LIBERO-AGENT: EVALUATING GENERAL-PURPOSE AGENTS FOR DIRECT EMBODIED MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

General-purpose agents can plan, use tools, and revise their behavior from feedback, but it remains unclear whether these capabilities transfer from digital environments to embodied manipulation. To investigate this question, we introduce **LIBERO-Agent**, an agent-native benchmark for evaluating these agents in robot manipulation tasks. Rather than asking agents to submit task-level Python control programs or operate through high-level robot skills, LIBERO-Agent provides an interactive robotic environment where agents can select which observations to inspect, process them with their own tools, and issue native action commands. LIBERO-Agent integrates 200 tasks into a common interaction framework and provides a 30-task primary suite that separates perception, short-horizon execution, and long-horizon composition. Results reveal a pronounced reliability gap: while agents perform well on perception and easy short-horizon tasks, their performance degrades substantially on hard short-horizon and long-horizon tasks. Richer observations improve short-horizon manipulation, while demonstration benefits depend on the agent and format. Among these agents, GPT-6 Astra achieves the strongest overall performance. Further analysis shows its major advantage lies in mechanism interaction, especially when sustained physical contact is needed, while its remaining failures stem from cross-stage interference and geometric errors.

1 INTRODUCTION

Large language models have evolved from conversational systems (Ouyang et al., 2022) into general-purpose agents that use tools and carry out complex tasks. Across mathematical reasoning (Cobbe et al., 2021), code generation (Jimenez et al., 2024; Huang et al., 2026), web browsing (Zhou et al., 2024; Koh et al., 2024), and computer use (Xie et al., 2024; Rawles et al., 2025), agents can plan toward a goal, revise their plans from feedback, and keep using tools over long interactions. These advances call for more demanding evaluation settings that test whether agents can autonomously organize perception, reasoning, and action throughout an interactive task.

Embodied robot manipulation (Liu et al., 2023; James et al., 2020; Chen et al., 2025; 2026) is a natural setting for such evaluation. An agent must ground instruction-relevant objects in the scene and decide further actions from visual and physical feedback. It must also handle potential occlusion and execution errors while tracking a changing task state under a time budget. Simulation makes these interactions reproducible and scalable. However, existing manipulation benchmarks often do not expose all of these choices to the evaluated agent: observations may be preselected, or physical execution may be abstracted behind benchmark-defined task-level action commands. When adapted to general-purpose agents, such scaffolding can decide what information the agent receives or how actions are composed, making it difficult to determine which embodied capabilities come from the agent itself rather than from the benchmark interface. Moreover, existing benchmarks are rarely structured to diagnose agent capabilities across perception, short-horizon execution, and long-horizon composition, making it difficult to localize where failures arise.

We define *agent-native embodied evaluation* as a setting that makes these choices part of the agent's responsibility: the benchmark specifies the available observations, native robot-control interface, and resource budgets, while leaving observation selection and processing, task decomposition, and action composition to the agent. To support this setting, we adapt LIBERO (Liu et al., 2023), a

