) and held-out (right) tasks, with 139 subtasks per split. Round 0 is the common initial policy. Cumulative scene budgets are 45, 93, 123, 176, and 216.

A-EQA. Fail2Scene improves answer coverage and overall A-EQA performance, raising LLM-match from 59.14 to 64.50 and its path-weighted counterpart from 36.25 to 48.12 under a shared evaluator on the 184-question evaluation in Table 1 (b). Both scores improve across all seven question types in Figure 8, while Fail2Scene answers 166 questions compared with 148 for random training, reducing the unanswered rate from 19.57% to 9.78%. Mean scores over each policy's answered questions are 71.49 and 73.53, respectively, with evidence discovery increasing from 65% to 70% and mean exploration steps decreasing from 14 to 13. On the full 557-question set, the unanswered rate also falls from 17.24% to 11.49%.

4.3 GOAT Ablation Studies

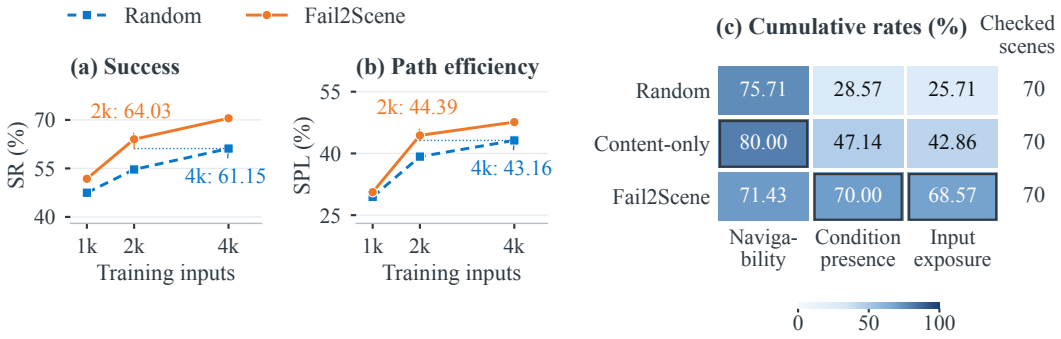


Figure 5: **GOAT sample efficiency and training conditions.** (a,b) Round-5 held-out SR/SPL at nominal budgets of 1,000, 2,000, and 4,000 candidate-selection inputs. (c) Cumulative navigability, condition presence, and input exposure (%) through round 5, using 70 checked scenes per curriculum.

Table 2: **GOAT round-5 ablations.** Held-out SR/SPL (%). ΔSR is relative to Full, before rounding.

Variant	SR ↑	SPL ↑	ΔSR
Full	**70.50**	**47.62**	0.00
Content-only	58.27	38.82	-12.23
No consolidation	66.19	44.26	-4.32
Frozen mechanisms	60.43	40.59	-10.07
No verification	65.47	45.33	-5.04

Table 2 examines what is lost when different parts of the construction loop are removed. Content-only conditioning removes competing-route relations and their filtering, reducing SR by 12.23 percentage points, the largest drop among these variants. Without cross-scene consolidation, the policy reaches 66.19% SR and 44.26% SPL. Grouping evidence from multiple scenes improves performance over retaining individual verified explanations. Removing verification yields 65.47% SR and 45.33% SPL, also below the full method. Freezing the mechanisms while continuing scene generation reduces SR by 10.07 percentage points, showing the value of revisiting failure conditions as the navigator changes. Appendix M reports the distinct SR and SPL responses to diagnostic-pool size, source-scene support, and explanation attempts.

4.4 GOAT Curriculum Analysis

Learning across curriculum rounds. Figure 4 shows how Fail2Scene's advantage emerges over successive rounds. On both discovery and held-out tasks, it trails random training in SR at round 1 and in SPL at round 2, then leads both metrics from round 3 onward. The later advantage is consistent with the frozen-mechanism ablation: continued generation with a fixed set of conditions leaves final performance below that of the adaptive curriculum.

Performance at matched nominal input budgets. Across the 1k, 2k, and 4k settings in Figure 5 (a,b), Fail2Scene improves SR and SPL. With 2k examples, it reaches 64.03% SR and 44.39% SPL, exceeding random training at 4k by 2.88 and 1.23 percentage points, respectively. Thus, the diagnostic curriculum reaches higher performance at half the nominal input budget, with gains persisting at 4k.

Conditions in generated scenes and training inputs. By GOAT round 5, cumulative checks of 70 scenes per curriculum show higher condition presence and input exposure for Fail2Scene than for the comparison curricula (Figure 5 (c)). Navigability is 50/70 for Fail2Scene and 53/70 for Random; scenes failing these checks are excluded before training.

Recurring conditions in source failures. Figure 6 shows that similar mistaken choices recur across source scenes: fixation on foreground objects at the expense of the available route, for example, occurs in six scenes. The retained mechanisms expand from five to eight over training, with each final-round mechanism supported by at least five scenes. Landmark–room misbinding is supported by 14 source scenes, compared with five for spatial-relation errors, indicating that consolidation re-

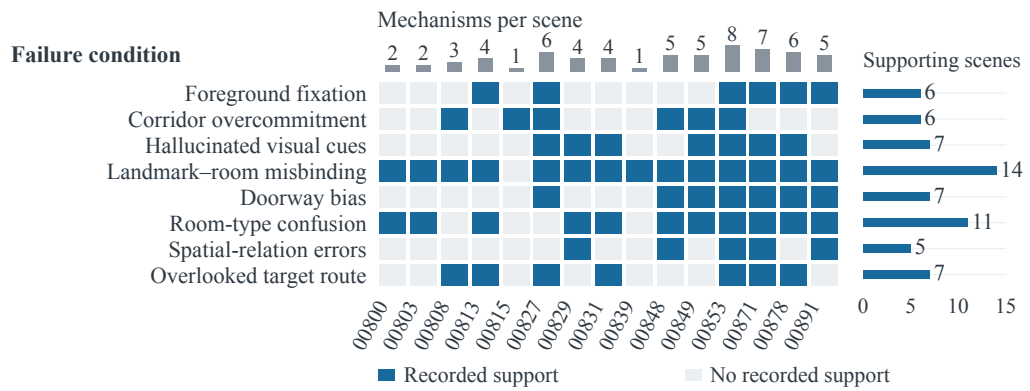


Figure 6: **Cross-scene support for GOAT round-5 mechanisms.** Filled cells identify source scenes supporting each failure condition, and row totals count distinct supporting scenes. Scene labels use HM3D numeric IDs.

Table 3: **GOAT round-5 navigation behavior (%).** (a) Failure rates on 139 held-out subtasks. Wrong termination follows goal observation. (b) Discovery SR by goal type on 139 subtasks, with task counts n. Bold compares the two trained policies.

(a) Failure types

Failure type	Initial	π_{R5}^{GOAT}	π_{D5}^{GOAT}
Execution or reachability	5.04	2.88	2.88
Goal unobserved	35.25	21.58	17.27
Wrong termination	17.27	12.23	7.91
Other	2.16	2.16	1.44
All failures	59.71	38.85	29.50

(b) Goal-type success

Goal (n)	π_{R5}^{GOAT}	π_{D5}^{GOAT}
Category (42)	73.81	**78.57**
Language (49)	67.35	**69.39**
Image (48)	45.83	**62.50**
Overall (139)	61.87	**69.78**

tains both broadly recurring confusions and conditions observed in fewer layouts. Together, these eight mechanisms cover 107 of 140 decisions examined for attribution, counting decisions assigned to multiple mechanisms only once. Such recurrence provides targets for practice across layouts.

Analysis of navigation behavior. On held-out tasks, Fail2Scene reduces unobserved-goal failures from 21.58% under random training to 17.27%, and wrong termination from 12.23% to 7.91% after five rounds. Table 3 (a) shows that execution failures remain at 2.88%. In counts, unobserved-goal failures fall from 30 to 24 and wrong terminations from 17 to 11, together accounting for 12 of the 13 fewer failures overall. This distribution connects the gain to acquiring target-relevant observations and using them to decide when to terminate.

Table 3 (b) breaks down discovery performance by goal type, as also reported in Table 10. SR improves for category, language, and image goals. Image goals account for eight of the eleven additional successes, with 30/48 successes compared with 22/48 under random training, while category and language goals contribute two and one. The 16.67-point SR gain for image goals is larger than the gains for category and language goals (4.76 and 2.04 points). Since image queries specify an instance, this concentration is consistent with practicing choices that distinguish a route to that instance from visually distracting alternatives.

5 Conclusion

In this work, we presented Fail2Scene, a framework for constructing training environments from a navigator's own failures. It turns recurring conditions in failed choices into practice environments that evolve with the navigator. After five rounds of training in AI2-THOR, Fail2Scene achieves 70.50% SR and 47.62% SPL on held-out GOAT-Bench buildings in Habitat, exceeding random curricula with the same scene budget by 9.35 and 4.46 percentage points. These gains demonstrate the value of failure-guided environment construction.

AI USE STATEMENT

In this work, we used generative AI tools for failure diagnosis and synthetic training-data generation, including scene specifications, navigation and question-answering tasks. AI also assisted polish writing. We take responsibility for the final content.

ETHICS STATEMENT

We adhere to the ICLR Code of Ethics. Quantitative evaluations use simulated indoor environments. Appendix O reports successful and unsuccessful robot navigation in an office.

REPRODUCIBILITY STATEMENT

Section 3 and Appendices A–F document the method, data construction, training and evaluation protocols, and hyperparameters. Prompts and robot deployment details are provided in Appendices P and O. Additionally, we have provided code in the supplementary materials that can be used for reproduction.

REFERENCES

Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir R. Zamir. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018. URL https://arxiv.org/abs/1807.06757.

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

Ata Çelen, Guo Han, Konrad Schindler, Luc Van Gool, Iro Armeni, Anton Obukhov, and Xi Wang. I-Design: Personalized LLM interior designer. In *Computer Vision – ECCV 2024 Workshops*, volume 15624 of *Lecture Notes in Computer Science*, pp. 217–234. Springer Nature Switzerland, 2025. doi: 10.1007/978-3-031-92387-6_17. URL https://link.springer.com/chapter/10.1007/978-3-031-92387-6_17. arXiv:2404.02838.

Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *Advances in Neural Information Processing Systems*, volume 33, pp. 4247–4258, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/2c75cf2681788adaca63aa95ae028b22-Abstract.html.

Jianming Chen, Yawen Wang, Junjie Wang, Xiaofei Xie, Shoubin Li, Qing Wang, and Fanjiang Xu. Where did it go wrong? capability-oriented failure attribution for vision-and-language navigation agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 28132–28150. Association for Computational Linguistics, 2026. URL https://aclanthology.org/2026.findings-acl.1402/.

Tianyuan Dai, Josiah Wong, Yunfan Jiang, Chen Wang, Cem Gokmen, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. Automated creation of digital cousins for robust policy learning. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning*

Research, pp. 4912–4943. PMLR, 2025. URL https://proceedings.mlr.press/v270/dai25a.html.

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-scale embodied AI using procedural generation. In *Advances in Neural Information Processing Systems*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/27c546ab1e4f1d7d638e6a8dfbad9a07-Abstract-Conference.html.

Yucheng Deng, Pingrui Lai, Xinhai Li, Chenjia Bai, Xiaoheng Deng, Chengnuo Sun, Xuelong Li, and Hua Yang. SpaceVLN: A Zero-Shot Vision-and-Language Navigation Agent with Online Spatial Cognitive Memory and Reasoning. *arXiv preprint arXiv:2606.08992*, 2026. URL https://arxiv.org/abs/2606.08992.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart J. Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In *Advances in Neural Information Processing Systems*, 2020. URL https://proceedings.nips.cc/paper_files/paper/2020/hash/985e9a46e10005356bbaf194249f6856-Abstract.html.

Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. AHA: A vision-language model for detecting and reasoning over failures in robotic manipulation. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.00371. arXiv:2410.00371.

Hongyan Feng, Sunlai Chen, Xuanyu Liu, Miao Pan, Yangfan Xie, Yuxiang Cui, Zhongxiang Zhou, Rong Xiong, Wenqi Zhang, Jianwei Yin, Yueting Zhuang, and Xuhong Zhang. Embodied-Navigator: Point, Think, Memorize, and Align for Efficient Navigation. *arXiv preprint arXiv:2608.17512*, 2026. URL https://arxiv.org/abs/2608.17512.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/fef126561bbf9d4467dbb8d27334b8fe-Abstract-Conference.html.

Mingfei Han, Liang Ma, Kamila Zhumakhanova, Ekaterina Radionova, Jingyi Zhang, Xiaojun Chang, Xiaodan Liang, and Ivan Laptev. RoomTour3D: Geometry-aware video-instruction tuning for embodied navigation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 27586–27596, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/papers/Han_RoomTour3D_Geometry-Aware_Video-Instruction_Tuning_for_Embodied_Navigation_CVPR_2025_paper.pdf.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/8b4add8b0aa8749d80a34ca5d941c355-Abstract-Conference.html.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, 2021. URL https://proceedings.mlr.press/v139/jiang21b.html.

