

RobotAgentBench

Towards Open-world Robot Autonomy

Lingjun Mao^{1,2,†,‡}, Jinglin Cao^{1,2,†,‡}, Lukun He^{1,2,†}, Wenpeng Xu^{1,2}, Yuchen Yan^{1,2}, Hengyu Jin¹, Zijun Zhang¹, Jinzhou Tang^{1,2}, Songyao Jin^{1,2}, Ziche Liu², Yifei Shao¹, Junbo Huang¹, Fang Nan^{1,2}, Ruobing Han^{1,2}, Ziqiao Xi¹, Ziming Xu¹, Shuang Liang^{1,2}, Sibozhu^{1,2}, Wenyi Wu^{1,2}, Xinyue Wang^{1,2}, Yiming Huang^{1,2}, Xinkai Zou^{1,2}, Gao Zhu³, Kun Zhou^{1,*}, Biwei Huang¹

¹ Aether AI, ² University of California San Diego, ³ University of California Davis

[†]Equal contribution; [‡]Co-project leaders; ^{*}Corresponding author

Recent advances in vision-language-action (VLA) and world-action models (WAMs) have significantly improved robotic manipulation, yet deploying robots in the open world demands a higher level of autonomy beyond executing predefined tasks. Robot agents offer a promising paradigm for this transition, integrating task discovery, contextual reasoning, physical execution, and adaptive recovery into a closed interaction loop. However, existing benchmarks largely evaluate isolated capabilities, failing to capture the complexity of open-world robot autonomy. We introduce RobotAgentBench (RAB), a comprehensive benchmark for evaluating robot agents in realistic, open-world household environments. Through an agent-assisted, human-in-the-loop real-to-sim pipeline, RAB constructs 50 diverse household layouts with interactive objects and physically grounded tasks, encompassing 90 atomic tasks, 18 short-horizon tasks, 8 long-horizon tasks, and 4 open-ended scenarios. RAB provides a multi-level evaluation of robot agent capabilities: (1) task-level assessments systematically examine five complementary dimensions—task discovery, context reasoning, manipulation, failure recovery, and robustness; (2) scenario-level evaluation assesses how agents integrate these capabilities to autonomously fulfill high-level user needs. Furthermore, RAB enables systematic comparisons of standalone VLA/WAM models and diverse robot agent architectures under a unified planning–execution–verification framework. Extensive experiments reveal substantial gaps between manipulation and autonomy, demonstrate the advantages of agentic architectures and recursive self-improvement, and highlight the remaining challenges toward open-world robot autonomy.

Keywords: Robot Agents, Embodied AI, Robot Autonomy, Open-Ended Evaluation

Date: October 27, 2026

Correspondence: Kun Zhou (franciskunzhou@gmail.com)

Code: <https://github.com/AetherLabsAI/RobotAgentBench>

Model: <https://huggingface.co/AetherLabs-AI/InternVLA-A1.5-RAB>

Project: <https://aetherlabsai.github.io/RobotAgentBench/>

1 Introduction

Embodied AI has advanced rapidly in recent years, with vision-language-action (VLA) and world-action models (WAMs) demonstrating increasingly strong manipulation capabilities across diverse tasks and environments [5, 19, 40, 48, 50, 52, 55]. These advances raise a fundamental question: *How*

far are we from deploying robots in the open world? Unlike predefined manipulation tasks, the real world involves numerous mundane yet complex challenges, often characterized by ambiguity, uncertainty, and constantly changing circumstances [13, 28, 37]. Consider a household robot preparing for guests: it must identify what needs to be cleaned, select appropriate food based on user preferences, coordinate multiple activities, and recover when manipulation attempts fail [32]. Such challenges are not exceptional, but intrinsic to everyday physical environments. Addressing them is essential for transforming robots into broadly useful assistants in human environments.

Operating in the open world demands a transition *from robot manipulation to robot autonomy*. Beyond executing explicit instructions, robots must autonomously discover relevant tasks, reason about environmental and user context, plan physical interactions, and adapt to unexpected outcomes. Robot agents provide a natural framework for this transition, integrating high-level reasoning and planning with physical execution and feedback-driven verification into a closed loop [2, 17, 41, 46]. Such agents must determine not only how to execute, but also what to do and how to adapt. Meanwhile, advances in multimodal foundation models, including GPT-6-Astra, and emerging agentic embodied systems [4, 8, 47, 54] are creating new opportunities to realize this integration. Together, these developments suggest that embodied AI is entering a new stage, shifting from standalone manipulation models toward autonomous robot agents capable of handling complex, dynamic physical environments.

However, existing benchmarks remain largely designed for robot manipulation rather than autonomy. Widely adopted benchmarks [6, 14, 21, 27, 33, 34, 56] primarily evaluate predefined tasks with explicit goals, overlooking autonomous task discovery and context-dependent decision-making. Recent efforts extend evaluation toward reasoning [7] and long-horizon planning [23, 31, 42, 43, 53], but often rely on stylized problems or abstracted physical interactions. More importantly, reasoning, planning, and manipulation are frequently evaluated in isolation, without capturing their interdependence during real-world operation. A robot capable of individual tasks may still struggle to determine what needs to be done, recover from failures, or adapt to human interventions. Evaluating open-world robot autonomy therefore requires moving beyond isolated capabilities toward a comprehensive assessment of the entire autonomous interaction loop.

To bridge this gap, we introduce RobotAgentBench (RAB), a benchmark for evaluating open-world robot autonomy in realistic household environments. We develop an agent-assisted, human-in-the-loop real-to-sim pipeline to reconstruct real-world scenes and expand them into 50 diverse layouts with interactive objects and physically grounded tasks. At its core, RAB provides 90 atomic tasks that can be flexibly composed into a wide range of household tasks, from which we construct 18 short-horizon and 8 long-horizon manipulation tasks as the foundation of our evaluation. Building upon these tasks, we introduce diverse variants and evaluation settings to systematically assess five complementary dimensions of robot autonomy: (1) task discovery, identifying what needs to be done; (2) context reasoning, interpreting implicit requirements; (3) manipulation, executing physical actions; (4) failure recovery, detecting and correcting execution errors; and (5) robustness, adapting to environmental disturbances. Beyond these task-level assessments, we introduce four open-ended household scenarios that require robot agents to integrate these capabilities to autonomously fulfill high-level user needs, enabling an end-to-end evaluation of open-world robot autonomy.

Using RobotAgentBench, we systematically evaluate standalone VLA/WAM models and diverse robot agent architectures, revealing five key findings:

- Standalone policies struggle with task discovery and contextual reasoning despite their manipu-

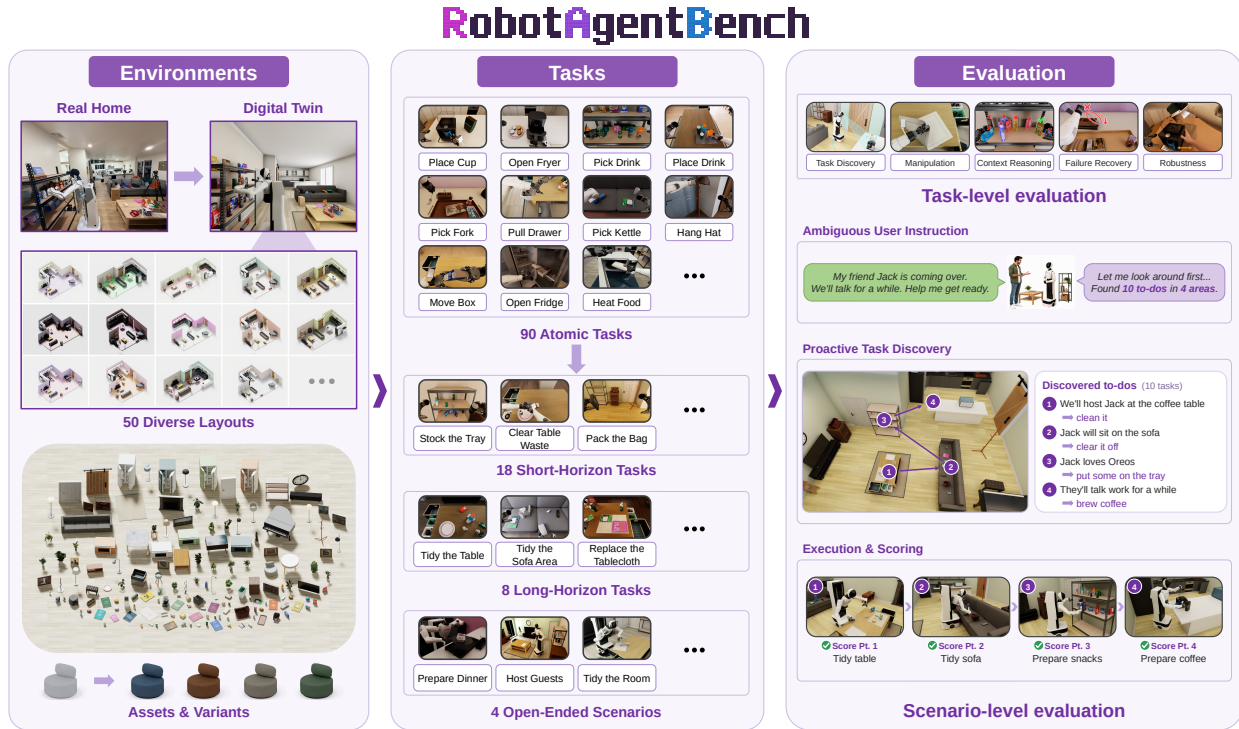


Figure 1. Overview of RobotAgentBench.

lation capabilities, highlighting the gap between manipulation and autonomy;

- Coding-agent harnesses substantially improve performance, particularly through hybrid code-based and learned-policy execution;
- Both planner capabilities and executor instruction-following abilities critically influence agent performance;
- Harness-level recursive self-improvement (RSI) progressively improves task- and scenario-level performance through reusable tool refinement and experience accumulation;
- Real-world evaluations reveal performance degradation but broadly consistent relative rankings, supporting the benchmark’s relevance to physical deployment.

Together, these findings highlight the promise and remaining challenges of advancing from robot manipulation toward open-world robot autonomy.

2 RobotAgentBench

We introduce *RobotAgentBench*, a benchmark designed to evaluate open-world robot autonomy beyond predefined manipulation tasks. We leverage a real-to-sim pipeline to construct diverse environments and collect data, then build the multi-level evaluation suite for systematic comparisons of diverse agent framework designs. Figure 1 illustrates the overview of our benchmark.

2.1 Real-to-Sim Pipeline for Environment and Data Construction

To enable realistic and scalable evaluation, we develop an agent-assisted, human-in-the-loop real-to-sim pipeline that starts from a physical room in a real home, reconstructs it in simulation,

expands the reconstructed scene into diverse interactive configurations, and collects demonstrations for training and evaluation.

2.1.1 Environment Construction

Starting from a physical room in a real home, we first reconstruct it in Isaac Sim and then expand it into diverse household environments. We subsequently collect real-robot demonstrations and reproduce them in simulation to generate trajectories.

Household Scene Reconstruction. We reconstruct the physical household environment in Isaac Sim through an iterative pipeline comprising real-world parameter collection, similar asset retrieval, and new asset reconstruction. First, we collect photographs and measurements of the room layout, furniture arrangements, and household objects to establish real-world references. Next, a coding agent retrieves similar 3D assets from online repositories and adjusts their geometry, scale, and appearance to match these references. For objects without suitable existing assets, we leverage the agent to directly reconstruct their geometry and textures from photographs. The resulting assets are assembled into the simulated environment with appropriate collision geometry and physical properties to support realistic interactions. Throughout this iterative process, human reviewers continuously validate visual fidelity and physical interaction consistency against the real-world environment, providing feedback to the agent for refining the reconstructed scene.

Environment Augmentation. Building upon the reconstructed home, we systematically diversify its spatial configurations and visual appearances. We generate 50 distinct layouts by modifying floor plans (*e.g.*, T-shaped and L-shaped configurations), rearranging functional areas, and varying furniture positions and orientations. Each layout is manually reviewed to eliminate geometric inconsistencies and implausible arrangements. We further increase environmental diversity through variations in furniture colors, textures, materials, and interior design styles.

2.1.2 Demonstration Collection

We follow a three-stage workflow comprising real-world demonstration collection, simulation alignment, and agent-assisted trajectory expansion to generate scalable, physically grounded data. First, we adopt the AgiBot G1 as the robot embodiment, matching its camera configurations and joint settings between simulation and the physical platform, and collect teleoperated demonstrations in the real home. Second, we replay these demonstrations in simulation, iteratively refining asset geometry and physical parameters to align interaction outcomes with the real world. Finally, building upon these reference trajectories, we employ a coding agent to generate additional successful trajectories across diverse object positions, environment layouts, and initial robot configurations. With access to the full simulator state, including object poses, the agent synthesizes and executes action programs to accomplish target tasks. For challenging interactions, we provide manually implemented control functions that the agent can invoke and compose. This workflow enables scalable trajectory generation while maintaining consistency with real-world demonstrations. Detailed observation and action spaces are provided in Appendix A.

2.2 Multi-level Evaluation of Robot Autonomy

Open-world robot autonomy requires robots to determine what needs to be done, execute physical tasks, and adapt to unexpected outcomes. To systematically evaluate these capabilities, RAB adopts a

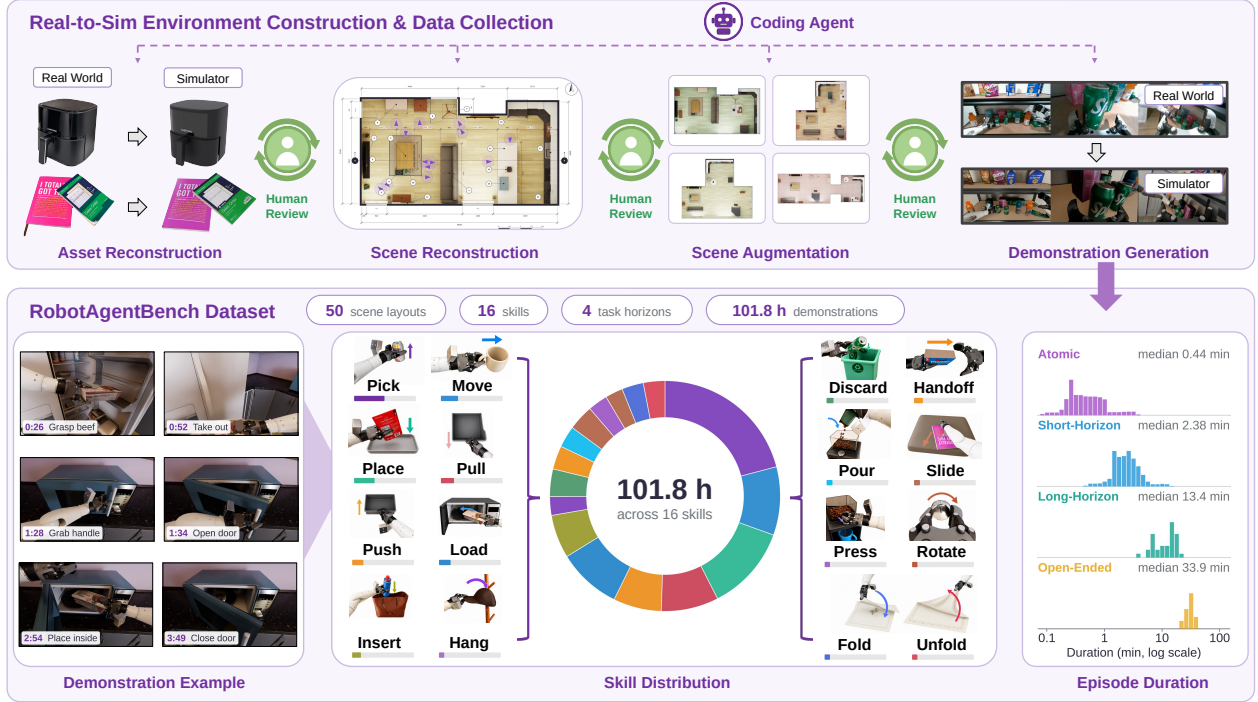


Figure 2. Real-to-sim pipeline and training data statistics for RobotAgentBench.

multi-level evaluation suite built upon a composable household task foundation. At the task level, we introduce dedicated evaluation settings to diagnose five complementary capabilities: task discovery, context reasoning, manipulation, failure recovery, and robustness. At the scenario level, we assess whether robot agents can integrate these capabilities to autonomously fulfill high-level user needs in open-ended household environments. Together, these two levels enable both fine-grained capability assessment and end-to-end evaluation of robot autonomy.

2.2.1 Composable Household Task Suite

We construct a foundational task suite comprising 90 atomic tasks involving everyday household objects and appliances. These tasks can be flexibly composed into diverse household tasks, from which we construct 18 representative short-horizon tasks and 8 long-horizon composite tasks. The suite covers common activities, such as retrieving food from a refrigerator, heating meals in a microwave, and using an air fryer. These tasks involve diverse manipulation skills, including pushing, pulling, bimanual coordination, and mobile manipulation, across rigid objects, deformable materials (*e.g.*, tablecloths), and granular materials (*e.g.*, coffee beans). Short-horizon tasks combine a few atomic tasks, such as picking up a hat from a sofa and hanging it on a coat rack, while long-horizon tasks involve multiple subgoals, such as tidying an entire sofa area. Building upon this task suite, we construct diverse evaluation settings by varying task instructions, contextual requirements, initial states, and environmental disturbances. These variants enable targeted assessments of different autonomy capabilities without requiring entirely new manipulation tasks. We further compose these tasks into open-ended household scenarios, where the tasks to be performed depend on the environment and user context. For each atomic task and short-horizon task, we collect 50 training trajectories; for each long-horizon task and open-ended scenario, we provide 10 complete demonstrations. In total, the

dataset comprises 101.8 hours of training data. Further details are provided in Figure 2.

2.2.2 Task-level Capability Evaluation

We design five complementary evaluation settings to examine how robots determine what to do, execute physical actions, and adapt to unexpected outcomes. Each setting focuses on a specific capability through controlled task configurations. We use capability-specific metrics: target-object coverage for task discovery, target-selection accuracy for context reasoning, and task progress scores for manipulation, failure recovery, and robustness.

Task Discovery. We evaluate whether robots can autonomously identify relevant tasks without explicit task instructions. We configure household scenes with task-relevant objects and provide scenario descriptions and user-specific context. The robot must explore the environment, interpret its observations, and infer which objects require attention. To reduce manipulation demands, the robot indicates inferred targets by touching the relevant objects with either gripper, without completing the underlying household tasks. We measure the fraction of target objects touched under a limited contact budget.

Context Reasoning. We assess whether robots can infer appropriate actions from implicit requirements rather than explicit object instructions. We construct four task settings involving drink selection, snack selection, waste selection, and sofa object selection. Given contextual information such as user preferences, intended uses, or classification rules, the robot must identify the appropriate target object. For example, waste selection requires identifying the object that belongs to the requested category under the supplied classification rules. We evaluate reasoning through contact with the correct target object, without requiring successful grasping or manipulation.

Manipulation. We evaluate physical execution through the 18 short-horizon and 8 long-horizon tasks in the foundational task suite. Given explicit task instructions, robots must execute the required physical interactions, assessing their instruction-following and manipulation capabilities across diverse objects, skills, and task horizons.

Failure Recovery. We evaluate whether robots can recognize and recover from execution failures. We construct five failure variants of the foundational tasks, including missed grasps and objects accidentally knocked over during manipulation. Starting from these failure states, robots must inspect the environment, identify the failure, and execute corrective actions to resume and complete the specified manipulation objective.

Robustness. We assess whether robots can adapt to unexpected environmental changes during execution. We construct six task variants involving human interventions and lighting changes, such as moving a coffee machine while the robot is pouring beans into it. Robots must respond to these disturbances and adjust their actions to complete the specified manipulation objective.

2.2.3 Scenario-level Integrated Evaluation

Beyond individual capabilities, open-world robot autonomy requires their integration into a closed loop of task discovery, reasoning, planning, execution, and adaptation. We therefore introduce four open-ended household scenarios: *preparing dinner*, *hosting guests*, *preparing to leave*, and *tidying the room*. Each scenario involves context-dependent combinations of tasks, with specific requirements

determined by the sampled environment and user information. Unlike task-level evaluation, robots do not receive explicit task lists. Instead, they must autonomously explore the environment, identify relevant tasks, interpret user needs, and plan their execution. This includes reasoning about task dependencies, such as retrieving food from the refrigerator before heating it in the microwave, and coordinating multiple physical interactions toward broader objectives.

We evaluate end-to-end autonomy using the final progress score aggregated over all task checkpoints in each scenario. To further analyze policy and agent behavior, we introduce four detailed metrics informed by the task-level capability dimensions: *Task Coverage*, the fraction of scenario tasks with recorded task-relevant contact or native operation-start evidence; *Task Progress*, the total process score earned across these engaged tasks divided by their total maximum process score; *Recovery Rate*, the fraction of observable physical-failure segments followed by success of the same task; and *Stagnation*, the fraction of execution time spent in repetitive, unproductive attempts or states with no task progress. Appendix D gives the operational definitions and aggregation rules.