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

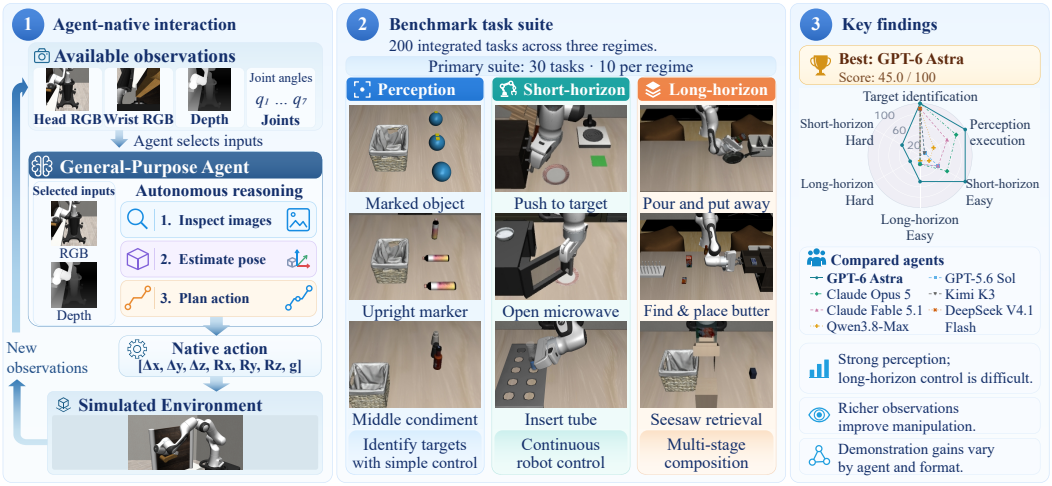

Figure 1: Overview of LIBERO-Agent: agent-native interaction, benchmark task suite, and key findings.

widely used simulation benchmark for robot manipulation, through a minimal interface that exposes episode control and bounded sequences of native end-effector commands. Agents independently choose which available observations to inspect, process them using their own tools, and compose actions without predefined task-level primitives. All interactions are subject to fixed action and time budgets.

Leveraging LIBERO's compatibility with related manipulation environments, we integrate representative tasks from several benchmarks into a common framework comprising *200* tasks. From this collection, we select 30 representative tasks as our primary evaluation benchmark and separate them into three aspects of embodied competence: perception, short-horizon manipulation, and long-horizon manipulation. The perception tasks focus primarily on target identification and therefore use only deliberately simple actions. They cover diverse perceptual signals involving object identity, appearance, state, spatial relations, and physical properties. The short-horizon tasks minimize perceptual ambiguity and long-term planning demands to focus on manipulation execution. Example tasks include pick-and-place, pushing, button pressing, knob rotation, and articulated-fixture manipulation. The long-horizon tasks compose these abilities across multiple stages, adding requirements for instruction decomposition, scene exploration, cross-stage constraints, and recovery from intermediate failures.

Using this benchmark, we compare seven general-purpose agents under a common protocol and study how different observation inputs and demonstration context affect manipulation. GPT-6 Astra achieves the highest Performance Score of 45.0/100. It succeeds on all perception and easy short-horizon tasks, but drops to 40% on hard short-horizon manipulation and 22% stable stage completion on hard long-horizon tasks. Richer observations improve short-horizon manipulation across Astra, Opus, and Fable, while demonstration benefits depend on both the agent and the representation format. Further analysis shows that Astra's advantage is strongest in mechanism interaction and is associated with sustained physical contact. Once local execution improves, failure modes shift toward cross-stage interference and geometric errors.

Our contributions are threefold:

- **An agent-native embodied evaluation protocol.** We introduce a minimal interface that leaves observation selection, information processing, task decomposition, and native action composition to the agent.
- **A capability-factorized manipulation benchmark.** Our framework supports *200* tasks, with a 30-task primary suite covering perception, short-horizon execution, and long-horizon composition.
- **An empirical analysis of embodied agent capabilities.** We compare seven agents and show that the strongest agent's advantage concentrates on mechanism interaction, where

sustained contact is strongly associated with success. Its remaining failures shift toward cross-stage interference and geometric errors.

2 Related Work

2.1 Benchmarks for General-Purpose Agents

Recent benchmarks increasingly evaluate language models as interactive agents. General-purpose suites such as AgentBench (Liu et al., 2024), AgentBoard (Ma et al., 2024), and GAIA (Mialon et al., 2024) test multi-step reasoning and tool use across diverse tasks. WebArena (Zhou et al., 2024), VisualWebArena (Koh et al., 2024), and WorkArena (Drouin et al., 2024) focus on evaluating agents under diverse executable web environments. OSWorld (Xie et al., 2024) and AndroidWorld (Rawles et al., 2025) extend this setup to desktop and mobile computer use. SWE-bench (Jimenez et al., 2024) and AppWorld (Trivedi et al., 2024) test interaction with codebases and application APIs. Together, these benchmarks emphasize multi-step interaction, feedback, and execution-based outcomes.

