EMBODIEDSMITH: SCALING EMBODIED DATA THROUGH RECURSIVE SELF-IMPROVEMENT FLYWHEEL IN SIMULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling robotic foundation models requires diverse training data and reliable evaluation environments. Simulation offers a scalable solution, yet existing generation pipelines remain constrained by predefined assets and skills, a disconnect between scene generation and task generation, and limited support for complex embodiments and physics. We introduce EmbodiedSmith, a framework for scalable embodied data generation through recursive self-improvement (RSI). EmbodiedSmith unifies asset, scene, and task generation in a pipeline that supports autonomous creation and language-driven customization. Its core is an agentic refinement loop: scene generation anticipates downstream task requirements, while task generation guides targeted scene edits, allowing scenes and tasks to iteratively improve one another. This joint refinement improves task generation success, including for long-horizon tasks. The framework further supports mobile manipulators, humanoids, and dexterous hands, as well as interactions involving deformable objects and fluids, broadening the range of behaviors and physical phenomena represented in generated data. Together, these capabilities provide a flexible simulation engine for both robot pretraining and evaluation. Extensive experiments validate the quality, diversity, and generation efficiency of the resulting data, while downstream policy experiments demonstrate that increased data diversity improves generalization.

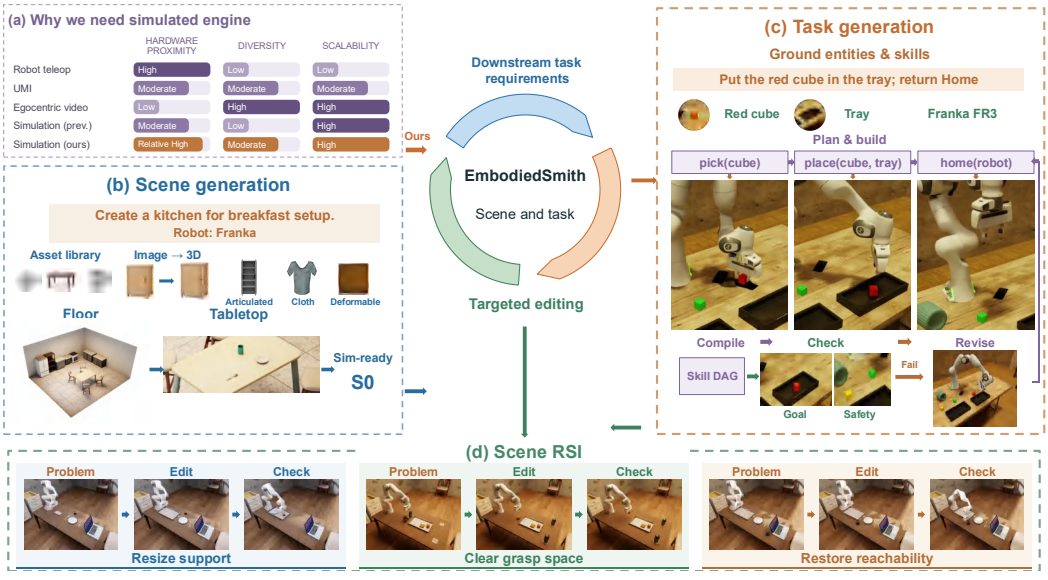


Figure 1: **Overview of our EmbodiedSmith.** Conditioned on a task and robot embodiment, EmbodiedSmith couples scene generation with task generation and execution, using feedback to iteratively refine scenes and replan.

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.

1 INTRODUCTION

Large-scale pretrained models have been validated as an effective path towards robot generalists that can perform everyday tasks across diverse environments (Kim et al., 2024). A growing body of work has focused on collecting large-scale robot datasets and training high-capacity robotic foundation models. These models have demonstrated meaningful generalization to novel objects, environments, and tasks, showing great promise towards developing broadly capable policies.

Simulation plays an important role in advancing large-scale pretrained models. First, as shown in Figure 1 (a), real-world robot data closely matches the target hardware but offers limited diversity and scalability, whereas egocentric and UMI data exhibit the opposite trade-off (Ye et al., 2026). Simulation **data** combines scalability and diversity with close alignment to the target hardware, making it an important data source for robot pretraining (Tian et al., 2026; Nasiriany et al., 2026; Authors, 2024). Second, simulation environments enable low-cost, fast, and reproducible large-scale **evaluation** (Nasiriany et al., 2026) and provide an ideal evolving environment for robotic agents. Thus, as recursive self-improvement (RSI) techniques advance (Weng, 2026; Xiao et al., 2026), simulation environments are becoming an essential component of robotics infrastructure (Liu et al., 2026; Authors, 2024).

However, existing simulation generation methods still face several limitations: (1) limited diversity and customizability of scenes, assets, and skills, where the scenes and assets are already available in the dataset and skills are typically predefined or rule-based (Nasiriany et al., 2026); (2) limited physical realism (Wang et al., 2026a) due to inadequate support for deformable objects and fluids, both of which are essential components of the physical world; (3) low task execution success rates, as scene generation is decoupled from robot task generation, leaving the feasibility of planned actions unverified. This issue becomes more pronounced as tasks grow more complex or longer-horizon; (4) limited support for complex embodiments, including mobile manipulators, dexterous hands, and humanoids. Task generation for these embodiments is challenging (Li et al., 2026), while missing these embodiments limits robots' ability to acquire human-like dexterous behaviors.

We introduce EmbodiedSmith, a framework for scalable embodied data generation through RSI. It is structured around four features:

- **High Data Diversity:** We design strong 3D generation methods to generate any novel object, layout, and scene. Besides, we construct effective agents that autonomously create and compose new skills to accomplish diverse task generation. These generation capabilities, spanning multiple levels of abstraction, are integrated into a unified pipeline that supports autonomous generation and language-driven customization.
- **Rich Physics Simulation:** We support assets and tasks involving more complex physics, including deformable objects and fluids, addressing gaps in existing data diversity and narrowing the gap between simulation and the real physical world.
- **RSI Data Generation:** Scene generation anticipates downstream task requirements, while task generation can revise the scene through targeted editing. This agentic loop enables scene and task generation to recursively refine one another. This process substantially improves task generation success rates, allowing our simulation engine to maintain high success rates even for long-horizon tasks.
- **Human-level Dexterity**: We extend robot task generation to mobile manipulators, humanoids, and dexterous hands, enabling EmbodiedSmith to generate data involving human-like dexterity.

By integrating these components, our EmbodiedSmith provides an effective, flexible, and comprehensive simulation data engine for robotic foundation models. It serves as an efficient source of pre-training data and enables systematic analysis of how task, environment, and embodiment diversity affect pre-trained models. It can also function as an effective evaluator, providing metrics and feedback that facilitate the iterative advancement of foundation models. Through extensive experiments, we validate the quality, diversity, and generation efficiency of the produced environments and demonstrations. Furthermore, experiments integrating the data with downstream policies demonstrate that greater data diversity improves model generalization.

2 Related Work

3D Scene Generation. 3D scene generation has been studied through learned generative models, procedural and foundation-model-guided pipelines, and agentic systems. Data-driven methods learn object distributions and spatial relations from 3D scene datasets: DiffuScene and Mixed Diffusion model indoor layouts using diffusion processes (Tang et al., 2024; Hu et al., 2026), while PhyScene introduces collision, accessibility, and interaction constraints to improve physical plausibility (Yang et al., 2024a). Procedural methods instead assemble environments using predefined grammars and geometric rules, as demonstrated by ProcTHOR and Infinigen Indoors (Deitke et al., 2022; Raistrick et al., 2024). To enable open-vocabulary control, foundation-model-guided methods translate user instructions into structured scene layouts through direct coordinate prediction (Feng et al., 2023), language-guided asset retrieval (Yang et al., 2024b), vision-language-based layout optimization (Sun et al., 2025a), hierarchical scene composition (Pun et al., 2025), or explicit constraint planning (Sun et al., 2025b). Agentic methods introduce iterative reasoning and refinement, including self-reflective tool use (Yang et al., 2025) and coordinated semantic and physical correction (Gao et al., 2025). SAGE, SceneSmith, and EmbodiedGen V2 extend this paradigm toward embodied simulation by coupling multi-agent generation with task-aware composition, physical validation, interactive assets, and simulator-ready deployment (Xia et al., 2026; Pfaff et al., 2026; Wang et al., 2025).

Simulated Robot Demonstration Generation. Humanoid simulation benchmarks increasingly cover locomotion and object interaction within the same embodiment. HumanoidBench (Sferrazza et al., 2024) provides a standardized suite of whole-body locomotion and dexterous manipulation tasks, while SIMPLE (Wei et al., 2026) combines contact-rich simulation and photorealistic rendering with motion-planning and teleoperation pipelines for demonstration collection. More directly, HumanoidGen (Jing et al., 2025) generates language-conditioned bimanual tasks and demonstrations through spatial constraints, and HumanoidMimicGen (Lin et al., 2026) adapts segmented source demonstrations to new configurations using whole-body planning. These works establish important benchmark and synthesis capabilities, but their reported settings center on curated task suites, annotated manipulation primitives, or source-demonstration-dependent skill sequences. Mapping newly instantiated scene–task combinations to executable, feasibility-checked whole-body demonstrations through a reusable skill interface remains less systematically examined.

3 Method

Given a task specification and robot embodiment, EmbodiedSmith generates validated robot demonstrations through the RSI pipeline in Figure 1. Section 3.1 constructs task-conditioned scenes through asset generation, scene generation, and scene edit. Section 3.2 grounds tasks into executable skill plans and refines them through generation and execution feedback. Section 3.3 extends execution to humanoid and dexterous embodiments through embodiment-specific adapters.

3.1 Environment Construction

EmbodiedSmith constructs task-conditioned simulation scenes through Asset Generation, Scene Generation, and feedback-driven Scene Edit.

Asset Generation. Existing asset libraries vary in semantics, coordinate conventions, scale, and geometry, while generated assets may have unreliable collision or physical properties. We therefore maintain an asset pool combining validated open-source and generated assets. Existing assets are normalized by a VLM from multi-view renderings and metadata and checked for semantics, geometry, scale, and scene compatibility. If retrieval fails, a scene-consistent reference image is refined and reconstructed by SAM3D (Team et al., 2025), then scaled, repaired, and equipped with collision geometry through adaptive convex decomposition. A VLM assigns physical parameters, while resting, contact, and perturbation tests validate the asset. Failed assets are repaired or regenerated, whereas validated assets are retained for reuse across scenes.