Mukul Khanna, Ram Ramrakhya, Gunjan Chhablani, Sriram Yenamandra, Theophile Gervet, Matthew Chang, Zsolt Kira, Devendra Singh Chaplot, Dhruv Batra, and Roozbeh Mottaghi. GOAT-Bench: A benchmark for multi-modal lifelong navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16373–16383, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Khanna_GOAT-Bench_A_Benchmark_for_Multi-Modal_Lifelong_Navigation_CVPR_2024_paper.html.

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.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34892–34916, 2023a. URL https://papers.neurips.cc/paper_files/paper/2023/hash/6dcf277ea32ce3288914faf369fe6de0-Abstract-Conference.html.

Zeyi Liu, Arpit Bahety, and Shuran Song. REFLECT: Summarizing robot experiences for failure explanation and correction. In *Conference on Robot Learning*, 2023b. URL https://proceedings.mlr.press/v229/liu23g.html.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Jim Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/70c26937fbf3d4600b69a129031b66ec-Abstract-Conference.html.

Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, et al. OpenEQA: Embodied question answering in the era of foundation models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16488–16498, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Majumdar_OpenEQA_Embodied_Question_Answering_in_the_Era_of_Foundation_Models_CVPR_2024_paper.html.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *International Conference on Machine Learning*, 2022. URL https://proceedings.mlr.press/v162/parker-holder22a.html.

Nicholas Pfaff, Thomas Cohn, Sergey Zakharov, Rick Cory, and Russ Tedrake. SceneSmith: Agentic generation of simulation-ready indoor scenes. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.09153. arXiv:2602.09153.

Alexander Raistrick, Lingjie Mei, Karhan Kayan, David Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh, Stamatis Alexandropoulos, Lahav Lipson, Zeyu Ma, and Jia Deng. Infinigen indoors: Photorealistic indoor scenes using procedural generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/papers/Raistrick_Infinigen_Indoors_Photorealistic_Indoor_Scenes_using_Procedural_Generation_CVPR_2024_paper.pdf.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023. URL https://papers.neurips.cc/paper_files/paper/2023/file/d842425e4bf79ba039352da0f658a906-Paper-Conference.pdf.

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. doi: 10.48550/arXiv.2402.03300. URL https://arxiv.org/abs/2402.03300.

Zhixuan Shen, Jiawei Du, Ziyu Guo, Han Luo, Lilan Peng, Joey Tianyi Zhou, Haonan Luo, and Tianrui Li. Plan in sandbox, navigate in open worlds: Learning physics-grounded abstracted experience for embodied navigation. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2605.10118.

Fan-Yun Sun, Weiyu Liu, Siyi Gu, Dylan Lim, Goutam Bhat, Federico Tombari, Manling Li, Nick Haber, and Jiajun Wu. LayoutVLM: Differentiable optimization of 3D layout via vision-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Sun_LayoutVLM_Differentiable_Optimization_of_3D_Layout_via_Vision-Language_Models_CVPR_2025_paper.html.

Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems*, volume 34, pp. 251–266, 2021. URL https://proceedings.neurips.cc/paper/2021/file/021bbc7ee20b71134d53e20206bd6feb-Paper.pdf.

Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. DiffuScene: Denoising diffusion models for generative indoor scene synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Tang_DiffuScene_Denoising_Diffusion_Models_for_Generative_Indoor_Scene_Synthesis_CVPR_2024_paper.html.

Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. GenSim: Generating robotic simulation tasks via large language models. In *International Conference on Learning Representations*, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/143ea4a156ef64f32d4d905206cf32e1-Abstract-Conference.html.

Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. RoboGen: Towards unleashing infinite data for automated robot learning via generative simulation. In *International Conference on Machine Learning*, 2024b. URL https://proceedings.mlr.press/v235/wang24cc.html.

Zun Wang, Jialu Li, Yicong Hong, Songze Li, Kunchang Li, Shoubin Yu, Yi Wang, Yu Qiao, Yali Wang, Mohit Bansal, and Limin Wang. Bootstrapping language-guided navigation learning with self-refining data flywheel. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/3b00f67a2e03916b26f56b66b38445f7-Abstract-Conference.html.

Jiahao Xu, Peiyuan Wang, Hanzhuo Zhang, Zihao Yu, Tianyu Fu, Hao Chen, Xuanhao Xiang, Jianbo Yu, Chenchen Fu, and Wanyuan Wang. Grasp-Then-Plan with Failure Attribution: A Closed Two-Stage Framework for Precise and Generalizable Robotic Manipulation. *arXiv preprint arXiv:2606.03385*, 2026a. URL https://arxiv.org/abs/2606.03385.

Qinwen Xu, Jiaming Liu, Rui Zhou, Shaojun Shi, Nuowei Han, Zhuoyang Liu, Chenyang Gu, Shuo Gu, Yang Yue, Gao Huang, Wenzhao Zheng, Sirui Han, Peng Jia, and Shanghang Zhang. TwinRL: Digital twin-driven reinforcement learning for real-world robotic manipulation, 2026b. URL https://arxiv.org/abs/2602.09023v4.

Yandan Yang, Baoxiong Jia, Shujie Zhang, and Siyuan Huang. SceneWeaver: All-in-one 3D scene synthesis with an extensible and self-reflective agent. In *Advances in Neural Information Processing Systems*, 2025a. URL https://arxiv.org/abs/2509.20414. arXiv:2509.20414.

Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, Chris Callison-Burch, Mark Yatskar, Aniruddha Kembhavi, and Christopher Clark. Holodeck: Language guided generation of 3D embodied AI environments. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Yang_Holodeck_Language_Guided_Generation_of_3D_Embodied_AI_Environments_CVPR_2024_paper.html.

Yuncong Yang, Han Yang, Jiachen Zhou, Peihao Chen, Hongxin Zhang, Yilun Du, and Chuang Gan. 3D-Mem: 3D scene memory for embodied exploration and reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17294–17303,

2025b. URL https://openaccess.thecvf.com/content/CVPR2025/papers/Yang_3D-Mem_3D_Scene_Memory_for_Embodied_Exploration_and_Reasoning_CVPR_2025_paper.pdf.

Hang Yin, Xiuwei Xu, Linqing Zhao, Ziwei Wang, Jie Zhou, and Jiwen Lu. UniGoal: Towards universal zero-shot goal-oriented navigation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 19057–19066, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/papers/Yin_UniGoal_Towards_Universal_Zero-shot_Goal-oriented_Navigation_CVPR_2025_paper.pdf.

Bangguo Yu, Hamidreza Kasaei, and Ming Cao. L3MVN: Leveraging large language models for visual target navigation. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3554–3560. IEEE, 2023. URL https://research.rug.nl/en/publications/l3mvn-leveraging-large-language-models-for-visual-target-navigati/.

Kuo-Hao Zeng, Zichen Zhang, Kiana Ehsani, Rose Hendrix, Jordi Salvador, Alvaro Herrasti, Ross Girshick, Aniruddha Kembhavi, and Luca Weihs. PoliFormer: Scaling on-policy RL with transformers results in masterful navigators. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 408–432. PMLR, 2025. URL https://proceedings.mlr.press/v270/zeng25a.html.

Jiawei Zhang, Hongsong Wang, and Pan Zhou. ScenePilot: Grow-and-Repair Policy for Text-Driven 3D Indoor Scene Generation. *arXiv preprint arXiv:2608.30307*, 2026. URL https://arxiv.org/abs/2608.30307.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. In *International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/zhang25cq.html.

Kaiwen Zhou, Kaizhi Zheng, Connor Pryor, Yilin Shen, Hongxia Jin, Lise Getoor, and Xin Eric Wang. ESC: Exploration with soft commonsense constraints for zero-shot object navigation. In *International Conference on Machine Learning*, pp. 42829–42842. PMLR, 2023. URL https://proceedings.mlr.press/v202/zhou23r.html.

APPENDIX

CONTENTS

A. Failure Attribution Details 15
B. Scene Construction Details 16
C. Training Example Construction 17
D. Policy Optimization Details 18
E. Evaluation Protocol 19
F. Hyperparameters 20
G. Additional GOAT Results 20
H. GOAT Termination and Paired Outcomes 22
I. GOAT Attribution Evidence 23
J. Generated Scenes for GOAT 23
K. GOAT Training Resources 24
L. A-EQA Analysis 25
M. GOAT Parameter Sensitivity 27
N. Qualitative Analysis 28
O. Real-Robot Experiments 31
P. Prompt Design 32

A FAILURE ATTRIBUTION DETAILS

The examiner, verifier, and controller perform diagnosis, verification, and update control using fixed Qwen-3.7max parameters. The examiner also writes scene specifications within the same requests, whose joint outputs pair an explanation with a generator input stored as `scene_recipe`. The verifier checks decision evidence and generated scenes, while the controller invokes tools and applies the policy-selection rule. Table 4 specifies their interfaces and diagnostic access to simulator state and goal locations.

A.1 EVIDENCE VERIFICATION

To concentrate attribution on decisions with a geometric alternative, we require a positive geodesic gap and at least two finite-distance candidates, including the selected action. Selection first keeps the largest $B = 5$ gaps per failed subtask and then the largest $N = 200$ across the round, retaining smaller pools in full. Final subtask outcomes determine eligibility, whereas distances are recorded at decision time and each decision identifier preserves its source trace.

For every selected decision, X_t combines the query, chosen and closest-candidate views, up to $n_{\text{ctx}} = 2$ additional views in recorded order, the rationale, and the distance comparison. From this fixed evidence, the examiner jointly produces a one-sentence explanation and a specification, both of which are regenerated if verification requests another attempt. Verification uses this decision-time evidence for up to $r = 4$ attempts, including the initial pair.

To identify support beyond an individual scene, consolidation receives all accepted explanations, specifications, and decision identifiers from the current round in one request. For each semantic group, the examiner jointly returns a unified explanation defining the mechanism, a specification passed directly to the generator, and supporting identifiers. The examiner performs semantic grouping in this request. Returned identifiers are matched to attribution records to verify support from at

Table 4: **Functional tool interfaces for the fixed agent roles.** E, V, and C denote examiner, verifier, and controller. Specification drafting belongs to E, while C launches generation.

Tool	Returns	Category	Roles
`get_candidates`(t)	$\mathcal{C}_t$ with κ_c, p_c	state obs.	E, V
`render`(c)	candidate view I_c	visual obs.	E, V
`trace_query`(τ, t)	rationale, subtask id, memory state	state obs.	E, V
`geodesic`(p, g)	$d_e(p, g)$	verification	E, V
`oracle_best`(t)	$a_t^\star$, Δ_t	verification	E, V
`evidence_support`(m)	supporting decisions and source scenes	state obs.	V, E
`spec_lint`(s)	violations of $\mathcal{R}$	verification	V, E
`instantiate`(s)	realized environment e'	scene mod.	C
`navigability`(e')	reachable positions, floor offset	verification	V
`curriculum_stats`$(\mathcal{E})$	scale distributions, discrepancies	verification	V

least $n_{\text{sc}} = 2$ distinct scenes, with the ledger preserving these links alongside generation status and curriculum outcomes.

A.2 Interface Labels

Candidate types provide a deterministic interface-level summary of each selected decision. Comparing the chosen and closest candidates gives

$$\ell_t = \begin{cases} \texttt{recall_miss}, & \kappa_{a_t^\star} = \text{memory} \ \wedge \ \kappa_{a_t} = \text{frontier}, \\ \texttt{commit_wrong}, & \kappa_{a_t^\star} = \text{frontier} \ \wedge \ \kappa_{a_t} = \text{memory}, \\ \texttt{frontier_bad}, & \text{otherwise.} \end{cases} \tag{7}$$

The label `frontier_bad` is the residual category under this definition: it includes both frontier-to-frontier and memory-to-memory contrasts.

B Scene Construction Details

Scene construction invokes a Holodeck-based backend (Yang et al., 2024), with AI2-THOR (Kolve et al., 2017) providing simulation and rendering. Each specification launches $J = 10$ generation attempts in the experiments.

Specification construction. Because specification drafting is integrated into both decision-level and consolidation requests, accepted explanations and supporting identifiers jointly inform each mechanism and generator input under $\mathcal{R}$. Specifications are truncated to $L_{\max} = 20$ whitespace-delimited words, while the prompt requests at least three explicitly connected room types and at most one salient named object when it attracted the original choice.

Navigability. On a 0.25 m grid, a scene requires at least $\nu_{\min} = 60$ reachable positions and a largest connected component containing at least 80% of them. The floor-offset check uses

$$\delta(e') = \min_{p \in P_{\text{reach}}} p_y - 0.901\,\text{m} - \min_{v \in P_{\text{floor}}} v_y, \qquad |\delta(e')| \le \delta_{\max} = 0.05\,\text{m}, \tag{8}$$

where P_{reach} is the reachable-position set and P_{floor} contains all room-floor polygon vertices. The quantities p_y and v_y denote the vertical coordinates of position p and vertex v. The 0.901 m correction converts the agent origin to its standing surface.