2.3 Robot Agent Framework for Evaluation

To systematically investigate the transition from standalone manipulation models to autonomous robot agents, RobotAgentBench supports the evaluation of both standalone VLA/WAM policies and diverse robot agent architectures. We introduce a unified planning-execution-verification framework for robot agents: (1) a *planner* that interprets environmental context and user requirements, decomposes tasks, and adapts plans based on execution feedback; (2) an *executor* that translates instructions into physical actions through learned policy models (e.g., VLA and WAM) or code-based control; and (3) a *verifier* that monitors execution outcomes, identifies failures, and guides corrective actions. By flexibly combining these components, we construct five representative agent architectures to systematically examine the roles of planning, execution, verification, and recursive self-improvement in robot autonomy. For clarity, we denote each architecture by its planner/executor/verifier (P/E/V) configuration, where C represents code-based control, L denotes a learned VLA/WAM policy, H denotes hybrid code-based and learned execution, and – indicates an absent component.

VLA/WAM-based Policy + Code-based Verifier (–/L/C). A VLA/WAM policy directly generates actions from observations and task instructions. A coding-agent verifier monitors execution feedback, selectively executes action prefixes, and introduces local corrections before querying the policy again [46].

Code-based Planner + VLA/WAM-based Policy (C/L/–). A coding-agent planner interprets task requirements, decomposes them into concrete subtask instructions, and delegates physical execution to a VLA/WAM policy. The planner dynamically revises its instructions based on execution feedback [2, 17, 41].

Code-based Planner + Code-based Policy + Code-based Verifier (C/C/C). A coding agent jointly handles planning, execution, and verification by generating executable programs from predefined control functions (e.g., `move_eef_to()`). It composes these functions to accomplish tasks and verifies execution outcomes to guide subsequent actions [16, 18, 26].

Code-based Planner + Hybrid Code/VLA/WAM-based Policy + Code-based Verifier (C/H/C). A coding agent serves as both planner and verifier, while the executor integrates code-based control and

learned VLA/WAM policies as unified callable tools. The agent dynamically selects and combines these tools according to task requirements and environmental feedback, leveraging their complementary capabilities.

Code-based Planner + Code-based Policy + Code-based Verifier + RSI (C/C/C + RSI). Building upon the fully code-based architecture, we introduce harness-level recursive self-improvement (RSI), enabling the coding agent to iteratively create, refine, and reuse control tools based on execution feedback. Successful improvements are retained across subsequent attempts, supporting progressive adaptation and experience accumulation [1].

3 Experiments

We conduct systematic experiments at both the task and scenario levels to evaluate standalone VLA/WAM-based policies and complete robot agents. Building on the unified robot agent framework we developed, we compare different harness designs and examine their effects on task performance. We further evaluate agents in the corresponding real-world environment to investigate performance gaps between simulation and reality.

3.1 Experimental Setup

Evaluated Models and Agents. For standalone policy baselines, we evaluate VLA models and WAMs. The VLA baselines include $\pi_{0.5}$ [19], Xiaomi Robotics-1 [48], InternVLA-A1.5 [30], MolmoAct2 [10], LingBot-VLA2 [51], HY-VLA [57], Galaxea G0.5 [29], and X-VLA [61]. The WAM baselines include τ_0 (low-level policy) [62], OpenWAM-Alpha [49], and LingBot-VA [25]. For complete robot agents, we compare five representative harness designs, using Codex powered by GPT-6 Astra [35] as the coding agent throughout and InternVLA-A1.5 as the learned policy wherever applicable. For the fully code-based design, we additionally evaluate Claude Code powered by Opus 5.5 [3] and Antigravity CLI powered by Gemini 4 Argon [22] to compare different coding agents under the same harness. For the RSI setting, we run ten iterations, evaluating 30 instances per task or scenario in each iteration, and report performance in the final iteration.

Evaluation Protocol. We evaluate all methods across the five task-level assessments and four open-ended scenarios. For each task or scenario, we evaluate 30 distinct instances across the full benchmark suite. Evaluation covers all four open-ended scenario families and the 50-layout catalog, with every applicable randomization axis enabled over its full benchmark-defined range: layout, appearance, lighting, scene and physical distractors, and robot-arm, mobile-base, and interaction-object initial states. The resulting comparisons measure performance across randomized instances within the benchmark scene, task, and context families. All methods interact with the same environment and 22-dimensional command interface; model-specific observation adapters are described in Appendix C. For open-ended scenarios, episodes terminate if the environment’s evaluator records no increase in task progress for 120 consecutive seconds of simulation time. Coding agents are limited to 200 model responses per episode, and each tool call executes at most 600 control steps, corresponding to 30 seconds of simulation time. We report the mean score under the corresponding evaluation metric for each task or scenario. Comparisons and rank orders refer to these observed instance means under the stated protocol.

Observation and Action Spaces. We adopt observation and action spaces consistent with the physical AgiBot G1 robot. All systems have access to the same observation sources: RGB images from one head-mounted and two wrist-mounted cameras, proprioceptive states, task instructions or contextual prompts, and a progressively revealed navigation map derived from static scene geometry under shared disclosure rules. Model inputs follow their native adapters; LingBot-VA conditions on RGB and language without proprioceptive state. The 22-dimensional action space controls the head, torso, both arms and grippers, and mobile base. Detailed specifications are provided in Appendix A.

3.2 Main Results

Table 1 summarizes the performance of standalone policy models and robot agents under both task-level and scenario-level evaluation.

Table 1. Performance on RobotAgentBench. Discovery, Reasoning, and Recovery denote Task Discovery, Context Reasoning, and Failure Recovery, respectively. Average is the mean of the five task-level scores. Gold, silver, and bronze circles mark the top three scores in Average and Open-Ended, ranked separately for standalone models and robot agents. Agent architectures follow the P/E/V notation defined in Section 2.3, where C denotes a coding agent, L a learned VLA/WAM-based policy, and – an absent component.

		Task-Level						Scenario-Level
Setting	Model / Agent	Manipulation	Discovery	Reasoning	Recovery	Robustness	Average	Open-Ended
Standalone VLA/WAM-based Policies								
VLA	$\pi_{0.5}$	13.1	1.2	16.7	20.3	8.1	11.9 3	1.1
	Xiaomi Robotics-1	18.1	0.5	15.8	5.8	21.6	12.4 2	0.4
	InternVLA-A1.5	11.3	0.7	10.8	23.2	44.2	18.0 1	1.0
	MolmoAct2	4.3	1.7	5.0	2.7	2.4	3.2	1.9 2
	LingBot-VLA2	6.5	0.3	1.7	6.6	10.2	5.1	0.5
	HY-VLA	10.6	1.0	15.0	3.3	7.6	7.5	0.8
	Galaxea G0.5	11.0	1.5	25.0	9.3	8.1	11.0	1.3 3
	X-VLA	9.9	0.4	5.0	6.8	7.6	5.9	0.3
WAM	τ_0 (low-level)	7.7	0.6	1.7	3.5	7.8	4.3	0.5
	OpenWAM-Alpha	13.4	1.2	15.0	4.6	8.6	8.6	2.0 1
	LingBot-VA	5.0	1.5	3.3	5.4	4.2	3.9	0.4
Robot Agents								
C/C/C	Codex	32.4	35.2	38.3	22.0	27.1	31.0	8.4
	Claude Code	20.6	24.5	60.0	44.2	17.1	33.3	9.2
	Antigravity CLI	29.2	36.5	80.0	27.8	25.0	39.7 3	9.8 3
–/L/C	Codex + InternVLA-A1.5	18.9	4.3	23.3	24.6	45.4	23.3	3.7
C/L/–	Codex + InternVLA-A1.5	16.5	4.0	40.0	26.8	44.8	26.4	9.1
C/H/C	Codex + InternVLA-A1.5	42.5	37.2	73.3	38.6	50.8	48.5 1	15.4 1
C/C/C + RSI	Codex	36.8	33.3	63.3	31.4	46.5	42.3 2	10.8 2

3.2.1 Task-Level Evaluation

We compare standalone VLA/WAM-based policies and complete robot agent systems across the five capability dimensions. Standalone policies generally underperform integrated agent systems,

Table 2. Detailed evaluation on open-ended scenarios. Coverage, Progress, and Recovery denote Task Coverage, Task Progress, and Recovery Rate, respectively. $\uparrow/\downarrow$ indicate higher/lower is better.

Setting	Model / Agent	Scenario Progress				Behavioral Metrics			
		Guests $\uparrow$	Dinner $\uparrow$	Leaving $\uparrow$	Tidying $\uparrow$	Coverage $\uparrow$	Progress $\uparrow$	Recovery $\uparrow$	Stagnation $\downarrow$
VLA	InternVLA-A1.5	0.3	1.8	1.2	0.7	2.3	4.5	0.6	88.3
WAM	OpenWAM-Alpha	0.9	3.4	2.1	1.6	3.1	2.7	0.2	84.6
C/C/C	Codex	11.2	3.6	1.4	17.4	13.4	15.2	15.4	33.7
C/C/C	Claude Code	6.4	12.8	8.1	9.5	12.8	16.2	18.8	18.7
C/C/C	Antigravity CLI	6.0	13.8	13.4	6.0	15.8	17.0	37.5	18.4
C/C/C + RSI	Codex	13.1	6.4	3.0	20.7	20.3	19.6	38.9	12.3
C/H/C	Codex + InternVLA-A1.5	15.8	14.3	13.9	17.6	15.2	25.1	30.7	12.1

highlighting a substantial gap between their current capabilities and the demands of real-world autonomy.

Standalone Policy Results. Standalone policies appear to primarily imitate demonstrated manipulation behaviors, with limited generalization to situations beyond the training distribution, such as recovering from execution failures. They struggle particularly with task discovery, which requires purposeful exploration rather than execution of a specified task. When initialized away from viewpoints commonly encountered in the demonstrations, these models often produce erratic movements instead of navigating to gather relevant observations. They also struggle with contextual reasoning, such as selecting a beverage that matches user preferences, with performance generally close to random guessing. Among the evaluated standalone models, InternVLA-A1.5 achieves the strongest overall performance, with notably higher robustness and failure recovery scores than other models, even surpassing some complete robot agents.

Robot Agent Results. Introducing coding agents substantially improves robot performance. Among fully code-based systems, Antigravity CLI has the highest reported mean score, followed by Claude Code. For Codex, RSI further improves performance through tool refinement and experience accumulation across task iterations. We also compare three ways of integrating coding agents with learned policies. Refining policy-generated actions (-/L/C) yields modest gains but remains constrained by the underlying policy’s capabilities. Using a coding agent as a planner (C/L/-) improves manipulation and robustness, with the largest gains in context reasoning, as the planner translates contextual information into explicit instructions for the executor. The best performance comes from combining code-based control with selectively invoked learned policies (C/H/C). For example, code-based control positions the gripper near an object, providing a favorable starting state for the policy to complete the grasp.

3.2.2 Scenario-Level Evaluation

Both standalone policies and robot agents perform poorly in open-ended scenarios. Table 2 provides a more detailed behavioral analysis. Due to their limited task discovery capabilities, VLA/WAM-based policies rarely explore their surroundings in response to contextual information. They spend most of their time idle or making random, unproductive movements, with occasional progress primarily when initialized near a relevant interaction site. Complete robot agents identify tasks more actively, and their planners often generate appropriate task instructions. However, frequent execution errors leave

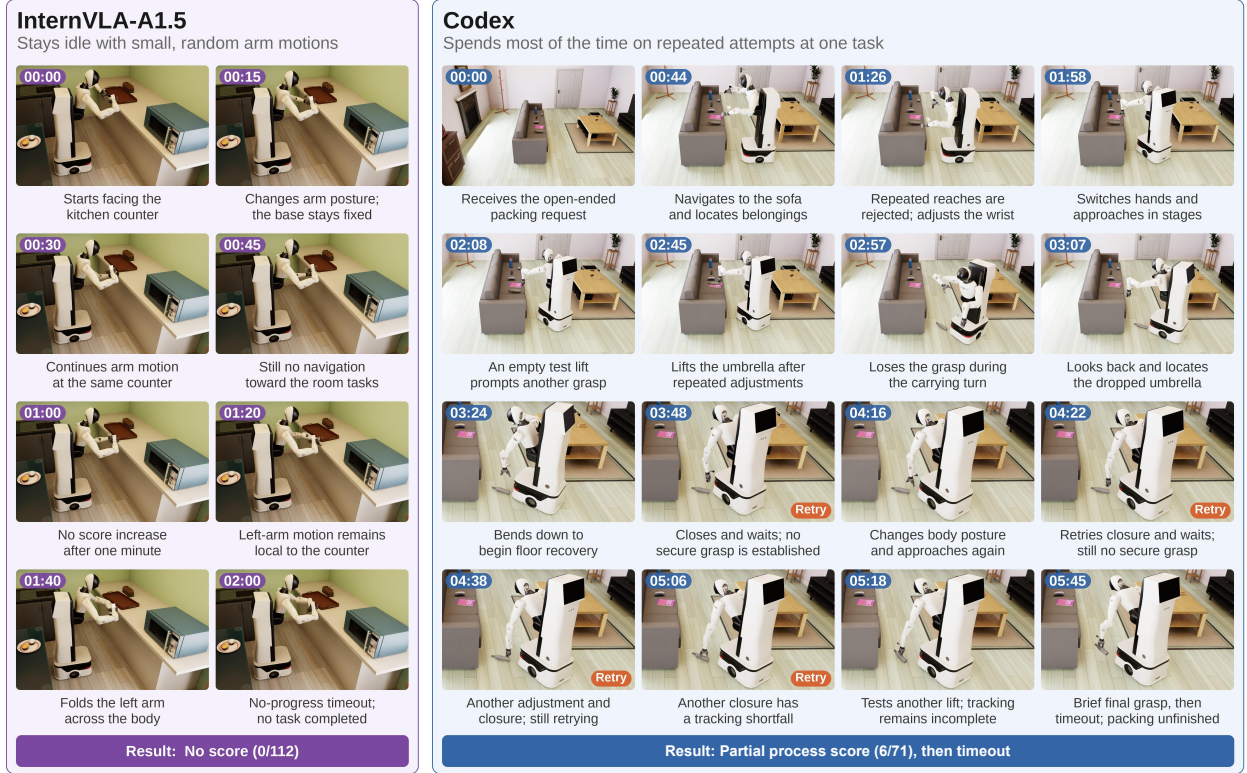


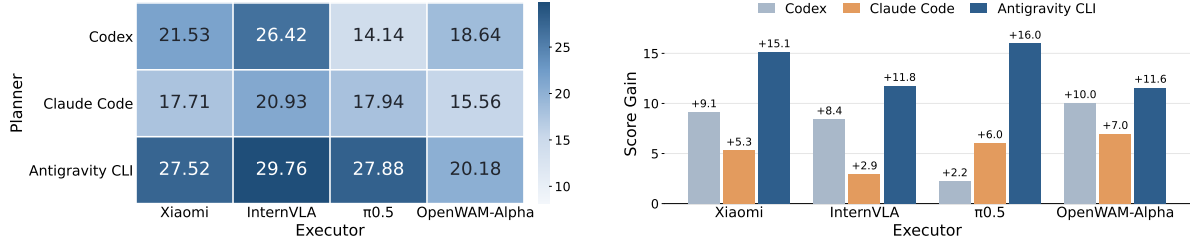
Figure 3. Open-ended scenario case studies. InternVLA-A1.5 (left) fails to initiate any task, while Codex (right) repeatedly attempts a single subtask without completing it before the episode times out.

them stuck on individual tasks, repeatedly retrying and adjusting their actions. Figure 3 illustrates these behaviors through representative case studies.

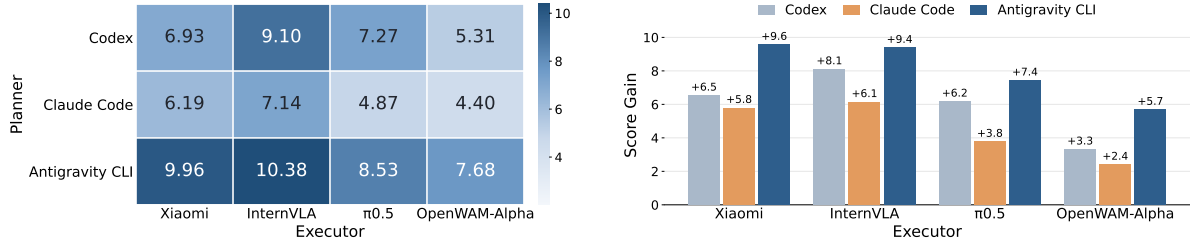
3.3 Agent Harness Analysis

Building on our evaluation of robot agent systems, we further investigate how different harness designs affect robot performance, including which coding agents are most effective as planners and which policy models integrate best with coding-agent harnesses. We also examine how agent performance evolves over multiple rounds of recursive self-improvement.

Which Coding Agent Makes a Better Planner? We select the three best-performing VLA models, Xiaomi Robotics-1, InternVLA-A1.5, and $\pi_{0.5}$, along with the best-performing WAM, OpenWAM-Alpha, as executors and pair each with different coding-agent planners. As shown in Figure 4, Antigravity CLI yields the largest performance gains across the evaluated policy executors. To investigate these differences, we randomly sample 10 open-ended episodes per planner and systematically compare their instruction styles in Figure 5. Antigravity CLI produces shorter instructions and switches them less frequently. Its instructions also contain fewer execution-specific details, such as grasping strategies or hand selection, and use simpler vocabulary. In contrast, Codex often specifies detailed manipulation procedures, including which hand to use. These observations suggest that concise, high-level instructions may be easier for VLA/WAM-based executors to follow, potentially because they better match the language encountered during training. Longer, more detailed instructions may instead introduce complexity that these policies struggle to interpret.



(a) Task-level performance across different planner-executor combinations.



(b) Scenario-level performance across different planner-executor combinations.

Figure 4. Performance comparison of planner-executor combinations on RobotAgentBench at the task level and in open-ended scenarios.

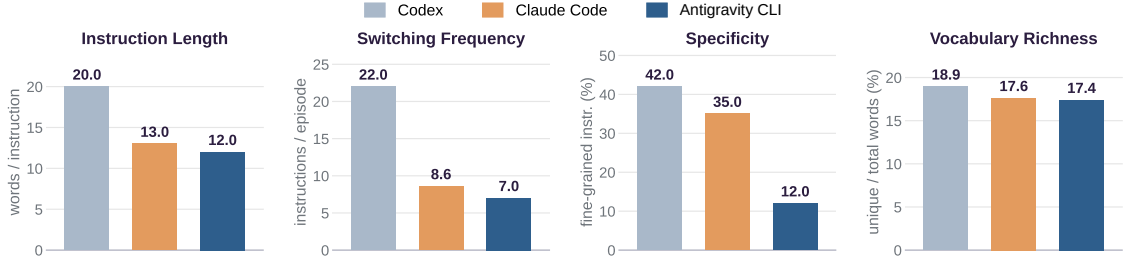


Figure 5. Planner instruction characteristics across three coding agents, based on 10 randomly sampled open-ended episodes per agent. Instruction length: average words per instruction; switching frequency: instruction changes per episode; specificity: proportion of instructions containing fine-grained execution details; vocabulary richness: ratio of unique words to total words.

Which Policy Model Makes a Better Executor? As shown in Figure 4, InternVLA-A1.5 achieves the best performance when paired with the same planner. This advantage stems in part from its stronger manipulation capabilities, especially robustness and failure recovery, which make it well suited to serving as a policy executor. We also observe that InternVLA-A1.5 follows revised planner instructions more reliably during replanning and error recovery. This behavior may be related to its extensive VQA training, which helps preserve its language understanding capabilities.

Does RSI Improve Robot Agent Performance? We evaluate three fully code-based agents, Codex, Claude Code, and Antigravity CLI, over ten RSI iterations, with 30 evaluation instances per task or scenario in each iteration. Agents can accumulate experience across tasks and refine their tools, implemented as reusable control functions. As shown in Figure 6, all three agents improve at both the task and scenario levels over successive RSI iterations, while the gaps between Codex and the other two agents gradually narrow. In their initial attempts, agents often make basic errors, such as attempting to reach a target pose in a single motion that violates joint limits. Over successive

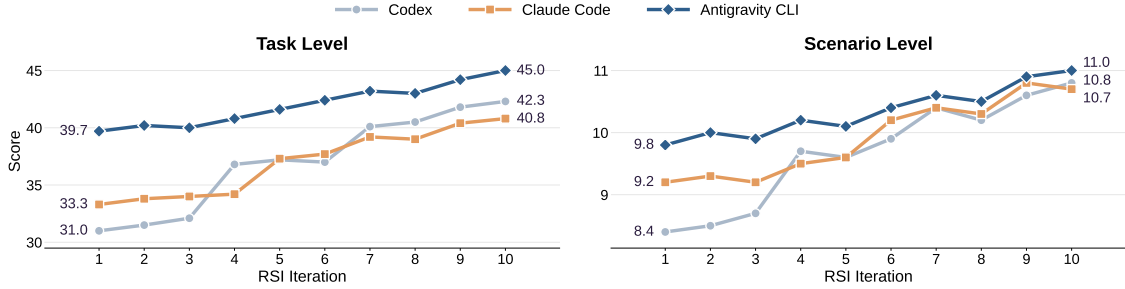


Figure 6. Task-level and scenario-level performance of three coding agents over successive RSI iterations

iterations, they refine their control functions to decompose such motions into smaller steps through feasible intermediate poses.