Physical Asset Generation. Physical asset generation extends the common asset pipeline through construction, physics validation, and task validation. In construction, articulated assets use an LLM to iteratively write and revise construction programs from validation feedback (Zhou et al., 2026), deformable assets use part-aware geometry with regional material properties, and fluid assets

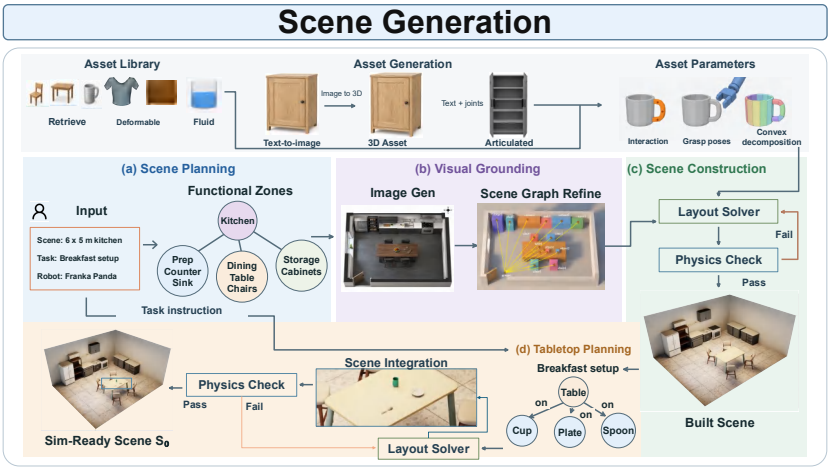


Figure 2: **Scene overview.** Our system transforms scene specifications, task instructions, and robot configurations into simulation-ready 3D scenes through functional zone planning, visual grounding, asset retrieval and generation, and physics-validated scene and tabletop layout optimization.

couple geometry with solver-specific parameters. Physics validation tests intended behavior and retains only improving revisions, while task validation checks collision-free robot motion using CuRobo (Sundaralingam et al., 2023). Failed assets are repaired on cloned instances, and only validated assets are released to the scene generator.

Scene Generation. Given a natural language description $\mathcal{T}$, task context, and robot embodiment configuration $\mathcal{E}$, our scene generator synthesizes an initial scene hypothesis $\mathcal{S}_0 = (\mathcal{G}, \mathcal{O}, \mathcal{X}, \mathcal{Q})$, where $\mathcal{G}$ denotes room boundaries and architectural openings, $\mathcal{O}$ represents object instances, $\mathcal{X}$ specifies 6-DoF poses, and $\mathcal{Q}$ designates support regions and robot interaction constraints. As illustrated in Figure 2, the process decouples open-ended semantic structuring from deterministic geometric solving. A VLM first parses the input into a typed relational graph encoding functional zones, static fixtures, orientations, support surfaces, and robot approach directions. Dedicated asset modules then instantiate its nodes with retrieved or generated assets.

Scene construction subsequently solves the layout hierarchically: ground fixtures are organized into local compositions and joint-optimized via Beam-DFS search, followed by wall fixture placement and parent-relative task-object positioning. The optimization enforces boundary limits, collision avoidance, clearances, connectivity, and workspace bounds, with reachability validated by CuRobo. The resulting $\mathcal{S}_0$ is therefore an initial geometry- and reachability-validated scene rather than a guaranteed executable environment. Downstream robot planning and execution further evaluate task feasibility and return feedback to the RSI loop (Figure 1). Environment-induced failures then trigger targeted Scene Edit, followed by revalidation, replanning, and re-execution.

Scene Edit. Scene Edit adapts validated environments to new tasks within the RSI loop without repeated scene synthesis. Given the scene, task, robot embodiment, and generation or execution feedback, the scene agent applies schema-constrained edits while preserving unrelated scene elements. Supported operations include task-object addition, replacement, rescaling, and reachability-aware repositioning within or across support surfaces. Asset replacement updates geometry, physical properties, grasp annotations, and simulator bindings according to the placement policy. Each edit can be previewed before application and produces a traceable revision identifying the physical, reachability, and export checks requiring revalidation before replanning.

3.2 Robot Trajectory Generation

With the task-conditioned scene S_0 from Section 3.1, the task generation grounds the task to scene entities and atomic skills, forming an executable Skill DAG that is validated through compilation, checking, and simulation. The robot task generation system consists of three core components. The **Task Plan Agent** interprets the task, makes the main generation decisions, and proposes repairs. The **Harness** manages the workflow, initiates independent reviews, and serves as the driver and central

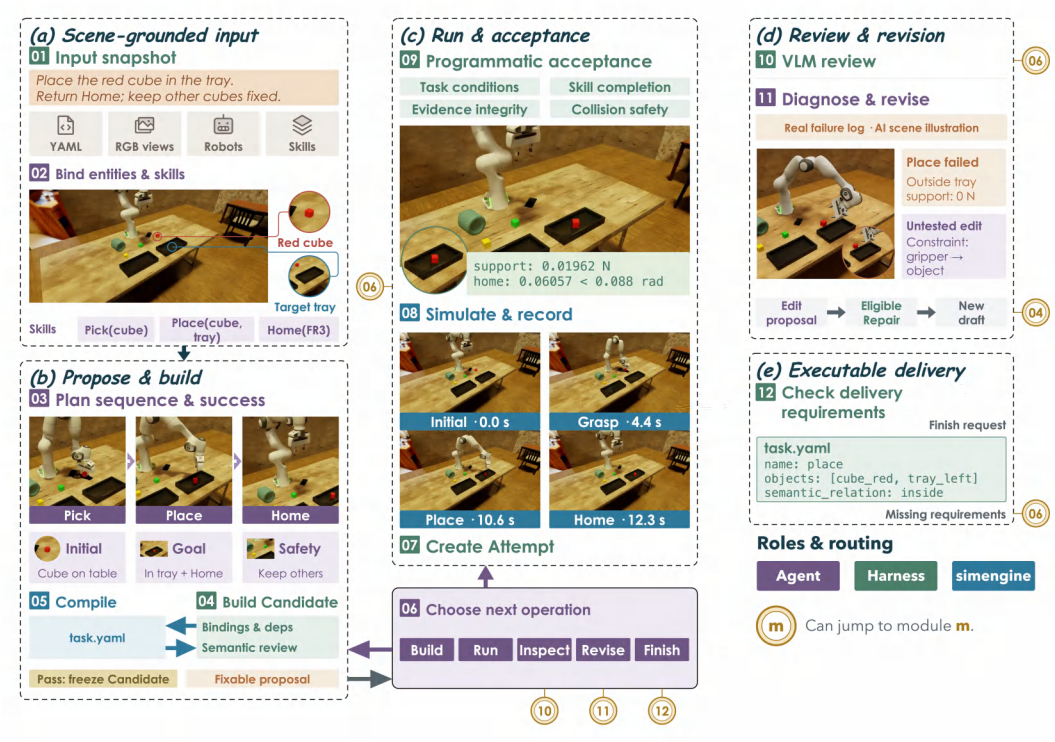


Figure 3: **Closed-loop robot task generation and validation.** (a) Instructions are grounded in the scene, robot configuration, and available skills. (b) The Agent proposes a skill sequence and success conditions, which are used to build and compile a candidate task configuration. (c) The candidate is executed in simulation and evaluated through programmatic acceptance checks. (d) Independent VLM review complements these checks, while failures trigger diagnosis, revision, and revalidation. (e) The executable configuration is delivered upon final acceptance.

scheduler of the overall framework. The **`simengine`** carries out simulation-based validation and records the execution.

Task Grounding and Planning. We first input the task prompt, the scene configuration files produced by the scene generation agent, the embodiment configuration files, a list of available atomic Skills, and an image of the scene from a global viewpoint to the Harness. As shown in Figure 3(a), the Harness first checks these inputs for consistency and freezes them as an input snapshot.Then it passes the input to the Agent, which maps task references to scene entities and binds the required atomic Skills. Based on these bindings, the Agent determines the robot placement, action parameters, and skill execution order, forming a directed acyclic graph of atomic skills **(Skill DAG**). After that, it translates task requirements into checkable success conditions and establishes an explicit correspondence among task requirements, Skill nodes, and validation evidence. For example, "place the cube in the tray" need to check both `Place` action and the final spatial relation between the cube and the tray. Together, these elements form a complete task proposal, as illustrated in step 03 of Figure 3(b).

Task Construction. As shown in step 06 of Figure 3, the Task Plan Agent has access to five high-level interaction tools: **`build`**, **`run`**, **`inspect`**, **`finish`**, and **`revise`**. For example, after forming a proposal, the Agent calls `build` to submit it to the Harness, as illustrated in step 04 of Figure 3(b). The Harness checks entity references, parameters, robot bindings, and task-requirement coverage, among other aspects, and produces the task configuration file.This configuration is then passed to `simengine` for parsing and compilation. The compiler returns node information if compilation succeeds, or failure logs otherwise. The Harness uses this feedback in its subsequent validation, which also includes geometric checks and semantic review. Only after all checks pass is the configuration frozen as a Candidate and returned to the Agent, completing the first successful `build`.

Execution and Validation. Once configured, the agent calls `run`. The Harness prepares the environment and launches a simulation for a specified Candidate, validation phase, and random seed, creating a new Attempt as shown in step 07 of Figure 3(c). During execution, the system records

Table 1: Quantitative comparison with prior scene-generation methods. Entries report mean ± sample standard deviation; ↑/↓ denote higher/lower is better, **bold** denotes the best result.

Method	CF↑	SL↑	PP↑	TF↑	VQ↑	COL↓	NAV↑	OOB↓	OPN↑
Holodeck	5.56±1.72	5.47±1.50	6.02±1.09	7.32±3.15	5.64±0.98	0.62±6.24	0.9967±0.0073	0.37±0.84	0.5997±0.2951
LayoutVLM	5.27±1.84	5.32±1.78	6.00±1.19	7.05±3.66	5.50±0.79	0.43±0.89	0.9954±0.0135	0.52±0.70	0.5417±0.4147
SAGE	5.20±1.98	5.13±1.60	5.92±1.08	7.07±3.31	5.15±0.78	0.35±1.09	0.9976±0.0081	0.03±0.18	0.8500±0.2311
EmbodiedGen V2	5.57±1.68	5.87±1.65	5.43±1.39	6.75±3.80	5.95±0.98	2.37±1.45	**0.9988±0.0052**	0.19±0.48	0.9145±0.1363
SceneSmith	5.34±1.44	6.18±1.38	6.59±1.01	**8.92±2.33**	5.52±0.96	2.05±1.87	0.9916±0.0217	**0.00±0.00**	0.9621±0.0742
Ours	**7.13±1.48**	**7.27±1.41**	**7.68±0.83**	8.33±2.88	**7.32±0.79**	**0.29±0.35**	0.9984±0.0130	**0.00±0.00**	**0.9875±0.0481**

object and robot states, Skill events, safety information, observations, actions, and video. Once the task finishes, this evidence is passed to the Harness for programmatic acceptance. Then, the check results and available raw failure evidence are passed to the Agent to inform its next tool call.

Independent Review and Finalization. If the first run passes these checks, the Agent does not immediately request `finish`. Instead, it invokes `inspect`, asking the Harness to initiate an independent VLM review and return the reviewer's feedback, as illustrated in step 10 of Figure 3(d). Only after considering this additional review and confirming task success does the Agent invoke `finish`, triggering the Harness's final delivery checks.If these checks pass, the Harness returns the final executable task configuration, as shown in Figure 3(e).