We bring this principle to robot manipulation, where benchmark-side perception and control abstractions can themselves supply part of the capability being evaluated. LIBERO-Agent therefore targets an *agent-native* setting that leaves observation processing, task decomposition, and native action composition to the agent itself.

2.2 Embodied Manipulation Benchmarks

Robot manipulation benchmarks provide scalable and reproducible testbeds for evaluating policies across diverse skills, scenes, and horizons (Yu et al., 2020; James et al., 2020; Mees et al., 2022; Liu et al., 2023; Gu et al., 2023; Nasiriany et al., 2024). More recent benchmarks broaden capability coverage toward language-conditioned reasoning (Zhang et al., 2025), memory-dependent tasks (Lei et al., 2026), robust bimanual generalization (Chen et al., 2025), and compositional long-horizon execution (Yang et al., 2026). These benchmarks have substantially advanced robot-policy evaluation, but typically assume a fixed observation interface. In parallel, foundation-model approaches such as Code as Policies (Liang et al., 2023) and VoxPoser (Huang et al., 2023) generate programs or spatial value maps executed through separate perception and control modules, separating high-level reasoning from low-level physical execution.

Among existing works, EmbodiedBench (Yang et al., 2025) and CaP-X (Fu et al., 2026) are most closely related to our setting. EmbodiedBench evaluates multimodal language models on manipulation, but reduces the burden of physical grounding and control through discretized actions and additional object-level perceptual information. CaP-X evaluates coding agents by synthesizing programs over predefined robot perception and control primitives, and its seven-task core benchmark primarily studies the effects of abstraction, interaction, and perceptual grounding. LIBERO-Agent instead evaluates agents through direct embodied interaction: agents select and process observations themselves and compose bounded native action sequences that act on the simulator. Our 30-task primary suite further factorizes evaluation into perception, short-horizon execution, and long-horizon composition, enabling capability diagnosis under the same interaction protocol.

3 Benchmark

3.1 Agent–Simulation Interaction Protocol

We evaluate agents through iterative interaction with a simulated manipulation environment. At the beginning of each episode, an agent receives the task instruction, the available observation modalities, and a fixed interaction budget. The agent may inspect and process the available observations using its own tools, maintain information from previous interactions, and decide what to inspect, how to process it, and what actions to take.

Interaction proceeds by alternating between agent decisions and environment execution. At each interaction step, the agent may submit a bounded sequence of robot actions. The simulation environment executes the sequence and returns updated observations. The agent can then use these

new observations to continue working toward the task goal. No task-level manipulation primitives or privileged simulator state are exposed to the agent. Thus, the benchmark fixes the interaction interface and budgets, while leaving observation selection and action composition to the agent.

Episodes terminate when the agent explicitly finishes the task or exhausts the interaction or time budget. Task success is determined independently by the simulation environment. For tasks with process-level requirements, private evaluators additionally track relevant physical events, ordering constraints, and required prior events.

3.2 Simulation Implementation

We implement the interaction protocol on LIBERO (Liu et al., 2023) through a minimal Model Context Protocol (MCP) interface with three tools. `start_episode` initializes an episode, returns the task instruction and resource budgets, and publishes the initial observation. `osc_sequence` executes a bounded sequence of robot actions and publishes the resulting observation, while `finish_episode` terminates the episode and returns the final evaluation outcome.

The simulation is built on MuJoCo (Todorov et al., 2012) with a 7-DoF Franka Emika Panda robot. Each `osc_sequence` call accepts a bounded sequence of native `OSC_POSE` actions. An action is represented by $[\Delta x, \Delta y, \Delta z, r_x, r_y, r_z, g] \in [-1, 1]^7$, corresponding to relative end-effector translation, rotation, and gripper control. Translation and rotation components are scaled by 0.05 m and 0.5 rad, respectively, and the gripper command uses $g = -1$ for opening and $g = +1$ for closing. Submitted actions are executed by the environment's native `OSC_POSE` controller.