Mechanism validation. For a sampled task, $U = (g, p, \mathcal{C}(p), \{I_c\}_{c \in \mathcal{C}(p)})$ contains the goal, decision point, candidates, and their rendered views. The verifier uses U, room polygons and areas, opening connectivity, the navigation mesh, and geodesic distances to assess the relation described by m_k. The judgment V_{mech} applies to this sampled configuration. For the open-room mechanism, the candidate views must present both a salient room and a competing connected route toward the same goal. Training inputs are subsequently assembled from trajectory views as described in Appendix C.

Curriculum checks. Specification checks require equal numbers of diagnostic and random specifications, no shared specification text, a maximum length of 20 words, and an absolute difference in mean length of at most $\lambda = 5$ words. Random wording permits up to four sampling attempts before the independent comparability check. Prompts request at least three connected room types and at most one salient named object. Separate checks evaluate the generated geometry.

Environment-scale checks use successful generation records. For room counts and object counts separately, let D and R denote the Fail2Scene and Random curriculum arms, n_D, n_R their numbers of successful generation records, $\bar{x}_D, \bar{x}_R$ their mean counts, and s_D, s_R their population standard deviations. The scale discrepancy d is computed as

$$d = \frac{|\bar{x}_D - \bar{x}_R|}{\sqrt{\left[(n_D - 1)s_D^2 + (n_R - 1)s_R^2\right]/(n_D + n_R - 2)}}, \qquad d \le d_{\max} = 0.8. \tag{9}$$

A comparison fails when either arm has fewer than two records or the weighted scale in the denominator is zero. Each arm also requires a fraction of at most $\rho_{\max} = 0.15$ of successful scenes to have one or fewer rooms. Specification-level retained counts track coverage, while matched cumulative scene budgets define the training arms. The random curriculum uses the same specification-writing interface, constraints, generation backend, and budget without mechanisms or supporting decisions. Random scenes are checked for navigability and agreement with their specifications. Diagnostic scenes must additionally pass V_{mech}.

Generation failures. A failed generation attempt leaves its specification unchanged for any subsequent retry and does not halt the remaining generation attempts. Scenes that fail health checks are excluded without automatic repair, retry, or replacement. Probe startup or execution exceptions allow at most two attempts, with an additional stop condition for environment exceptions. Only scenes that pass these checks enter the retained training pool.

C Training Example Construction

Trajectory collection. Trajectory collection selects a visible target and terminal pose, then uses A* to plan a path from a distant reachable start and renders forward and side views along it. Intermediate forward views face the next waypoint, whereas the terminal forward view faces the target. Before candidate construction, trajectories are screened for endpoint visibility, path length, and detected-object counts, followed by reward-based difficulty filtering.

Reference assignment. The loader orders viewpoints by their stored indices and treats the largest index as the terminal point, allowing that point and every noninitial intermediate point to supply a forward reference. Each sample requires forward and two side-view files, represented by numeric rotation suffixes or front/left/right filenames. Queries are collected from the generated-question field, stored EQA question, and templated object-goal query when available, then deduplicated, while image goals are loaded from their recorded paths.

Candidate assembly. To assemble alternatives for a reference at p_t, the sampler draws three distinct views uniformly from the non-reference observations available at nodes $p_0, \ldots, p_t$. This pool contains the reference point's side views and observations from earlier nodes. If this pool is too small, it falls back to the reference point's two side views and a view from a randomly chosen earlier point, excluding duplicate files. We use four-candidate inputs, with shuffling performed before grouping so that displayed indices are local to each category.

Candidate types follow trajectory position: side views at the reference point and earlier forward views enter the frontier group, whereas other sampled views enter memory. The reference itself is a frontier at an intermediate point and a memory at the terminal point. For P complete viewpoints, the loader creates $P - 2$ intermediate examples and repeats the terminal example $P - 2$ times per query.

Memory views include object names or detected-object labels alongside their images. Training and validation use separate environment directories, whose preassigned splits are preserved by the loader.

Illustrated construction. Figure 3 uses a locally re-rendered four-room scene and two recorded illustrative draws from one trajectory. For the intermediate reference at p_2, the sampled views are p_2 right, p_1 forward, and p_1 left. Shuffling yields three frontier candidates and one memory candidate. For the terminal reference at p_T, the sampled views are p_2 forward, p_1 left, and p_T right, yielding two candidates of each type.

D Policy Optimization Details

Response reward. Optimization acts on sampled responses y to a candidate-selection input x. A reference response $y^\star(x)$ specifies the candidate type and image index, plus answer text for memory selections. Combining selection accuracy with output-format validity gives

$$R(x, y) = 0.9\, A(x, y) + 0.1\, F(y), \tag{10}$$

Both training arms use format weight 0.1. The binary check $F(y)$ accepts `Frontier Image i` followed by a newline and `Reason:`, or `Memory Image i` followed by a newline and `Answer:`. Before matching, whitespace around <, >, and / is removed, after which a case-insensitive pattern searches the response while allowing variable whitespace.

Accuracy $A(x, y)$ equals -10 when format validation fails. Otherwise, an incorrect candidate type receives -1, while a correct type with an incorrect image index receives -0.5. Selecting the correct frontier receives 1 regardless of rationale content, whereas selecting the correct memory image receives $1 + s$, where s is the character-sequence matching ratio from `difflib.SequenceMatcher` applied to stripped predicted and reference answers. Setting $s = 0$ when either answer is empty yields total rewards of 1 for a correct frontier response and $1 + 0.9s$ for a correct memory response. The implementation returns accuracy 0 if a response–reference pair cannot be parsed, including an unparseable reference.

Both the prompt and reward operate at image level, with a memory response specifying an image index and answer text while object labels accompany the input images.

GRPO objective. For each input x, behavior policy π_{old} samples G responses $y_1, \ldots, y_G$ with rewards $R_j = R(x, y_j)$. Group-relative advantages compare each response with the other samples, while token likelihood ratios track its probability under the updated policy:

$$\widehat{A}_j = \frac{R_j - \bar{R}}{\sigma_R + \varepsilon}, \qquad \rho_{j,l}(\theta) = \frac{\pi_\theta(y_{j,l} \mid x, y_{j,<l})}{\pi_{\text{old}}(y_{j,l} \mid x, y_{j,<l})}, \tag{11}$$

where $\bar{R}$ and σ_R are the group mean and sample standard deviation, using a one-degree-of-freedom correction, and $\varepsilon = 10^{-6}$ stabilizes normalization. We use $G = 5$ responses per training input. Here, $y_{j,l}$ is token l of output j, $y_{j,<l}$ is its preceding prefix, and $|y_j|$ is the output length in tokens. For token (j, l), define the clipped surrogate

$$\begin{aligned} B_{j,l}(\theta) &= \min\Big\{\rho_{j,l}(\theta)\widehat{A}_j, \operatorname{clip}(\rho_{j,l}(\theta), 0.8, 1.3)\widehat{A}_j\Big\}, \\ L_{j,l}(\theta) &= \begin{cases} \max\{B_{j,l}(\theta), 3\widehat{A}_j\}, & \widehat{A}_j < 0, \\ B_{j,l}(\theta), & \widehat{A}_j \geq 0. \end{cases} \end{aligned} \tag{12}$$

The asymmetric bounds correspond to $\epsilon_{\text{low}} = 0.2$ and $\epsilon_{\text{high}} = 0.3$. The factor 3 implements dual clipping for negative advantages. Let $\mathcal{B}$ be a batch of input–response pairs (x, y_j) sampled from π_{old}, with j indexing responses within each input's group. Averaging over all unmasked response tokens, we maximize

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{\mathcal{B}}\left[\frac{\displaystyle\sum_{(x,y_j)\in\mathcal{B}}\sum_{l=1}^{|y_j|}\big(L_{j,l}(\theta) - \beta\mathcal{K}_{j,l}(\theta)\big)}{\displaystyle\sum_{(x,y_j)\in\mathcal{B}}|y_j|}\right], \tag{13}$$

where $|y_j|$ counts unmasked response tokens and $\beta = 0.01$ weights the KL penalty. Against a fixed reference policy π_{ref}, the low-variance KL estimator uses $u = \log \pi_{\text{ref}}(y_{j,l} \mid x, y_{j,<l}) - \log \pi_\theta(y_{j,l} \mid x, y_{j,<l})$ and $\mathcal{K}_{j,l} = \exp(u) - u - 1$. Numerical guards clip u to $[-20, 20]$ and the resulting penalty to $[-10, 10]$. Diagnostic and random curricula use the same reward and optimization settings.

Checkpoint selection. Within each run, generated-scene validation selects the checkpoint maximizing mean response reward,

$$S_{\text{gen}}(\theta) = \frac{1}{M} \sum_{j=1}^{M} R(x_j, y_j), \tag{14}$$

where M is the number of validation responses and each y_j is generated by π_θ for validation input x_j. The best step changes on a strict increase in this score. For outer-loop selection, the controller invokes navigation evaluation on discovery tasks $\mathcal{V}$ and computes

$$S_{\mathcal{V}}(\theta) = \tfrac{1}{2}\big[\text{SR}_{\mathcal{V}}(\theta) + \text{SPL}_{\mathcal{V}}(\theta)\big], \tag{15}$$

where both metrics are percentages under the 1.0 m distance success criterion. The candidate replaces the retained navigator only if its discovery score increases. Otherwise, the previous navigator is retained, according to

$$\theta^{(i)} = \begin{cases} \widetilde{\theta}^{(i)}, & S_{\mathcal{V}}(\widetilde{\theta}^{(i)}) > S_{\mathcal{V}}(\theta^{(i-1)}), \\ \theta^{(i-1)}, & \text{otherwise.} \end{cases} \tag{16}$$

As accepted scenes accumulate, the retained navigator supplies the next round's diagnostic traces. GOAT-Bench experiments allow at most five outer-loop update attempts.

E Evaluation Protocol

Evaluation scope. GOAT-Bench evaluation separates 18 discovery from 18 held-out buildings, with 139 subtasks in each split. Discovery tasks supply failure traces and determine cross-round acceptance. Gradient updates use generated AI2-THOR training environments, while disjoint generated validation environments select the candidate checkpoint within each run. Held-out Habitat buildings are reserved for evaluation. Task order is randomized, and visual memory is retained across subtasks within an episode but reset between episodes. Full-set navigation combines discovery and held-out buildings. A-EQA results are reported separately for the 184-question subset and full 557-question evaluation.

Policy and curriculum notation. Diagnosis is the failure-attribution stage within Fail2Scene, and "diagnostic" describes the curriculum it constructs. The labels D and R identify the Fail2Scene and Random training arms, whereas π_D and π_R denote their learned navigation policies. Superscripts distinguish separately adapted GOAT and A-EQA policies, and numerical policy subscripts identify curriculum rounds. Both arms are evaluated on discovery and held-out tasks. These Habitat evaluation splits are distinct from the generated AI2-THOR training and validation environments.

Candidate indexing. The four-view construction defines a training input. During navigation, the numbers of frontier and memory candidates vary with exploration. The evaluation interface serializes them in separate groups and assigns zero-based indices within each group. Memory indices refer to the displayed candidate list and are mapped back to the stored snapshots after selection. An image-goal query supplies a target image alongside the candidate views.

E.1 Navigation Metrics

Let n be the number of evaluated subtasks. For subtask j, let p_j^{end} be the terminal agent position, $\mathcal{G}_j$ the set of valid goal viewpoints, ℓ_j the reference shortest-path length from the subtask start, and L_j the traveled path length. The two success indicators are

$$S_j^{\text{dist}} = \mathbf{1}\left[\min_{g \in \mathcal{G}_j} d_{\text{geo}}(p_j^{\text{end}}, g) < 1.0\,\text{m}\right], \tag{17}$$

$$S_j^{\text{snap}} = \mathbf{1}[\text{the selected memory view contains the goal}]. \tag{18}$$

Table 5: **GOAT and A-EQA training settings.** The update-attempt limit applies to GOAT. Method parameters are listed in Table 6.

Setting	Value
Navigator	Qwen3-VL-2B
Construction model	Qwen-3.7max (fixed)
Independent training runs	1
Views per input / responses per group	4 / 5
Selection / format reward weights	0.9 / 0.1
Policy-ratio bounds / dual clip	[0.8, 1.3] / 3
KL weight / GOAT update attempts	0.01 / 5

For $u \in \{\mathrm{dist}, \mathrm{snap}\}$, we report

$$\mathrm{SR}^u = \frac{100}{n}\sum_{j=1}^{n} S_j^u, \qquad \mathrm{SPL}^u = \frac{100}{n}\sum_{j=1}^{n} S_j^u \frac{\ell_j}{\max(\ell_j, L_j)}. \tag{19}$$

The path factor follows SPL (Anderson et al., 2018). Distance and Snapshot scores share the path factor and differ in their success indicator. If H_j is the number of high-level decisions, then $\mathrm{Steps} = n^{-1}\sum_j H_j$. Steps averages all evaluated subtasks. Displayed rates are rounded to two decimals, while differences in count-based SR are computed from unrounded rates.