3.4 Sim–Real Performance Gap

To assess performance differences between simulation and reality, we evaluate a subset of tasks and four representative scenario contexts, one from each open-ended scenario category, in both environments. We match the number of real-world and simulation training demonstrations for these tasks, using 50 demonstrations per atomic action and short-horizon task, and 10 per long-horizon task and open-ended scenario. We evaluate InternVLA-A1.5, OpenWAM-Alpha, and Codex powered by GPT-6 Astra. Each system is evaluated on 30 instances per task or scenario in each environment; the training-demonstration counts above are separate from these evaluation trials. As shown in Table 3, both standalone models and the coding agent generally exhibit lower performance in the real world, reflecting the greater complexity of physical interactions and the increased visual clutter in real environments. The paired results characterize sim–real performance changes for the evaluated systems and matched task contexts. Figure 7 presents representative case studies from the real-world evaluation.

Table 3. Performance comparison between simulation and the real world on matched evaluation subsets. Green upward and red downward arrows indicate increases and decreases relative to simulation, respectively.

Model / Agent	Environment	Task-Level						Scenario-Level
		Manipulation	Discovery	Reasoning	Recovery	Robustness	Average	Open-Ended
InternVLA-A1.5	Sim	11.8	1.2	12.5	20.8	42.7	17.8	1.4
	Real	10.7 ↓	1.7 ↑	10.0 ↓	15.0 ↓	30.0 ↓	13.5 ↓	1.0 ↓
OpenWAM-Alpha	Sim	14.5	0.8	12.5	5.5	9.2	8.5	2.4
	Real	10.0 ↓	0.0 ↓	15.0 ↑	5.3 ↓	10.5 ↑	8.2 ↓	0.0 ↓
Codex	Sim	35.8	38.4	45.8	24.1	26.8	34.2	8.7
	Real	33.0 ↓	33.3 ↓	44.2 ↓	25.0 ↑	24.0 ↓	31.9 ↓	6.0 ↓

Task-level example (robustness) Single task under human perturbations



Scenario-level example (open-ended) Long-horizon household scenario



Figure 7. Real-world evaluation examples with the AgiBot G1.

4 Related Works

Robot Manipulation Benchmarks. Robot manipulation benchmarks evaluate increasingly diverse and complex tasks, yet largely retain predefined goals and task-specific demonstrations. RLBench [21], Meta-World [56], ManiSkill2 [14], and LIBERO [27] assess individual manipulation skills through dedicated tasks, while RoboCasa [34], BEHAVIOR [24, 45], and RoboTwin [6, 33] extend coverage to diverse household activities and environments. VLABench [58] and LIBERO-Long [27] extend evaluation to longer-horizon tasks, providing complete demonstrations that allow policies to learn entire task sequences through imitation. More recent benchmarks, including RoboDojo [7], RoboCerebra [15], and RoboMME [9], assess reasoning, planning, and memory alongside physical execution, but their narrowly specified tasks offer limited coverage of context-dependent household assistance. *RobotAgentBench* evaluates diverse agentic capabilities through task-level assessments and open-ended household scenarios, with training demonstrations primarily teaching reusable skills that agents must select and compose according to the current situation.

Embodied Planning Benchmarks. Beyond robot manipulation benchmarks, some work abstracts physical execution into predefined callable functions, focusing on evaluating the planning and reasoning capabilities of models in simulated embodied activities. ALFRED [42], ALFWorld [43], RealFRED [23], and EmbodiedBench [53] assess household instruction following and planning, while WorldLines [59] emphasizes persistent world state and memory. TEACH [36], DialFRED [11], KnowNo [39], AmbiK [20], and Ask-to-Act [38] further evaluate interaction and clarification under incomplete or ambiguous instructions. However, by abstracting away physical control, these benchmarks provide a limited assessment of how planning and reasoning must adapt to execution outcomes, particularly when recovering from failures in real-world tasks.

Robot Agents Robot agents integrate reasoning and physical execution through different architectures. SayCan [2], Inner Monologue [17], and Hi Robot [41] combine high-level planning by language or vision-language models with learned skills or policies. Code as Policies [26], ProgPrompt [44], Instruct2Act [16], and VoxPoser [18] use generated programs to connect perception, planning, and control. Harness VLA [60] combines learned policies with analytic control primitives within an agentic framework. Recent approaches further enable execution-time correction of policy actions [46] and feedback-driven improvement of tools and control code [1, 12]. *RobotAgentBench* provides a unified benchmark for systematically evaluating these robot agent designs.

5 Conclusion

In this work, we introduce *RobotAgentBench* (RAB), a comprehensive benchmark for evaluating open-world robot autonomy beyond predefined manipulation tasks. Built upon an agent-assisted real-to-sim pipeline, RAB provides multi-level evaluations spanning five fundamental capabilities and open-ended household scenarios, enabling systematic assessment of robot agents from task discovery and contextual reasoning to physical execution and adaptive recovery. Through extensive evaluations of VLA/WAM models and diverse robot agent architectures, we reveal a substantial gap between manipulation capabilities and autonomous task completion. Our findings demonstrate the benefits of coding-agent harnesses, hybrid execution, and recursive self-improvement, while highlighting persistent challenges in planning–execution coordination, failure recovery, and real-world deployment. Despite recent advances, reliable autonomy in open-ended scenarios remains far from solved.

We hope RAB serves as a systematic testbed for understanding these limitations and advancing the transition from robot manipulation toward open-world robot autonomy. Building on RAB, we aim to extend agent-level recursive self-improvement (RSI) into the physical open world. Through autonomous exploration and interaction, robot agents could discover new tasks, learn from failures, and continuously refine their planning and execution strategies. Accumulating and transferring reusable experience across tasks and environments could pave the way toward continually self-improving robot autonomy with minimal human intervention.

References

- [1] 2026. *Physical RSI 1.0: Recursive Self-Harness for Scaling Embodied Skills*. <https://yanming03.github.io/PhysicalRSI/v2/>.
- [2] Ahn, Michael, Brohan, Anthony, Brown, Noah, Chebotar, Yevgen, Cortes, Omar, David, Byron, Finn, Chelsea, Fu, Chuyuan, Gopalakrishnan, Keerthana, Hausman, Karol, *et al.* 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.
- [3] Anthropic. 2026. *Introducing Claude Opus 5.5*. Accessed: 2026-10-24.
- [4] Bjorck, Johan, Li, Zhiqi, Man, Yunze, Wang, Jing, Cheng, An-Chieh, Liu, Sifei, Wang, Shihao, Yu, Zhiding, Badki, Abhishek, Birchfield, Stan, *et al.* 2026. Vesta: A Generalist Embodied Reasoning Model. *arXiv preprint arXiv:2606.20905*.
- [5] Cai, Junhao, Cai, Zetao, Cao, Jiafei, Chen, Yilun, He, Zeyu, Jiang, Lei, Li, Hang, Li, Hengjie, Li, Yang, Liu, Yufei, *et al.* 2026. Internvla-a1: Unifying understanding, generation and action for robotic manipulation. *arXiv preprint arXiv:2601.02456*.
- [6] Chen, Tianxing, Chen, Zanzin, Chen, Baijun, *et al.* 2025. RoboTwin 2.0: A Scalable Data Generator and Benchmark with Strong Domain Randomization for Robust Bimanual Robotic Manipulation. *arXiv preprint arXiv:2506.18088*.
- [7] Chen, Tianxing, Chen, Yue, Li, Zixuan, Tang, Junyuan, Su, Kailun, Lu, Haoran, Wan, Weijie, Chen, Baijun, Liu, Songling, Yan, Haowen, *et al.* 2026a. RoboDojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*.

-
- [8] Chen, Yitong, Huai, Zezheng, Li, Sixian, Wang, Yubang, Zhang, Haozhe, Zhang, Yifei, Chen, Hechang, Gong, Jingjing, Jiang, Yu-Gang, & Qiu, Xipeng. 2026b. ETA: A new agentic paradigm for embodied tasks. *arXiv preprint arXiv:2608.03924*.
 - [9] Dai, Yinpei, Fu, Hongze, Lee, Jayjun, Liu, Yuejiang, Zhang, Haoran, Yang, Jianing, Finn, Chelsea, Fazeli, Nima, & Chai, Joyce. 2026. Robomme: Benchmarking and understanding memory for robotic generalist policies. *arXiv preprint arXiv:2603.04639*.
 - [10] Fang, Haoquan, Duan, Jiafei, Clay, Donovan, Wang, Sam, Liu, Shuo, Huang, Weikai, Fan, Xiang, Tsai, Wei-Chuan, Chen, Shirui, Wang, Yi Ru, *et al.* 2026. Molmoact2: Action reasoning models for real-world deployment. *arXiv preprint arXiv:2605.02881*.
 - [11] Gao, Xiaofeng, Gao, Qiaozi, Gong, Ran, Lin, Kaixiang, Thattai, Govind, & Sukhatme, Gaurav S. 2022. DialFRED: Dialogue-Enabled Agents for Embodied Instruction Following. *IEEE Robotics and Automation Letters*, 7(4), 10049–10056.
 - [12] Ge, Shijia, Zhou, Alex, Zeng, Jianshu, Wan, Yexing, Wu, Di, Zheng, Zelin, Wang, Yazhe, Jia, Zhiqi, Shangguan, Xuan, Zhu, Jay, *et al.* 2026. Make Code as Policy Great Again: Frontier Agents Write, Call, and Evolve Robot Tools. *arXiv preprint arXiv:2609.39018*.
 - [13] Gong, Xicheng, Sun, Guozheng, Xu, Peiran, & Mu, Yadong. 2026. RePlan-Bot: Multi-Level Replanning for Embodied Instruction Following. *arXiv preprint arXiv:2605.25851*.
 - [14] Gu, Jiayuan, Xiang, Fanbo, Li, Xuanlin, Ling, Zhan, Liu, Xiqiang, Mu, Tongzhou, Tang, Yihe, Tao, Stone, Wei, Xinyue, Yao, Yunchao, Yuan, Xiaodi, Xie, Pengwei, Huang, Zhiao, Chen, Rui, & Su, Hao. 2023. ManiSkill2: A Unified Benchmark for Generalizable Manipulation Skills. *In: International Conference on Learning Representations (ICLR)*.
 - [15] Han, Songhao, Qiu, Boxiang, Liao, Yue, *et al.* 2025. RoboCerebra: A Large-scale Benchmark for Long-horizon Robotic Manipulation Evaluation. *In: Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*.
 - [16] Huang, Siyuan, Jiang, Zhengkai, Dong, Hao, Qiao, Yu, Gao, Peng, & Li, Hongsheng. 2023a. Instruct2act: Mapping multi-modality instructions to robotic actions with large language model. *arXiv preprint arXiv:2305.11176*.
 - [17] Huang, Wenlong, Xia, Fei, Xiao, Ted, Chan, Harris, Liang, Jacky, Florence, Pete, Zeng, Andy, Tompson, Jonathan, Mordatch, Igor, Chebotar, Yevgen, *et al.* 2022. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*.
 - [18] Huang, Wenlong, Wang, Chen, Zhang, Ruohan, Li, Yunzhu, Wu, Jiajun, & Fei-Fei, Li. 2023b. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*.
 - [19] Intelligence, Physical, Black, Kevin, Brown, Noah, Darpinian, James, Dhabalia, Karan, Driess, Danny, Esmail, Adnan, Equi, Michael, Finn, Chelsea, Fusai, Niccolo, *et al.* 2025. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*.

-
- [20] Ivanova, Anastasiia, Bakaeva, Eva, Volovikova, Zoya, *et al.* 2025. AmbiK: Dataset of Ambiguous Tasks in Kitchen Environment. *Pages 33216–33241 of: Annual Meeting of the Association for Computational Linguistics (ACL)*.
 - [21] James, Stephen, Ma, Zicong, Arrojo, David Rovick, & Davison, Andrew J. 2020. RL Bench: The Robot Learning Benchmark & Learning Environment. *IEEE Robotics and Automation Letters*, 5(2), 3019–3026.
 - [22] Kavukcuoglu, Koray. 2026. *Gemini 4 Argon: Our Next Era of Frontier Intelligence*. Accessed: 2026-10-24.
 - [23] Kim, Taewoong, Min, Cheolhong, Kim, Byeonghwi, Kim, Jinyeon, Jeung, Wonje, & Choi, Jonghyun. 2024. ReALFRED: An Embodied Instruction Following Benchmark in Photo-Realistic Environments. *In: European Conference on Computer Vision (ECCV)*.
 - [24] Li, Chengshu, Zhang, Ruohan, Wong, Josiah, Gokmen, Cem, Srivastava, Sanjana, Martín-Martín, Roberto, Wang, Chen, Levine, Gabriel, Lingelbach, Michael, Sun, Jiankai, *et al.* 2023. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. *Pages 80–93 of: Conference on Robot Learning*. PMLR.
 - [25] Li, Lin, Zhang, Qihang, Luo, Yiming, Yang, Shuai, Wang, Ruilin, Han, Fei, Yu, Mingrui, Gao, Zelin, Xue, Nan, Zhu, Xing, *et al.* 2026. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*.
 - [26] Liang, Jacky, Huang, Wenlong, Xia, Fei, Xu, Peng, Hausman, Karol, Ichter, Brian, Florence, Pete, & Zeng, Andy. 2023. Code as policies: Language model programs for embodied control. *Pages 9493–9500 of: 2023 IEEE International conference on robotics and automation (ICRA)*. IEEE.
 - [27] Liu, Bo, Zhu, Yifeng, Gao, Chongkai, Feng, Yihao, Liu, Qiang, Zhu, Yuke, & Stone, Peter. 2023. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. *In: Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*.
 - [28] Liu, Xinyi, Sani, Mohammadreza Fani, Zhou, Zewei, Wirbel, Julius, Zarrin, Bahram, & Galeazzi, Roberto. 2025. RoboPilot: Generalizable Dynamic Robotic Manipulation with Dual-thinking Modes. *arXiv preprint arXiv:2510.00154*.
 - [29] Liu, Yicheng, Dong, Zibin, Ye, Baijun, Yuan, Tianyuan, Jiang, Tao, Yang, Anqi, Cao, Shicheng, Liu, Haonan, Sun, Yue, Guo, Zihan, *et al.* 2026. G0. 5: One autoregressive stream for robot reasoning and action. *arXiv preprint arXiv:2608.11739*.
 - [30] Ma, Haoxiang, Cai, Junhao, Xu, Xiaoxu, Li, Hao, Yang, Yuyin, Tian, Yang, Cao, Jiafei, Zhu, Hongrui, Qiu, Zherui, Yang, Yuqiang, *et al.* 2026. InternVLA-A1. 5: Unifying Understanding, Latent Foresight, and Action for Compositional Generalization. *arXiv preprint arXiv:2607.04988*.
 - [31] Mao, Lingjun, Ren, Jiawei, Zhou, Kun, Chen, Jixuan, Ma, Ziqiao, & Qin, Lianhui. 2025. DeliveryBench: Can Agents Earn Profit in Real World? *arXiv preprint arXiv:2512.19234*.
 - [32] Mao, Lingjun, He, Lukun, Cao, Jinglin, Xu, Wenpeng, *et al.* 2026 (oct). *Building a Real-world Autonomous Robotic System with Causality-driven Agent and World Model*. Aether AI Research.

-
- [33] Mu, Yao, Chen, Tianxing, Peng, Shijia, Chen, Zanzin, Gao, Zeyu, Zou, Yude, Lin, Lunkai, Xie, Zhiqiang, & Luo, Ping. 2024. Robotwin: Dual-arm robot benchmark with generative digital twins (early version). *Pages 264–273 of: European Conference on Computer Vision*. Springer.
 - [34] Nasiriany, Soroush, Maddukuri, Abhiram, Zhang, Lance, Parikh, Adeet, Lo, Aaron, Joshi, Abhishek, Mandlekar, Ajay, & Zhu, Yuke. 2024. RoboCasa: Large-Scale Simulation of Everyday Tasks for Generalist Robots. *In: Robotics: Science and Systems (RSS)*.
 - [35] OpenAI. 2026. *GPT-6 Astra: A New Generation of Intelligence*. Accessed: 2026-10-24.
 - [36] Padmakumar, Aishwarya, Thomason, Jesse, Shrivastava, Ayush, Lange, Patrick, Narayan-Chen, Anjali, Gella, Spandana, Piramuthu, Robinson, Tur, Gokhan, & Hakkani-Tur, Dilek. 2022. TEACH: Task-Driven Embodied Agents that Chat. *In: AAAI Conference on Artificial Intelligence*.
 - [37] Pchelintsev, Svyatoslav, Patratskiy, Maxim, Onishchenko, Anatoly, Korchemnyi, Alexandr, Medvedev, Aleksandr, Vinogradova, Uliana, Galuzinsky, Ilya, Postnikov, Aleksey, Kovalev, Alexey K, & Panov, Aleksandr I. 2025. Lera: Replanning with visual feedback in instruction following. *Pages 19218–19225 of: 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.
 - [38] Ramrakhya, Ram, Chang, Matthew, Puig, Xavier, *et al.* 2025. Grounding Multimodal LLMs to Embodied Agents that Ask for Help with Reinforcement Learning. *arXiv preprint arXiv:2504.00907*.
 - [39] Ren, Allen Z., Dixit, Anushri, Bodrova, Alexandra, Singh, Sumeet, Tu, Stephen, Brown, Noah, Xu, Peng, Takayama, Leila, Xia, Fei, Varley, Jake, Xu, Zhenjia, Sadigh, Dorsa, Zeng, Andy, & Majumdar, Anirudha. 2023. Robots That Ask For Help: Uncertainty Alignment for Large Language Model Planners. *In: Conference on Robot Learning (CoRL)*.
 - [40] Shao, Yifei, Huang, Junbo, Xi, Ziqiao, Nan, Fang, Han, Ruobing, Yan, Yuchen, Mao, Lingjun, Cao, Jinglin, Xu, Wenpeng, Zhang, Zijun, He, Lukun, Xu, Ziming, Liang, Shuang, Zhou, Kun, & Huang, Biwei. 2026. *CausalWAM: A Causal World Action Model for Generalizable Robot Control*.
 - [41] Shi, Lucy Xiaoyang, Ichter, Brian, Equi, Michael, Ke, Liyiming, Pertsch, Karl, Vuong, Quan, Tanner, James, Walling, Anna, Wang, Haohuan, Fusai, Niccolo, *et al.* 2025. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*.
 - [42] Shridhar, Mohit, Thomason, Jesse, Gordon, Daniel, Bisk, Yonatan, Han, Winson, Mottaghi, Roozbeh, Zettlemoyer, Luke, & Fox, Dieter. 2020. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. *In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
 - [43] Shridhar, Mohit, Yuan, Xingdi, Côté, Marc-Alexandre, Bisk, Yonatan, Trischler, Adam, & Hausknecht, Matthew. 2021. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. *In: International Conference on Learning Representations (ICLR)*.
 - [44] Singh, Ishika, Blukis, Valts, Mousavian, Arsalan, Goyal, Ankit, Xu, Danfei, Tremblay, Jonathan, Fox, Dieter, Thomason, Jesse, & Garg, Animesh. 2022. Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*.

-
- [45] Srivastava, Sanjana, Li, Chengshu, Lingelbach, Michael, Martín-Martín, Roberto, Xia, Fei, Vainio, Kent Elliott, Lian, Zheng, Gokmen, Cem, Buch, Shyamal, Liu, Karen, *et al.* 2022. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. *Pages 477–490 of: Conference on robot learning*. PMLR.
 - [46] Team, Galbot, Chen, Xuchuan, Cheng, Xiaoqian, Deng, Yu, Ding, Lihe, Dong, Shaocong, Gao, Xiangjun, Jia, Haozhe, Li, Zekai, Li, Zhoujian, *et al.* 2026a. Systematically Exploring the Capabilities of GPT-6 Astra as Embodied Policies. *arXiv preprint arXiv:2609.38537*.
 - [47] Team, Gemini Robotics, Abdolmaleki, Abbas, Abeyruwan, Saminda, Ainslie, Joshua, Alayrac, Jean-Baptiste, Arenas, Montserrat Gonzalez, Balakrishna, Ashwin, Batchelor, Nathan, Bewley, Alex, Bingham, Jeff, *et al.* 2025. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*.
 - [48] Team, Xiaomi Robotics, Guo, Jun, Jin, Piaopiao, Li, Jason, Li, Peiyan, Li, Yingyan, Liu, Futeng, Peng, Wanli, Qin, Optimus, Su, Yifei, *et al.* 2026b. Xiaomi-Robotics-1: Scaling Vision-Language-Action Models with over 100K Hours of Real-World Trajectories. *arXiv preprint arXiv:2607.15330*.
 - [49] Wang, Yuran, Huang, Siqiao, Li, Mingleyang, Zhang, Chenhao, Liang, Jiaqi, Jin, Weiyang, Chen, Yue, Chi, Xuemin, Zhou, Donghao, Yu, Qize, *et al.* 2026. Openwam: An open, modular exploration towards systematic world-action model pretraining. *arXiv preprint arXiv:2609.07398*.
 - [50] Wu, Wei, Wang, Fangjing, Lu, Fan, Sun, He, Liu, Shi, Wang, Yunnan, Yan, Yibin, Wang, Yong, Ma, Shuailei, Wang, Xinyang, *et al.* 2026a. From Foundation to Application: Improving VLA Models in Practice. *arXiv preprint arXiv:2607.06403*.
 - [51] Wu, Wei, Wang, Fangjing, Lu, Fan, Sun, He, Liu, Shi, Wang, Yunnan, Yan, Yibin, Wang, Yong, Ma, Shuailei, Wang, Xinyang, Liu, Yibin, Yang, Shuai, Zhou, Tianxiang, Zhang, Kejia, Zhou, Lei, Su, Cheng, Xue, Nan, Tan, Bin, Zhang, Han, Zhang, Youchao, Liao, Fei, Zhu, Xing, Shen, Yujun, & Zheng, Kecheng. 2026b. *From Foundation to Application: Improving VLA Models in Practice*.
 - [52] Xu, Ziming, Liang, Shuang, Han, Ruobing, Xi, Ziqiao, Rao, Mingxing, Zhou, Kun, Zhang, Zijun, Yan, Yuchen, Wei, Yufan, Huang, Junbo, *et al.* 2026. CausalWM: Causal Chain-of-Thought Reasoning for Embodied World Model. *arXiv preprint arXiv:2609.23184*.
 - [53] Yang, Rui, Chen, Hanyang, Zhang, Junyu, Zhao, Mark, Qian, Cheng, Wang, Kangrui, Wang, Qineng, Koripella, Teja Venkat, Movahedi, Marziyeh, Li, Manling, Ji, Heng, Zhang, Huan, & Zhang, Tong. 2025a. EmbodiedBench: Comprehensive Benchmarking Multi-modal Large Language Models for Vision-Driven Embodied Agents. *In: International Conference on Machine Learning (ICML)*.
 - [54] Yang, Zhejian, Chen, Yongchao, Zhou, Xueyang, Yan, Jiangyue, Song, Dingjie, Liu, Yinuo, Li, Yuting, Zhang, Yu, Zhou, Pan, Chen, Hechang, *et al.* 2025b. Agentic robot: A brain-inspired framework for vision-language-action models in embodied agents. *arXiv preprint arXiv:2505.23450*.
 - [55] Ye, Seonghyeon, Ge, Yunhao, Zheng, Kaiyuan, Gao, Shenyuan, Yu, Sihyun, Kurian, George, Indupuru, Suneel, Tan, You Liang, Zhu, Chuning, Xiang, Jiannan, *et al.* 2026. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*.