By default, observations available to agents include third-person and wrist-view RGB images, metric depth, camera calibration, and robot state such as joint and end-effector states, velocities, commanded torques, and force/torque measurements. These observations are published to an episode-local workspace, where agents may inspect files, perform geometric computation, and store information across interactions. Each agent rollout is subject to fixed action-submission and wall-clock budgets. A separate evaluation process maintains private task checkers and event traces.

Leveraging LIBERO's compatibility with related manipulation benchmarks, we integrate tasks and assets from LiLo-VLA (Yang et al., 2026), RoboMemArena (Lei et al., 2026), and VLABench (Zhang et al., 2025), resulting in a 200-task pool under a shared interaction interface.

3.3 Task Design

From the 200-task pool supported by the simulation runtime, we select 30 representative tasks as the primary evaluation suite. We organize them into three task groups: **Perception-Focused Manipulation**, **Short-Horizon Manipulation**, and **Long-Horizon Manipulation**, with ten tasks in each group. Together, they separate three questions: can the agent identify the right target, execute the intended physical interaction, and compose these abilities across multiple stages? Short- and long-horizon tasks are further divided into easy and hard subsets based on difficulty.

Perception-Focused Manipulation. These tasks simplify physical interaction so that the main challenge lies in identifying task-relevant information. The ten tasks cover object pose, spatial relations, color, texture, instance markings, shape, semantic category, size, reflectance, and physical properties. Representative examples include identifying an upright marker, an object between two others, a different object, and the heaviest among visually identical objects. Perception is evaluated through embodied execution rather than question answering: agents must act on their judgments under a simple control requirement.

Short-Horizon Manipulation. These tasks specify both the relevant object and manipulation objective so that the main question is whether a correct target and goal can be turned into a reliable physical state change. We further divide them into easy and hard subsets according to execution difficulty: easy tasks allow relatively direct interactions and broader tolerances, whereas hard tasks require tighter geometric alignment, direction-sensitive motion, sustained contact, or more dexterous interaction. The easy subset consists of pick-and-place, button pressing, opening and closing a drawer, and pouring. The hard subset contains direction-sensitive pushing, opening a microwave, operating a stove control, precise tube insertion, and wiping a spill.

Long-Horizon Manipulation. These tasks compose the perception and manipulation abilities above across multiple stages, while adding planning and task-state tracking. We divide them into easy and hard subsets according to cross-stage difficulty: easy tasks mainly sequence familiar subgoals with limited cross-stage dependence, whereas hard tasks require stronger goal inference, causal dependencies, persistent physical constraints, or mechanism reasoning. Representative easy tasks include repeated pouring or transferring multiple objects, while hard tasks include reconstructing a scene from a goal image and using a mechanism such as a rotary latch or seesaw to complete the task. These tasks test whether agents can maintain a coherent task state and satisfy constraints over extended execution.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Agents and primary evaluation. We evaluate seven general-purpose agents: GPT-6 Astra (OpenAI, 2026a), GPT-5.6 Sol (OpenAI, 2026b), Claude Opus 5 (Anthropic, 2026b), Claude Fable 5.1 (Anthropic, 2026a), DeepSeek-V4.1-Flash (Xu et al., 2026), Kimi K3 (Team et al., 2026), and Qwen3.8-Max (Alibaba Cloud, 2026). Each evaluated agent is a model–harness pair, with reasoning effort set to high where supported. Our primary evaluation uses the 30-task suite in Section 3.3 with the full observation profile and no demonstrations. We evaluate each task with three rollouts using the same task-specific initial state and simulator seed. Each rollout has a 1,800-second wall-clock limit and a fixed action-submission budget. A single `osc_sequence` submission may contain between 1 and 50 native `OSC_POSE` actions. Third-person and wrist-view RGB and depth observations are rendered at 256×256 resolution.