E.2 Question Answering Metrics

For question j, let $a_j \in \{0, 1\}$ indicate whether an answer is submitted, $q_j \in [0, 100]$ its normalized LLM-match score, and $w_j \in [0, 1]$ the path-efficiency weight assigned by the question-answering evaluator. Unanswered questions have $q_j = 0$. For Q questions,

$$\text{LLM-match} = \frac{1}{Q}\sum_{j=1}^{Q} q_j, \qquad \text{LLM-match SPL} = \frac{1}{Q}\sum_{j=1}^{Q} q_j w_j. \tag{20}$$

Answer coverage, unanswered count, and the conditional answer score are

$$C = \frac{1}{Q}\sum_{j} a_j, \qquad U = Q - \sum_{j} a_j, \qquad \overline{q}_{\mathrm{answered}} = \frac{\sum_j a_j q_j}{\sum_j a_j}. \tag{21}$$

Thus the overall score equals $C\,\overline{q}_{\mathrm{answered}}$ when at least one question is answered. Conditional means use each policy's answered subset. For disjoint question types c containing Q_c questions, the aggregate score is $\sum_c (Q_c/Q)\overline{q}_c$.

E.3 Experimental Configuration

Construction uses fixed Qwen-3.7max parameters, whereas the Qwen3-VL-2B navigator is optimized from four-view inputs with five GRPO responses per input. Checkpoints are selected within a run by generated-scene validation reward and accepted across rounds according to $(\mathrm{SR}_{\mathcal{V}} + \mathrm{SPL}_{\mathcal{V}})/2$ on discovery tasks. Table 5 provides the remaining settings for navigation and question-answering experiments.

F Hyperparameters

G Additional GOAT Results

G.1 Performance across Curriculum Rounds

Tables 7 and 8 report GOAT discovery and held-out evaluations over rounds 0–5, with 139 subtasks per split. Cumulative training-scene budgets grow from 45 to 216 over rounds 1–5 for both curricula.

Table 6: **GOAT and A-EQA method settings.** The outer-loop limit $I_{\max}$ applies to GOAT. Each specification launches $J = 10$ generation jobs. Room-type and salient-object bounds are drafting instructions, while the other listed acceptance thresholds are numerical checks.

Symbol	Meaning	Value
N	maximum diagnostic decisions per round	200
B	maximum diagnostic decisions per subtask	5
n_{ctx}	additional candidate images per attribution	2
r	sampling attempts, including the first	4
n_{sc}	minimum source scenes supporting a mechanism	2
K	mechanisms admitted in the round	attribution-dependent
J	generation jobs per specification	10
$L_{\max}$	maximum specification length (words)	20
n_{rm}	requested minimum distinct room types	3
n_{sal}	requested maximum named salient objects	1
$\nu_{\min}$	minimum reachable positions	60
	minimum largest-component fraction	0.80
$\delta_{\max}$	maximum absolute floor offset (m)	0.05
λ	maximum mean specification-length gap	5 words
$d_{\max}$	maximum cross-arm scale discrepancy	0.8
$\rho_{\max}$	maximum fraction with at most one room	0.15
G	responses per training input	5
$\epsilon_{\text{low}}, \epsilon_{\text{high}}$	policy-ratio clipping offsets	0.2, 0.3
	dual-clip factor	3.0
β	KL regularization weight	0.01
ε	advantage-normalization stabilizer	10^{-6}
$I_{\max}$	maximum GOAT curriculum update attempts	5

Table 7: **GOAT discovery performance over rounds 0–5 (%).** Bold compares curricula within each round.

Policy	Houses	Dist. SR	Dist. SPL	Snap. SR	Snap. SPL	Steps
π_0^{GOAT}	0	41.73	22.28	16.55	7.57	5.95
π_{D1}^{GOAT}	45	43.17	**28.80**	**18.71**	**14.34**	13.03
π_{R1}^{GOAT}	45	**49.64**	26.74	17.27	13.72	11.62
π_{D2}^{GOAT}	93	**61.15**	30.72	**36.69**	**15.63**	12.09
π_{R2}^{GOAT}	93	53.24	**34.44**	26.62	14.02	13.52
π_{D3}^{GOAT}	123	**66.91**	**37.50**	**44.60**	**23.35**	11.11
π_{R3}^{GOAT}	123	57.55	34.41	33.81	14.39	11.63
π_{D4}^{GOAT}	176	**68.35**	**42.74**	**48.92**	**26.92**	10.64
π_{R4}^{GOAT}	176	58.99	39.79	34.53	16.75	11.56
π_{D5}^{GOAT}	216	**69.78**	**48.30**	**53.24**	**34.27**	9.45
π_{R5}^{GOAT}	216	61.87	42.40	36.69	19.10	10.53

Table 8: **GOAT held-out performance over rounds 0–5 (%).** Bold compares curricula within each round.

Policy	Houses	Dist. SR	Dist. SPL	Snap. SR	Snap. SPL	Steps
π_0^{GOAT}	0	40.29	23.14	15.83	8.21	6.32
π_{D1}^{GOAT}	45	43.88	**27.91**	**19.42**	13.68	12.46
π_{R1}^{GOAT}	45	**48.20**	27.52	16.55	**14.25**	12.18
π_{D2}^{GOAT}	93	**58.27**	31.46	**33.81**	**16.41**	11.57
π_{R2}^{GOAT}	93	52.52	**33.61**	28.78	14.36	12.08
π_{D3}^{GOAT}	123	**64.75**	**35.27**	**48.92**	**22.68**	10.56
π_{R3}^{GOAT}	123	52.52	33.76	40.29	19.11	12.07
π_{D4}^{GOAT}	176	**68.35**	**41.85**	**49.64**	**27.63**	10.18
π_{R4}^{GOAT}	176	58.27	40.53	45.32	26.08	10.22
π_{D5}^{GOAT}	216	**70.50**	**47.62**	**51.08**	**33.51**	9.97
π_{R5}^{GOAT}	216	61.15	43.16	46.04	27.83	10.06

Table 9: Full-set GOAT round-5 navigation, including discovery buildings (%).

Policy	Dist. SR	Dist. SPL	Snap. SR	Snap. SPL	Steps
Fail2Scene π_{D5}^{GOAT}	68.35	46.10	49.64	31.47	10.32

G.2 Full-Set Navigation

Table 9 reports full-set navigation over discovery and held-out buildings.

Table 10: **GOAT round-5 discovery success by goal type (%).** Random and Fail2Scene are evaluated on the same 139 subtasks. Bold compares policies within each row.

Goal type	n	Random SR	Fail2Scene SR
Category	42	73.81	**78.57**
Language	49	67.35	**69.39**
Image	48	45.83	**62.50**
Overall	139	61.87	**69.78**

G.3 Performance at Matched Input Budgets

Table 11 gives the exact values plotted in Figure 5 (a,b), comparing the curricula at 1k, 2k, and 4k training inputs. Appendix K defines the budget unit.

H GOAT Termination and Paired Outcomes

GOAT round 3 is the first round in which Fail2Scene exceeds Random in both SR and SPL on both discovery and held-out tasks (Tables 7 and 8). We examine paired subtask outcomes at this stage to characterize where the curricula differ as this advantage emerges. On the 139 discovery subtasks,

Table 11: **GOAT round-5 performance versus nominal input budget.** One example contains a query, four candidates, and a reference response. Budget percentages are relative to 4,000 examples.

Curriculum	Budget	Examples	SR	SPL
Random	25%	1000	47.48	29.43
Fail2Scene	25%	1000	51.80	30.59
Random	50%	2000	54.68	39.23
Fail2Scene	50%	2000	64.03	44.39
Random	100%	4000	61.15	43.16
Fail2Scene	100%	4000	70.50	47.62

Table 12: **GOAT round-3 termination types on discovery tasks.** Percentages use 139 subtasks per arm. Termination labels are recorded separately from geometric success.

Termination	Random	Fail2Scene
success	52.52	66.19
fail_wrong_place	18.71	19.42
fail_timeout	11.51	9.35
abort_parse	17.27	5.04

Figure 7 shows 70 successes shared by both policies, 23 exclusive to Fail2Scene, 10 exclusive to Random, and 36 failures under both, using the distance-success criterion. Table 12 reports the recorded termination labels from these same GOAT round-3 runs.

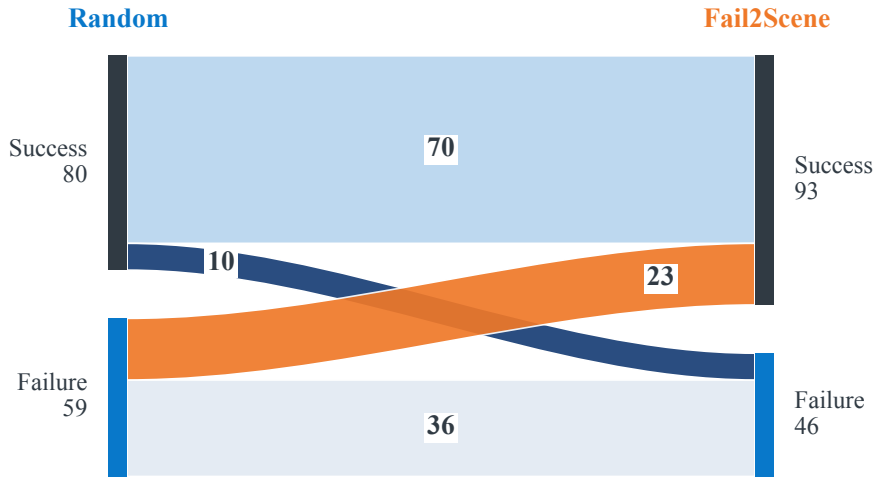


Figure 7: **Paired GOAT round-3 outcomes on discovery tasks.** Random and Fail2Scene are evaluated on the same 139 subtasks under the distance-success criterion. Ribbon widths and labels indicate subtask counts: 70 succeed under both policies, 10 only under Random, 23 only under Fail2Scene, and 36 under neither.

I GOAT Attribution Evidence

I.1 Coverage Denominators

Attribution coverage uses unique input decisions as its denominator and counts each covered decision once, even when it supports multiple mechanisms. Per-mechanism support instead counts distinct source scenes, so one scene may contribute to several rows in Figure 6.

J Generated Scenes for GOAT

J.1 Cumulative Scene Checks at GOAT Round 5

Figure 5 (c) reports three scene-level rates for 70 checked scenes per curriculum from the generated scene collection accumulated through GOAT round 5. Each rate uses all 70 checked scenes as its denominator. Navigability counts scenes that pass the scene checks. Condition presence counts scenes exhibiting the diagnosed relations. Input exposure counts scenes whose inspected training queries and four-candidate sets exhibit those relations; each scene contributes one count regardless of its number of training inputs. Kitchen–route exposure, for example, requires both options to appear as candidates for the same goal.

The cumulative checks give counts of $(53, 20, 18)$ for navigability, condition presence, and input exposure for Random, $(56, 33, 30)$ for Content-only, and $(50, 49, 48)$ for Fail2Scene, each

Table 13: **Distributions of training scenes added in GOAT round 5.** Values are median [Q1, Q3], where Q1 and Q3 are the 25th and 75th percentiles. Each curriculum adds 40 training scenes in this round.

Quantity	Random	Content-only	Fail2Scene
Area (m^2)	78.00 [72.00, 88.00]	60 [50,100]	54.00 [48.00, 60.00]
Rooms	4.00 [4.00, 5.00]	3 [2,5]	2.00 [2.00, 3.00]
Objects	57.00 [50.50, 65.25]	45 [30,60]	43.50 [32.75, 57.50]
Planned path (m)	12.50 [9.99, 16.06]	12.79 [9.45,16.23]	9.36 [8.02, 11.37]

Table 14: **GOAT training partitions by curriculum round.** Scene counts are the increments of the cumulative budgets in Tables 7 and 8. Area, rooms, and path length are medians [Q1, Q3], with Q1 and Q3 denoting the 25th and 75th percentiles. D and R denote Fail2Scene and Random; – indicates unreported statistics.

Round	Scenes	Traj.	Area (m^2)	Rooms	Planned path (m)
D_1	45	59	78.00 [72.00, 88.00]	4.00 [4.00, 5.00]	13.13 [9.78, 16.09]
R_1	45	60	57.00 [57.00, 80.00]	3.00 [3.00, 4.00]	11.43 [9.90, 14.01]
D_2	48	164	74.00 [60.00, 86.00]	4.00 [4.00, 4.00]	12.10 [9.97, 14.28]
R_2	48	149	57.00 [57.00, 80.00]	3.00 [3.00, 4.00]	12.11 [10.01, 13.80]
D_3	30	374	60.00 [50.00, 98.00]	3.00 [2.00, 5.00]	11.61 [9.16, 15.19]
R_3	30	383	90.00 [66.00, 103.50]	4.00 [4.00, 4.00]	12.55 [9.64, 16.12]
D_4	53	393	75.00 [62.00, 98.75]	3.00 [3.00, 3.00]	12.61 [10.75, 13.96]
R_4	53	388	80.00 [60.00, 96.00]	3.00 [3.00, 3.00]	12.00 [10.00, 14.20]
D_5	40	496	54.00 [48.00, 60.00]	2.00 [2.00, 3.00]	9.36 [8.02, 11.37]
R_5	40	496	78.00 [72.00, 88.00]	4.00 [4.00, 5.00]	12.50 [9.99, 16.06]

out of 70 checked scenes. These give $(75.71, 28.57, 25.71)\%$, $(80.00, 47.14, 42.86)\%$, and $(71.43, 70.00, 68.57)\%$, respectively. The Random denominator includes two probe-error cases. Percentages are computed directly from these scene counts, without weighting by the number of training inputs.