-
- [56] Yu, Tianhe, Quillen, Deirdre, He, Zhanpeng, Julian, Ryan, Hausman, Karol, Finn, Chelsea, & Levine, Sergey. 2020. Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning. *In: Conference on Robot Learning (CoRL)*.
 - [57] Zhang, He, Xiang, Lingzhu, Lin, Haitao, Huang, Zeyu, Wang, Minghui, Zhong, Dingyan, Dong, Yubo, Wu, Yihao, Rao, Yongming, Zhang, Dongsheng, *et al.* 2026a. Hy-embodied-0.5-vla: From vision-language-action models to a real-world robot learning stack. *arXiv preprint arXiv:2606.14409*.
 - [58] Zhang, Shiduo, Xu, Zhe, Liu, Peiju, *et al.* 2025. VLABench: A Large-Scale Benchmark for Language-Conditioned Robotics Manipulation with Long-Horizon Reasoning Tasks. *In: IEEE/CVF International Conference on Computer Vision (ICCV)*.
 - [59] Zhang, Yehang, Su, Jianchong, Huang, Haojian, *et al.* 2026b. WorldLines: Benchmarking and Modeling Long-Horizon Stateful Embodied Agents. *arXiv preprint arXiv:2606.18847*.
 - [60] Zhang, Yixian, Zhang, Huanming, Gao, Feng, Li, Xiao, Liu, Zhihao, Nie, Yi, Zhu, Chunyang, Qiu, Jiaying, Yan, Yuchen, Liu, Jiyuan, *et al.* 2026c. Harness vla: Steering frozen vlas into reliable manipulation primitives via memory-guided agents. *arXiv preprint arXiv:2607.08448*.
 - [61] Zheng, Jinliang, Li, Jianxiong, Wang, Zhihao, Liu, Dongxiu, Kang, Xirui, Feng, Yuchun, Zheng, Yinan, Zou, Jiayin, Chen, Yilun, Zeng, Jia, *et al.* 2026. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *Pages 60580–60606 of: International Conference on Learning Representations*, vol. 2026.
 - [62] Zhou, Pengfei, Chen, Shengcong, Chen, Di, Wang, Jiaxu, Jin, Rongjun, Zhu, Bingwen, Pan, Yike, Gu, Songen, Wang, Kuanning, Nan, Shufeng, Qiu, Xingyu, Qiu, Chenhao, Yang, Pu, Cai, Yunuo, Gao, Jianxiong, Li, Yifan, Fu, Yanwei, Yue, Xiangyu, Chen, Zhi, & Luo, Jianlan. 2026. τ_0 -WM: A Unified Video-Action World Model for Robotic Manipulation.

A Benchmark Setup

This section describes the information available to evaluated systems, the robot action space, and the shared simulation settings. Task and scenario definitions are provided in Appendix B.

A.1 Provided Information

The evaluation interface provides all systems with access to the same information sources: task instructions and context, camera observations, robot-state feedback, and a progressively revealed navigation map. The same information disclosure rules apply to all systems. How this information is represented, integrated, and used depends on the model architecture, input adapter, and agent design. Model-specific inputs are described in Appendix C.

Task instructions and context. Each episode provides a task instruction or a high-level scenario request. Context-dependent settings additionally supply information such as user preferences, household routines, and storage arrangements. The information provided for each task family is described in Appendix B.

Visual observations. The benchmark records synchronized RGB observations from one head-mounted camera at 1280×800 and two wrist-mounted cameras at 848×480 . Learned policies receive updated observations after executing an action prefix; the prediction horizon and executed prefix length are separate quantities. Code-based agents can request observations through the evaluation interface.

Robot state. We use the AgiBot G1 as the robot embodiment. Its state is represented by a 23-dimensional vector containing twenty joint channels and the mobile base’s world-frame position (x, y) and unwrapped yaw. The joint channels comprise fourteen arm joints, two torso/waist joints, two head joints, and two gripper channels. Torso lift is expressed in metres, while angular channels are expressed in radians. Table 4 summarizes the state and action representations.

Navigation map. Map-assisted evaluations provide a read-only top-down navigation map at a resolution of 0.04 m. The map is derived from static scene geometry rather than reconstructed from RGB images. It is progressively revealed using the measured head-camera pose, field of view, and static occlusion, with a small local region around the robot treated as known. The same generation and revelation rules apply across participating systems; revealed regions may differ because systems choose different viewpoints and trajectories. The map distinguishes known free floor, static obstacles, and unexplored regions, and displays the robot’s measured position, heading, and executed trajectory. The complete map, generated movable objects, task labels, and target poses are not exposed. No planned routes or automatic navigation actions are provided. Systems must choose and execute their own movements and use camera observations to assess movable objects.

A.2 Action Space

Each action is a 22-dimensional whole-body command containing fourteen arm-joint targets, two torso/waist targets, two head-joint targets, two continuous gripper targets, and two mobile-base commands specifying forward velocity and yaw rate. Joint commands specify target positions, while base commands specify velocities. The channel definitions, units, and nominal ranges are summarized in Table 4.

A.3 Simulation and Execution Settings

Scene construction. We first construct a digital twin of a real home, matching its layout, furniture dimensions, and appearance. We then create additional layouts by modifying floor plans, including T-shaped and L-shaped configurations, and rearranging functional areas. All layouts are manually reviewed for geometric validity and plausible furniture arrangements. The resulting catalog contains 50 distinct layouts, with additional visual variation in furniture colours, textures, materials, and interior styles.

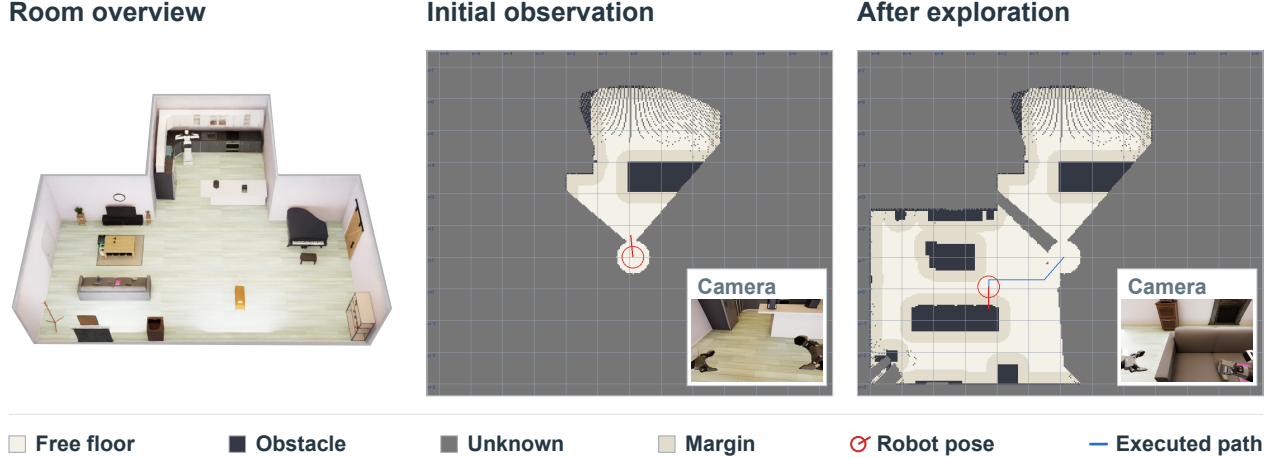


Figure 8. Room overview (left) and progressively revealed maps with corresponding camera views (right). The room overview is for illustration and is not an agent input.

Table 4. Robot state and action spaces. Joint entries represent measured positions in the state and position targets in the action. Base position and unwrapped yaw are expressed in the world frame; base actions specify forward velocity and yaw rate. Ranges are nominal command bounds, not empirical state bounds; the arm intervals summarize envelopes over joint-specific limits. Base-speed bounds describe the demonstration-collection controller.

State space (23 dimensions)				Action space (22 dimensions)			
Dim.	Component	Unit	Nominal range	Dim.	Component	Unit	Nominal range
0–6	Left arm joints	rad	$[-\pi, \pi]$	0–6	Left arm joints	rad	$[-\pi, \pi]$
7–13	Right arm joints	rad	$[-\pi, \pi]$	7–13	Right arm joints	rad	$[-\pi, \pi]$
14	Torso lift	m	$[0, 0.55]$	14	Torso lift	m	$[0, 0.55]$
15	Waist pitch	rad	$[0, \pi/2]$	15	Waist pitch	rad	$[0, \pi/2]$
16	Head yaw	rad	$[-\pi/2, \pi/2]$	16	Head yaw	rad	$[-\pi/2, \pi/2]$
17	Head pitch	rad	$[-0.349, 0.524]$	17	Head pitch	rad	$[-0.349, 0.524]$
18	Left gripper	rad	$[0, \pi/4]$	18	Left gripper	rad	$[0, \pi/4]$
19	Right gripper	rad	$[0, \pi/4]$	19	Right gripper	rad	$[0, \pi/4]$
20	Base x	m	—	20	Base forward velocity	m/s	$[-0.18, 0.18]$
21	Base y	m	—	21	Base yaw rate	rad/s	$[-0.25, 0.25]$
22	Base yaw (unwrapped)	rad	—				

Physics and control. Robot parameters, camera calibration, asset properties, and scene settings are resolved before each episode and remain fixed across its subtasks. Physics integration runs at 60 or 240 Hz, depending on the configuration, while the action interface operates at 20 Hz. The integration rate and backend remain unchanged within an episode. Cloth manipulation may use a dedicated soft-body solver. Grasping relies on physical contact and friction between the gripper pads and objects.

Continuous execution and scoring. Task transitions do not reset the scene, object states, or accumulated progress. Task processes and appliance dynamics persist as the robot moves between workspaces. Progress is tracked throughout the episode, while placement credit is updated using the current physical state, so disturbing an earlier placement can reduce its final credit. At episode termination, all registered tasks are finalized using their task-specific scoring rules. Task-specific completion criteria are provided in Appendix B.

B Task and Scenario Descriptions

The following sections describe the tasks and scenarios used to evaluate robot autonomy. Input descriptions refer to the environment interface; individual policy inputs follow the adapters in Appendix C. Manipulation includes 18 short tasks and 8 long tasks. Task discovery and context reasoning assess whether agents identify relevant objects through gripper contact. Failure recovery evaluates execution from predefined failure states, while robustness evaluates adaptation to disturbances during execution. Four open-ended household scenarios combine task discovery, reasoning, and manipulation under high-level user requests. The figures illustrate task settings and procedures using demonstrations and diagnostic renders, rather than evaluated-agent results.

B.1 Manipulation

Task overview. We evaluate manipulation through 18 short tasks and 8 long tasks, covering both individual manipulation skills and their composition into more complex household activities.

Input and setting. The agent receives a task instruction, camera observations, and robot-state feedback. Each episode starts from a task-specific scene. Long tasks require continuous execution across subgoals without resetting the environment.

Completion and scoring. Performance is measured using task-specific process scores. All 18 short tasks and 8 long tasks contribute to the manipulation score.

Short tasks

1. Prepare coffee

Request. Prepare a cup of coffee.

Setting. A coffee machine, an empty cup, and a bag of coffee beans are on the countertop.

Task. Place the cup upright under the coffee outlet and release it. Pour beans into the hopper, then return the bag upright to the countertop and release it. Press the button to start brewing, press it again to stop dispensing, and retrieve the same cup.

Completion. Prepare the coffee and retrieve the cup while keeping it upright and avoiding excessive spillage.

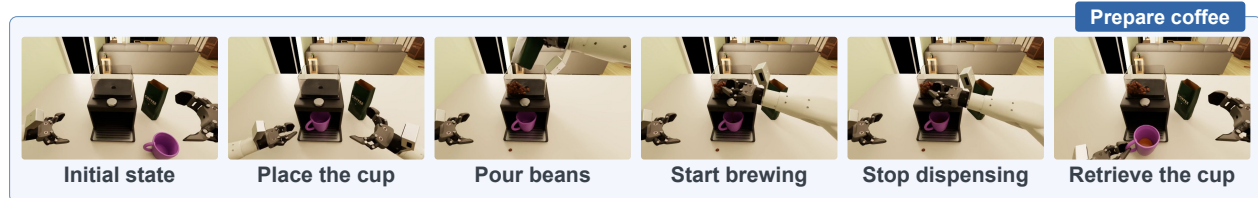


Figure 9. Procedure for the “Prepare coffee” task.

2. Load and unload the air fryer

Request. Load the designated pastry into the air fryer, then take it out.

Setting. The air fryer basket is initially closed. Two pastries are on a plate, one of which is designated as the target.

Task. Pull out the basket, place the target pastry inside, and push the basket closed. Then remove the same basket completely, set it down securely, and retrieve the same pastry.

Completion. Load the target pastry, remove and set down the basket, and retrieve the pastry.

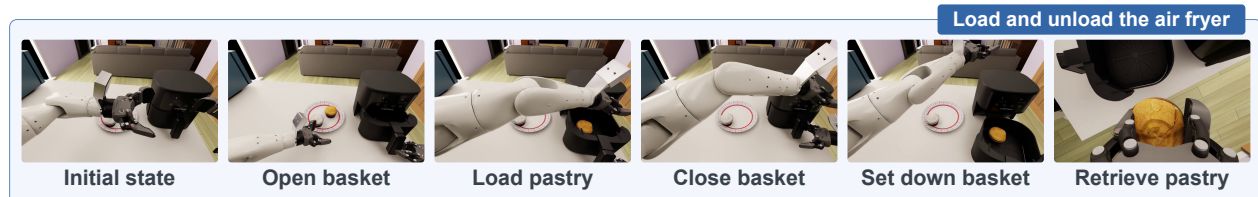


Figure 10. Procedure for the “Load and unload the air fryer” task.

3. Stock the tray

Request. Bring the requested product from the shelf to the small wooden tray.

Setting. The target product is on the snack shelf, and the small wooden tray is on the coffee table.

Task. Pick up the requested product, carry it to the coffee table, and place it in the small wooden tray.

Completion. Leave the requested product in the small wooden tray after releasing it.

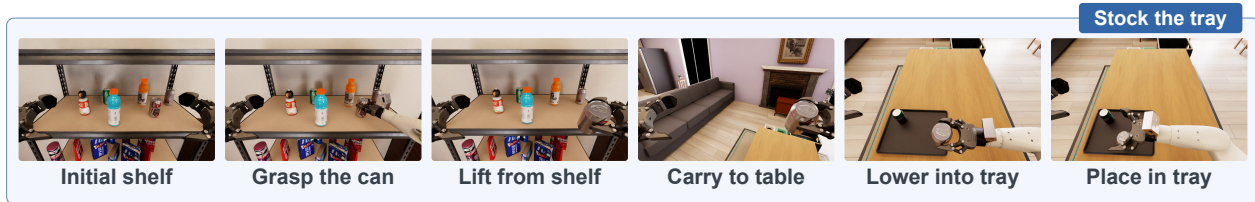


Figure 11. Procedure for the “Stock the tray” task.

4. Retrieve items from the drawer

Request. Take the requested object out of the drawer and put it in the specified container.

Setting. A wallet or sunglasses case is inside the initially closed coffee-table drawer. The destination is specified as either the open tote bag or the item-storage basket.

Task. Open the drawer, grasp the target, lift it fully clear of the drawer rim, carry it to the specified container, and release it inside. Closing the drawer afterward is not required.

Completion. Complete the retrieval and leave the same object released inside and supported by the specified container.

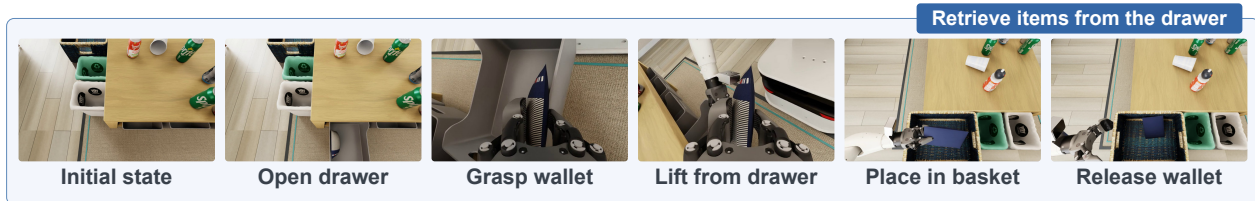


Figure 12. Procedure for the “Retrieve items from the drawer” task.

5. Store items in the drawer

Request. Put the requested sofa object into the coffee-table drawer and close the drawer.

Setting. A wallet or green notebook is on the sofa, and the coffee-table drawer is initially closed.

Task. Pick up the requested object from the sofa, carry it to the coffee table, open the drawer, place and release the object inside, and close the drawer.

Completion. The drawer must be closed, with the same object released inside and supported by its interior.

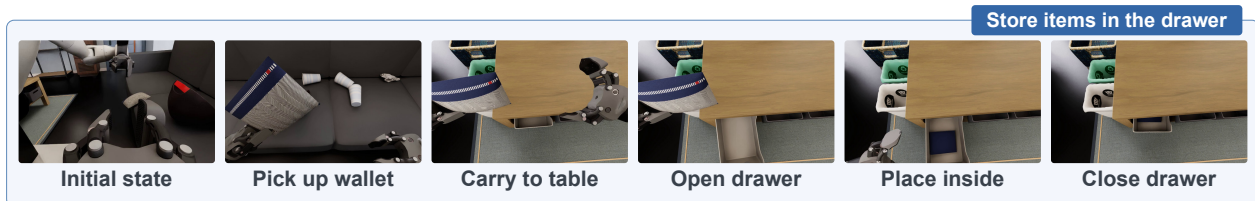


Figure 13. Procedure for the “Store items in the drawer” task.

6. Hang the cap

Request. Take the cap from the sofa and hang it on the coat rack.

Setting. A cap rests on the sofa, and a coat rack with hooks is elsewhere in the room.

Task. Grasp the cap, carry it to the rack, position it on a suitable hook, and release it.

Completion. Leave the cap hanging from and supported by a valid rack hook after release. Merely placing it near the rack is insufficient.

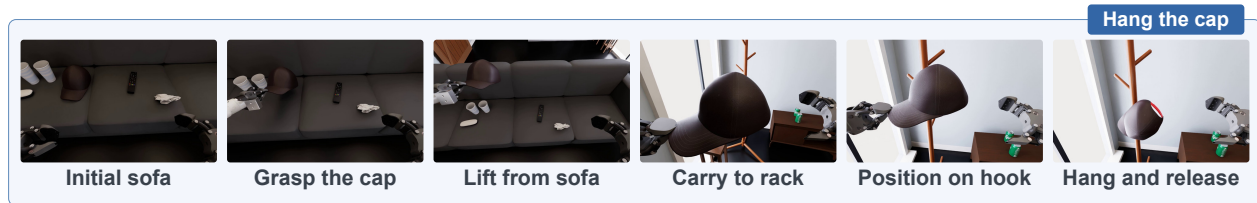


Figure 14. Procedure for the “Hang the cap” task.

7. Clear table waste

Request. Clear the designated rubbish from the coffee table into the appropriate bins.

Setting. Designated waste items are on the coffee table, with waste and recycling bins nearby.

Task. Pick up each designated item and release it inside the appropriate waste or recycling bin. Items may be cleared in any order.

Completion. All designated items must be inside their correct bins and no longer held by the robot. Removing them from the table alone is insufficient.