Repair and Revalidation. If the task does not pass, the Agent invokes `revise` to propose changes, such as adjusting robot placement, tuning action parameters, or correcting grasp poses. As shown in step 11 of Figure 3(d), the Harness first checks whether the proposed repair is allowed. Upon approval, it copies the original Candidate or failed draft into a new draft and applies the approved changes.The Agent then invokes `build` to recompile and check the revised draft. If these checks pass, the Harness freezes it as a new Candidate, which must undergo validation again.

3.3 Embodiments for Human-level Dexterity

Humanoid Robots. To execute the generated task plan on the Unitree G1 humanoid (Unitree Robotics, 2024), an execution adapter sits beneath the shared skill DAG and decouples control without altering atomic skills: locomotion targets are translated into planar velocity and heading references for learned balance-walking policies (Luo et al., 2025), while arm goals are resolved via the G1 kinematics and cuRobo (Sundaralingam et al., 2023). At 50 Hz, the adapter merges 15-DoF lower-body and 14-DoF arm commands into a 29-DoF body action, dispatching hands separately. Confining embodiment-specific mappings entirely to the adapter keeps upper-level task skills fully reusable across morphologies (He et al., 2025; Li et al., 2025).

Dexterous-hand Robots. For dexterous manipulation, we deploy Sharpa Wave hands on the Split ALOHA platform (6-DoF arm and 22-DoF hand per side). A task-specific contact solver maps object interaction contracts into collision-checked hand poses and finger configurations. During execution, cuRobo plans arm trajectories while finger joints are commanded explicitly, with PhysX recording per-link contact telemetry rather than aggregated gripper readings. Each task enforces a fail-closed contact contract (e.g., designated index-pad contact and phase-wise force gates). Integrated with the five-tool agent loop above, this pipeline enables automatic contact-rich trajectory generation.

4 Experiments

We design our experiments to answer three questions: **(Q1)** Does our task-oriented environment generation pipeline better satisfy perceptual and geometric requirements? (Section 4.1) **(Q2)** Does our RSI framework enable more reliable generation of challenging and diverse robot tasks? (Section 4.2) **(Q3)** Do our generated diverse demonstrations improve model performance? (Section 4.3)

4.1 Enviroment Generation

Evaluation setup. We compare our method with five representative scene generation baselines: Holodeck Yang et al. (2024b), LayoutVLM Sun et al. (2025a), SAGE Xia et al. (2026), EmbodiedGen V2 Wang et al. (2026b), and SceneSmith Pfaff et al. (2026). All methods are evaluated on the same set of 20 indoor scene specifications spanning diverse room types and functional requirements, with three independent generations per scene for each method. We evaluate the generated environments



Figure 4: Qualitative comparison across indoor-scene generation methods.

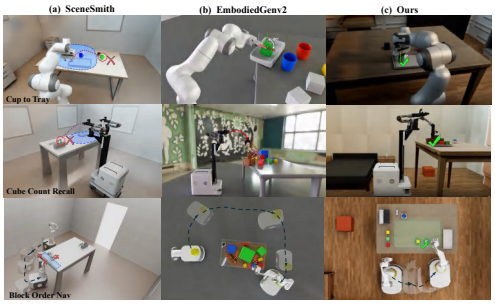


Figure 5: **Qualitative comparison.** Blue regions indicate reachable workspaces. Red regions indicate execution difficulties or failures; green tick indicate successful interactions;red fork indicate failure interactions

from both perceptual and geometric perspectives. GPT-5.5 (OpenAI) is provided with four rendered views of each scene and scores five perceptual metrics on a 1–10 scale: Content / Function (CF), Spatial Layout (SL), Physical Plausibility (PP), Text Fidelity (TF), and Visual Quality (VQ). We additionally report four geometry-based metrics: Collision (COL), Navigability (NAV), Out of Bound (OOB), and Opening Clearance / Openness (OPN). Detailed definitions of all metrics and used assets are provided in Appendix B.1 and B.2.

Quantative Results. Table 1 demonstrates that our method achieves SOTA performance on CF, SL, PP, VQ, COL, OOB, and OPN. Specifically, considering perceptual metrics, CF reaches 7.13 ± 1.48, compared to 5.57 ± 1.68 for the strongest baseline, representing a 28.0% relative improvement. SL and PP reach 7.27 ± 1.41 and 7.68 ± 0.83, outperforming the strongest baselines by 17.6% and 16.5%, respectively, while VQ reaches 7.32 ± 0.79, improving over EmbodiedGen V2 (5.95 ± 0.98) by 23.0%. SceneSmith obtains the highest TF score of 8.92 ± 2.33, whereas our method yields a competitive score of 8.33 ± 2.88. Crucially, these gains in functional completeness and spatial organization do not occur at the expense of geometric usability. Our method achieves the lowest COL score of 0.29 ± 0.35, produces no out-of-bounds objects, maintains a NAV score of 0.9984 ± 0.0130 (comparable to the top baseline result of 0.9988 ± 0.0052), and achieves the highest OPN score of 0.9875 ± 0.0481. This combination is particularly crucial for embodied applications, as richer and more functionally complete scenes frequently introduce collisions, blocked openings, or fragmented traversable space. In contrast, our approach enhances scene content and organization while preserving navigability, valid boundaries, and access to openings.

Qualitative Results. Figure 4 further illustrates this distinction: Holodeck and SAGE often generate relatively sparse scenes, LayoutVLM frequently leaves large unused areas or weakly structured layouts, and EmbodiedGen V2 and SceneSmith generally create richer scenes but still exhibit locally inconsistent arrangements. Conversely, our method produces denser yet structured environments with clear functional regions and preserved circulation space.

4.2 Robotic Demonstration Generation

We evaluate robotic trajectory generation through two questions: **(1) Does our scene generation method better support downstream demonstration generation than the baselines? (2) How do execution-feedback-driven refinement and robot placement adaptation in our agent affect the success rate and cost of demonstration generation?**

Experimental setup. Our main quantitative benchmark comprises 16 tasks: 10 for fixed-base manipulation and 6 for mobile manipulation, evenly split between short- and long-horizon tasks within each category. A demonstration is accepted only if it satisfies predefined task completion predicates and passes trajectory data-integrity checks. The complete task list, horizon assignments, and acceptance criteria are provided in the Appendix C.

Demonstration Generation Comparison. To evaluate how reliably each generated scene supports demonstration generation, we run 10 independent agent sessions from the same initial scene state

Table 2: **Per-task demonstration generation performance on fixed-base and mobile manipulation.** Short- and long-horizon tasks are distinguished by background color, with long-horizon tasks shaded in orange. Darker cells indicate the best result for each metric. ✗ indicates that EmbodiedGen V2 does not support generating the articulated assets required by the task. **m.** denotes memory tasks.

Category	Task	SceneSmith + Planner			EmbodiedGen V2 + Planner			EmbodiedSmith		
		ASR↑	Top-5↑	Token↓	ASR↑	Top-5↑	Token↓	ASR↑	Top-5↑	Token↓
Fixed-base manipulation	Open Drawer	32%	40%	**8782**	✗	✗	✗	**78%**	**80%**	10885
	Cup to Tray	19%	24%	9672	35%	40%	8721	**64%**	**72%**	**4288**
	Click Button	37%	44%	69814	89%	**100%**	**9235**	**100%**	**100%**	16193
	Basket to Tray	69%	78%	43879	0%	0%	**13838**	**70%**	**82%**	23221
	Open Cabinet	0%	0%	713248	✗	✗	✗	**35%**	**42%**	**20429**
	Cup Pairing (**m.**)	0%	0%	**62518**	12%	20%	138551	**51%**	**58%**	120071
	Block Order (**m.**)	7%	10%	201026	0%	0%	87644	**83%**	**88%**	**80386**
	Cube Count (**m.**)	3%	6%	186377	0%	0%	177621	**79%**	**84%**	**90316**
	Button History (**m.**)	15%	26%	67662	71%	80%	93251	**83%**	**92%**	**27611**
	Object Gesture (**m.**)	59%	**82%**	**57626**	0%	0%	88343	**62%**	74%	72532
Mobile manipulation	Open Drawer	0%	0%	161643	✗	✗	✗	**43%**	**62%**	**16389**
	Cup to Tray	22%	44%	118139	0%	0%	12315	**79%**	**96%**	**5396**
	Lift Cover	7%	14%	140514	40%	80%	22415	**88%**	**100%**	**4979**
	Book Stow(**m.**)	0%	0%	90168	✗	✗	✗	**17%**	**34%**	**26385**
	Cup Pairing(**m.**)	0%	0%	137257	0%	0%	221043	**52%**	**62%**	**73892**
	Block Order(**m.**)	0%	0%	124671	0%	0%	219762	**93%**	**100%**	**50283**
Average		21%	24%	137062	21%	27%	91061	**67%**	**76%**	**40204**

to produces 10 simulation-ready task configuration as described in Section 3.2 for each task.We then evaluate it using the same set of 10 simulation seeds, which are not used during generation or refinement.

We report **ASR** (mean success rate across all candidates and tasks), **Top-5 ASR** (mean success rate of the five best-performing candidates per task) and **Token** (mean VLM output tokens per session). Sessions that produce no executable task plan are assigned zero success and included in the token accounting. As shown in Table 2, EmbodiedSmith outperforms each baseline in ASR on its supported tasks. For our method, the mobile setting performs better on *Cup to Tray* but worse on *Open Drawer*, suggesting that navigation may improve reachability, while grasp-feasible placement may still be unsuitable for joint-constrained opening motions.

To understand these performance differences, we qualitatively examine the generated scenes in Figure 5. Some SceneSmith scenes have unsuitable support-surface heights or widely separated interaction locations, limiting reachability despite robot placement adaptation. Scenes generated by EmbodiedGen V2 exhibit inconsistent object scales and cluttered layouts that hinder manipulation. In contrast, our method better aligns scene layouts with task requirements and the robot's workspace, facilitating successful demonstration generation.

Ablations of RSI Design. We evaluate two ablations on the same initial scenes. *w/o Feedback* retains initial robot placement selection and task plan but disables revision based on execution feedback. *w/o Placement Validation* selects robot placement solely from three-view observations and object world coordinates, without considering kinematic or collision constraints or validating placement feasibility, but retains execution feedback for program revision.All other settings and the evaluation protocol follow those used in the demonstration generation comparison above.

As shown in Table 3, removing execution feedback reduces ASR to almost 0% across all task categories, despite lower token consumption. Removing placement validation particularly affects long-horizon fixed-base tasks; smaller drops on mobile tasks suggest that navigation may partially compensate for suboptimal initial placement.

4.3 Generated Data for Policy Training

We design experiments to verify that the generated robot trajectories can effectively improve the performance and generalization ability of trained models, thereby demonstrating the capability of our engine to provide large-scale training data.

Table 3: **Agent Workflow Ablation.** We evaluate the effects of execution feedback and robot placement validation across fixed-base and mobile manipulation tasks. Token denotes the mean cumulative VLM output tokens per session.