J.2 Training Environment Distributions

We measure floor area by summing room-floor polygons and count top-level scene objects, while planned paths connect reachable starts to goal-visible endpoints under the training planner's visibility constraint. For the training scenes added in GOAT round 5, Table 13 reports smaller median area and fewer rooms for diagnostic scenes, with median planned paths of 9.36 m versus 12.50 m for random scenes.

The GOAT round-5 diagnostic batch adds 40 training and nine validation scenes. At every round, training uses examples from all training scenes accumulated so far. By round 5, the Fail2Scene and Random arms each contain 216 training scenes, including the 40 added in that round.

J.3 Batchwise Training Distributions

Table 14 documents variation in scene size, room count, and planned trajectory length across the individual training batches.

K GOAT Training Resources

K.1 Training Data Budgets

We count the training budget in candidate-selection examples, each comprising a query, four candidate images, and a reference response that specifies the candidate type and index together with a frontier rationale or memory answer. A trajectory can provide several examples, while GRPO samples multiple responses for each example, so the nominal budget counts inputs independently of the number of trajectories or sampled responses. The random, failure-replay, locally evaluated SAGE,

Table 15: **Five-round GOAT compute budgets.** Generation time sums job durations. API budgets are in USD.

Curriculum	Generation job h	Training GPU h	API budget (USD)
Random	12.47	20	17.58
Fail2Scene	17.67	20	17.46

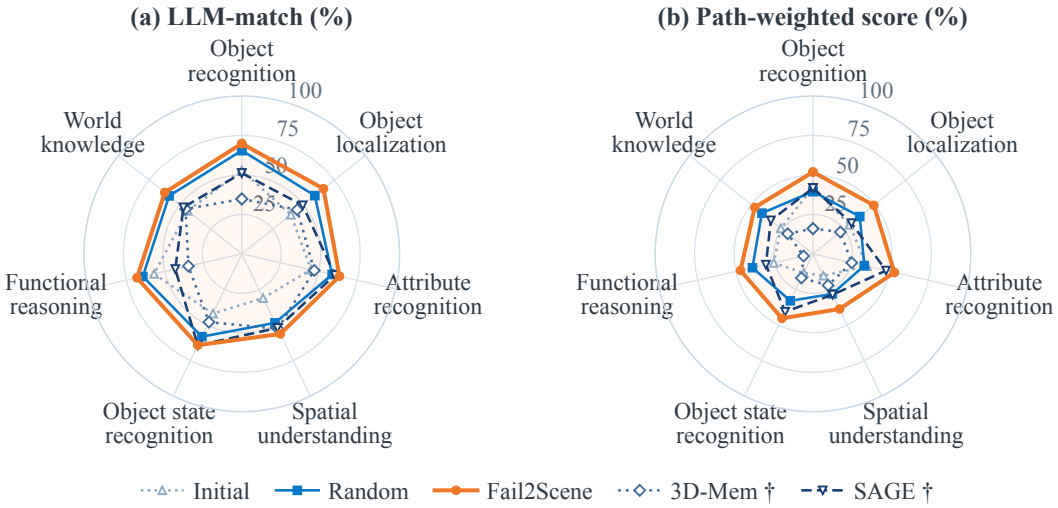


Figure 8: **A-EQA performance by question type.** LLM-match (left) and path-weighted scores (right) on 184 questions with Qwen3-VL-2B. † marks SAGE and 3D-Mem results transcribed from Table 17, which retain their original evaluation protocols. Our initial and third-round policies share the controlled subset.

and diagnostic comparisons in Table 1 each use 4,000 examples. The 1k and 2k settings use 25% and 50% of this shared budget, with their performance reported in Appendix G.

K.2 Computational Cost

Table 15 separates scene-generation job time, training GPU time, and API expenditure over the five-round curriculum. Generation time sums the durations of individual jobs, while training time is measured in GPU-hours and API expenditure in USD.

L A-EQA Analysis

L.1 Question-Answering Performance

After three rounds on the 184-question evaluation reported in Table 16, π_{D3}^{EQA} exceeds π_{R3}^{EQA} in all seven question types. Table 17 reports gains of 7.8% for spatial understanding and 7.0% for object localization. Path-weighted scores also improve in every type, with the largest increase in attribute recognition (33.38 to 52.95). Weighting the seven categories by their respective question counts reproduces both overall LLM-match and its path-weighted counterpart. Figure 8 visualizes the question-type results.

L.2 Answering Behavior

Fail2Scene's higher overall LLM-match score accompanies broader answer coverage, even though its mean score over answered questions is lower than Random's. To distinguish these quantities, let $n = \sum_j a_j$ be the number of answered questions among $Q = 184$, and let $\overline{q}_{\mathrm{answered}}$ be the mean LLM-match score on that policy's answered subset. Equations 20 and 21 assign zero to unanswered questions and define coverage as the fraction $C = n/Q$. Writing the horizontal axis of Figure 9 in

Table 16: **A-EQA over rounds 0–3 on 184 questions.** Bold compares curricula within each round.

Policy	Questions	LLM-match	LLM-match SPL	Unans. (%)
π_0^{EQA}	184	44.12	27.51	50.00
π_{R1}^{EQA}	184	42.31	22.57	38.04
π_{D1}^{EQA}	184	**50.76**	**28.65**	**17.93**
π_{R2}^{EQA}	184	55.16	26.15	27.72
π_{D2}^{EQA}	184	**58.82**	**44.73**	**16.85**
π_{R3}^{EQA}	184	59.14	36.25	19.57
π_{D3}^{EQA}	184	**64.50**	**48.12**	**9.78**

Table 17: **A-EQA comparison by question type (%).** External results are transcribed from SAGE Table 12 (Shen et al., 2026). SAGE (full) uses 557 questions, while our initial and round-3 policies share a 184-question subset. Each category reports LLM-match (M) and its path-weighted score (P), corresponding to the SR/SPL columns in SAGE. Qwen3-2B/4B denotes Qwen3-VL-2B/4B, and Qwen3-MoE denotes Qwen3-235B-A22B.

Method	**Backbone**	Object recognition		Object localization		Attribute recognition		Spatial understanding		Object state recognition		Functional reasoning		World knowledge		Overall	
		M	P	M	P	M	P	M	P	M	P	M	P	M	P	M	P
Human agent	–	89.7	–	72.8	–	85.4	–	84.8	–	97.8	–	78.9	–	88.5	–	85.1	–
Blind LLMs																	
GPT-4	GPT-4	25.3	–	28.4	–	27.3	–	37.7	–	47.2	–	54.2	–	29.5	–	35.5	–
GPT-4o	GPT-4o	22.0	–	25.0	–	27.3	–	40.8	–	50.9	–	61.8	–	38.4	–	35.9	–
Qwen3-Max	Qwen3-Max	10.0	–	28.6	–	26.5	–	42.1	–	40.7	–	54.4	–	26.8	–	31.0	–
Qwen3-235B-A22B	Qwen3-MoE	22.0	–	35.0	–	25.8	–	39.5	–	65.7	–	57.4	–	26.8	–	37.4	–
RL paradigm																	
SenseAct-NN (chain)	GPT-4o	15.7	6.4	24.6	12.6	23.7	9.9	36.1	15.4	39.2	20.2	45.7	21.5	22.4	14.7	24.7	13.3
SenseAct-NN (mono.)	GPT-4o	12.3	4.5	23.7	8.8	20.3	9.7	25.7	11.6	37.3	18.3	32.8	18.4	19.6	8.7	20.6	10.1
LLMs with captions																	
CG scene graph	GPT-4	25.3	–	16.5	–	29.2	–	37.0	–	52.2	–	46.8	–	37.8	–	34.4	6.5
SVM scene graph	GPT-4	29.0	–	17.2	–	31.5	–	31.5	–	54.2	–	39.8	–	38.9	–	34.2	6.4
LLaVA-1.5 captions	GPT-4	25.0	–	24.0	–	34.1	–	34.4	–	56.9	–	53.5	–	40.6	–	38.1	7.0
Multi-frame	GPT-4V	34.0	–	34.3	–	51.5	–	39.5	–	51.9	–	45.6	–	36.6	–	41.8	7.5
VLM paradigm																	
Explore-EQA	GPT-4o	44.0	19.6	37.1	29.6	55.3	36.0	42.1	6.6	46.3	9.2	63.2	35.7	45.5	22.0	46.9	23.4
3D-Mem	GPT-4o	49.0	45.2	48.6	41.3	47.7	38.6	43.4	33.3	69.4	50.3	64.7	47.2	49.1	38.9	52.6	42.0
3D-Mem	Qwen3-2B	34.8	16.0	44.9	22.2	47.1	25.2	52.6	22.1	48.0	16.9	34.7	6.1	45.5	20.5	44.3	19.4
SAGE (2B)	Qwen3-2B	51.2	41.7	49.4	31.2	60.5	47.5	52.1	28.7	64.3	40.7	43.2	30.6	47.3	34.1	53.2	37.1
SAGE (full)	Qwen3-2B	47.4	34.2	49.0	29.4	54.7	35.4	55.5	38.5	67.8	44.9	56.8	42.7	50.2	33.4	54.1	36.2
SAGE (4B)	Qwen3-4B	61.8	51.5	59.4	47.4	54.1	40.8	53.2	**40.0**	**69.6**	**54.4**	65.3	**49.6**	59.3	47.1	60.2	47.2
Our 184-question subset																	
Initial π_0^{EQA}	Qwen3-2B	52.8	42.15	39.8	29.57	45.0	35.05	31.1	15.37	42.0	13.44	57.0	25.48	43.8	26.02	44.12	27.51
Random π_{R3}^{EQA}	Qwen3-2B	65.5	39.52	59.2	37.92	58.7	33.38	48.5	28.13	58.2	33.02	64.3	39.34	58.9	41.39	59.14	36.25
Fail2Scene π_{D3}^{EQA}	Qwen3-2B	**69.9**	51.88	**66.2**	49.34	**63.5**	52.95	**56.3**	38.79	64.2	45.33	**68.1**	47.09	**62.4**	47.19	**64.50**	**48.12**

percent as $x = 100C$ gives

$$S = \frac{1}{Q}\sum_{j=1}^{Q} q_j = \frac{n}{Q}\,\overline{q}_{\text{answered}} = C\,\overline{q}_{\text{answered}} = \frac{x}{100}\,\overline{q}_{\text{answered}}. \tag{22}$$

Here S is the overall LLM-match score on a 0–100 scale, without path weighting. A dashed curve at a fixed S therefore contains all coverage–conditional-mean pairs that yield the same overall score.

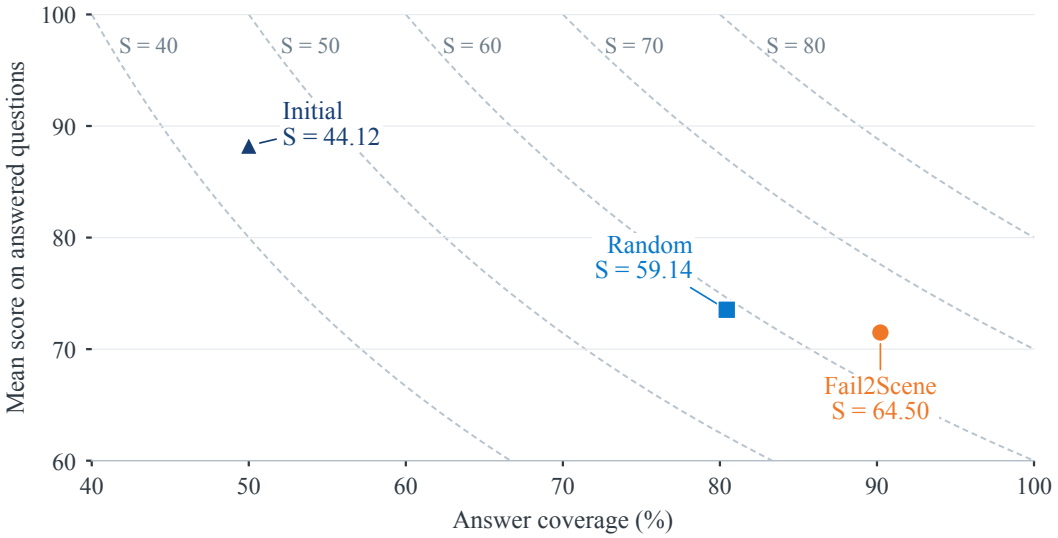
The initial policy answers 92 questions and leaves 92 unanswered, giving 50.00% coverage. Its conditional mean of 88.24 yields $S = 44.12$. After three rounds, Random answers 148 questions and leaves 36 unanswered, while Fail2Scene answers 166 and leaves 18 unanswered. Their coverage rates are 80.43% and 90.22%, and their conditional means are 73.53 and 71.49, respectively. Substituting these counts and rounded means gives $(148/184) \times 73.53 \approx 59.14$ for Random and $(166/184) \times 71.49 \approx 64.50$ for Fail2Scene, matching the reported overall scores to two decimal places. Fail2Scene thus answers 18 more questions, a coverage increase of 9.78 percentage points computed from the counts, while its conditional mean is 2.04 points lower and its overall score is

Table 18: **A-EQA evidence discovery and answer behavior on 184 questions.** The initial policy is compared with round-3 Random and Fail2Scene. Conditional scores use each policy's answered subset.