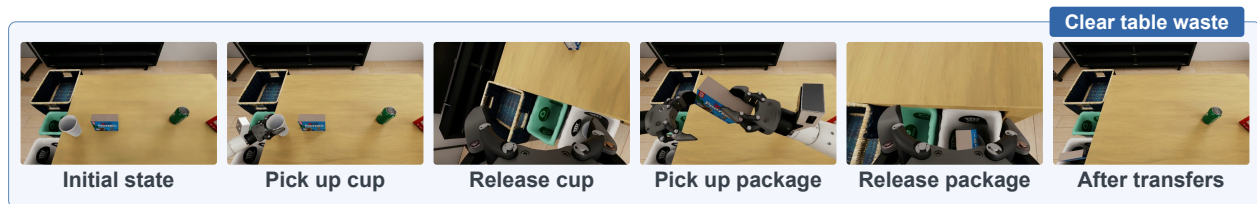


Figure 15. Procedure for the “Clear table waste” task.

8. Move food to the microwave

Request. Move the designated food box from the refrigerator into the microwave and close both doors.

Setting. The target food box is inside the refrigerator among other food boxes. The refrigerator and microwave doors are initially closed.

Task. Open the refrigerator, retrieve the target box, and close the refrigerator door. Carry the box to the microwave, open its door, place and release the same box on the plate inside, and close the door. Heating is not required.

Completion. Complete the transfer and leave the same food box inside the microwave, with both appliance doors closed.

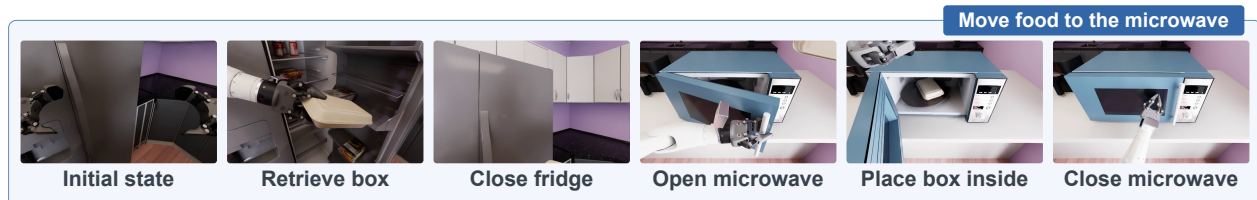


Figure 16. Procedure for the “Move food to the microwave” task.

9. Clear objects from the cloth

Request. Move all designated objects off the tablecloth onto the permitted tabletop area.

Setting. The target objects initially rest on the tablecloth. A separate area on the same coffee table is designated for placing them.

Task. Move each designated object off the cloth and into the tabletop area specified by the instruction, in any order. Leave the cloth in place and keep other reserved tabletop areas clear.

Completion. All targets must be released and supported by the permitted tabletop area outside the cloth, with the cloth left in place.

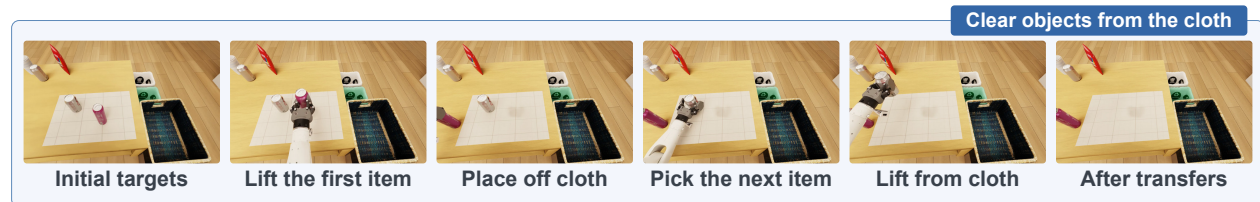


Figure 17. Procedure for the “Clear objects from the cloth” task.

10. Pack the bag

Request. Put the requested personal object into the tote bag.

Setting. The target object rests at its specified source location. The destination tote bag is fixed in place and already open.

Task. Pick up the target, carry it to the tote bag, guide it through the opening, and release it inside. Opening or moving the bag is not part of the task.

Completion. Leave the same object released inside and supported by the bag, rather than on its rim or the surrounding surface.

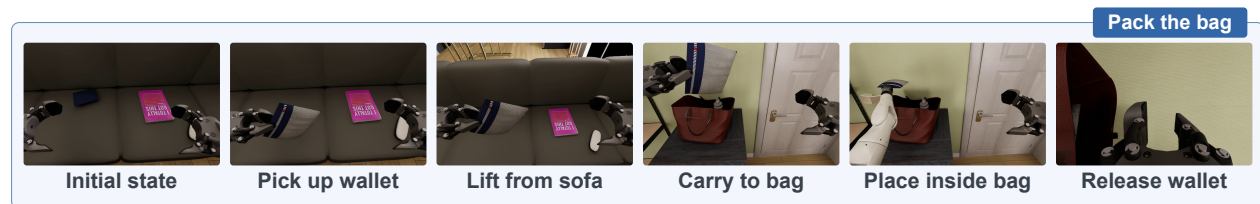


Figure 18. Procedure for the “Pack the bag” task.

11. Place cutlery on the tray

Request. Place the designated utensils neatly on the large meal tray.

Setting. The requested utensils are in a cutlery holder beside the large meal tray.

Task. Grasp each designated utensil, extract it from its holder slot, carry it to the large meal tray, and release it inside. Follow the requested arrangement, such as placing the utensils neatly in parallel.

Completion. All requested utensils are placed on the tray in the specified arrangement.

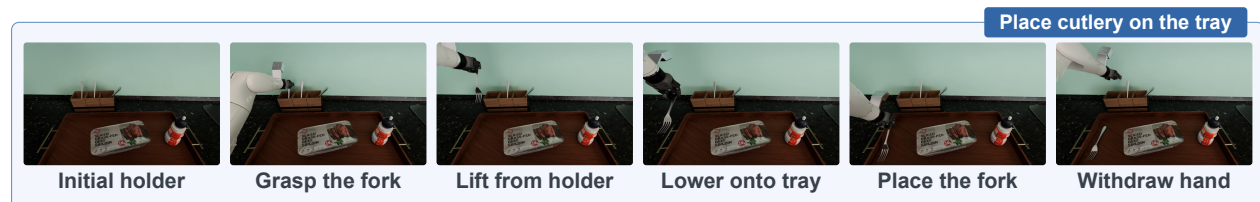


Figure 19. Procedure for the “Place cutlery on the tray” task.

12. Deliver the tray

Request. Carry the large meal tray to the coffee table.

Setting. The large meal tray rests on its initial support and may be empty or already loaded with food boxes, cups, or utensils. The destination area on the coffee table is available.

Task. Grasp and lift the tray, carry it steadily to the designated area on the coffee table, lower it onto the tabletop, and release it. Preparing food or loading the tray beforehand is not part of this transport task.

Completion. Place and release the tray on the designated area of the coffee table, keeping any required contents on the tray.

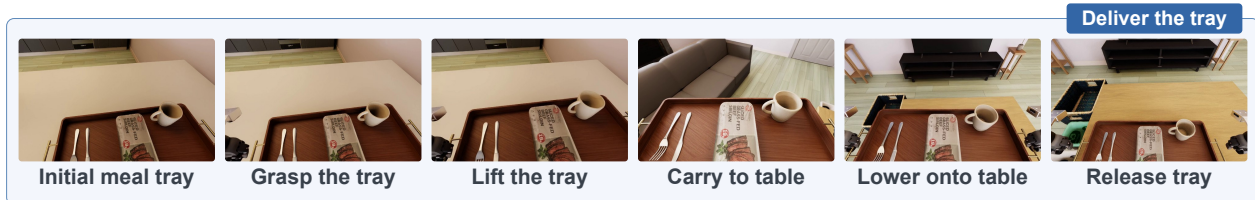


Figure 20. Procedure for the “Deliver the tray” task.

13. Move sofa items to the table

Request. Move the requested object from the sofa to the coffee table.

Setting. A designated object, such as a book, remote control, wallet, or sunglasses case, rests on the sofa. The destination is the permitted object area on the coffee table.

Task. Pick up the requested object from the sofa, carry it to the coffee table, and release it in the permitted object area, keeping clear of the tablecloth and fixed wooden tray.

Completion. Leave the same object released in the permitted area and supported by the coffee-table surface.

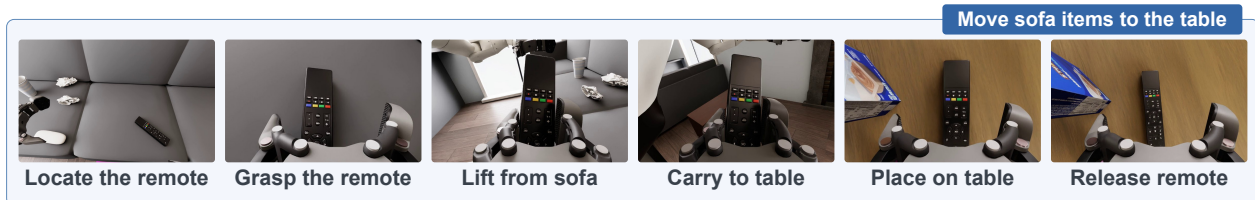


Figure 21. Procedure for the “Move sofa items to the table” task.

14. Store items in the basket

Request. Put the requested tabletop object into the item-storage basket.

Setting. A wallet or sunglasses case rests on the specified table. The destination is the item-storage basket beside the waste bins.

Task. Pick up the target from the specified table, carry it to the item-storage basket, guide it through the opening, and release it inside. The tote bag and waste bins are not valid destinations.

Completion. Leave the same object released inside and supported by the designated item-storage basket.

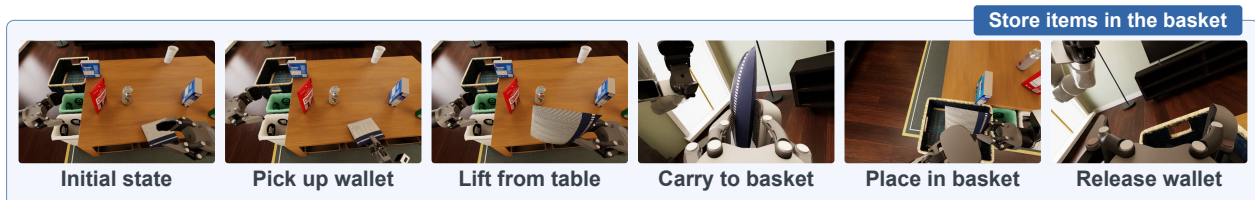


Figure 22. Procedure for the “Store items in the basket” task.

15. Heat food and retrieve the plate

Request. Heat the food in the microwave, then take out the plate.

Setting. A food box rests on a plate inside the closed microwave. The microwave is initially off.

Task. Turn the control knob to start heating and wait for the light to turn off. Open the door, grasp and remove the plate while keeping the food box on it, and close the door. Take out the plate and food together rather than removing only the box.

Completion. Finish heating, retrieve the plate with the food box remaining on it, and close the microwave door.

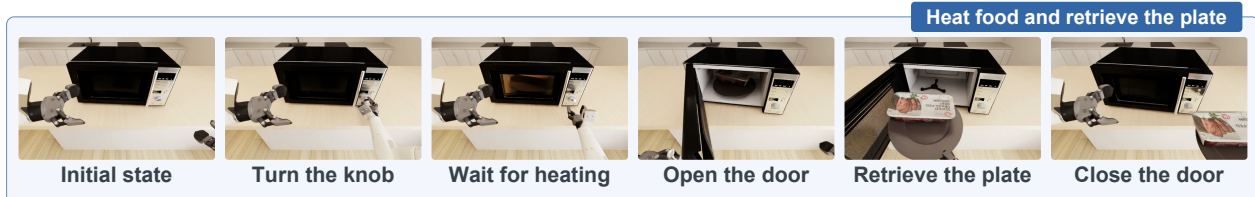


Figure 23. Procedure for the “Heat food and retrieve the plate” task, illustrated using atomic-action demonstrations.

16. Wipe the table

Request. Pick up the brush, clean the stains on the coffee table, and return the brush to its original place.

Setting. A brush and one or more stains are on the coffee table. The brush is initially resting on the tabletop.

Task. Grasp and lift the brush, then wipe across the stained areas with the brush face in contact with the tabletop. After cleaning, return the brush to its original position and release it.

Completion. Remove the stains and return the brush to its original place.

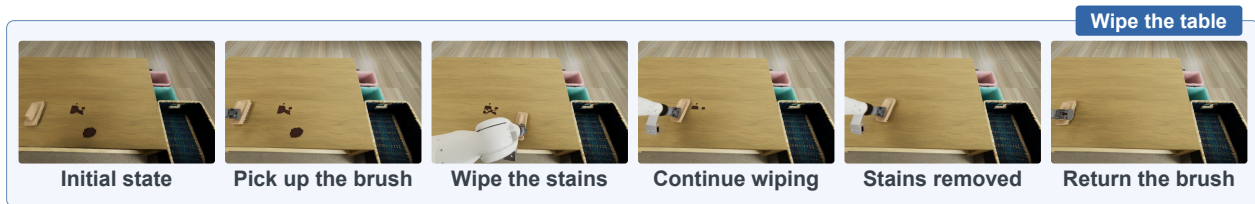


Figure 24. Procedure for the “Wipe the table” task.

17. Spread the tablecloth

Request. Take the tablecloth from the cloth basket and spread it on the coffee table.

Setting. A tablecloth is stored in the designated cloth basket beside the coffee table. The intended tabletop region is clear of objects.

Task. Grasp the cloth, lift it out of the basket, and unfold it above the table. Lower and spread it over the designated tabletop region, adjust its position as needed, and release it.

Completion. Leave the tablecloth unfolded and spread over the designated tabletop area.

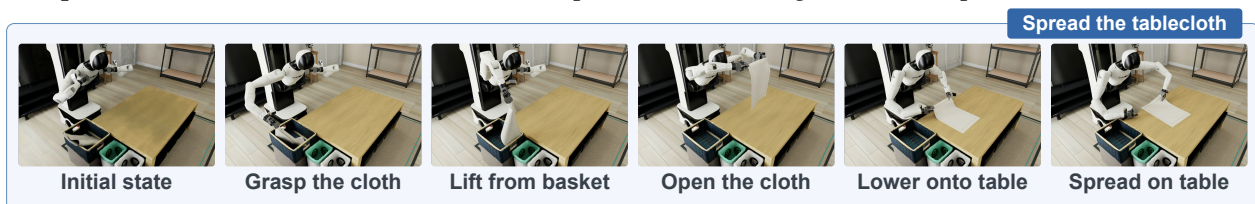


Figure 25. Procedure for the “Spread the tablecloth” task.

18. Collect the tablecloth

Request. Collect the tablecloth from the coffee table and put it in the cloth basket.

Setting. A tablecloth lies on the coffee table, with no objects resting on it. The designated cloth basket is beside the table.

Task. Grasp an edge of the cloth, gather it, and lift the entire cloth clear of the tabletop. Move it over the designated cloth basket, lower it inside, and release it.

Completion. Place the entire tablecloth inside the designated basket, leaving no part on the table.

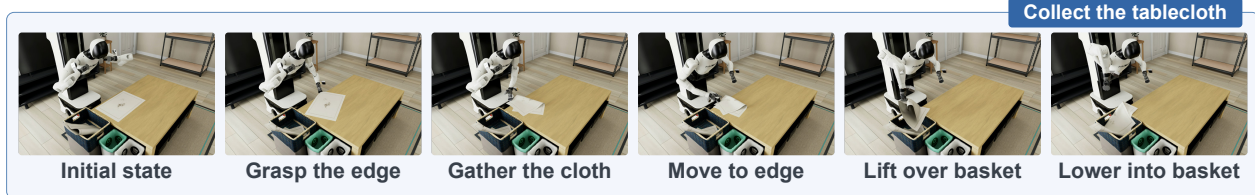


Figure 26. Procedure for the "Collect the tablecloth" task.

Long tasks

1. Stock the tray (Multi)

Request. Bring all requested snacks and drinks from the shelf to the small wooden tray.

Setting. Multiple products are requested by type and quantity. They are on the shelf, and the small wooden tray is on the coffee table.

Task. Select the requested products, carry them to the coffee table, and release them inside the small wooden tray. Track the remaining items and avoid displacing products already delivered. The large meal tray is not the destination.

Completion. Place all requested products in the small wooden tray.

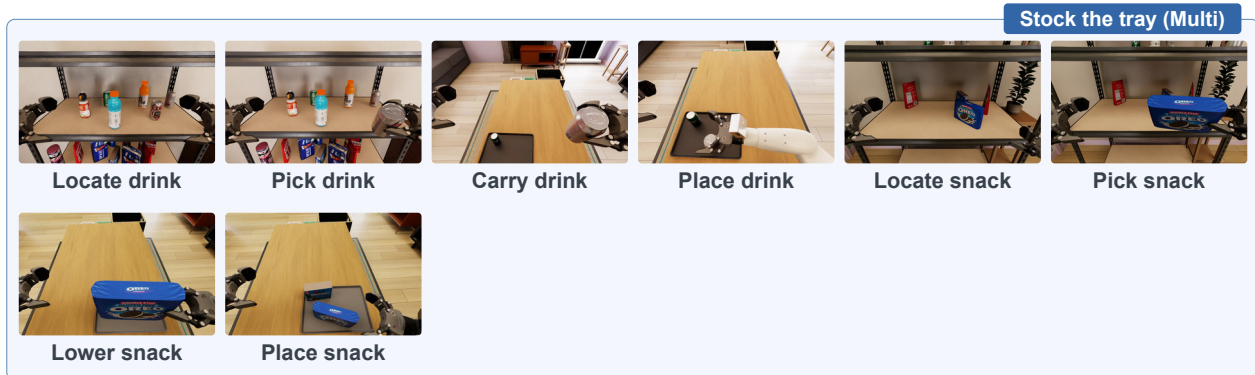


Figure 27. Drink and snack delivery components of the "Stock the tray (Multi)" task, illustrated using separate demonstrations.

2. Tidy the sofa area

Request. Put the book on the coffee table, store the wallet in the coffee-table drawer and close it, and hang the cap on the coat rack.

Setting. A book, a wallet, and a cap are on the sofa. The coffee table, its initially closed drawer, and the coat rack are available.

Task. Transfer the book to the designated coffee-table surface. Open the drawer, place and release the wallet inside, and close the drawer. Hang and release the cap on a valid coat-rack hook. Complete these transfers without undoing earlier placements.

Completion. The book rests on the designated table surface, the wallet remains inside the closed drawer, and the cap is supported by the coat-rack hook after release. All three conditions must remain satisfied at the end.

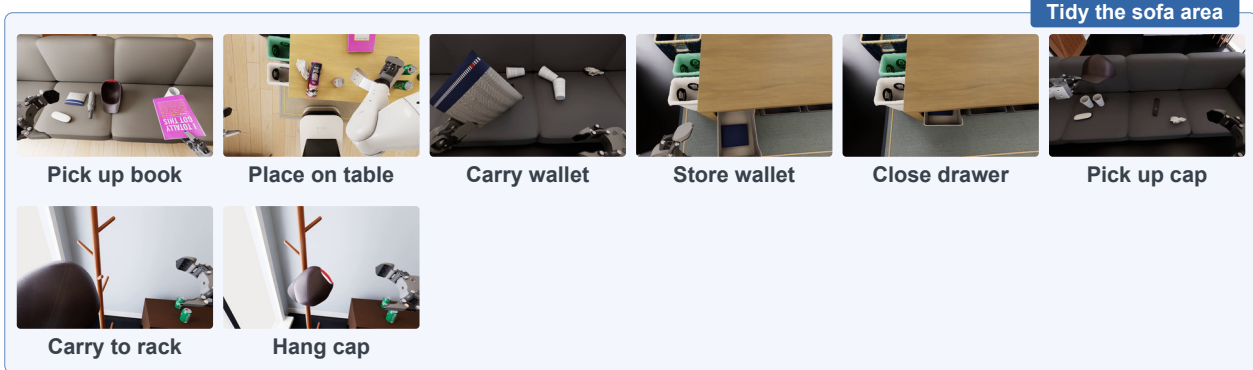


Figure 28. Book transfer, wallet storage, and cap hanging for the “Tidy the sofa area” task, illustrated using separate demonstrations.

3. Tidy the table

Request. Put the designated table waste in the correct bins and the requested belongings in the item-storage basket.

Setting. Specified waste items and reusable belongings are on the table. The waste bins and the item-storage basket are available.

Task. Identify the requested waste items and dispose of each in the assigned bin. Pick up the specified wallet or sunglasses case from the table and place it in the item-storage basket. Do not discard reusable belongings or use the tote bag as the storage destination. Preserve earlier placements while completing the remaining transfers.

Completion. Every designated waste item is in its correct bin, and every requested belonging is inside and supported by the item-storage basket. These conditions must remain satisfied at the end.