	Fixed-Short			Fixed-Long			Mobile-Short			Mobile-Long		
Method	**ASR↑**	**Top-5↑**	**Token↓**	**ASR↑**	**Top-5↑**	**Token↓**	**ASR↑**	**Top-5↑**	**Token↓**	**ASR↑**	**Top-5↑**	**Token↓**
w/o Feedback	4.0%	8.0%	**3721**	0.0%	0.0%	**6529**	0.0%	0.0%	**2979**	0.0%	0.0%	**5448**
w/o Placement Validation	48.6%	56.8%	63819	13.2%	22.4%	208316	67.3%	79.6%	13294	47.6%	**67.3%**	40898
Ours	**69.4%**	**75.2%**	15003	**71.6%**	**79.2%**	78183	**70.0%**	**86.0%**	8921	**54.0%**	65.3%	50186

Table 4: **Task success rates (%) under simple and randomized evaluation settings.** We compare $\pi_{0.5}$ and Fast-WAM fine-tuned on the *clean* or *mixed* dataset.

Task	π0.5 (Clean)			π0.5 (Mixed)			FastWAM (Clean)			FastWAM (Mixed)		
	Simple	**Random**	**Average**	**Simple**	**Random**	**Average**	**Simple**	**Random**	**Average**	**Simple**	**Random**	**Average**
Spatially Guided Placement	81%	5%	43.0%	64%	47%	55.5%	34%	1%	17.5%	25%	12%	18.5%
Cuboid Placement	88%	11%	49.5%	65%	59%	62.0%	41%	2%	21.5%	28%	18%	23.0%
Block Stacking	30%	4%	17.0%	39%	22%	30.5%	11%	0%	5.5%	15%	5%	10.0%
Unstack and Place	69%	7%	38.0%	31%	18%	24.5%	26%	0%	13.0%	12%	4%	8.0%
Drawer Opening	35%	5%	20.0%	46%	30%	38.0%	13%	1%	7.0%	18%	9%	13.5%
Sequential Object Placement	56%	4%	30.0%	43%	37%	40.0%	21%	0%	10.5%	16%	7%	11.5%
Bottle Righting	64%	8%	36.0%	54%	45%	49.5%	27%	1%	14.0%	22%	10%	16.0%
Bowl Nesting	68%	13%	40.5%	64%	56%	60.0%	29%	2%	15.5%	26%	15%	20.5%
Microwave Door Closing	78%	12%	45.0%	67%	61%	64.0%	37%	3%	20.0%	29%	18%	23.5%
Sponge Wiping	43%	3%	23.0%	35%	27%	31.0%	16%	0%	8.0%	13%	4%	8.5%
Average	61.2%	7.2%	34.2%	50.8%	40.2%	45.5%	25.5%	1.0%	13.3%	20.4%	10.2%	15.3%

Training Data Generation. We selected two target scenes, a study room and a kitchen, and planned each scene from scratch. For each scene, we designed five scene-appropriate tasks covering common atomic skills such as pick, place, and open, with task lengths ranging from three to six atomic skills. It is worth noting that, after determining the tasks, we directly used prompts to generate all task plans, which could be executed successfully without any manual debugging or intervention. Detailed prompt descriptions for the generated tasks are provided in the Appendix E.

We construct two equal-duration datasets: **clean**, comprising 20 hours of fixed-layout demonstrations without randomization, and **mixed**, combining 10 hours of these demonstrations with 10 hours of randomized demonstrations featuring varying tabletop layouts.

Experimental Setup. We select $\pi_{0.5}$ and FastWAM as representative VLA and WAM models, respectively, train them on data collected in a study and a kitchen, and evaluate their performance in two settings with different generalization requirements. Each scene dataset contains five tasks, with more details provided in the appendix. We define two evaluation settings. The **simple** setting uses the same scene layouts as those used to collect the *clean* dataset, with no randomization. The **random** setting uses scene randomization controlled by random seeds.

Evaluation Results. Table 4 shows models trained on the *clean* dataset perform well in the *simple* setting, but their success rates drop sharply in the *random* setting. This suggests that training on fixed-layout data may encourage reliance on scene-specific image–trajectory associations rather than manipulation strategies that generalize across layouts. Models trained on the *mixed* dataset achieve higher average success rates than their *clean*-trained counterparts, with particularly pronounced gains in the *random* setting. These results show that increasing the diversity of our generated data can substantially improve generalization to randomized scenes, supporting its potential use as a source of large-scale training data.

5 Conclusion

We presented EmbodiedSmith, a framework for scalable embodied data generation through RSI. By integrating asset, scene, and task generation, EmbodiedSmith supports diverse tasks, complex embodiments, and physical interactions. Its coupled environment and task generation loop enables mutual refinement, improving overall generation success rates. Experiments demonstrate the quality, diversity, and generation efficiency of the resulting environments and demonstrations. These results suggest that jointly refining simulation environments and executable task plans offers a promising path toward scalable training data and evaluation infrastructure for generalist robot models.

AI USE STATEMENT

In this work, we used generative AI tools to generate synthetic datasets, implement methods, propose or refine hypotheses, design or provide feedback on research methodologies and experiments, assist with translation, help develop theoretical models or conceptual frameworks, and clean or reformat datasets. We did not use generative AI tools to support qualitative or thematic data analysis or to interpret experimental results. The formulation of mathematical claims, provision of critical ingredients for proving mathematical claims, and assistance with writing proofs were not applicable to this work. Additionally, we used generative AI tools for tasks for which disclosure is recommended, including suggesting experimental parameters, creating or editing software code, creating research artifacts, drafting parts of the manuscript, summarizing or analyzing existing literature, identifying research topics or gaps, brainstorming, sourcing and searching for information, improving manuscript readability, identifying relevant literature, and formatting references. We reviewed all AI-assisted work.

Generative AI was used to support the development of conceptual frameworks, research hypotheses, methodology, and experimental design. These suggestions were reviewed and discussed by the authors before being incorporated into the work. Relevant ideas were also checked against the existing literature and, where appropriate, evaluated empirically. AI-assisted implementation and code were reviewed and validated by authors through the applicable unit, integration, simulation, and end-to-end tests. Synthetic environments, assets, task plans, and robot demonstrations generated by the proposed agent framework were subjected to the geometric, physical, kinematic, task-success, and data-integrity validation procedures described in the paper rather than accepted directly from model outputs. AI-assisted data cleaning and reformatting were performed during the proposed data generation pipeline through inspectable scripts or structured transformations. For translation and manuscript preparation, the authors checked AI-assisted text against the intended technical meaning and revised it where necessary.

For recommended-disclosure tasks, literature summaries, research-gap suggestions, brainstorming outputs, and candidate references generated with AI assistance were treated only as aids to discovery. Relevant scientific claims and citations were verified against the original papers or other primary sources before inclusion. AI-assisted manuscript drafts, edits, reference formatting, software code, and other research artifacts were reviewed and revised by the authors before use. Generative AI was not used to determine the interpretation of experimental results or the scientific conclusions reported in the paper. Reported quantitative results were obtained from the implemented experimental pipeline rather than generated or altered by an AI assistant. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies automated generation of simulation environments and robot demonstrations. Generative models used within EmbodiedSmith may produce incorrect scene interpretations, physically invalid assets, or infeasible robot tasks. Generated outputs are therefore subjected to the geometric, physical, kinematic, task-success, and safety validation procedures described in the paper. Nevertheless, demonstrations generated in simulation should not be assumed to be safe for direct deployment on physical robots without additional hardware-specific validation and safety mechanisms. External assets and resources are subject to their respective licenses, and any released assets or generated data will retain appropriate provenance and licensing information where applicable. The experiments reported in this paper do not involve human subjects or sensitive personal data. We view EmbodiedSmith as research infrastructure for model training and evaluation rather than as a replacement for real-world safety validation.

REPRODUCIBILITY STATEMENT

We provide the information required to reproduce EmbodiedSmith and its experimental evaluation throughout the main paper, appendix, and supplementary materials. Section 3 describes the scene-construction, task-generation, execution, validation, and feedback-driven refinement pipeline, including the shared skill and agent interfaces and embodiment-specific execution mechanisms. Appendix B documents the asset library and scene-evaluation metrics, while Appendix C provides the robotic task definitions and success criteria, supported skills and agent tools, agent

configuration, held-out evaluation protocol, robot configurations, and demonstration-recording procedure. Appendix D further specifies the randomization and data-generation protocol used for downstream policy training and evaluation. Experimental settings, evaluation metrics, and held-out evaluation procedures are reported in the corresponding experimental and appendix sections. Additional robot-evaluation details and extended scene generation implementation details are provided in the supplementary material.

REFERENCES

Genesis Authors. Genesis: A generative and universal physics engine for robotics and beyond, December 2024. URL https://github.com/Genesis-Embodied-AI/genesis-world.

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 S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 5982–5994. Curran Associates, Inc., 2022. doi: 10.52202/068431-0433. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/27c546ab1e4f1d7d638e6a8dfbad9a07-Paper-Conference.pdf.

Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models. *Advances in Neural Information Processing Systems*, 36:18225–18250, 2023.

Jialin Gao, Donghao Zhou, Mingjian Liang, Lihao Liu, Chi-Wing Fu, Xiaowei Hu, and Pheng-Ann Heng. Disco-layout: Disentangling and coordinating semantic and physical refinement in a multi-agent framework for 3d indoor layout synthesis, 2025. URL https://arxiv.org/abs/2510.02178.

Tairan He, Wenli Xiao, Toru Lin, Zhengyi Luo, Zhenjia Xu, Zhenyu Jiang, Jan Kautz, Changliu Liu, Guanya Shi, Xiaolong Wang, Linxi Fan, and Yuke Zhu. HOVER: Versatile neural whole-body controller for humanoid robots. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025. doi: 10.1109/ICRA55743.2025.11128549.

Siyi Hu, Diego Martin Arroyo, Stephanie Debats, Fabian Manhardt, Luca Carlone, and Federico Tombari. Mixed diffusion for 3d indoor scene synthesis. In *2026 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 1262–1272, 2026. doi: 10.1109/WACV61042.2026.00129.

Zhi Jing, Siyuan Yang, Jicong Ao, Ting Xiao, Yu-Gang Jiang, and Chenjia Bai. HumanoidGen: Data generation for bimanual dexterous manipulation via LLM reasoning. In *Advances in Neural Information Processing Systems*, volume 38, San Diego, CA, USA and Mexico City, Mexico, 2025. doi: 10.52202/085713-5220. URL https://papers.neurips.cc/paper_files/paper/2025/hash/e4ef7454447baa15a424314e6284441b-Abstract-Conference.html.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Chengshu Li, Mengdi Xu, Arpit Bahety, Hang Yin, Yunfan Jiang, Huang Huang, Josiah Wong, Sujay Garlanka, Cem Gokmen, Ruohan Zhang, et al. Momagen: Generating demonstrations under soft and hard constraints for multi-step bimanual mobile manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 112425–112446, 2026.