Metrics. We report *stable* performance across three rollouts. For perception and short-horizon tasks, let $y_{i,r} \in \{0, 1\}$ denote full-task success for task i in rollout r. For a long-horizon task i with K_i stages, let $c_{i,r}$ denote the number of completed stages in rollout r. Stable success rate and stable stage completion over a task set $\mathcal{T}$ are

$$\mathrm{SR}_{\mathrm{stable}}(\mathcal{T}) = \frac{100}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} \prod_{r=1}^{3} y_{i,r}, \qquad \mathrm{SC}_{\mathrm{stable}}(\mathcal{T}) = \frac{100}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} \frac{\min_{r \in \{1,2,3\}} c_{i,r}}{K_i}, \tag{1}$$

respectively. Perception judgment accuracy (Judg. Acc.) follows the same stable criterion using correct target identification instead of full-task success. Thus, $\mathrm{SR}_{\mathrm{stable}}$ requires full success in all three rollouts, while $\mathrm{SC}_{\mathrm{stable}}$ preserves partial progress while requiring it to be reproducible. Tasks are equally weighted within each subset. The 100-point overall Performance Score is

$$\mathrm{Score} = 5J + 10P + 5M_{\mathrm{easy}} + 10M_{\mathrm{hard}} + 20L_{\mathrm{easy}} + 50L_{\mathrm{hard}}, \tag{2}$$

where J and P denote stable perception judgment and task success, M denotes stable short-horizon success, and L denotes stable long-horizon stage completion, with all components normalized to $[0, 1]$. The weighting emphasizes harder and longer-horizon tasks. Resource usage is averaged over all 90 primary episodes per agent.

4.2 MAIN RESULTS

Correct identification does not ensure reliable physical execution. Table 1 reveals a pronounced gap between correct target identification and successful execution. GPT-5.6 Sol, DeepSeek V4.1 Flash, Kimi K3, and Qwen3.8-Max all achieve 90% stable judgment accuracy, yet their perception-focused task success is only 10%, 0%, 10%, and 30%, respectively. The same weakness carries over to short-horizon manipulation: their easy-task success rates are only 40%, 0%, 0%, and 20%, and none succeeds stably on any hard short-horizon task. Thus, these agents often identify *what* to manipulate but fail to turn that decision into a reliable physical state change.

The gap narrows but persists for Opus and Fable. Claude Opus 5 achieves 100% judgment accuracy but 80% task success, while Claude Fable 5.1 reaches 90% and 60%, respectively. Their easy short-horizon success also drops to 60% and 40%, and both score 0% on the hard subset. Only GPT-6 Astra achieves 100% on both perception-focused metrics and easy short-horizon manipulation. Overall, target identification is therefore already strong for most agents, while reliable physical execution remains much harder.

Table 1: Stable performance on the primary 30-task suite. Metrics follow the stable criterion in Section 4.1. Judg. Acc.: target-identification accuracy; SR: success rate; SC: stage completion; M: million tokens. Score denotes the 100-point Performance Score. Bold marks the best task-performance result in each column.

Agent	Perception		Short-Horizon		Long-Horizon		Overall	Resource Usage	
	Judg. Acc.	Task SR	Easy SR	Hard SR	Easy SC	Hard SC	Score	Avg. Time	Avg. Tokens
GPT-6 Astra	**100.0%**	**100.0%**	**100.0%**	**40.0%**	**50.0%**	**22.0%**	**45.0**	11.5 min	1.38M
GPT-5.6 Sol	90.0%	10.0%	40.0%	0.0%	16.7%	0.0%	10.8	22.0 min	3.44M
Claude Opus 5	**100.0%**	80.0%	60.0%	0.0%	16.7%	0.0%	19.3	22.0 min	5.49M
Claude Fable 5.1	90.0%	60.0%	40.0%	0.0%	16.7%	0.0%	15.8	22.4 min	3.01M
DeepSeek-V4.1-Flash	90.0%	0.0%	0.0%	0.0%	0.0%	0.0%	4.5	21.2 min	0.39M
Kimi K3	90.0%	10.0%	0.0%	0.0%	0.0%	0.0%	5.5	19.0 min	3.22M
Qwen3.8-Max	90.0%	30.0%	20.0%	0.0%	10.0%	0.0%	10.5	23.9 min	4.80M

Table 2: Observation ablation: stable short horizon success rates (%). Level 1: RGB + joint angles + end-effector pose; Level 2: Level 1 + dynamic proprioception; Level 3: Level 2 + calibrated depth. Bold marks the best nonzero result per agent and column.