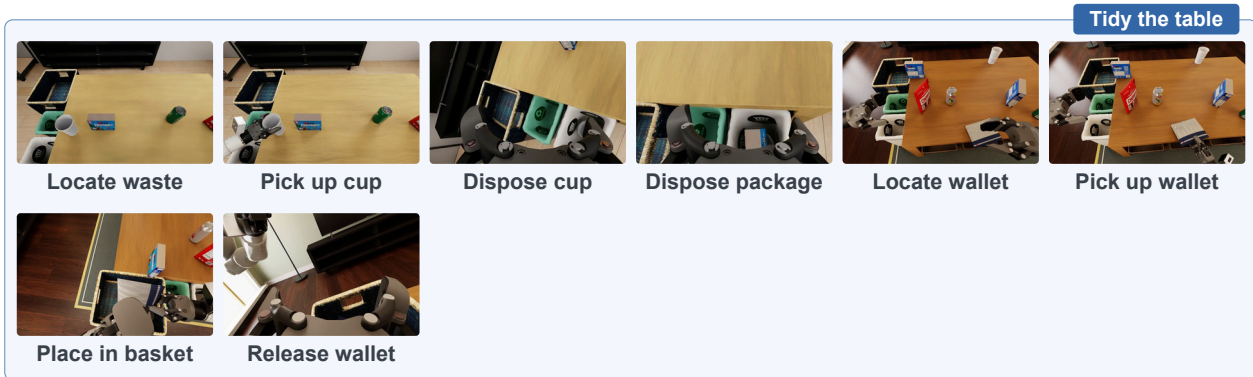


Figure 29. Waste disposal and belonging storage for the “Tidy the table” task, illustrated using separate demonstrations.

4. Replace the tablecloth

Request. Replace the used tablecloth with a clean one.

Setting. The used cloth covers the table. A clean cloth and the cloth basket are available from the start.

Task. Clear obstructions when needed, remove the used cloth, fold it, and put it away. Retrieve the clean cloth, spread it over the intended tabletop region, and release it.

Completion. The used cloth is folded and stored, and the clean cloth is released and spread over the intended tabletop region.

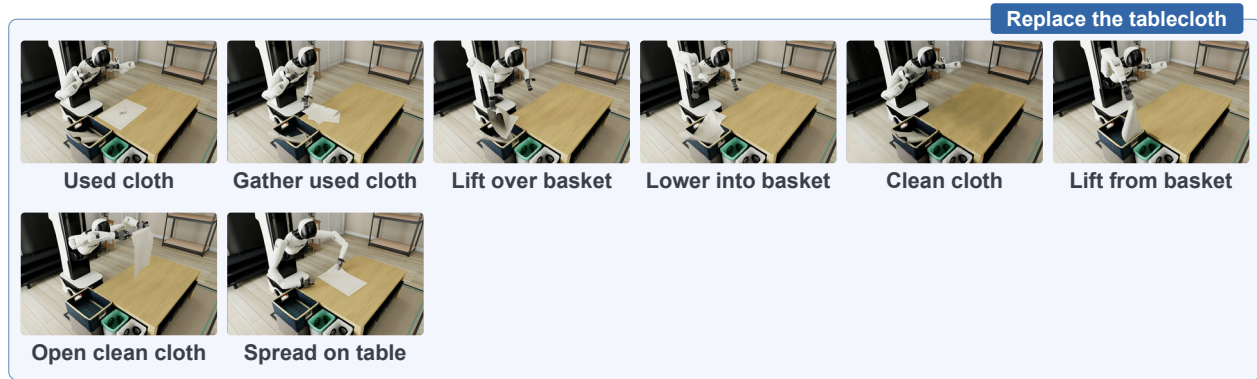


Figure 30. Cloth collection and spreading components of the “Replace the tablecloth” task, illustrated using separate demonstrations.

5. Find and pack an item

Request. Find the requested object and put it in the tote bag.

Setting. A wallet or sunglasses case is placed somewhere in the room before the episode begins. The instruction identifies the target but does not reveal its location. The destination is the designated open tote bag.

Task. Search the room, locate the requested object, pick it up, and carry it to the bag. Guide the same object through the opening and release it inside. Search and packing take place in the same scene without resetting the object.

Completion. Locate the requested object, complete its pickup and packing sequence, and leave it inside and supported by the designated bag.

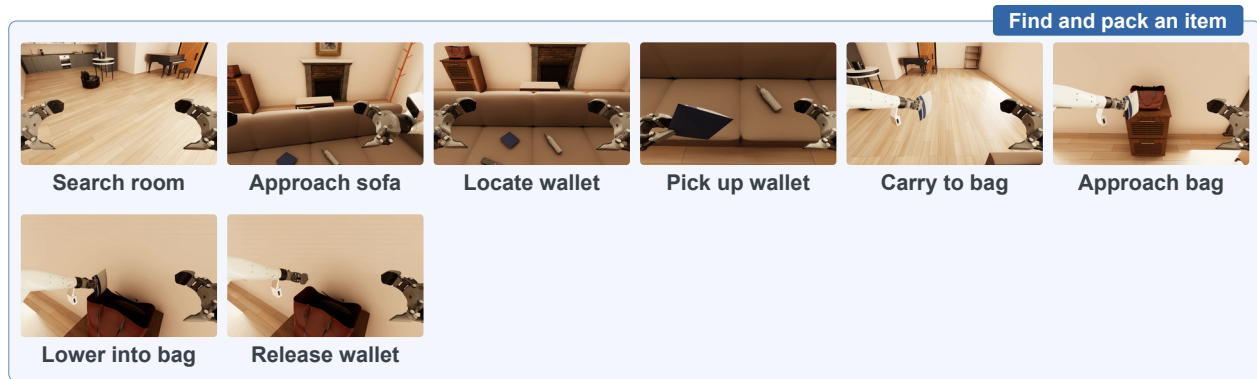


Figure 31. Search and packing of a wallet, shown as an excerpt from one continuous recording.

6. Prepare a meal

Request. Take the requested food from the refrigerator, heat it in the microwave, and place it with the requested cutlery on the large meal tray.

Setting. The requested food box is in the refrigerator. A microwave, a large meal tray, and the requested cutlery are available.

Task. Retrieve the food box and close the refrigerator. Heat the food in the microwave, retrieve it, and close the microwave door. Transfer the heated food box from the microwave plate to the large meal tray, and add the requested cutlery. Leave the microwave plate off the tray.

Completion. The heated food box and all requested cutlery are on the large meal tray. Delivering the tray is not required.

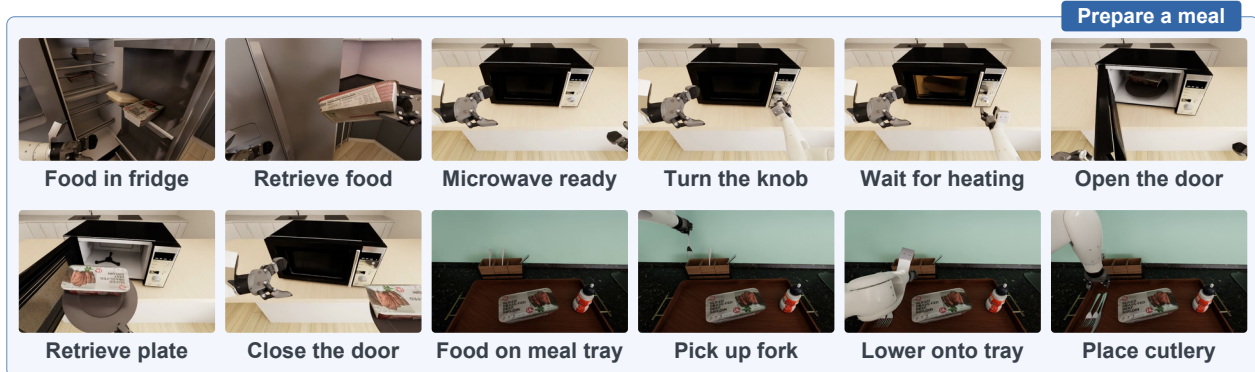


Figure 32. Refrigerator, microwave, and cutlery components of the “Prepare a meal” task, illustrated using separate demonstrations.

7. Prepare coffee and pastry

Request. Prepare a cup of coffee and load and retrieve the designated pastry using the air fryer.

Setting. A cup and coffee beans are available at the coffee machine. The target pastry is on a plate beside the air fryer, whose basket is initially closed.

Task. Position the cup under the coffee outlet, pour the beans, return and release the bean bag, operate the machine, and retrieve the coffee cup. For the pastry, pull out the basket, place the target inside, and close it. Then remove the basket completely, set it down securely, and retrieve the same pastry. This task instance requires neither pastry heating nor returning the pastry to the plate.

Completion. Prepare the coffee and retrieve the cup, and complete the pastry loading and retrieval sequence. The cup and pastry do not need to be held simultaneously at the end.

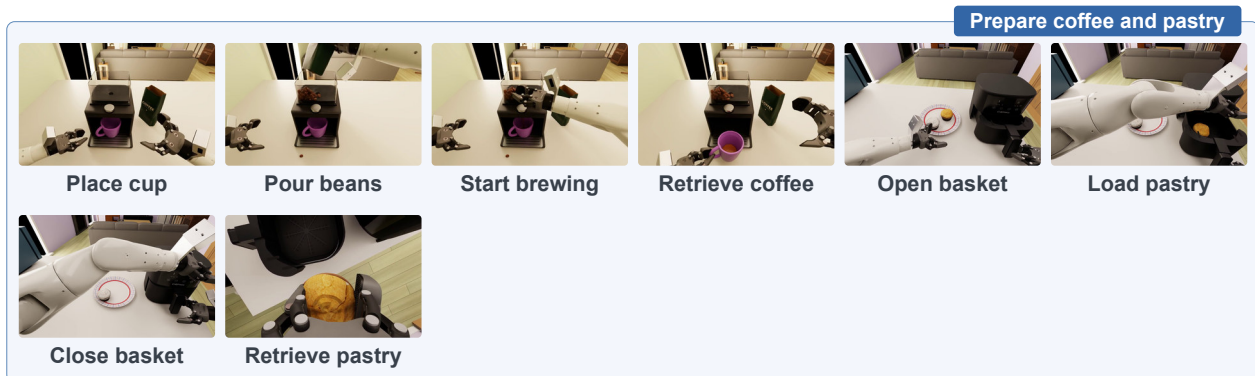


Figure 33. Coffee preparation and pastry retrieval for the “Prepare coffee and pastry” task, illustrated using separate demonstrations.

8. Pack bags and baskets

Request. Collect the listed objects and put each in its assigned bag or item-storage basket.

Setting. A wallet and a sunglasses case are placed at specified locations in the room, each assigned to a bag or item-storage basket.

Task. Pick up each requested object from its source, carry it to the assigned container, and release it inside. Keep track of the remaining objects and preserve earlier placements. The assignments may use the same container type for all targets or a mixture of bags and baskets.

Completion. Complete all requested transfers and leave every target inside and supported by its assigned container at the end.

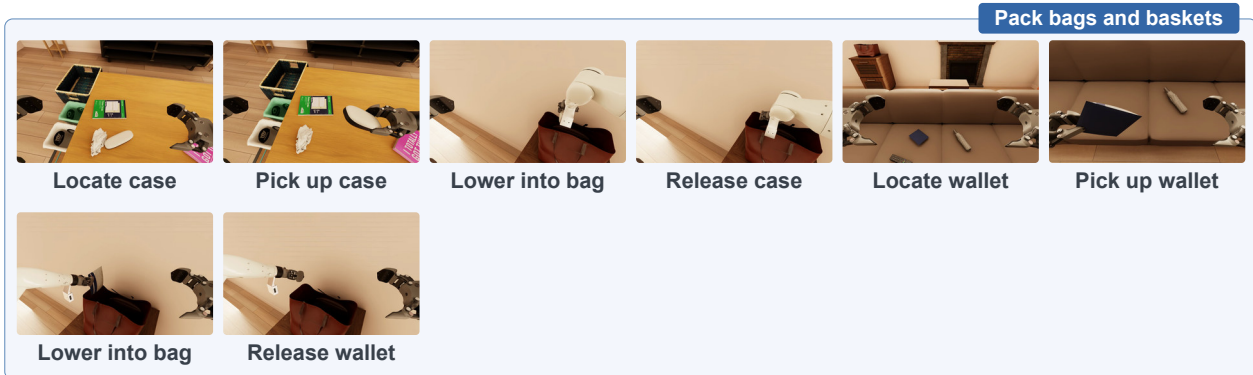


Figure 34. Successive packing of a glasses case and a wallet, shown as excerpts from one continuous recording.

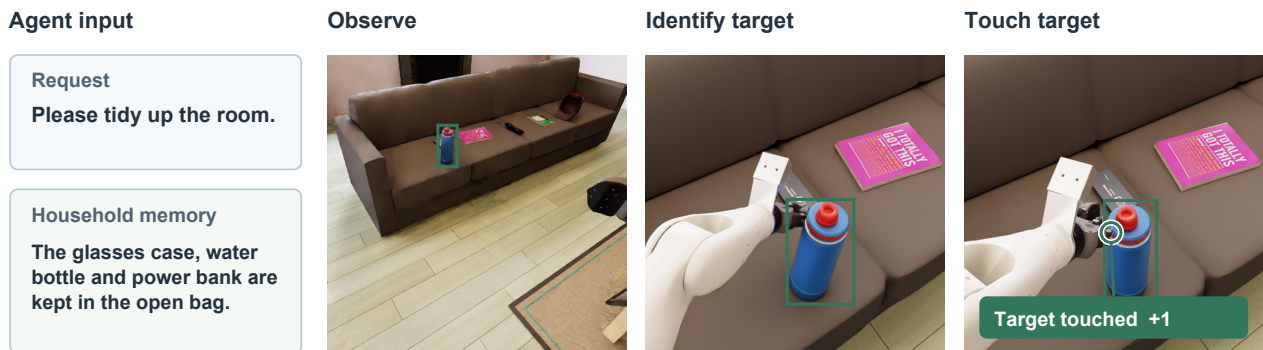
B.2 Task Discovery

Identify task-relevant objects

Task overview. We evaluate the agent’s ability to infer which household tasks are needed and identify the objects involved across four scenarios: dinner preparation, hosting guests, preparing to leave, and room tidying.

Input and setting. The agent receives household memory, a high-level scenario request, and camera observations. It explores the home, infers which objects need to be handled, and indicates them by touching each with either gripper. The request does not explicitly list the target objects.

Completion and scoring. Performance is measured by object coverage: the number of distinct target objects touched divided by the total number of targets, N . Each episode allows contact with at most $N + 2$ distinct physical objects, including targets and non-targets. Repeated contacts with the same object count once, and non-target contacts receive no credit. Only gripper-finger contacts count. The episode ends when all targets are touched, the touch budget is exhausted, or the time limit is reached. Grasping or completing the inferred household tasks is not required.



Touch budget: $N+2$ distinct objects

Figure 35. Task Discovery: identifying and touching a task-relevant object.

B.3 Context Reasoning

Task overview. We evaluate the agent’s ability to infer which object meets a contextual requirement across four tasks: drink selection, snack selection, waste selection, and sofa object selection.

Input and setting. The agent receives a task instruction, contextual information, and camera observations. The context describes user preferences, intended uses, classification rules, or updated requests. The agent must use this information to identify the appropriate object among the visible candidates.

Completion and scoring. The agent selects an object by touching it with either gripper. Touching the correct candidate earns 1 point and ends the episode; touching an incorrect candidate earns 0 and ends the episode. Simultaneous correct and incorrect contacts also fail. Episodes that time out without selecting a candidate receive zero credit. Contacts with background furniture do not count as selections. Grasping, lifting, carrying, placing, and disposal are not required.

1. Drink selection

Setting. Several drinks are stocked on a shelf. The supplied situation specifies the user’s requirements, such as flavour, sugar content, or caffeine preference, rather than a target package.

Task. Resolve the current request and select the available drink that satisfies its requirements. When the context contains competing or revised requests, follow the confirmed one.

Example. The illustrated request asks for sweet-tasting cola with zero sugar. Diet Coke is the correct choice; unsweetened sparkling water does not satisfy the cola requirement.

Task

I enjoy sweet-tasting cola, but my current preference is zero sugar. Do not confuse that with water that has no sweet taste at all. Select one shelf item that fits this situation.

Completion

Touch the Diet Coke can.



Figure 36. Context Reasoning: drink selection.

2. Snack selection

Setting. A shelf contains snacks with different flavours, textures, and ingredients, together with other stocked products. The situation describes the desired snack properties or the request that currently takes priority.

Task. Infer the requested properties and select the matching snack from the visible inventory. A broadly similar product is insufficient when it conflicts with a stated requirement.

Example. The illustrated request calls for a crunchy savoury snack with a smoky, grill-like flavour. Barbecue-flavoured crisps satisfy it; sour-cream-and-onion crisps do not match the requested flavour.

<p>Task</p> <p>For movie night, I want a crunchy savoury snack with a smoky grill-like flavour. A chewy meat stick is not what I have in mind. Select one shelf item that fits this situation.</p> <p>Completion</p> <p>Touch the barbecue-flavoured crisps.</p>	
--	--

Figure 37. Context Reasoning: snack selection.

3. Waste selection

Setting. Several candidate waste objects are placed on the coffee table. The instruction requests one category, and the accompanying context explicitly defines the recyclable and non-recyclable categories for this evaluation.

Task. Apply the supplied classification rules and touch the sole candidate in the requested category. The task tests category-based selection, not transport to a bin.

Example. The illustrated instruction requests the only recyclable item. Under the supplied rules, the dry cardboard box is correct and the used toothbrush is incorrect.

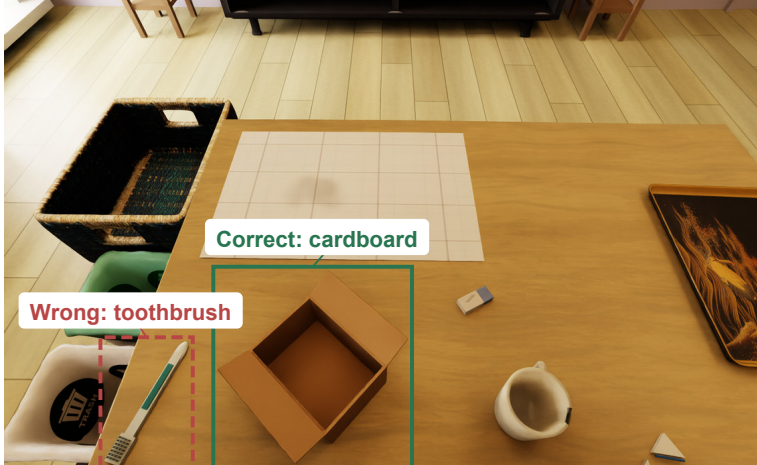
<p>Task</p> <p>Touch the only recyclable item on the coffee table with either gripper.</p> <p>For this task, clean empty glass containers, metal cookie tins, dry cardboard and newspapers are recyclable.</p> <p>Completion</p> <p>Touch the cardboard box.</p>	
--	--

Figure 38. Context Reasoning: waste selection.

4. Sofa object selection

Setting. Eight personal objects are placed on the sofa: a TV remote, a wallet, a sunglasses case, a power bank, a cap, an umbrella, a water bottle, and a green notebook. The context supplies relevant object-use facts, and the scenario describes the user’s current need.

Task. Infer which object serves that need and touch it. Follow the latest request when an earlier plan is cancelled or superseded; no sofa-to-table transfer is required.

Example. The illustrated user needs rain protection over the shoulders, not only the top of the head. The umbrella is correct; the power bank cannot meet this need.

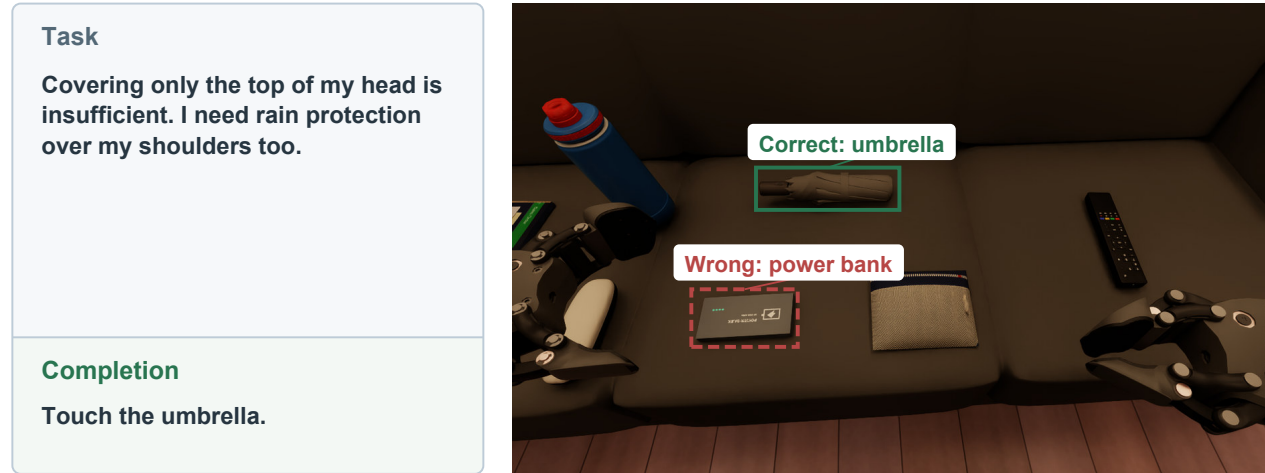


Figure 39. Context Reasoning: sofa object selection.

B.4 Failure Recovery

Task overview. We evaluate the agent’s ability to assess a failed action state, adjust its actions, and complete the task. The five settings cover missed grasps of a shelf drink, a coffee-bean bag, a TV remote, and a used cup, together with a tipped-cup state.