Jialong Li, Xuxin Cheng, Tianshu Huang, Shiqi Yang, Ri-Zhao Qiu, and Xiaolong Wang. AMO: Adaptive motion optimization for hyper-dexterous humanoid whole-body control. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.061.

Kevin Lin, Ajay Mandlekar, Caelan Reed Garrett, Nikita Chernyadev, Yu Fang, Runyu Ding, Yuqi Xie, Justin Tran, Linxi Fan, and Yuke Zhu. HumanoidMimicGen: Data generation for loco-manipulation via whole-body planning. *arXiv preprint arXiv:2605.27724*, 2026. URL https://arxiv.org/abs/2605.27724. Presented at the ICRA 2026 Workshop on Synthetic Data for Robot Learning.

Yibin Liu, Yaxing Lyu, Daqi Gao, Zhixuan Liang, Weiliang Tang, Shilong Mu, Xiaokang Yang, and Yao Mu. From passive observer to active critic: Reinforcement learning elicits process reasoning for robotic manipulation. *arXiv preprint arXiv:2603.15600*, 2026.

Zhengyi Luo, Ye Yuan, Tingwu Wang, Chenran Li, Sirui Chen, Fernando Castañeda, Zi-Ang Cao, Jiefeng Li, and Yuke Zhu. GR00T Whole-Body Control. https://github.com/NVlabs/GR00T-WholeBodyControl, 2025. Software repository, accessed 2026-09-12.

Soroush Nasiriany, Sep Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations*, volume 2026, pp. 98643–98667, 2026.

OpenAI. GPT-5.5. OpenAI API Documentation. URL https://developers.openai.com/api/docs/models/gpt-5.5. Accessed: 2026-09-26.

Nicholas Pfaff, Thomas Cohn, Sergey Zakharov, Rick Cory, and Russ Tedrake. Scenesmith: Agentic generation of simulation-ready indoor scenes. In *Forty-third International Conference on Machine Learning*, 2026.

Hou In Derek Pun, Hou In Ivan Tam, Austin T Wang, Xiaoliang Huo, Angel X Chang, and Manolis Savva. Hsm: Hierarchical scene motifs for multi-scale indoor scene generation. *arXiv preprint arXiv:2503.16848*, 2025.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21783–21794, June 2024.

Carmelo Sferrazza, Dun-Ming Huang, Xingyu Lin, Youngwoon Lee, and Pieter Abbeel. HumanoidBench: Simulated humanoid benchmark for whole-body locomotion and manipulation. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024. doi: 10.15607/RSS.2024.XX.061. URL https://www.roboticsproceedings.org/rss20/p061.html.

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 *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 29469–29478, 2025a.

Wenzhuo Sun, Mingjian Liang, Wenxuan Song, Xuelian Cheng, and Zongyuan Ge. Roomplanner: Explicit layout planner for easier llm-driven 3d room generation, 2025b. URL https://arxiv.org/abs/2511.17048.

Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, Nathan Ratliff, and Dieter Fox. CuRobo: Parallelized collision-free robot motion generation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8112–8119, 2023. doi: 10.1109/ICRA48891.2023.10160765.

Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 20507–20518, 2024.

SAM 3D Team, Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, Aohan Lin, Jiawei Liu, Ziqi Ma, Anushka Sagar, Bowen Song, Xiaodong Wang, Jianing Yang, Bowen Zhang, Piotr Dollár, Georgia Gkioxari, Matt Feiszli, and Jitendra Malik. Sam 3d: 3dfy anything in images. 2025. URL https://arxiv.org/abs/2511.16624.

Yang Tian, Yuyin Yang, Yiman Xie, Zetao Cai, Xu Shi, Ning Gao, Hangxu Liu, Xuekun Jiang, Zherui Qiu, Feng Yuan, et al. Interndata-a1: Pioneering high-fidelity synthetic data for pre-training generalist policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 976–985, 2026.

Unitree Robotics. Unitree g1 humanoid robot. `https://www.unitree.com/g1/`, 2024. Official product documentation, accessed 2026-09-12.

Shuai Wang, Yaxin Feng, Xuekun Jiang, Shihan Tian, Ningyu Yan, Xing Shen, Chaoyang Lyu, Hui Wang, Yunsong Zhou, Hanqing Wang, et al. Gauge: A measurement-grounded benchmark for physical fidelity in simulation engines and video world models. *arXiv preprint arXiv:2608.05948*, 2026a.

Xinjie Wang, Liu Liu, Yu Cao, Ruiqi Wu, Wenkang Qin, Dehui Wang, Wei Sui, and Zhizhong Su. Embodiedgen: Towards a generative 3d world engine for embodied intelligence, 2025. URL `https://arxiv.org/abs/2506.10600`.

Xinjie Wang, Liu Liu, Taojun Ding, Andrew Choi, Chaodong Huang, Mengao Zhao, Ziang Li, Jackson Jiang, Chunlei Yu, Shengxiang Liu, Wei Xu, and Zhizhong Su. Embodiedgen v2: An agentic, simulation-ready 3d world engine for embodied ai. *arXiv preprint arXiv:2607.07459*, 2026b.

Songlin Wei, Zhenhao Ni, Jie Liu, Zhenyu Zhao, Junjie Ye, Hongyi Jing, Junkai Xia, Xiawei Liu, Michael Leong, Liang Heng, Di Huang, and Yue Wang. SIMPLE: Simulation-based policy learning and evaluation for humanoid loco-manipulation. In *Conference on Robot Learning (CoRL 2026)*, 2026.

Lilian Weng. Harness engineering for self-improvement. *Lil'Log, July*, 4, 2026.

Hongchi Xia, Xuan Li, Zhaoshuo Li, Qianli Ma, Jiashu Xu, Ming-Yu Liu, Yin Cui, Tsung-Yi Lin, Wei-Chiu Ma, Shenlong Wang, Shuran Song, and Fangyin Wei. Sage: Scalable agentic 3d scene generation for embodied ai. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Wenli Xiao, Jia Xie, Tonghe Zhang, Haotian Lin, Letian Fu, Haoru Xue, Jalen Lu, Yi Yang, Cunxi Dai, Zi Wang, et al. Enpire: Agentic robot policy self-improvement in the real world. *arXiv preprint arXiv:2606.19980*, 2026.

Yandan Yang, Baoxiong Jia, Peiyuan Zhi, and Siyuan Huang. Physcene: Physically interactable 3d scene synthesis for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16262–16272, 2024a.

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 (NeurIPS)*, 2025.

Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, et al. Holodeck: Language guided generation of 3d embodied ai environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16227–16237, 2024b.

Yifan Ye, Yankai Fu, Yaoxu Lv, Bohan Hou, Jun Cen, Lingdong Kong, Duo Zheng, Tianxing Chen, Jiaming Liu, Ziang Cao, et al. Data pyramid for embodied manipulation. *arXiv preprint arXiv:2607.24744*, 2026.

Matt Zhou, Ruining Li, Xiaoyang Lyu, Zhaomou Song, Zhening Huang, Chuanxia Zheng, Christian Rupprecht, Andrea Vedaldi, and Shangzhe Wu. Articraft: An agentic system for scalable articulated 3d asset generation. *arXiv preprint arXiv:2605.15187*, 2026.

A ABSTRACT

In this appendix, we provide additional information:

- Section B details scene generation, including the asset library, and scene-evaluation metrics.
- Section C provides details of robotic demonstration generation, including task definitions and success criteria, robot skills, agent tools, evaluation protocols, ablation settings, robot configurations, and demonstrations across embodiments.
- Section D describes the hierarchical randomization and feedback-driven data-generation procedure used for policy training and evaluation.
- Section E provides additional robot evaluation details, including task generation prompts, data-collection statistics, training hyperparameters, and representative evaluation rollouts.

B DETAILS OF ENVIROMENT GENEARTION

B.1 USED ASSETS LIBRARY

To preserve the intended behavior of each system, each baseline retains its native asset source and generation pipeline, while all methods receive identical input specifications and matched random seeds whenever supported.

Our method utilizes a self-constructed asset library as the asset index for scene composition. This library contains simulation-ready assets with standardized geometric representations and semantic metadata, enabling the pipeline to retrieve objects based on both scene semantics and downstream physical requirements. Given an input specification, the pipeline progressively selects suitable assets, determines their functional roles and spatial relationships, and instantiates them using the proposed scene generation strategy.

B.2 EVALUATION METRICS

We evaluate generated environments using five perceptual metrics and four geometry-based metrics.

Perceptual Metrics. For each scene, GPT-5.5 (OpenAI) receives four rendered views and assigns a score from 1 to 10 for each metric. **Content / Function (CF)** measures whether the scene contains the objects and functional components required by the intended room type. **Spatial Layout (SL)** evaluates object placement, orientation, spacing, functional zoning, and circulation space. **Physical Plausibility (PP)** assesses visually perceived grounding, support, alignment, and obvious floating or interpenetration artifacts. **Text Fidelity (TF)** measures consistency with the input specification in terms of object categories, quantities, attributes, materials, spatial relations, and room type. **Visual Quality (VQ)** evaluates materials, textures, lighting, clarity, scale consistency, and rendering artifacts.

Geometry-based Metrics. Collision (COL) is the number of objects involved in actual mesh intersections. **Navigability (NAV)** is the fraction of valid floor area contained in the largest continuously traversable region under robot-width constraints. **Out of Bound (OOB)** counts objects whose geometry extends beyond valid room or building boundaries. **Opening Clearance / Openness (OPN)** is the fraction of door and window sides that remain unobstructed.

C ROBOTIC DEMONSTRATION GENERATION DETAILS

C.1 TASK DEFINITIONS AND SUCCESS CRITERIA

The main evaluation comprises 16 tasks: ten fixed-base manipulation tasks and six mobile manipulation tasks. The task table groups them as Easy and Long horizon. Table 7 summarizes the scenes used in this evaluation, with shared scenes listed once. Table 8 provides the task instructions, scene assignments, group labels, and completion conditions.

Success is determined by predefined task-specific completion conditions. For multi-stage tasks, a successful episode must satisfy both the required intermediate-event conditions and the final-state

Table 5: **Robot Skills.** We summarize the parameterized atomic skills supported by the simulation execution layer.

Name	Description
navigate	Move the mobile base to a working location before local manipulation.
pick	Execute pre-grasp, approach, gripper closure, and lift for a target object.
manualpick	Execute a grasp with explicitly adjusted grasp pose parameters.
dexpick	Execute a grasp using a selected predefined grasp pose.
dynamicpick	Plan and execute a grasp for a moving target.
place	Transport a held object to a target, descend, release it, and retreat.
dexplace	Place a held object within a specified geometric target region.
open	Plan and execute opening motion for an articulated object such as a drawer or door.
close	Plan and execute closing motion for an articulated object such as a drawer or door.
artpreplan	Execute planned approach poses for an articulated-object interaction.
rotate	Plan and execute rotation of an articulated joint to a target state.
press	Approach and press a designated button or contact region.
move	Move an object toward another object along a planar trajectory.
rotate_obj	Reorient a grasped object while maintaining its attachment.
approach_rotate	Move a held object toward a target while adjusting its orientation.
flip	Flip a grasped object and release it.
goto_pose	Plan arm motion to a specified end-effector pose.
gripper_action	Open or close the gripper independently of arm motion.
arm_motion	Move the arm to a joint or relative end-effector target, including a home pose.
joint_ctrl	Directly control selected robot joints.
observe_hold	Hold the robot pose during a specified observation interval.
wait	Hold the end-effector pose for a specified number of simulation steps.
pour_water	Tilt a held container to transfer fluid and check the simulated outcome.

conditions. For recall tasks, any specified observation, occlusion, or intermediate manipulation stages must be completed before the final task outcome is evaluated. The task-completion conditions remain fixed during feedback-driven refinement. Task-specific thresholds for joint opening, lifting height, and pose matching are specified in the corresponding evaluation configurations.