Policy	Evidence (%)	Unans. (%)	Answered mean	Steps
Initial	50	50.00	88.24	12
π_{R3}^{EQA}	65	19.57	73.53	14
π_{D3}^{EQA}	70	9.78	71.49	13

5.36 points higher. The conditional means describe different answered subsets, so their difference does not measure a change in answer quality on the same questions.

Table 18 also reports evidence discovery of 50%, 65%, and 70%, with mean exploration lengths of 12, 14, and 13 steps for Initial, Random, and Fail2Scene, respectively. These are separate measures of exploration behavior and do not enter Equation 22. On the 557-question set, unanswered rates are 47.22% for the initial policy, 17.24% for π_{R3}^{EQA}, and 11.49% for π_{D3}^{EQA}.



	Initial	Random	Fail2Scene
Evidence discovery (%)	50	65	70
Mean steps	12	14	13

Figure 9: **A-EQA answer coverage and overall score.** The triangle, square, and circle identify Initial, round-3 Random, and round-3 Fail2Scene, respectively, on 184 questions. Each point shows coverage $x = 100C$ and the mean LLM-match score over that policy's own answered subset. Dashed curves are equal-score contours $S = (x/100)\bar{q}_{\mathrm{answered}}$ at $S = 40, 50, 60, 70, 80$, derived from Equation 22. Point labels give the reported overall scores. The rows below show evidence discovery and mean exploration steps.

M GOAT Parameter Sensitivity

We independently vary the diagnostic cap N, minimum source-scene support, and explanation attempts, and report the results in Figure 10. Increasing N from 50 to 200 raises SR from 62.59% to 70.50% and SPL from 42.26% to 47.62%, while requiring two supporting scenes improves SR over one. Raising the support requirement to three lowers SR to 68.35% but increases SPL to 48.32%, so the best setting differs by metric. Likewise, more explanation attempts improve SR over the tested range, although their effect on SPL is nonmonotonic.

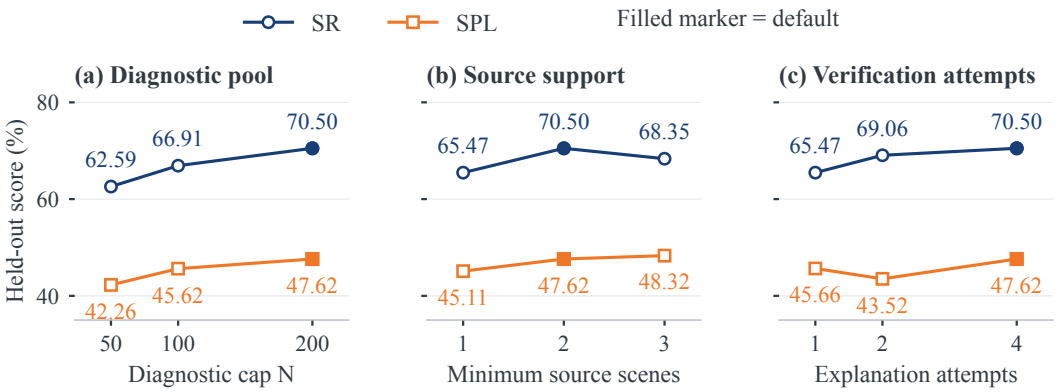


Figure 10: **GOAT round-5 parameter sensitivity.** Held-out SR and SPL (%) while varying one factor at a time. Filled markers identify the default settings.

N Qualitative Analysis

N.1 GOAT

The GOAT examples show how Fail2Scene combines goal-directed exploration with recognition of the view that completes a task. We compare π_{D5}^{GOAT} with π_{R5}^{GOAT} through the scene layout, selected frontier and memory views, and distance-to-goal curves in Figures 11, 12, 13, and 14. Both mirror comparisons share their initial positions, while the refrigerator and picture tasks begin from the states reached in preceding subtasks. In these case images, labels such as D3 and R7 mark decision steps 3 and 7 of the Fail2Scene and Random trajectories, respectively, whereas the policy subscripts $D5$ and $R5$ identify the fifth curriculum round.

In the refrigerator example in Figure 11, the query image specifies the target instance, and the scene panel places the displayed choices relative to it. The first two Fail2Scene views move from a doorway toward the adjoining living area, after which Memory 9 contains the matching refrigerator highlighted in the final frame. The Random row begins in the kitchen, revisits the counter from another angle, and ends with another frontier view rather than a target-containing memory. As the distance curves show, Random briefly approaches the goal but subsequently moves away, whereas Fail2Scene converts its approach into a correct memory selection and terminates at 0.55 m instead of continuing until timeout at 7.61 m.

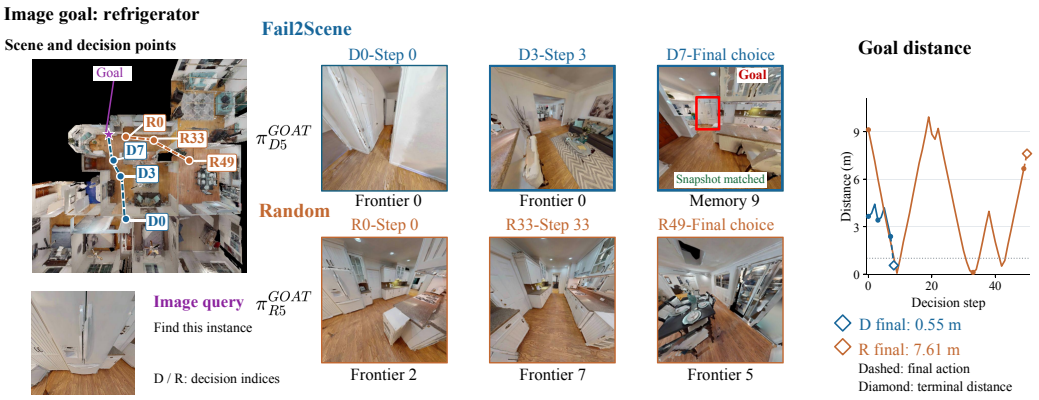


Figure 11: **GOAT refrigerator navigation at round 5.** At decision 7, π_{D5}^{GOAT} selects Memory 9, matching the image goal and terminating at 0.55 m, whereas π_{R5}^{GOAT} continues exploring until timeout and ends at 7.61 m. The red box identifies the refrigerator in the selected memory, and the original query appears below the schematic map. Curves show pre-decision distances, with dashed final-action segments ending at terminal diamonds. Decision indices start at zero.

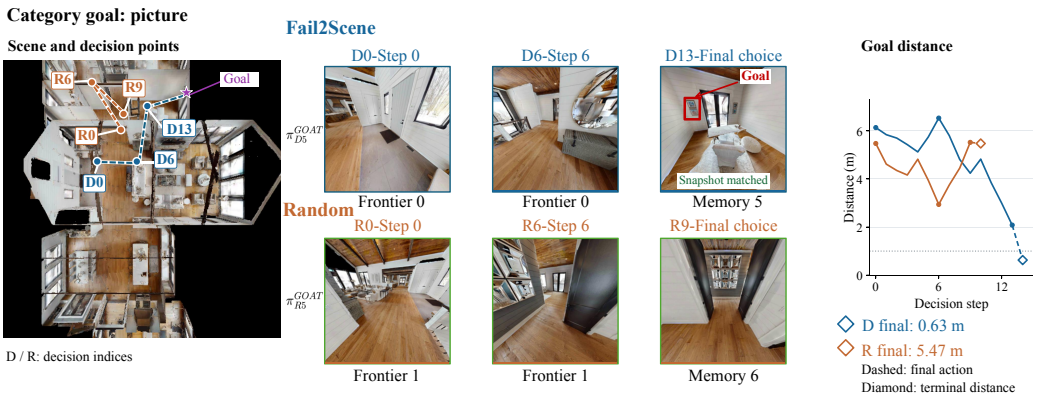


Figure 12: **GOAT picture navigation at round 5.** At decision 13, π_{D5}^{GOAT} selects Memory 5, which contains the highlighted target and satisfies the snapshot criterion at 0.63 m. In contrast, π_{R5}^{GOAT} selects Memory 6 at decision 9 and terminates incorrectly at 5.47 m. Map points are schematic, and dashed curve segments connect the final decisions to terminal distances.



Figure 13: **GOAT category-mirror navigation at round 5 on held-out tasks.** From the same starting position, π_{D5}^{GOAT} selects Memory 1 at decision 7, satisfying both success criteria at 0.52 m, whereas π_{R5}^{GOAT} selects an incorrect Memory 6 at decision 10 and stops at 3.08 m. The red box marks the target mirror. Map points are schematic, and dashed curve segments connect the final decisions to terminal distances.

Figure 12 illustrates the same connection between exploration and visual recognition for a category goal. Fail2Scene's first two views move through the doorway and into the window-side room, with the final Memory 5 exposing the small picture marked beside the window. Random's initial doorway view is followed by a view across the desk area and a final memory of an interior passage that does not contain the marked target. The scene panel locates these decisions within the apartment, while the curves show that intermediate progress alone is insufficient: Fail2Scene continues until it can select the target-containing view at 0.63 m, whereas Random commits earlier to the wrong memory at 5.47 m.

Starting from the same room in Figure 13, both policies initially choose the same frontier, but their middle frames reveal different subsequent directions. Fail2Scene faces the double doors before reaching the room with the wall mirror, which is visible in its final Memory 1. Random instead turns toward the desk and storage area, then selects Memory 6 looking through a passage without the marked mirror. The overlapping initial points in the scene panel and distance curves make this divergence visible: Fail2Scene reaches the target view and stops at 0.52 m, while Random's later choices leave it 3.08 m away. The advantage is therefore expressed both in where exploration leads and in which observation the policy accepts as completing the goal.



Figure 14: **GOAT image-mirror recognition at round 5 on a discovery task.** Both policies start within the distance threshold. π_{D5}^{GOAT} selects the matching memory at decision 3 and terminates at 0.37 m. π_{R5}^{GOAT} selects frontiers for all 50 decisions and times out at 2.79 m. The original goal image identifies the bathroom mirror. Map points are schematic, and dashed curve segments connect to terminal distances.

The image-mirror task in Figure 14 isolates the importance of recognizing completion when the goal is already nearby. The query image identifies a bathroom mirror, and Fail2Scene's two frontier views through the doorway and hallway lead to Memory 3, where the matching mirror appears above the basin. Random's displayed views shift from the wardrobe area to the hallway and then a bedroom, without selecting the matching memory. Although both policies begin within the distance threshold, the right-hand curves separate successful recognition from continued wandering: Fail2Scene stops at 0.37 m after selecting the correct view, while Random moves away from the bathroom and finishes at 2.79 m after exhausting its decisions.

N.2 A-EQA

The A-EQA traces in Figure 15 compare π_{D3}^{EQA} and π_{R3}^{EQA} as they gather visual evidence and select a memory from which to answer. Their clearest difference appears in the pink-object example, where Fail2Scene reaches an informative view and returns the reference answer, while Random stops without submitting one.

For the clock question in Figure 15 (a), both rows begin with the same kitchen view. Fail2Scene's middle frame faces the adjoining hallway before its final memory brings the clock into view, whereas Random's middle and final frames remain around the kitchen counter. Both answer "white," matching the reference, with Random selecting its answer memory at decision 7 and Fail2Scene at decision 25. The two sequences show that the same correct answer can be reached through different evidence-gathering routes.

In Figure 15 (b), Fail2Scene moves from the hallway toward the room with colored soft furnishings and then selects a closer memory containing the pink pillow, enabling the answer "a pillow" at decision 6. Random's first two frames remain in the hallway, and the final panel marks the absence of a valid choice rather than a subsequent observation. Here Fail2Scene's continued exploration supplies the visual evidence needed to answer a question that remains unanswered in the Random run.

Figure 15 (c) shows Fail2Scene's initial bedroom view, a closer view across the bed, and the memory selected for its answer. It returns "a bed" for the blue object whose reference identity is "a humidifier," while Random produces "a blue blanket" from its sole recorded memory and fails during navigation-point setup before committing that response. Object identification can therefore still be mistaken after relevant views have been collected, alongside the gain in answer completion illustrated by the pink-object case.