Input and setting. The agent receives the task instruction, camera observations, and robot-state feedback. Each episode starts from a predefined failed state, with the target still in the scene but not held. The figures pair a normal-operation reference (left) with a recovery initial state (right), using separate recordings.

Completion and scoring. Performance is measured using task-specific process scores during recovery.

1. Shelf pickup (Missed grasp)

Setting. The requested drink remains upright on the shelf after a missed grasp, with an empty gripper nearby.

Task. Adjust the grasp, lift the same drink clear of the shelf, and hold it steadily for three seconds. Tray delivery is not required.

Example. The illustrated target is a juice bottle. The left image shows it held by the gripper; in the right image, it remains on the shelf beside the empty hand.



Figure 40. Failure Recovery: normal shelf pickup (left) and a missed-grasp state (right).

2. Pour coffee beans (Missed grasp)

Setting. The coffee-bean bag remains upright on the counter beside the coffee machine after a missed grasp, with an empty gripper nearby.

Task. Adjust the grasp, pick up the bag, pour the beans into the hopper, and return the bag upright to the counter.

Example. The left image shows the bag held during normal pickup. In the right image, the empty gripper is beside the bag, which remains on the counter.



Figure 41. Failure Recovery: normal bean-bag pickup (left) and a missed-grasp state (right).

3. Move sofa items to the table (Missed grasp)

Setting. The TV remote remains in the sofa area after a missed grasp, with an empty gripper nearby. The coffee table is the destination.

Task. Adjust the grasp, pick up the same remote, and release it fully within and supported by the coffee tabletop. This recovery variant accepts any valid position on that tabletop; the standard sofa-to-table task specifies a restricted placement area.

Example. The left image shows the remote held after a normal pickup. In the right image, the remote remains in the sofa area and the nearby gripper is empty.

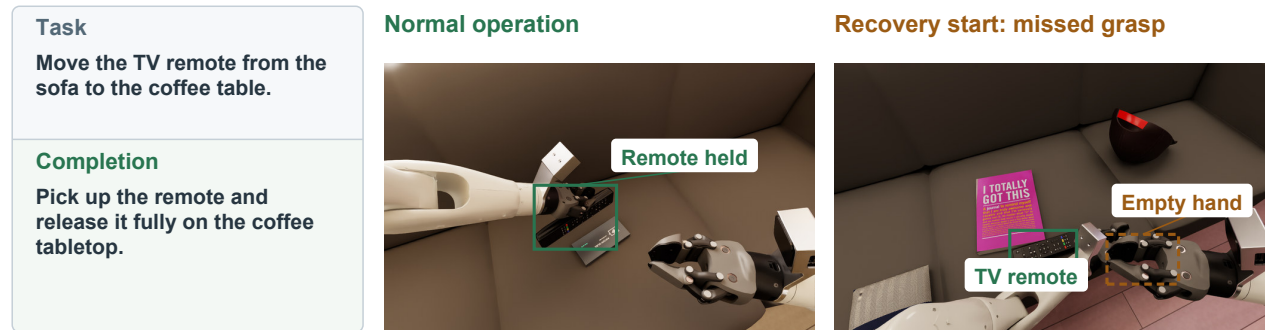


Figure 42. Failure Recovery: normal TV-remote pickup (left) and a missed-grasp state (right).

4. Clear table waste (Missed grasp)

Setting. The used cup remains upright on the coffee table after a missed grasp, with an empty gripper nearby.

Task. Adjust the grasp, pick up the same cup, and release it inside its assigned waste bin.

Example. The left image shows a normal cup pickup. In the right image, the cup remains upright on the table beside the empty gripper.

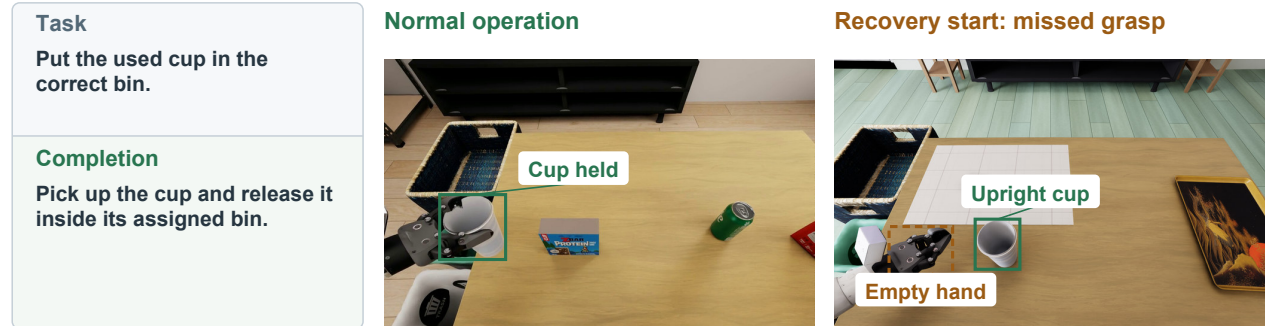


Figure 43. Failure Recovery: normal cup pickup (left) and a missed-grasp state (right).

5. Clear table waste (Tipped cup)

Setting. The used cup lies on its side on the coffee table and is not held by either gripper.

Task. Adapt the pickup to the cup's current position and orientation, then release the same cup inside its assigned waste bin.

Example. The left image shows a normal cup handover between the two grippers. The right image shows the tipped cup on the table, with both grippers empty.

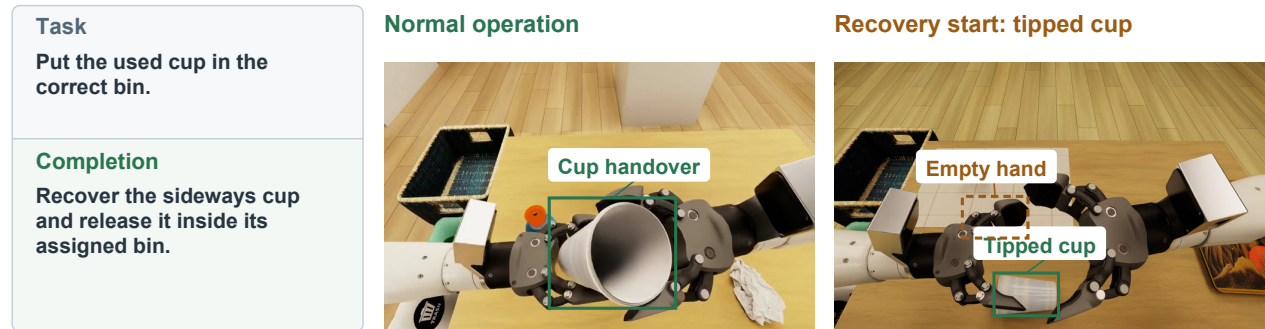


Figure 44. Failure Recovery: normal cup handover (left) and a tipped-cup state (right).

B.5 Robustness

Task overview. We evaluate the agent's ability to adapt to disturbances during manipulation. The six settings cover shelf-target relocation, shelf lighting changes, coffee-bag relocation, coffee-machine displacement, combined bag and machine movement, and lighting changes during button operation.

Input and setting. The agent receives the task instruction, camera observations, and robot-state feedback. During execution, a human hand moves a task object or the scene illumination changes. The agent must respond to the changed conditions while continuing the same task.

Completion and scoring. Performance is measured using task-specific process scores.

1. Shelf pickup (Relocated)

Setting. One product on the shelf is designated as the target. As the robot approaches, a visible human hand moves the target to another position on the same shelf level and releases it.

Task. Locate the product at its new position, grasp it, and lift it clear of the shelf. Tray placement is not required.

Example. The hand moves the target bottle away from the robot's planned grasp position. The agent must update its approach to pick up the same bottle.

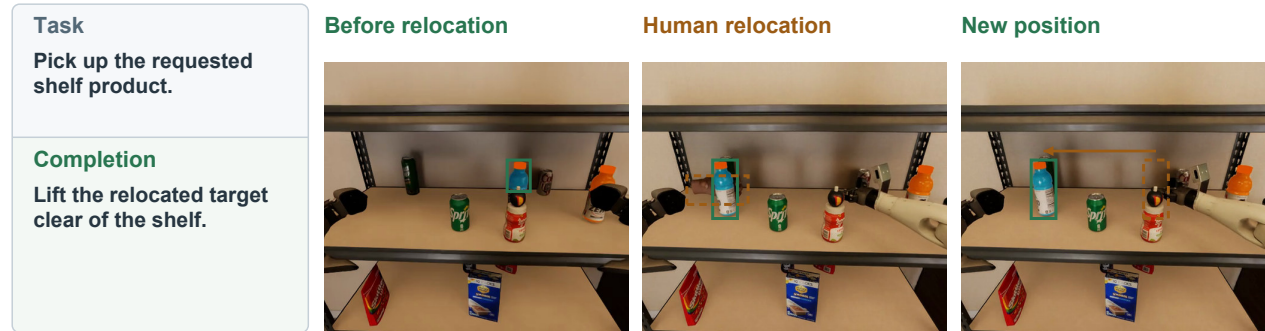


Figure 45. Robustness: shelf-target relocation by a human hand.

2. Shelf pickup (Lighting)

Setting. A single requested product is on the shelf. Light intensity alternates between its original value and one-quarter of that value at intervals of two to three seconds. Object positions remain unchanged.

Task. Locate and pick up the requested product under the changing illumination, lifting it clear of the shelf and holding it steadily for three seconds. Tray placement is not required.

Example. The figure shows the same shelf under normal and reduced illumination. The target product remains in place, but its appearance changes with the lighting.

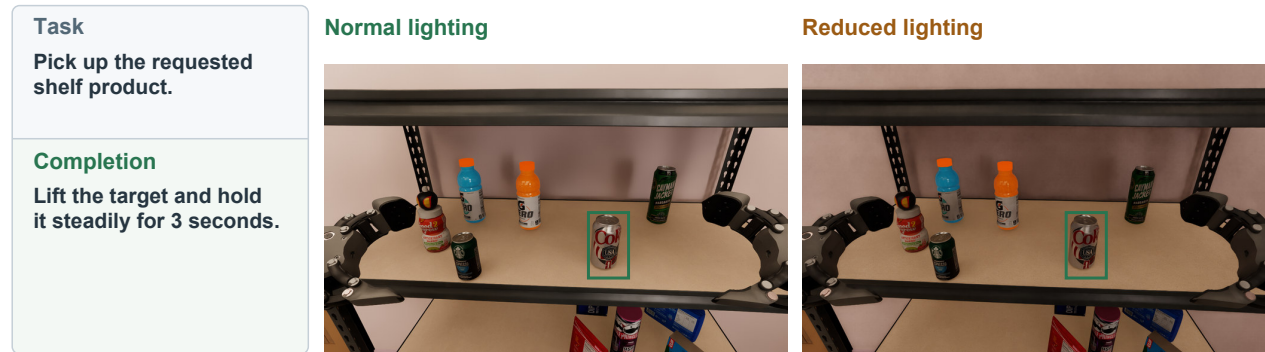


Figure 46. Robustness: the same shelf under normal and reduced illumination.

3. Pour coffee beans (Bag moved)

Setting. The coffee-bean bag is upright on the counter beside the machine. As the robot approaches to grasp it, a human hand moves the bag to another position on the counter and releases it.

Task. Locate and grasp the relocated bag, pour the beans into the hopper, and return the bag upright to the counter. Preparing or retrieving a cup of coffee is not required.

Example. The bag moves before the gripper reaches it. The agent must adjust the pickup to the new bag position before continuing with the pour.

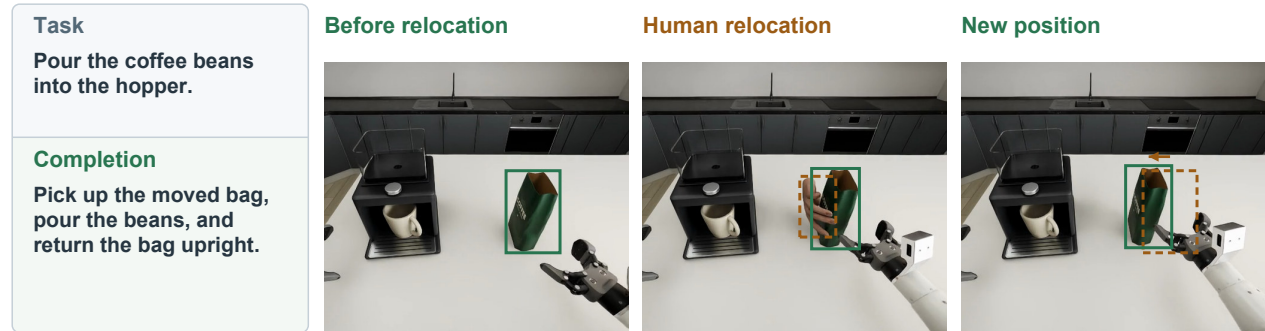


Figure 47. Robustness: coffee-bag relocation before pickup.

4. Pour coffee beans (Machine moved)

Setting. The robot holds the coffee-bean bag above the hopper. A human hand pushes the coffee machine while the bag is tilted for pouring.

Task. Track the displaced hopper, adjust the bag's position to continue pouring into it, and return the bag upright to the counter.

Example. The hopper moves away from the bag opening during the pour. The agent must realign the bag with the new hopper position rather than continue pouring at the old one.

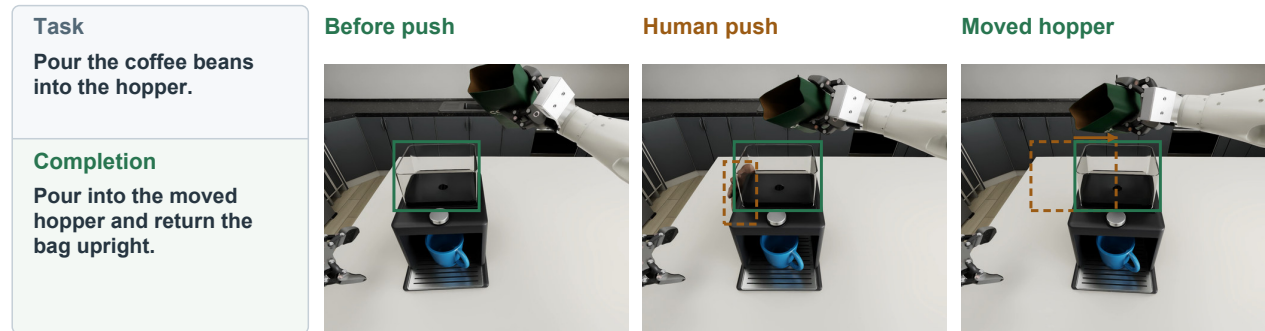


Figure 48. Robustness: coffee-machine displacement during pouring.

5. Pour coffee beans (Both moved)

Setting. Two disturbances occur in sequence: a human hand first relocates the bag before pickup, then pushes the coffee machine during pouring.

Task. Adjust the pickup to the relocated bag, realign the pour after the machine moves, and return the bag upright to the counter.

Example. Finding the moved bag resolves the first disturbance, but the agent must update its actions again when the hopper shifts during pouring.

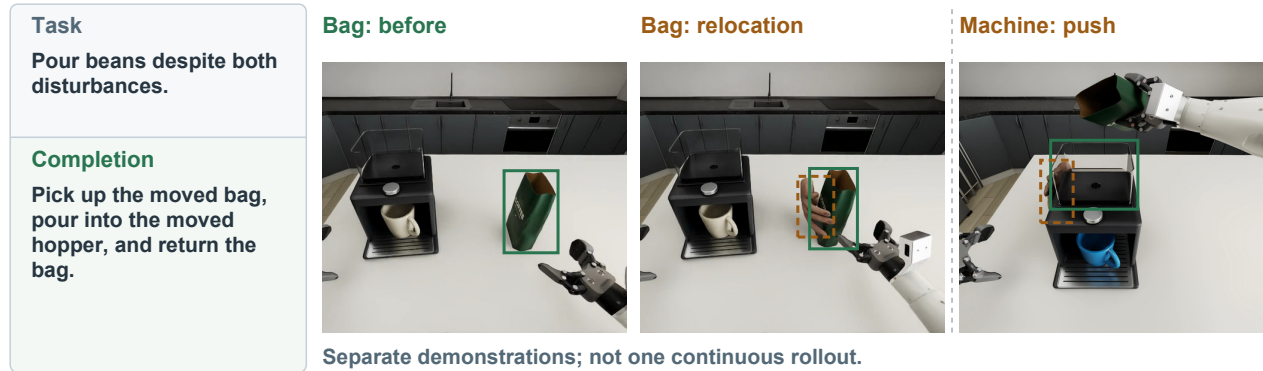


Figure 49. Robustness: bag relocation and machine displacement, illustrated using separate demonstrations.

6. Operate the coffee-machine button (Lighting)

Setting. The coffee machine and cup are ready for dispensing. Illumination changes during button operation, including changes in brightness, side lighting, and brief flashes.

Task. Locate and press the machine button to start dispensing, then press it again to stop after collecting the required amount of coffee.

Example. Side lighting or a brief flash changes the button’s appearance. The agent must still identify and operate the same control to start and stop dispensing.

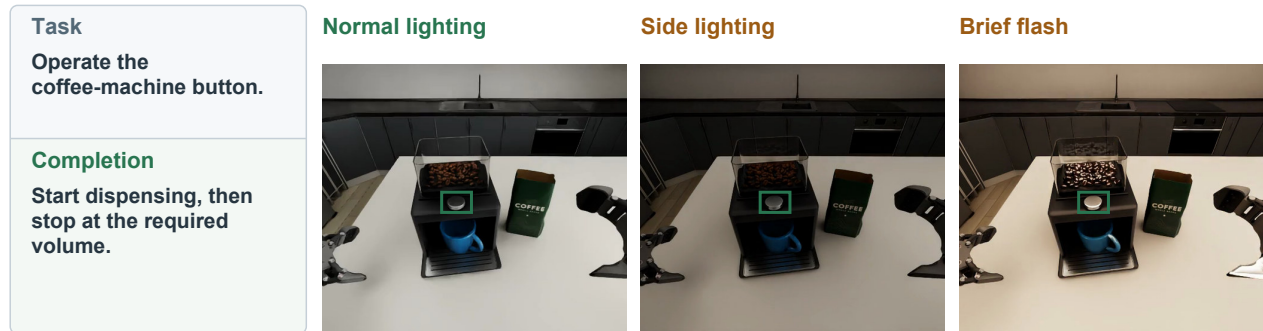


Figure 50. Robustness: the coffee-machine button under normal lighting, side lighting, and a brief flash.

B.6 Open-ended Scenarios

Task overview. We evaluate the agent’s ability to infer, plan, and execute household tasks from a high-level request in four scenarios: preparing dinner, hosting guests, preparing to leave, and tidying the room. Each episode combines multiple subtasks that must be completed while preserving earlier results.

Input and setting. Each room is generated from a selected layout and a scenario-specific task specification defining the required objects, appliances, source locations, destinations, and initial states. All subtasks share the same physical scene and object instances, without resets between subtasks. The agent receives household memory, a scenario request, camera observations, and robot-state feedback, but not the underlying subtask list or target bindings. Household memory describes personal preferences, routines, and storage arrangements. The agent combines this information with scene observations to identify the required tasks and plan their execution, respecting dependencies such as preparing food before serving it.

Completion and scoring. Native subtask scores reward verified physical progress, from target contact through manipulation and completion. Partial execution receives credit. The episode score is the sum of its subtask scores, normalized by the sum of their maxima and reported as a percentage. Completed placements remain subject to checking, so disturbing previously placed objects can reduce the score.

The episode terminates after 120 seconds of simulation time without an increase in the aggregate score. Only a score increase resets this shared timer; switching tasks or changing locations does not. Scores are finalized at termination. This timeout limits inactivity rather than total episode duration.

1. Prepare dinner

Setting. The request identifies the person joining for dinner. Household memory describes their usual meal and accompaniments, while the scene shows the available food, serving equipment, and clutter in the eating area.

Task. Prepare the appropriate meal and accompaniments, serve them together on the large meal tray at the coffee table, and clear misplaced belongings and waste from the seating and serving areas.

Example. Ryan's usual dinner includes a warmed beef meal, a fork and knife, and freshly prepared coffee. The illustrated example combines meal and coffee preparation, tray serving, and clearing a book, a remote, and a used cup.