C.2 Robot Skills

The task agent composes parameterized atomic skills into a task DAG. Table 5 summarizes the robot skills supported by the simulation execution layer. Each skill node specifies an execution resource and goal parameters, together with a target entity when applicable.

C.3 Agent Tools

A lightweight harness exposes the five task-level tools in Table 6. The agent uses them to ground scene entities, build and execute skill plans, revise failed plans from execution feedback, and submit successful rollouts for final validation.

C.4 Agent Configuration and Evaluation Protocol

We use `gpt-5.6-sol` with `high` reasoning effort as the task generation agent for all generation experiments. For each task and generated scene, we run 10 independent agent sessions from the same initial scene state. Each session attempts to produce a candidate comprising an initial robot placement, a skill plan, and executable task-success criteria. In the fixed-scene evaluation, feedback-driven refinement may revise the plan and initial robot placement within a maximum budget of 30 rounds. The task-success criteria, scene assets, and object layout remain unchanged.

Table 6: **Agent Tools.** We summarize the five task-level tools exposed by the harness for planning, execution, feedback-driven repair, and final validation.

Name	Description
`inspect`	Provide object identities, poses, support relations, interaction regions, and visual observations so the agent can ground targets and select an initial robot placement.
`build`	Organize grounded interactions into a skill DAG with target entities, execution resources, goal parameters, and dependencies. Define success criteria for terminal states, required events, safety constraints, and prohibited events. Before execution, the harness checks object references, skill parameters, robot compatibility, dependency structure, and consistency with the task instruction.
`run`	Execute the skill DAG in physics simulation with embodiment-specific motions. The harness returns skill outcomes, object and attachment states, safety events, and task checks.
`repair`	Use failed-rollout evidence to revise affected object bindings, dependencies, robot working poses, grasp poses, approach directions, or motion parameters. The revised plan is revalidated while task-success criteria remain fixed.
`finish`	Submit a successful rollout for final checks of terminal state, required events, runtime safety, and data integrity. The harness retains accepted trajectories as demonstrations.

The final plan and placement are frozen before evaluation under 10 shared held-out simulation seeds that are excluded from generation and refinement. The scene is reset before each evaluation trial, and the frozen candidate is executed without further agent-driven refinement or scene-configuration edits.

Across seeds, task-enabled randomization can vary initial object and robot poses and, when configured, observation conditions such as lighting. These variations can produce different navigation and manipulation trajectories for the same frozen task plan.

C.5 Evaluation Metrics and Ablation Settings

For each candidate, we compute the fraction of successful trials over the ten held-out seeds. ASR averages candidate success rates within each task and then across tasks. Top-5 ASR averages the five highest candidate success rates within each task and then across tasks. Token is the mean cumulative VLM output tokens per agent session. Sessions that do not produce an executable plan contribute zero to the success rates and remain in the token accounting.

The agent ablations use the same initial scenes and held-out evaluation protocol. *w/o Feedback* disables plan revision based on execution feedback. *w/o Placement Validation* selects the initial robot placement from three rendered views and object world coordinates without kinematic or collision-based placement validation; execution-feedback-driven revision and normal navigation remain enabled. Both ablations retain the original task-success criteria.

C.6 Robot Configurations and Demonstration Recording

The quantitative benchmark uses the robot configuration specified for each task. Fixed-base tasks keep the base pose fixed during execution, whereas mobile manipulation tasks allow base navigation to a suitable working location. Arm trajectories are planned with CuRobo using scene collision geometry. Within each task, all scene-generation methods use the same robot configuration and low-level control interfaces.

All quantitative-benchmark episodes are simulated in *Isaac Sim 6.0.1*, with both the physics and rendering time steps set to $30Hz$. During execution, we record temporally aligned visual observations, robot proprioceptive states, object states, and control actions, together with execution-stage annotations. Task-success logs and skill-execution outcomes are used to verify required intermediate events and final-state conditions. Only episodes that satisfy the task-success criteria and pass trajectory-integrity checks are included in the demonstration dataset, while failed episodes are retained for diagnosis but excluded from the dataset.

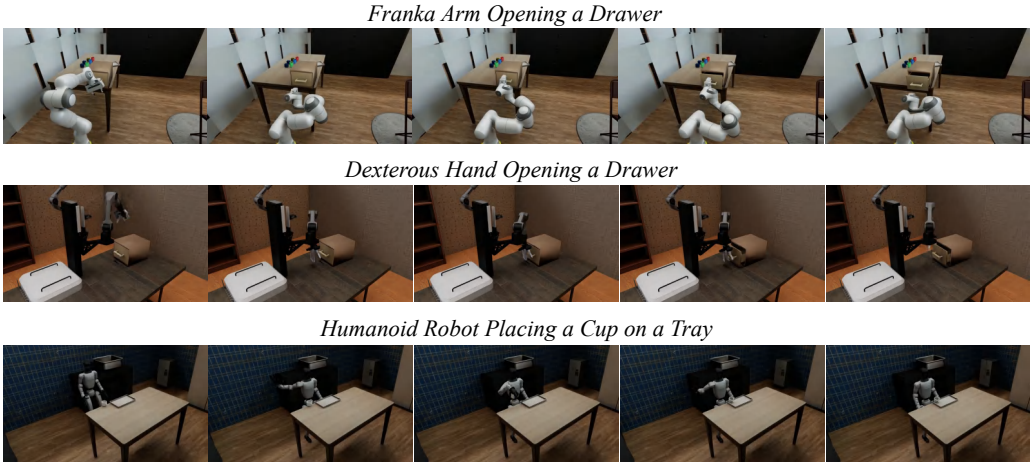


Figure 6: **Embodiedment Showcase**

C.7 Demonstrations across embodiments.

Beyond the quantitative evaluations above, we present expert demonstrations using single-arm, dual-arm, humanoid, and dexterous-hand configurations. Figure 6 shows keyframes spanning the initial state, intermediate interactions, and task completion. During execution, EmbodiedSmith records synchronized visual observations, robot states, and control actions, converting successful trajectories into demonstration data for policy learning.

Table 7: **Scene Prompts.** We list the full scene generation prompts used in the demonstration-generation evaluation. Prompts shared by multiple tasks appear once.

Scene	Prompt
Kitchen with Cups	A modern minimalist rectangular kitchen is arranged in a top-down layout. The floor is covered with large light grayish-white square tiles, and the center of the room remains largely open to provide a wide, unobstructed area for movement and operation. Along the left wall stands a large silver-gray double-door refrigerator, while a smaller dark gray storage cabinet or mini fridge is placed in the lower-left corner. On the upper-left wall, there are white wall cabinets as well as a set of light wood-paneled cabinets. Along the upper wall, a continuous light gray kitchen countertop is installed, with a black glass induction cooktop embedded slightly to the right of center. The cooktop features multiple circular heating zones. On the left side of the countertop, a small number of utensils and a blue compartment organizer are placed. On the upper-right wall, there is a gray open wall shelf divided into multiple storage compartments. In the lower-right area, a white freestanding kitchen appliance or storage cabinet is placed, with a front panel featuring a round knob, several buttons, and a black control module. At the bottom center of the room is a white sink cabinet with a rectangular sink embedded in the countertop, and cabinet doors and drawers below. All large furniture and appliances are arranged along the edges of the room so as not to occupy the center, creating a spacious pathway suitable for robot navigation and manipulation. The overall color palette consists of white, light gray, dark gray, and light wood tones, with soft and even indoor lighting. In the middle of the room, a tabletop workspace contains three cups, three cubes, and one tray. The cups are colored red, blue, and green, while the cubes are colored yellow, white, and black. Each cup is positioned directly in front of its assigned cube, forming the initial cup-to-cube spatial relations: the red cup is directly in front of the yellow cube, the blue cup is directly in front of the white cube, and the green cup is directly in front of the black cube. The tray is placed in front of the cups. The drawers are positioned near the edges of the workspace and are spatially separated from the cup-cube pairs.