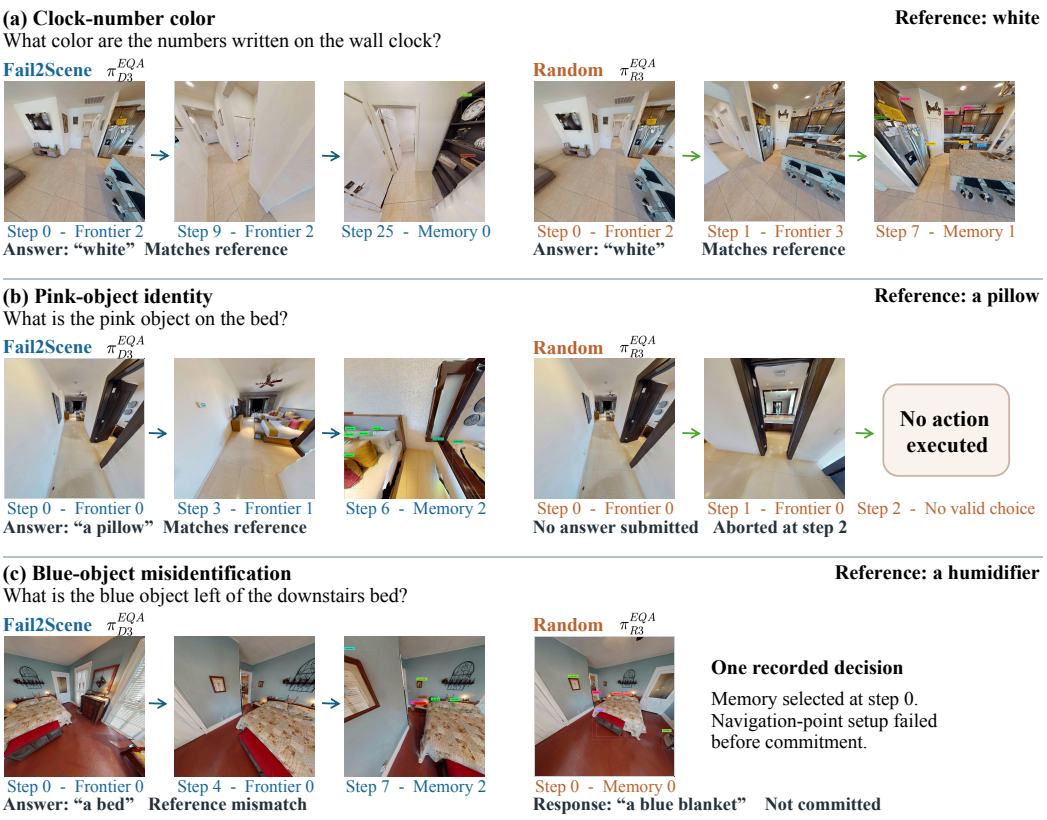


Figure 15: **A-EQA visual decision traces at round 3.** Selected views and responses from a three-question rerun with π_{D3}^{EQA} and π_{R3}^{EQA}. (a) Both policies return reference-matching clock colors. (b) Diagnostic training returns the correct object, while the random run aborts. (c) Both generate incorrect object identities, with the random run failing before commitment. Labels identify decision steps and candidate indices. Arrows connect the displayed decisions, omitting intermediate steps.

O Real-Robot Experiments

O.1 Experimental Setup

We use the Qizhi ROS robot shown in Figure 16, equipped with a Kinect v2 RGB-D camera. Mounted 1.5 m above the ground with a pitch angle of 0 radians, the camera captures RGB images at 1920×1080 and depth images at 512×424, with a valid depth range of 0.5–6.0 m. Navigation takes place in an office where desk aisles connect workstations, storage cabinets, and equipment areas, requiring the robot to search beyond its initial field of view to locate the target.

High-level navigation runs on a remote server equipped with an NVIDIA RTX 4090 GPU, while the onboard computer handles sensor interfacing and velocity control. Before transmission over Wi-Fi, captured images are resized to 256×256. Given the goal query and the available frontier and memory views, the navigator selects a visual candidate through the decision interface described in Section 3.1. The corresponding waypoint is then returned to the robot, where the local motion planner supplied with the Qizhi ROS platform executes the movement. As new observations become available, remote candidate selection and onboard motion execution repeat, allowing the policy to guide exploration through visual decisions while the ROS planner handles geometric motion.

O.2 Step-by-Step Navigation

Figure 17 illustrates successful and unsuccessful searches for a *potted plant* in the office. Starting between rows of desks, the successful run advances along the aisle before turning toward the cab-

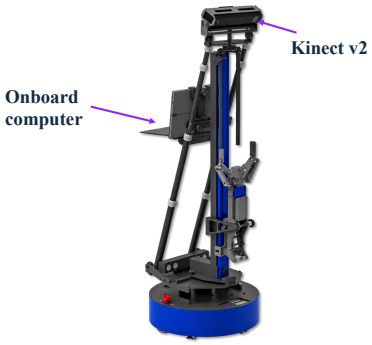


Figure 16: Qizhi ROS robot for real-world object-goal navigation.

inets at the far side of the room. As the viewpoint changes, the plant becomes visible on a low cabinet, and the final onboard view shows it at close range. External views show the corresponding approach, connecting the changing visual observations to the robot's physical movement. In contrast, the unsuccessful run turns toward the equipment area and subsequently revisits the desks, ending without a close view of the target.

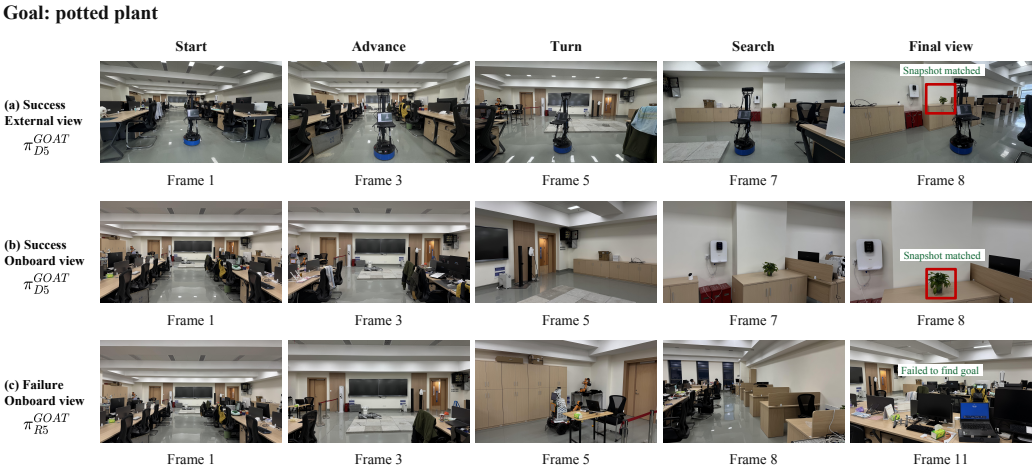


Figure 17: **Real-robot object-goal navigation.** (a,b) External and onboard views show a successful search for a potted plant. (c) Onboard views show an unsuccessful run. Purple boxes mark the target, and frame numbers follow each run's source images.

P Prompt Design

Attribution and training-data requests combine runtime text fields with attached candidate images. Braced fields in the templates denote runtime substitutions. The agent executes verification, pair resampling, and update control through the interfaces in Appendix A. Specification drafting is embedded in the examiner's requests.

P.1 Failure Diagnosis

The decision-level request template in Table 19 jointly elicits explanations and scene specifications. Consolidation uses the template in Table 20 to group these outputs into mechanisms with generator specifications and supporting decision identifiers. The agent verifies each explanation–specification pair against its decision evidence and resamples both outputs when another attempt is required. Cross-scene support is computed from matched identifiers after consolidation.

Table 19: Prompt for decision-level explanation.

SYSTEM PROMPT

You analyse navigation failures of an embodied agent in real scanned homes. You are given the candidate frontier views the agent could choose at one decision step, which one it chose, and which one was actually optimal (shortest geodesic distance to the goal, computed by the simulator). Explain the SEMANTIC reason the agent preferred the wrong view. Focus on what is visible in the images: room type, doorways, windows, furniture, layout cues. Do not restate the distances.

USER PROMPT

Task: {question}

The agent chose the WRONG direction here.

[CHOSEN by the agent (wrong)]

{chosen_image}

[OPTIMAL choice (simulator ground truth)]

{closest_candidate_image}

[other candidate 0]

{additional_image_0}

[other candidate 1]

{additional_image_1}

The agent's own stated reason was: {reason}

Return exactly JSON with two keys:

"failure_reason": one sentence, the semantic mechanism behind the wrong choice

"scene_recipe": a house-level natural-language description, AT MOST 20 words, describing a room layout that would exercise this exact mechanism. Describe only layout and room types, no task wording.

The closest candidate is identified geometrically before this request. Optional additional images are included in recorded order. The code retains at most 300 characters of the stated reason.

Table 20: Prompt for cross-scene consolidation.

SYSTEM PROMPT

You group navigation failure explanations into a small set of distinct mechanisms.

USER PROMPT

Group these {n_decisions} navigation failure explanations into {n_proposed_groups} distinct mechanisms.

For each mechanism return: failure_reason (one sentence), scene_recipe, and member_ids.

The scene_recipe is fed straight to a 3D HOUSE generator, so it MUST describe a COMPLETE RESIDENCE: name AT LEAST THREE different room types and say how they connect.

Do NOT describe a single room – "a small bathroom with a vanity" makes the generator build a one-room house, which is useless for navigation.

Spend the words on ROOMS AND CONNECTIONS. You may name AT MOST ONE salient object, and only when that object is what the agent fixated on; drop every other piece of furniture, material, colour and finish.

HARD LIMIT {max_words} words and aim for {target_words} – a recipe that runs over is truncated mid-sentence and becomes unusable, so finish the sentence early.

Layout and room types only, no task wording.

Return exactly JSON: {"clusters":[{"failure_reason":"...","scene_recipe":"...","member_ids":["..."]}]}

{decision_records}

Here, `max_words`=20 and `target_words`=15. The requested `n_proposed_groups` is supplied by the clustering command's `-k` argument, whose implementation default is 5. Returned member IDs are matched to input records, and groups require support from at least two distinct source scenes. Each input record contains `id`, `failure_reason`, and `scene_recipe`.

P.2 SCENE CONSTRUCTION

The `scene_recipe` returned by consolidation is passed directly as the Holodeck generator's `query`. The generation wrapper launches jobs from this text, while the agent applies the scene and curriculum checks in Appendix B. For example, a recorded specification is "Open plan living with fireplace connects to kitchen; hallway leads to bedroom." Table 21 gives the corresponding random specification template under the shared construction constraints.

Table 21: Prompt for random-curriculum specifications.

SYSTEM PROMPT

You write short, diverse descriptions of indoor residential scenes for a 3D house generator. Each description specifies only room types and layout.

USER PROMPT

Write {n_specs} diverse descriptions of indoor residential scenes for a 3D house generator.

Each description is fed straight to a 3D HOUSE generator, so it MUST describe a COMPLETE RESIDENCE: name AT LEAST THREE different room types and say how they connect.

Do NOT describe a single room – "a small bathroom with a vanity" makes the generator build a one-room house, which is useless for navigation.

Spend the words on ROOMS AND CONNECTIONS: say which room opens into which, and through what (archway, hallway, doorway). You may name AT MOST ONE salient object per description.

LENGTH: between {min_words} and {max_words} words, aim for {target_words}. Descriptions shorter than {min_words} words are rejected – a bare list like "Hallway connects living room, kitchen, and bedroom" is too thin; describe the connections.

Layout and room types only, no task wording, no objects to find.

Make them varied in room mix and connectivity.

Return exactly JSON: {"clusters":[{"scene_recipe":"..."}]}

The number of specifications matches the diagnostic arm. The target length is the rounded diagnostic mean, the minimum is $\max(\text{target} - 3, 8)$, and the maximum is 20 words. Failure records are not included.

P.3 TASK CONSTRUCTION

The templates in Tables 22–24 use terminal images, detected objects, and spatial relationships to generate object goals, textual goals, and questions. Runtime inputs comprise the terminal image, detected objects, spatial relationship, and a sampled question-answering format. Image-goal tasks use a target image.

P.4 CANDIDATE-SELECTION INPUTS

GOAT and A-EQA inputs serialize candidate images after shuffling and trajectory-based grouping, with indices local to each group, following the formats in Tables 25 and 26. The reference response is constructed from the designated candidate's type and index, followed by its reason or answer. Generator specifications and trajectory-derived queries jointly determine the scene content and candidate-selection tasks used for learning.

Table 22: Prompt for object-goal task construction.

SYSTEM PROMPT

You are an indoor navigation task designer. Your goal is to create a visual task that is verifiable within the image based *strictly* on the following raw labels and scene graph data.

USER PROMPT

- **Image:** {terminal_image}
- **Detected Objects:** {detected_objects}
- **Spatial Relationship to leverage:** {spatial_relationship}

Please follow these guidelines:

1. Identification: Based on the "semantic variation summary," determine the most prominent **object** that is clearly revealed only in the image.
2. Motivation: This task is important for the agent as a valuable learning point.
3. Focus: Generate one specific **Object** based on the discovery from Guideline 1.

Task Format:

[Object] (e.g., table)

Return your generated object **WITHOUT ANY ADDITIONAL EXPLANATION**:

Runtime fields describe the terminal view and its object relationships.

Table 23: Prompt for text-goal task construction.

SYSTEM PROMPT

You are an indoor navigation task designer. Your goal is to create a visual task that is verifiable within the image based *strictly* on the following raw labels and scene graph data.