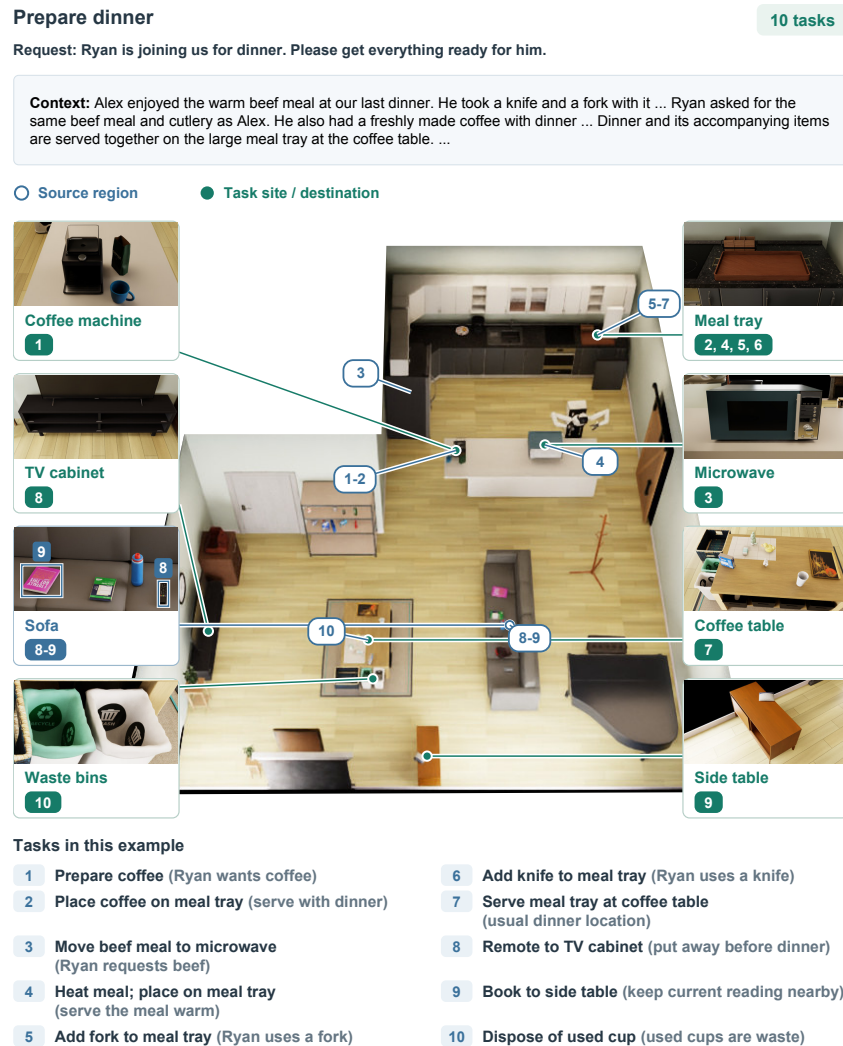


Figure 51. An example of dinner preparation with coffee.

2. Host guests

Setting. The request identifies the visitor but does not name the refreshments to prepare. Household memory records the visitor's usual snack and drink. Products are stocked on the shelf, and belongings or used packaging may occupy the seating and serving areas.

Task. Select the visitor's usual refreshments and serve them together on the wooden tray. Return misplaced belongings to their designated locations and dispose of waste in the appropriate bins. Other stocked products should remain on the shelf.

Example. Alex usually has Sprite and Oreos. The illustrated example combines serving these refreshments with moving the book to the side table, hanging the cap, returning the remote to the TV cabinet, storing the wallet in the basket, and disposing of a used can and cup.

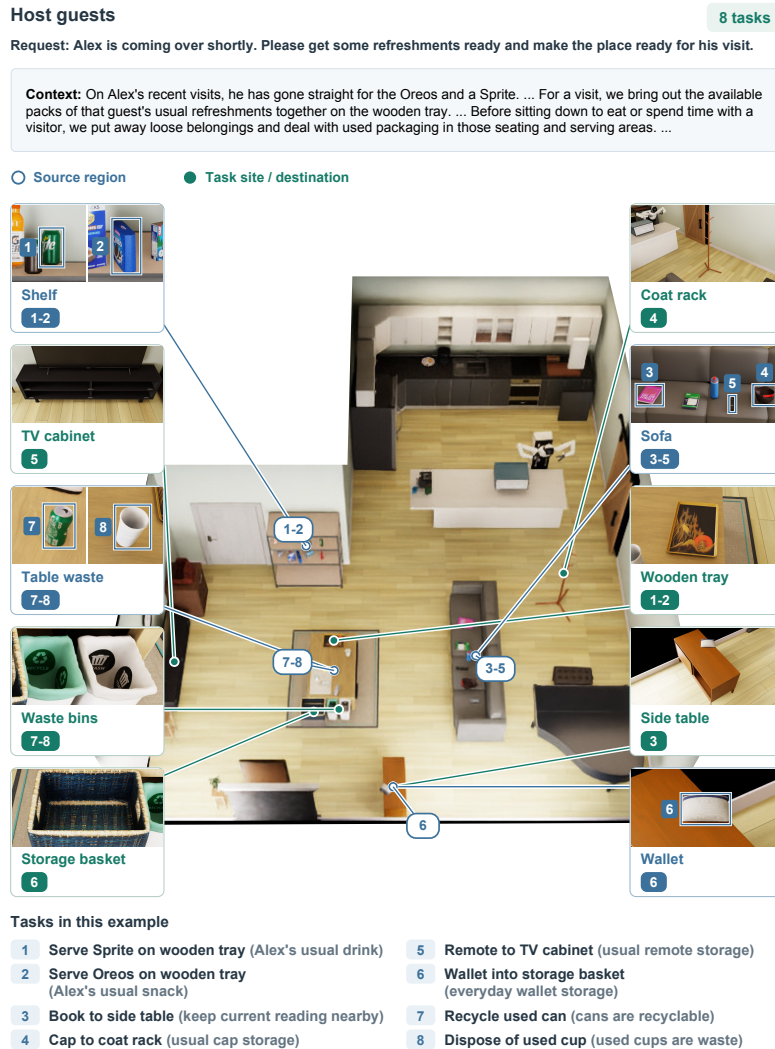


Figure 52. An example of preparing for a guest visit.

3. Prepare to leave

Setting. The request describes who is going out and the purpose of the outing. Household memory specifies the belongings needed for that activity and where other objects belong. The scene shows which items remain unpacked or misplaced.

Task. Collect the required belongings and pack them in the open bag. Put away other misplaced objects, remove waste, and clean any table stains. Items already packed or correctly stored need no further transfer.

Example. Emma needs her wallet and power bank when working away from home. In the illustrated example, the water bottle and glasses case also need returning to their usual storage location in the open bag. The remaining tasks are to return the remote, store the notebook and close the drawer, and dispose of paper waste and a snack wrapper.

Prepare to leave

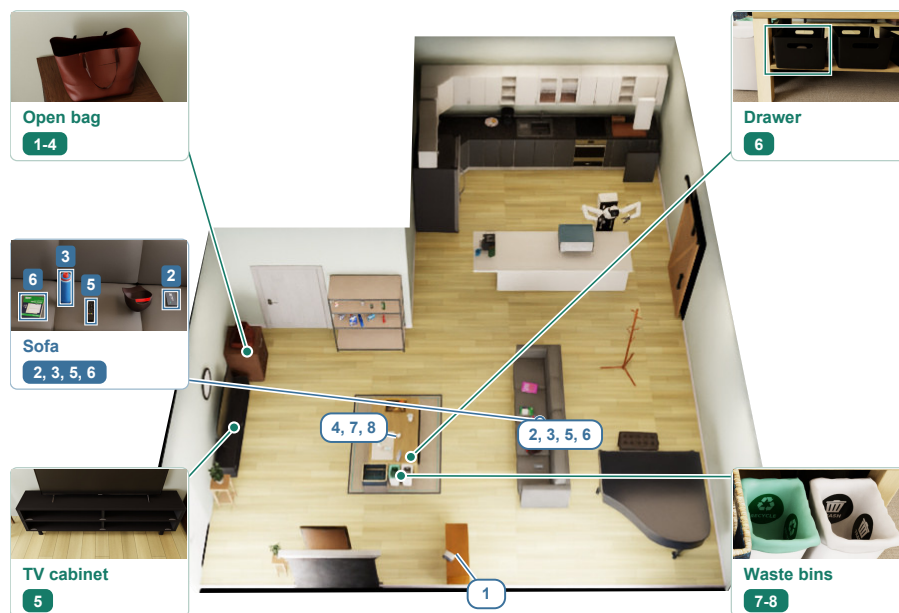
8 tasks

Request: Emma is going out to work away from home today. Could you help her get ready?

Context: Emma uses the open bag when she works away from home. She checks for her wallet and power bank before leaving ... The glasses case, water bottle and power bank are kept in the open bag. ... The TV remote belongs on the TV cabinet. ... The green notebook is kept in the coffee-table drawer, with the drawer closed afterward. ...

○ Source region

● Task site / destination



Tasks in this example

- | | |
|--|---|
| 1 Pack wallet in open bag (Emma's work checklist) | 5 Remote to TV cabinet (usual remote storage) |
| 2 Pack power bank in open bag (Emma relies on her phone) | 6 Notebook into drawer; close it (usual notebook storage) |
| 3 Water bottle into open bag (usual bottle storage) | 7 Dispose of paper waste (paper goes in waste) |
| 4 Glasses case into open bag (usual case storage) | 8 Dispose of snack wrapper (wrappers go in waste) |

Figure 53. An example of preparing to leave.

4. Tidy the room

Setting. Personal belongings and waste are scattered around the room, and the coffee table may have stains. Household memory specifies where belongings should be stored and how waste should be sorted.

Task. Return misplaced belongings to their designated locations, dispose of waste in the appropriate bins, and clean any table stains with the brush, returning it afterward. Objects already in their correct locations should remain undisturbed.

Example. The illustrated example includes returning a book, cap, remote, notebook, wallet, and glasses case to their designated locations, together with disposing of a used cup and recycling an empty can. No table cleaning is required in this example.

Tidy the room

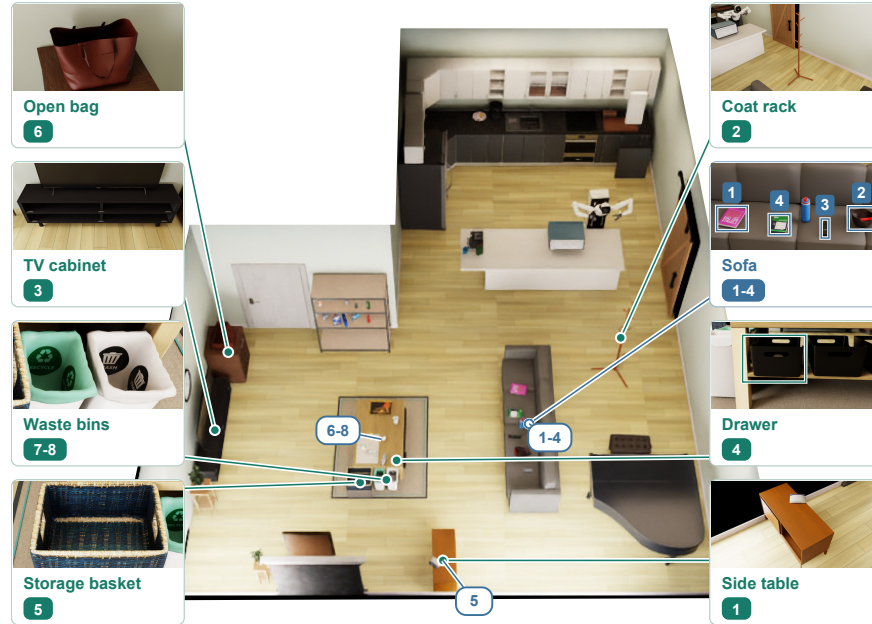
8 tasks

Request: Please tidy up the room.

Context: Books that are still being read belong on the small side table ... The TV remote belongs on the TV cabinet. Caps go on the coat rack ... The green notebook is kept in the coffee-table drawer ... the wallet stays in the storage basket beside the bins. The glasses case, water bottle and power bank are kept in the open bag. ... Empty drink cans go in the recycling bin. ...

○ Source region

● Task site / destination



Tasks in this example

- | | |
|---|--|
| 1 Book to side table (keep current reading nearby) | 5 Wallet into storage basket (everyday wallet storage) |
| 2 Cap to coat rack (usual cap storage) | 6 Glasses case into open bag (usual case storage) |
| 3 Remote to TV cabinet (usual remote storage) | 7 Dispose of used cup (used cups are waste) |
| 4 Notebook into drawer; close it (usual notebook storage) | 8 Recycle used can (cans are recyclable) |

Figure 54. An example of room tidying.

C Policy training and model-specific adaptation

This section describes how we fine-tune and adapt the eleven policy baselines to the AgiBot G1, covering shared training settings and model-specific configurations.

C.1 Shared training setup

Training data and split. We fine-tune all eleven policy baselines using demonstrations from the 101.8-hour RobotAgentBench simulation dataset, covering atomic actions, short tasks, long tasks, and open-ended household scenarios. All models use the same episode-level training/validation split.

Data representation and preprocessing. Demonstrations follow a unified format with 23-dimensional robot states and 22-dimensional whole-body actions, as defined in Table 4. We adapt the data to each model’s required input format, including image resizing, normalization, and multi-view arrangement. State and action normalization statistics are computed from the training split. Image augmentations are applied only during training. Model-specific adaptations are described below.

Fine-tuning and sampling. We follow each model’s native fine-tuning recipe, retaining its prescribed frozen components. In selected continuation phases, 80% of training samples come from a combined pool of short- and long-task demonstrations, and 20% from atomic-action demonstrations.

Training-budget convention. Table 5 reports trainable parameter counts, final-phase global batch sizes, and total optimizer updates. Global batch sizes include gradient accumulation, and each training step corresponds to one optimizer update.

Model	Trainable (B)	Final-phase batch	Optimizer updates
$\pi_{0.5}$	3.35	1,024	16,000
Xiaomi Robotics-1	4.72	56	70,000
InternVLA-A1.5	2.68	64	120,000
MolmoAct2	5.05	32	100,000
LingBot-VLA2	6.38	256	60,000
LingBot-VA	5.09	128	16,000
HY-VLA	4.53	128	18,000
Galaxea G0.5	2.86	128	28,000
X-VLA	0.88	768	33,500
τ_0 low-level policy	2.50	256	35,000
OpenWAM-Alpha	6.02	1,280	49,203

Table 5. Fine-tuning configurations. Parameter counts cover the trainable policy scope in billions. Batch sizes describe the final phase, rather than the full training history.

C.2 Model-specific settings

$\pi_{0.5}$. We fine-tune from `pi05_base` for 16,000 optimizer updates, with a final-phase global batch size of 1,024. The G1 adapter maps the benchmark’s state and action channels to the model’s native format.

Xiaomi Robotics-1. We fine-tune from the released Xiaomi Robotics-1 base checkpoint for 70,000 optimizer updates with a global batch size of 56, adapting the model to the G1 whole-body state and action interface.

InternVLA-A1.5. We fine-tune from `InternRobotics/InternVLA-A1.5-base` for 120,000 optimizer updates with a global batch size of 64. We retain the model’s native training objectives and frozen Wan teacher.

MolmoAct2. We fine-tune from `allenai/MolmoAct2` for 100,000 optimizer updates, with a final-phase global batch size of 32. Training retains both continuous and discrete action objectives, with LoRA disabled.

LingBot-VLA2. We fine-tune from `robbyant/lingbot-vla-v2-6b` for 60,000 optimizer updates, with a final-phase global batch size of 256. The adapter maps the benchmark’s state and action channels to the model’s native representation.

LingBot-VA. We fine-tune from `robbyant/lingbot-va-base` for 16,000 optimizer updates with a global batch size of 128. The policy conditions on RGB observations and language without proprioceptive input, and its action output is adapted to the 22 G1 command channels.

HY-VLA. We fine-tune from `tencent/Hy-Embodied-0.5-VLA-UMI` for 18,000 optimizer updates, with a final-phase global batch size of 128. We retain its six-frame image-history mechanism, sampling historical images every five source frames.

Galaxea G0.5. We fine-tune from the released G0.5 base policy for 28,000 optimizer updates, with a final-phase global batch size of 128. We retain its released action-tokenizer codec and adapt the whole-body channels to its native action representation.

X-VLA. We fine-tune from 2toINF/X-VLA-Pt for 33,500 optimizer updates, with a final-phase global batch size of 768. The backbone remains frozen for the first 1,000 updates, following the initial training recipe.

τ_0 low-level policy. We fine-tune the low-level action policy from sii-research/tau-0-v1a for 35,000 optimizer updates, with a final-phase global batch size of 256. This configuration evaluates the action policy rather than the complete world-model hierarchy.

OpenWAM-Alpha. We fine-tune from the released OpenWAM-Alpha Pretrain Foundation Model for 49,203 optimizer updates, with a final-phase global batch size of 1,280. Training retains the joint video-and-action objective and frozen auxiliary encoders. The three camera views are combined into a single image canvas.

D Detailed Analysis of Open-ended Scenarios

We complement scenario scores with behavioural metrics and an analysis of how agents distribute their interactions and respond to execution difficulties.

D.1 Behavioural metrics

Task Coverage. Task Coverage measures the proportion of required tasks physically engaged by the agent:

$$\text{Coverage} = 100 \times \frac{\sum_e |I_e|}{\sum_e |T_e|},$$

where T_e is the required task set in episode e and $I_e \subseteq T_e$ is the engaged subset. Engagement requires eligible gripper-finger contact with a registered target or a task-specific operation-start event. Each task counts once; workspace selection and background monitoring do not count.

Task Progress. Task Progress measures process-score attainment among engaged tasks:

$$\text{Progress} = 100 \times \frac{\sum_e \sum_{t \in I_e} p_{e,t}}{\sum_e \sum_{t \in I_e} p_{e,t}^{\max}},$$

where $p_{e,t}$ and $p_{e,t}^{\max}$ are the process score at termination and its maximum. Target-contact credit is included, while final-state score channels and unengaged tasks are excluded.

Recovery Rate. Recovery Rate is the percentage of observable physical-failure segments followed by task-specific success of the same task at a later control step. Repeated failures before recovery form one segment. API errors, formatting errors, and termination-only settlement events are excluded.

For these three metrics, numerators and denominators are pooled across scenarios before computing percentages. Zero denominators are reported as undefined.

Stagnation. Stagnation measures the fraction of simulation time spent in repeated physical execution without progress. An interval requires at least three executions that advance simulation; only time beyond a 30-second grace period counts. Task-score increases, manipulation progress, or newly explored map area interrupt the interval. Successful navigation exceeding 2 cm translation or 0.035 rad rotation is excluded.

State loops also qualify when a cycle of at most eight calls repeats at least three times over at least 30 seconds without task or map progress; the first cycle is excluded. Matching tolerances are 2 cm for base position, 0.035 rad for joints and base yaw, and 1 cm/0.052 rad for hand position/orientation. Overlapping intervals count once. Time follows the 20 Hz simulation clock, excluding API waiting.

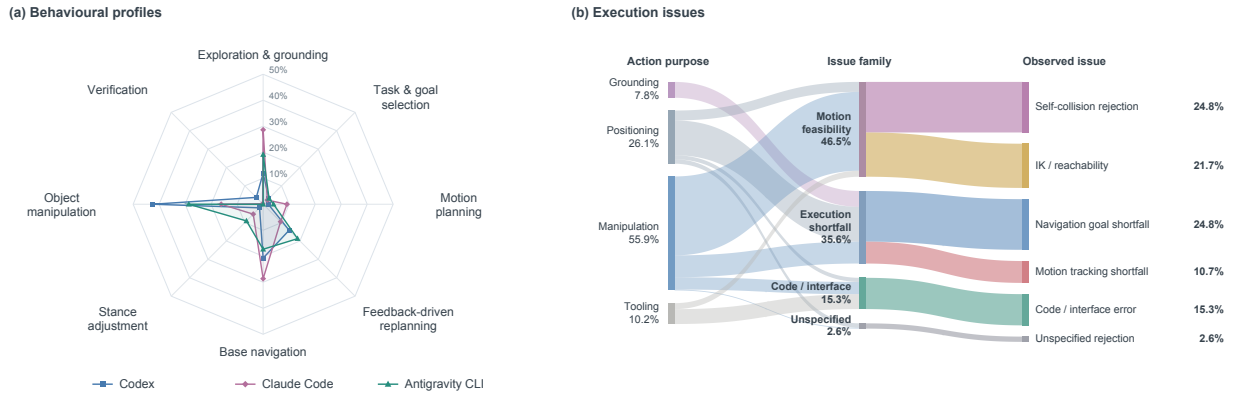


Figure 55. Behavioural patterns and execution issues in open-ended scenarios. (a) Scenario-balanced task-event distributions. (b) Identified execution issues grouped by action purpose and issue family. Percentages are rounded independently.

D.2 Behavioural patterns and execution issues

Analysis protocol. We analyse interaction traces of Codex, Claude Code, and Antigravity CLI across the four open-ended scenarios. Figure 55(a) groups submitted action attempts and deduplicated public planning statements into eight behavioural categories. Compound scripts count once, and verification includes only standalone checks. Task-event shares exclude setup and unclassified events and are averaged equally across scenarios after within-episode normalization.

Panel (b) summarizes identified execution issues. Calls are weighted inversely by each episode’s submitted-action count, with equal agent and scenario weights before normalization over issue-bearing calls. Each call receives at most one issue label. Navigation shortfalls require residual translation exceeding 2 cm or rotation exceeding 0.02 rad. These percentages describe issue composition rather than the overall error rate.

Behavioural patterns. Object manipulation accounts for 42.4% of Codex’s task events, compared with 16.1% for Claude Code and 28.6% for Antigravity CLI. Claude Code instead concentrates more interactions on exploration, grounding, and base navigation, which together account for 57.3% of its events.

For all three agents, feedback-driven replanning is more frequent than explicit task and goal selection. Observed adjustments include smaller approach steps and revised wrist orientations, illustrating the recurring need to update actions using execution feedback.

Execution difficulties. Motion-feasibility issues and execution shortfalls together account for 82.1% of identified issues, compared with 15.3% for code or interface errors. These difficulties span both positioning and object manipulation, suggesting that reliable execution requires more than correct tool calls.

Agents need to coordinate navigation, stance adjustment, and manipulation, and revise unsuccessful approaches using execution feedback. Such revisions should be followed by verification that the intended physical progress has occurred.