Scene	Prompt
Study with Drawer	Create a modern minimalist rectangular study room or workspace in a top-down layout. The floor is covered with light natural wood flooring, with the wood grain running continuously along the long side of the room. Along the top wall, place a large white work desk with a wide tabletop and slightly rounded edges. On the desk, arrange a small number of office supplies and electronic devices in a neat but slightly random manner, including a dark gray rectangular device, a small white device, a slender red stationery item, a round cup or mug, a gray-black upright organizer, and a brownish-red cup, with appropriate spacing between all items.Along the left wall, place a light beige open bookshelf or storage cabinet with horizontal shelves. In the center-left area of the room, place a low dark reddish-brown rectangular storage box or cover-like object with rounded corners, a top surface featuring dark texture and visible scratches, and a light gold or light wood decorative band running across the middle. In the center-right area of the room, place a large light gray circular rug with a darker gray border and a slightly mottled texture.Maintain sufficient spacing between the work desk, bookshelf, rectangular storage box, and circular rug, so that the center of the room forms a continuous, unobstructed open area suitable for robot navigation and manipulation. Use soft, even indoor lighting, and adopt a color palette dominated by light wood, white, beige, gray, and dark brown tones.A tabletop workspace contains 3 cups, 3 cubes, 1 tray. The cups are red, blue, and green, and the cubes are yellow, white, and black. Each cup is positioned directly in front of its assigned cube, forming the initial cup-to-cube spatial relations: the red cup is directly in front of the yellow cube, the blue cup is directly in front of the white cube, and the green cup is directly in front of the black cube. The tray is positioned in front of the cups. The drawers are positioned near the edges of the workspace and are spatially separated from the cup-cube pairs.
Living Room with Basket	Create a modern minimalist rectangular living room in a top-down layout. Use light natural wood flooring with a consistent wood-grain direction. Arrange most large furniture along the edges of the room and preserve a large, continuous, unobstructed open area in the center for robot navigation and manipulation.Place a light gray three-seat sofa along the bottom side of the room. The sofa should have three clearly separated seat cushions and a simple modern design.Place a rectangular light-wood table slightly left of the center of the room. Use the table as the main tabletop manipulation workspace. Scatter several small everyday objects across the tabletop with clear spacing between them, including a blue rectangular package or book, several small circular objects, a yellow handle-shaped object, a small frame-like object, and a few small red, white, and dark-colored items. Keep the tabletop objects sparse and clearly separated so that each object is individually accessible for robotic manipulation.Along the left wall, place a long and narrow light-wood storage cabinet or sideboard. Place a few small dark decorative objects or containers on top of it. In the lower-left area, place a low dark-brown rectangular storage box or small piece of furniture with a lighter vertical decorative strip.Add additional dark or brown wooden furniture along the upper and right edges of the room, while keeping these objects away from the central navigation area.Use a warm modern color palette dominated by light wood, white, light gray, dark brown, and small black accents. Use soft and even indoor lighting. Keep the overall scene clean, spacious, and suitable for robotics simulation, tabletop manipulation, object rearrangement, and indoor navigation tasks.In the middle of the room, there is a tabletop workspace containing a wooden basket with a handle and a shallow tray.
Bedroom with Button	Create a modern minimalist rectangular bedroom in a top-down layout. Cover the floor with light natural wood flooring, with the wood grain running continuously along the long side of the room. Keep the central area of the room largely open and unobstructed for robot navigation and manipulation. Place a large dark gray double bed in the upper-left area of the room. The bed should have a rectangular shape with slightly rounded edges and a dark gray to charcoal-colored mattress and frame. Along the right wall, place a large brown wooden three-door wardrobe. Each wardrobe door should have recessed rectangular panel details and a long curved white or silver handle. Use a light gray frame around the wardrobe. At the bottom center of the room, place a low light-wood bench, storage cabinet, or bed-end furniture piece. Place a dark gray or black low cabinet in the lower-left area, and a dark brown wooden storage cabinet or bedside cabinet in the lower-right area. In the upper-right area, place a small orange-brown square ottoman or padded stool. Add a small dark wooden cabinet or furniture piece nearby along the upper wall. Arrange all large furniture close to the room boundaries and preserve a wide, continuous open area in the center. Use a warm modern color palette dominated by light wood, dark gray, brown, beige, and small orange-brown accents. Use soft and even indoor lighting. Ensure the scene is clean, spacious, and suitable for robotics simulation, indoor navigation, object retrieval, and wardrobe interaction tasks.In the middle of the room, there is a table with a blue circular button placed on top of it.
Kitchen Counter with Cabinet	The scene contains a long kitchen-style counter measuring approximately 2.0 m in length and 0.65 m in depth. A microwave is placed on one end of the counter at a height reachable by the robot. The microwave has a single graspable hinged door that can be opened and closed. A food container is placed on the countertop near the microwave. At the opposite end of the counter, place a small storage cabinet with a single graspable hinged door. Two graspable storage boxes are placed on the countertop near the cabinet. A storage basket is positioned on the counter a short distance away from the cabinet. The microwave area and cabinet area should be sufficiently separated so that the robot must reposition its mobile base when moving between them. A continuous unobstructed navigation corridor must remain along the entire front side of the counter.

Scene	Prompt
Bedroom with Pose References	Create a modern minimalist rectangular bedroom in a top-down layout. Cover the floor with light natural wood flooring, with the wood grain running continuously along the long side of the room. Keep the central area of the room largely open and unobstructed for robot navigation and manipulation. Place a large dark gray double bed in the upper-left area of the room. The bed should have a rectangular shape with slightly rounded edges and a dark gray to charcoal-colored mattress and frame. Along the right wall, place a large brown wooden three-door wardrobe. Each wardrobe door should have recessed rectangular panel details and a long curved white or silver handle. Use a light gray frame around the wardrobe. At the bottom center of the room, place a low light-wood bench, storage cabinet, or bed-end furniture piece. Place a dark gray or black low cabinet in the lower-left area, and a dark brown wooden storage cabinet or bedside cabinet in the lower-right area. In the upper-right area, place a small orange-brown square ottoman or padded stool. Add a small dark wooden cabinet or furniture piece nearby along the upper wall. Arrange all large furniture close to the room boundaries and preserve a wide, continuous open area in the center. Use a warm modern color palette dominated by light wood, dark gray, brown, beige, and small orange-brown accents. Use soft and even indoor lighting. Ensure the scene is clean, spacious, and suitable for robotics simulation, indoor navigation, object retrieval, and wardrobe interaction tasks. In the middle of the room, there is a table. On the tabletop, three reference objects — a red cube, a yellow cube, and a blue cylinder — are arranged in a row as the reference region, each at a distinct position and orientation. An opaque rectangular memory cover with a graspable handle on its top surface is placed beside the reference row; it is large enough to be moved over the reference region and lifted vertically to reveal the objects underneath. On the other side of the table, an empty tray serves as the work region, and the three matching work objects — identical copies of the red cube, the yellow cube, and the blue cylinder — are placed next to the tray.
Cup Pairing Workspace	A tabletop workspace contains 3 cups, 3 cubes, one tray, a book, and a notebook. The cups have colors red, blue, and green. The cubes have colors yellow, white, and black. Each cup is positioned directly in front of its assigned cube, forming the initial cup-to-cube spatial relation: the red cup is directly in front of the yellow cube, the blue cup is directly in front of the white cube, and the green cup is directly in front of the black cube. The tray is positioned in front of the cups. The drawer are positioned near the edges of the workspace and are spatially separated from the cup-cube pairs.
Cube Counting Workspace	A tabletop workspace contains 3 cups, one tray, a book, a notebook, and multiple cubes. The cups have colors red, blue, and green. The three cup-pair cubes have colors yellow, white, and black. Each cup is positioned directly in front of its assigned cube, forming the initial cup-to-cube spatial relation: the red cup is directly in front of the yellow cube, the blue cup is directly in front of the white cube, and the green cup is directly in front of the black cube. The tray is positioned in front of the cups, and it contains three cubes: two red cubes and one blue cube. An opaque rectangular cover with a graspable top handle is placed beside the tray, large enough to be moved over the tray and hide the cubes inside. A basket is placed at one corner of the workspace, and three matching spare cubes — two red and one blue — are placed next to the basket. The drawer are positioned near the edges of the workspace and are spatially separated from the cup-cube pairs.
Block Order Workspace	The scene contains a large rectangular table measuring approximately 2.0 m in width, 1.0 m in depth, and 1.18 m in height.All designated entry approach zones and circulation clearances must remain completely unobstructed. In particular, if a western-side entry clearance is reserved for the door, this area must remain entirely free of cabinets, carts, lamps, storage units, or any other fixtures. Optional storage fixtures may only be placed within the designated fixture zones, outside both the door-swing envelope and the continuous primary circulation path.On the tabletop, place an opaque brown rectangular memory cover measuring approximately 0.30 m long, 0.10 m wide, and 0.10 m high. The cover should function as a bottom-open shell, allowing it to be lifted vertically to reveal objects underneath.A graspable handle is mounted at the center of the top surface of the cover. The handle measures approximately 0.10 m long, 0.03 m wide, and 0.10 m high, and should be modeled as part of, or rigidly attached to, the cover. The memory cover should be represented as a graspable RigidObject.Place eight blocks on the tabletop. The blocks are divided into four color groups, with two blocks of the same color in each group. Four of the blocks — one of each color — are arranged into a rear row as the reference blocks, and the other four blocks — one of each color, matching the reference set — are placed in a tray at the front edge of the table as the work set. In front of the rear row, four equally spaced empty slots form the front row.Position the rectangular memory cover directly behind the rear row of blocks. Its long axis should be parallel to the direction of the block arrangement, so that the cover is aligned consistently with the two-row block layout.
Button Sequence Workspace	The scene contains a large rectangular table measuring approximately 2.0 m in width, 1.0 m in depth, and 1.18 m in height.All designated entry approach zones and circulation clearances must remain completely unobstructed. In particular, if a western-side entry clearance is reserved for the door, this area must remain entirely free of cabinets, carts, lamps, storage units, or any other fixtures. Optional storage fixtures may only be placed within the designated fixture zones, outside both the door-swing envelope and the continuous primary circulation path.On the tabletop, three circular push buttons — red, yellow, and blue — are mounted in a row near the center of the table. A single green cube is placed directly next to the red button. A tray is placed at the front edge of the table.

Scene	Prompt
Mobile Cup-Cube Workspace	A tabletop workspace contains 3 cups, 3 cubes, one tray, a book, and a notebook. The cups have colors red, blue, and green. The cubes have colors yellow, white, and black. Each cup is positioned directly in front of its assigned cube, forming the initial cup-to-cube spatial relation: the red cup is directly in front of the yellow cube, the blue cup is directly in front of the white cube, and the green cup is directly in front of the black cube. The three cup-cube pairs are distributed along the long side of the workspace with sufficient spacing between them. The tray is positioned in front of the cups. The drawer is positioned near one edge of the workspace and is spatially separated from the cup-cube pairs. A continuous unobstructed navigation corridor is maintained along the front side of the workspace, allowing a mobile robot to reposition between the drawer, cup-cube pairs, and tray.
Mobile Block Memory Table	The scene contains a large rectangular table measuring approximately 2.0 m in width, 1.0 m in depth, and 1.18 m in height. All designated entry approach zones and circulation clearances must remain completely unobstructed. In particular, if a western-side entry clearance is reserved for the door, this area must remain entirely free of cabinets, carts, lamps, storage units, or any other fixtures. Optional storage fixtures may only be placed within the designated fixture zones, outside both the door-swing envelope and the continuous primary circulation path. On the tabletop, place an opaque brown rectangular memory cover measuring approximately 0.30 m long, 0.10 m wide, and 0.10 m high. The cover should function as a bottom-open shell, allowing it to be lifted vertically to reveal objects underneath. A graspable handle is mounted at the center of the top surface of the cover. The handle measures approximately 0.10 m long, 0.03 m wide, and 0.10 m high, and should be modeled as part of, or rigidly attached to, the cover. The memory cover should be represented as a graspable RigidObject. Place eight blocks on the tabletop. The blocks are divided into four color groups, with two blocks of the same color in each group. Four blocks form a reference set and four blocks form a matching work set. Place the reference set toward one end of the table and the matching work set toward the opposite end, while preserving sufficient free tabletop space for rearrangement. Position the rectangular memory cover directly behind the reference blocks. Maintain a continuous unobstructed navigation corridor along the front side of the table so that the mobile robot can move between the reference and work areas.

D Task Randomization for Policy Training and Evaluation

To broaden training-data coverage and improve adaptability to environmental changes, we design a hierarchical, configurable randomization and data-generation mechanism with three levels: asset composition, spatial layout, and observation and initial conditions. The asset level samples distractor instances and counts from a filtered pool. The layout level samples the positions and orientations of designated task objects and distractors, subject to object-clearance, support-region, and reserved manipulation-space constraints. The observation and initial-condition level perturbs lighting, background textures, camera poses, and the initial robot-base pose. Each dimension's activation, sampling range, and perturbation magnitude are configured according to task requirements.

We organize these variations through a big-group–small-group–episode hierarchy. Each big group fixes an asset composition and contains small groups with different initial object layouts. Episodes within a small group share asset identities and object layouts while resampling enabled observation conditions and robot-base perturbations. This structure separately controls asset-composition coverage, layout coverage, and episodes per layout. Combinations of distractor categories, counts, and spatial arrangements are sampled automatically, without manually enumerating layouts. Figure 7 illustrates collection examples for the task in Section 4.3 with distractor, lighting, and texture randomization enabled.