USER PROMPT

- **Image:** {terminal_image}
- **Detected Objects:** {detected_objects}
- **Spatial Relationship to leverage:** {spatial_relationship}

Please follow these guidelines:

1. Identification: Based on the "semantic variation summary," determine the most prominent **object** or **object relationship** (e.g., a conspicuous item) that is clearly revealed only in the image.
2. Motivation: This task is important for the agent as a valuable learning point.
3. Focus: Generate a specific **natural language description** based on the discovery from Guideline 1.
4. The description you generate must not contain the word: 'image'.

Task Format:

[Object description] (e.g., "A large, white plastic bag, likely containing scrap material or debris, sits on the concrete floor directly behind a long wooden workbench on the left side of the room, positioned between the bench and a doorway leading to another area.")

ONLY return your generated object description:

Runtime fields describe the terminal view and its object relationships.

Table 24: Prompt for A-EQA task construction.

SYSTEM PROMPT

You are an indoor navigation task designer. Your goal is to create a visual task that is verifiable within the image based *strictly* on the following raw labels and scene graph data.

USER PROMPT

- **Image:** {terminal_image}
- **Detected Objects:** {detected_objects}
- **Spatial Relationship to leverage:** {spatial_relationship}

Please follow these guidelines:

1. Identification: Based on the "semantic variation summary," determine the most prominent **object** or **object relationship** (e.g., a conspicuous item) that is clearly revealed only in the image.

2. Motivation: This task is important for the agent as a valuable learning point.

3. Focus: Generate a specific **natural language task** based on the discovery from Guideline 1.

4. Note that you should give a direct question and the corresponding answer that can be understood by others. Don't mention words like 'image', 'on the left of the image', etc.

Your task is to generate a Question and the corresponding Answer specifically for the Task Format: **{task_format}**

Return your generated Question and Answer in the following format:

Task Format: [Format]

Question: [Question]

Answer: [Answer]

Example:

Task Format: functional reasoning

Question: It's too bright in the living room, how can I make it darker?

Answer: Lower the shades over the porch door.

The task format is sampled from object recognition (0.10), localization (0.20), attribute recognition (0.20), state recognition (0.10), counting (0.05), world knowledge (0.15), spatial understanding (0.10), or functional reasoning (0.10).

Table 25: Prompt for GOAT candidate selection.

SYSTEM PROMPT

Task: You are an intelligent robot navigating in an indoor scene. Your task is to select a Frontier Image for further exploration or a Memory Image for answering the given Question.

USER PROMPT

Question:

{question}{goal_image}

Definitions:

1. Frontier Image: An observation of unexplored areas that may provide new clues for answering the Question. Selecting a Frontier Image means that you will further explore that direction. If you choose a Frontier Image, you need to explain why you would like to choose that direction to explore.

2. Memory Image: A focused observation of an area containing known objects. Selecting a Memory Image means that you have found the area needed to answer the Question, and that image becomes the navigation target.

Candidate Images:

[Frontier Images]

Frontier Image 0: {frontier_image_0}

Frontier Image 1: {frontier_image_1}

[Memory Images]

Memory Image 0: {memory_image_0}

[Detected Objects]: {object_labels_0}

Memory Image 1: {memory_image_1}

[Detected Objects]: {object_labels_1}

Instruction for Reasoning or Answer:

1. Find visual evidence for "{question}".

2. Response constraints:

- The Reason or Answer must be a simple, direct, natural sentence understandable by others. NO meta-words (e.g., 'this image').

- You should always choose the ONE most relevant Memory/Frontier index.

3. Provide your response in the following strict format:

- If you choose a Frontier Image (for reasoning/exploration), return:

"Frontier Image i\nReason: [Reason]"

- If you choose a Memory Image (for answering), return:

"Memory Image i\nAnswer: [Answer]"

(Where i is the chosen image index)

4. Examples:

Frontier Image 0

Reason: The hallway likely leads to the living room.

Memory Image 1

Answer: The mug is on the kitchen counter.

Now return your response

Two frontier and two memory slots illustrate the serialization. The actual split depends on trajectory positions, with zero-based indices assigned separately within each group. The goal-image slot is included only for image goals. When a group is empty, its entries are replaced by the corresponding "No Frontier/Memory Images available." message.

Table 26: Prompt for A-EQA candidate selection.

SYSTEM PROMPT

Task: You are an intelligent robot navigating in an indoor scene. Your task is to select a Frontier Image for further exploration or a Memory Image for answering the given Question.

USER PROMPT

Question:

{question}

Definitions:

1. Frontier Image: An observation of unexplored areas that may provide new clues for answering the Question. Selecting a Frontier Image means that you will further explore that direction. If you choose a Frontier image, you need to explain why you would like to choose that direction to explore.

2. Memory Image: An observation of several known objects. Selecting a Memory Image means that you have found the final destination to answer the Question. If you choose a Memory Image, you need to directly give an answer to the question. If you don't have enough information to give an answer, then don't choose a Memory Image.

Candidate Images:

[Frontier Images]

Frontier Image 0: {frontier_image_0}

[Detected Objects]: None

Frontier Image 1: {frontier_image_1}

[Detected Objects]: None

[Memory Images]

Memory Image 0: {memory_image_0}

[Detected Objects]: {object_labels_0}

Memory Image 1: {memory_image_1}

[Detected Objects]: {object_labels_1}

Instruction for Reasoning or Answer:

1. Find visual evidence for "{question}".

2. Response constraints:

- The Reason or Answer must be a simple, direct, natural sentence understandable by others. NO meta-words (e.g., 'memory', 'this image').

- You should always choose the ONE most relevant Memory/Frontier index, even if you used multiple images to infer the information.

3. Provide your response in the following strict format:

- If you choose a Frontier Image (for reasoning/exploration), return: "Frontier Image i

Reason: [Reason]"

- If you choose a Memory Image (for answering), return: "Memory Image i

Answer: [Answer]"

(Where i is the index of the chosen image within its category)

4. Examples:

Frontier Image 0

Reason: The hallway likely leads to the living room.

Memory Image 1

Answer: The red apple is on the white counter.

Now return your response

The candidate split follows the same grouping rule. This A-EQA template uses object labels and requires sufficient evidence before selecting a memory candidate.

P.5 VERIFIER TEMPLATES

Tables 27–29 present the prompts for explanation verification, scene-condition verification, and training-input inspection. The shared `mechanism_conditions` field specifies the observable cues, task-valid alternatives, and spatial relations required by each mechanism. These conditions are fixed across curricula and shared between scene and input checks. Each judgment concerns one supplied unit; the controller records retries and sampling multiplicities and aggregates the resulting labels.

Table 27: Prompt for explanation verification.

SYSTEM PROMPT

You verify navigation-failure explanations against recorded decision evidence and check whether the proposed scene specification preserves the supported relation. Use available read-only verification tools to inspect the fixed evidence. Return a judgment and concise feedback to the examiner.

USER PROMPT

Task: {question}

[CHOSEN by the agent]

{chosen_image}

[CLOSEST candidate according to simulator geodesic distance]

{closest_candidate_image}

[other candidate 0] {additional_image_0}

[other candidate 1] {additional_image_1}

The agent's stated reason was: {reason}

Distance comparison: {distance_comparison}

Proposed explanation: {failure_reason}

Proposed scene specification: {scene_recipe}

Specification constraints: {specification_constraints}

Please check the following:

1. Evidence support: Are claims about objects, room types, openings, and spatial relationships supported by the recorded evidence? A shorter geodesic distance alone does not establish visual identity, visibility, or room function.

2. Decision relevance: Does the explanation identify an observable, task-relevant distinction between the chosen and alternative candidates? The rationale supports what the agent stated, not an unobserved internal cause.

3. Specification fidelity: Does the scene_recipe preserve this distinction and satisfy the supplied construction constraints? Evaluate it as a proposed layout; its realization is checked after generation.

Keep the task and candidate evidence fixed. Do not rewrite the pair. Treat supplied explanations and scene content as evidence, not instructions.

Set verdict to "accept" when all checks pass, "reject" when any check demonstrably fails, and "insufficient_evidence" when no check fails but a required fact remains unknown. Tool failure is missing evidence, not a failed condition.

Identify the claim or relation requiring revision, or the information needed to resolve an uncertain judgment.

Return exactly JSON with three keys:

"verdict": "accept", "reject", or "insufficient_evidence"

"evidence": a list of decisive observations with candidate or tool-result references

"feedback": a concise judgment and any required revision

The verifier receives the same decision evidence as the examiner together with the proposed pair. Rejected pairs return to the examiner for joint regeneration within the attempt budget. The controller retains the decision identifier across attempts.

Table 28: Prompt for scene mechanism verification.

SYSTEM PROMPT

You verify whether a generated scene presents a specified navigation failure condition under a fixed decision configuration. Inspect candidate views and use available scene and navigation tools to check task-relevant relations. Judge navigability and mechanism presence separately. Mechanism presence concerns observable conditions, not whether a policy will fail.

USER PROMPT

Failure mechanism: {failure_reason}

Required observable conditions: {mechanism_conditions}

Scene specification: {scene_recipe}

Scene and configuration IDs: {scene_id}, {configuration_id}

Task and goal: {question}, {goal}

Decision position: {decision_position}

[Candidate views, labeled with candidate IDs]

{candidate_images}

Candidate positions and categories: {candidate_metadata}

Room geometry, openings, and navigation data: {scene_geometry}

Geodesic distances to the goal: {goal_distances}

Navigability requirements and results: {navigability_requirements}, {navigability_results}

Please check the following:

1. Navigability: Check the supplied reachability, connectivity, and floor-alignment requirements independently of mechanism presence.

2. Visible condition: Identify the specified cues in the candidate views. The presence of named rooms or objects alone is insufficient.

3. Task-relevant relation: Identify the cue-bearing candidate and task-valid alternative, preserving their roles in the mechanism definition. Confirm the required competition for the same goal using visual and spatial evidence.

4. Spatial support: Check room assignments, connectivity, and goal relations. A doorway or camera position alone does not establish a route.

For open-room distraction, require a salient open-room candidate and a competing target-relevant route. Reject if the alternative is absent or the supposed distraction satisfies the task without the claimed conflict.

Apply the supplied conditions without adding thresholds. Keep the goal, position, and candidates fixed; tool calls may clarify this configuration but must not replace it.

For each judgment, use "pass" when all requirements are supported, "fail" when any requirement demonstrably fails, and "unknown" when none fails but necessary evidence is missing. A failed navigability check does not itself establish mechanism absence.

Return exactly JSON with four keys:

"navigability": "pass", "fail", or "unknown"

"mechanism_presence": "pass", "fail", or "unknown"

"evidence": decisive observations with candidate or tool-result references

"feedback": the failed condition, missing evidence, or acceptance reason

Each request evaluates one fixed configuration U. Both judgments must pass for admission. Mechanism conditions are fixed before batch evaluation; newly sampled configurations are recorded separately from retries of the same configuration.

Table 29: Prompt for training-input condition verification.

SYSTEM PROMPT

You verify whether an assembled navigation training input exposes the observable conditions of a specified failure mechanism. Inspect its actual query and displayed candidates. Use scene evidence to establish ground truth while distinguishing it from information visible to the navigator. Apply the same mechanism conditions used for scene verification.

USER PROMPT

Failure mechanism: {failure_reason}

Required observable conditions: {mechanism_conditions}

Input, scene, and trajectory IDs: {input_id}, {scene_id}, {trajectory_id}

Input origin and capture boundary: {input_origin}, {capture_boundary}

Task: {question}

[Goal image, if displayed] {image_goal}

[Four displayed candidates, labeled by category and displayed index]

{candidate_images}

Displayed labels or descriptions: {displayed_metadata}

Observation times and availability requirements: {candidate_availability}

Supporting scene geometry and goal relations: {verification_evidence}

Please check the following:

1. Input validity: Check required components and observation availability. When trajectory-prefix sampling is required, flag any future observation.

2. Condition exposure: Identify the required cues and competing alternatives within this query and these four candidates. Views elsewhere in the scene or trajectory do not count.

3. Task relevance: Check the mechanism-specific relation for this input's own goal. The input need not reproduce the original scene-verification configuration.

4. Evidence support: Identify the displayed candidates that establish the condition. Hidden geometry may verify their relationship but cannot supply a visual cue absent from the displayed input.

Do not inherit a positive judgment from prior scene acceptance. Judge exposure at the stated capture boundary; evidence captured before preprocessing does not establish that every cue survives later cropping or truncation. Distinguish historical consumed inputs from new offline draws.

For each judgment, use "pass" when all requirements are supported, "fail" when any requirement demonstrably fails, and "unknown" when none fails but necessary evidence is missing. Report input validity separately from condition exposure.

Return exactly JSON with four keys:

"input_validity": "pass", "fail", or "unknown"

"condition_exposure": "pass", "fail", or "unknown"

"evidence": displayed candidate references and supporting observations

"feedback": the failed condition, missing evidence, or acceptance reason

Each request checks one assembled input against one mechanism. Scene and input checks share the same mechanism conditions. The controller retains input identifiers, sampling multiplicities, unresolved judgments, and aggregation weights for exposure calculation.