Since geometric plausibility does not guarantee task completion, we incorporate execution feedback into grouped collection. The system replans and executes each randomized scene, counting only trajectories that pass task verification and are successfully saved toward the success quota. Once a small group reaches its quota, a new layout is sampled under the same asset composition until the big group contains enough complete small groups. Failures trigger progressively broader resampling: an individual failure preserves the object layout while episode conditions are resampled and execution is replanned; consecutive episode failures reaching a threshold trigger layout replacement; and consecutive small-group failures or exhaustion of the candidate big group's budget trigger asset-composition replacement. Rejected candidates do not count toward the target number of completed groups, and the system continues sampling new combinations to meet the collection target.

Table 8: **Task Prompts and Scene Assignments.** We list the task instruction, generated scene, and success condition for ten fixed-base and six mobile manipulation tasks, grouped into easy and long-horizon tasks.

Task	Scene	Task prompt	Completion condition
Fixed-base manipulation / *Easy*			
Open Drawer	Study with Drawer	Pull the drawer fully open.	The drawer reaches its full-opening threshold.
Cup to Tray	Kitchen with Cups	Pick the green cup and place it on the tray.	The correct cup is released in the valid tray region and supported.
Click Button	Bedroom with Button	Press the blue circular button.	Robot interaction activates the correct button.
Basket to Tray	Living Room with Basket	Pick up the basket and place it on the tray.	The basket is released in the valid tray region and supported.
Open Cabinet	Kitchen Counter with Cabinet	Open the cabinet door.	The cabinet door reaches its hinge-opening threshold.
Fixed-base manipulation / *Long horizon*			
Cup Pairing Recall	Cup Pairing Workspace	Remember cup–block pairs; move all three cups to the tray, then restore them.	All cups visit the tray; original pairings and relative positions are restored.
Block Order Recall	Block Order Workspace	Observe and cover the reference row; reconstruct its order with matching work blocks.	The covered reference's left-to-right order is reproduced in the work region.
Cube Count Recall	Cube Counting Workspace	Count the tray's cubes, cover the tray, and transfer matching cubes to the basket.	The basket receives exactly the remembered number of matching cubes.
Button History Recall	Button Sequence Workspace	Press the cube-adjacent button, then another; move the cube to the tray; press the unused button.	The ordered actions are completed; the last button differs from both earlier buttons.
Object Gesture Recall	Bedroom with Pose References	Observe reference poses, cover the reference region, and arrange matching work objects.	Corresponding positions and orientations match within the pose tolerances.
Mobile manipulation / *Easy*			
Open Drawer+Nav	Mobile Cup-Cube Workspace	Navigate to the drawer and pull it fully open.	Required navigation completes and the drawer reaches its full-opening threshold.
Cup to Tray+Nav	Mobile Cup-Cube Workspace	Navigate to the red cup and transfer it to the tray.	Required navigation completes; the red cup is released in the valid tray region.
Lift Cover+Nav	Mobile Block Memory Table	Navigate to the brown cover, grasp its top handle, and lift vertically.	The cover remains held and rises by at least the lift threshold.
Mobile manipulation / *Long horizon*			
Book Stow & Deliver	Mobile Cup-Cube Workspace	Move to the drawer, pull it fully open, place the notebook inside, and close it. Then move to the tray and place the book on it.	The notebook remains in the closed drawer and the book is released on the tray after the required navigation.
Cup Pairing Nav	Mobile Cup-Cube Workspace	Recall cup–block pairs; navigate to move all cups to the tray, then restore them.	All cups visit the tray; navigation completes and the original pairings are restored.
Block Order Nav	Mobile Block Memory Table	Observe the reference order; navigate to the opposite work area and arrange matching blocks.	Matching blocks reproduce the reference's left-to-right order in the work area.

E Robot Evaluation Experiment Details

The data collection statistics for the ten evaluation tasks are summarized in Table 9. The training hyperparameters of the two evaluated models are reported in Table 10. Representative real-time inference processes for all ten tasks are shown in Figure 8.

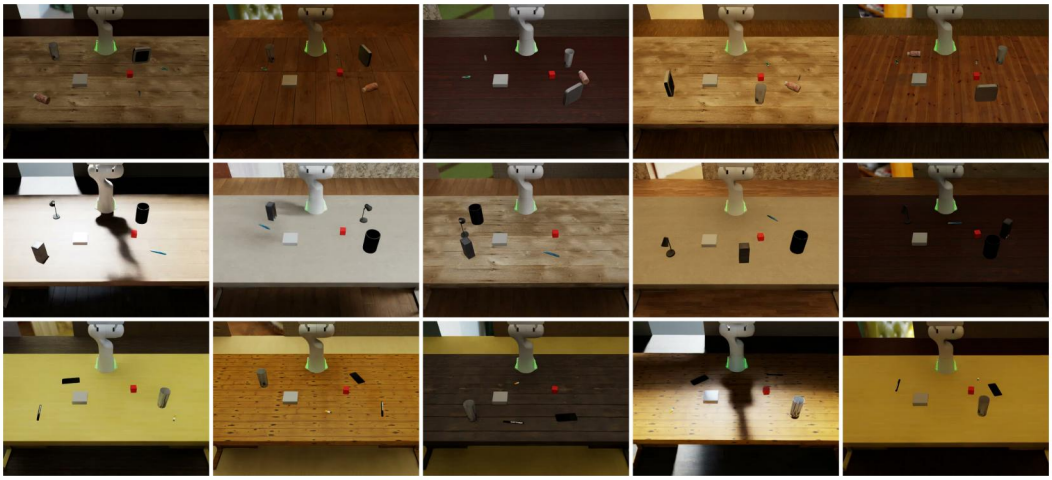


Figure 7: Data of Block stacking with randomization

Table 9: Task statistics for trajectory collection.

Task	**Atomic Actions**	**Avg. Duration (s)**	**Success Rate (%)**
Spatially Guided Placement	3	13.40	88.55
Cuboid Placement	3	14.80	97.32
Block Stacking	4	13.37	97.09
Unstack and Place	6	25.99	85.25
Drawer Opening	3	19.60	79.38
Sequential Object Placement	6	31.98	80.32
Bottle Righting	4	22.74	85.61
Bowl Nesting	4	18.75	70.56
Microwave Door Closing	6	16.64	90.43
Sponge Wiping	5	49.81	75.54

Table 10: Training hyperparameters. Learning-rate schedules use 10k warmup steps for $\pi_{0.5}$ and 5k for FastWAM. FastWAM retains a 100k-step cosine schedule.

Hyperparameters	$\pi_{0.5}$	**FastWAM**
Batch Size (Total)	256	64
Learning Rate	5×10^{-5}	1×10^{-4}
Learning Rate Schedule	Constant	Cosine Decay
Training Steps (Max.)	20k	70k

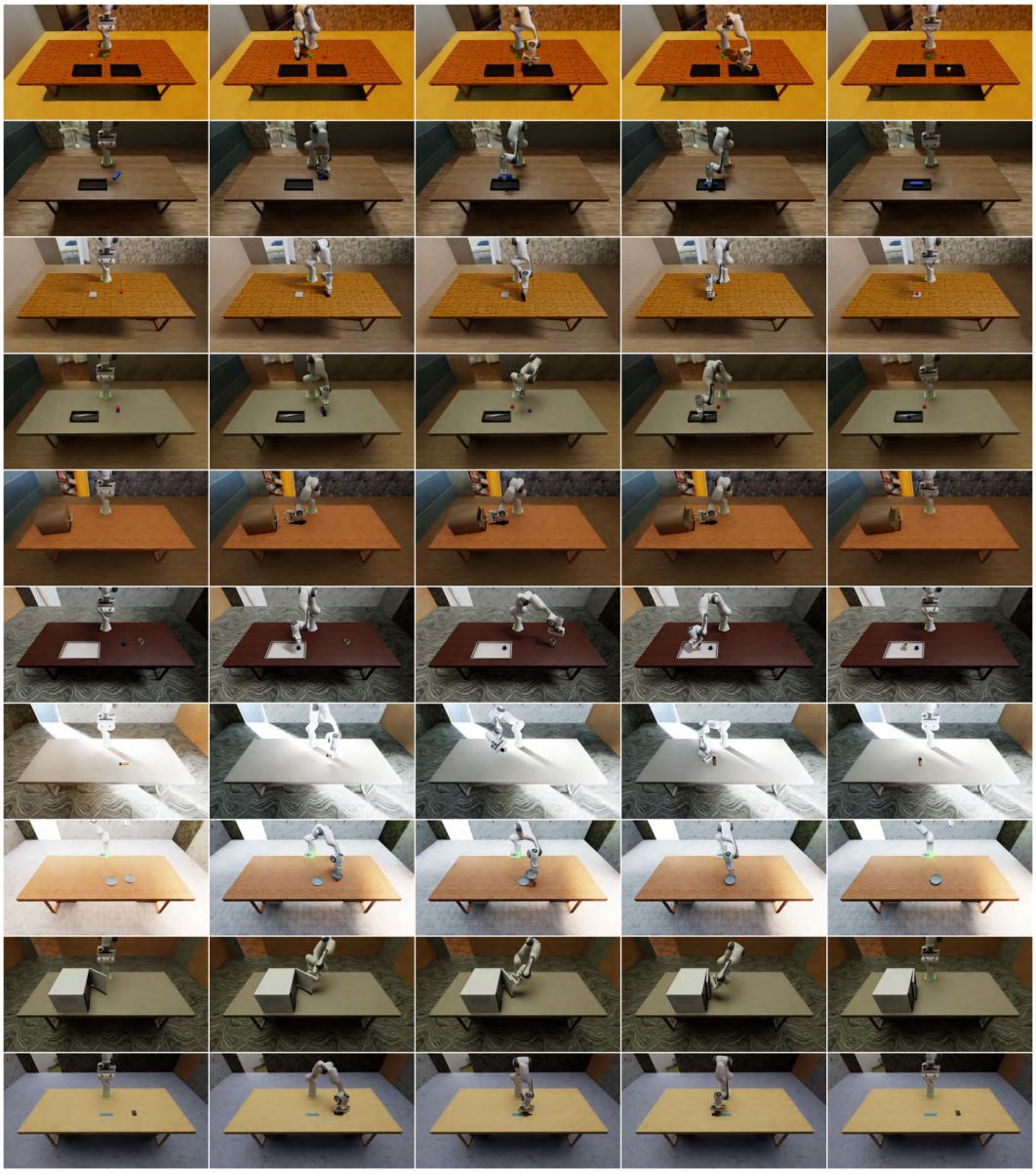


Figure 8: **Representative rollouts of the ten evaluation tasks.** Each row corresponds to one task, while the columns show representative execution stages in temporal order from left to right. For clearer visualization of task execution, tabletop distractors are disabled in these illustrative rollouts.