Agent	Observation	Easy	Hard	Overall
Astra	Level 1	40.0	20.0	30.0
	Level 2	80.0	20.0	50.0
	Level 3	**100.0**	**40.0**	**70.0**
Opus	Level 1	20.0	0.0	10.0
	Level 2	20.0	0.0	10.0
	Level 3	**60.0**	0.0	**30.0**
Fable	Level 1	0.0	0.0	0.0
	Level 2	0.0	0.0	0.0
	Level 3	**40.0**	0.0	**20.0**

Table 3: Demonstration ablation: stable long horizon stage completion (%). None: no demo; Video: successful execution (MP4); Traj.: actions from another successful episode of the same task. Bold marks the best nonzero result per agent and column.

Agent	Context	Easy	Hard	Overall
Astra	None	50.0	22.0	36.0
	Video	**93.3**	**39.0**	**66.2**
	Video + Traj.	90.0	26.0	58.0
Opus	None	16.7	0.0	8.3
	Video	11.7	0.0	5.8
	Video + Traj.	**35.0**	0.0	**17.5**
Fable	None	16.7	0.0	8.3
	Video	30.0	**5.0**	17.5
	Video + Traj.	**30.8**	**5.0**	**17.9**

Performance degrades with interaction difficulty and task horizon. Reliable execution on simpler tasks does not necessarily extend to harder physical interactions. Astra falls from 100% success on easy short-horizon tasks to 40% on the hard subset, while every other agent drops to 0%. The drop on hard short-horizon tasks shows that tighter alignment, direction-sensitive motion, sustained contact, and mechanism interaction already pose major challenges before multi-stage composition is introduced. Long-horizon tasks further widen this gap. Astra retains 50.0% stable stage completion on the easy subset and 22.0% on the hard subset. Sol, Opus, and Fable each reach only 16.7% on easy long-horizon tasks, Qwen reaches 10.0%, and DeepSeek and Kimi make no stable progress. On the hard subset, every non-Astra agent scores 0%. These results suggest that longer horizons compound the difficulty of manipulation by requiring reliable execution across successive stages. Even Astra, which consistently solves the simpler tasks, struggles to sustain this reliability over longer sequences.

Overall performance varies widely and is not explained by greater resource use. Overall, Astra achieves the highest Performance Score at 45.0/100, followed by Opus at 19.3 and Fable at 15.8. Sol and Qwen obtain similar scores of 10.8 and 10.5, respectively, while Kimi and DeepSeek score 5.5 and 4.5. Astra's performance advantage is accompanied by shorter episodes and lower token consumption than most other agents. It averages 11.5 minutes per episode, compared with 19.0–23.9 minutes for the other agents. Its average token usage is only 1.38M, below most other agents. Taken together, Astra's higher success rates and lower resource use suggest that it completes tasks more efficiently, with less need for prolonged interaction or additional tokens.

4.3 Observation and Demonstration Context

Richer observations improve manipulation performance. To assess how different observation modalities affect manipulation success, we evaluate Astra, Opus, and Fable on the ten short-horizon tasks. We compare three cumulative observation profiles. Level 1 (L1) provides RGB observa-

tions, joint angles, and end-effector pose; Level 2 (L2) adds dynamic proprioception, including joint torques, end-effector velocity, and force/torque measurements; and Level 3 (L3) further adds metric depth and camera intrinsics and extrinsics. The evaluation follows the protocol in Section 4.1, with high reasoning effort and no demonstrations.

Table 2 shows that richer observations improve overall success, but the gains are mostly concentrated on easier tasks. Adding dynamic proprioception raises Astra's overall success from 30% to 50%, driven by an increase from 40% to 80% on easy tasks, while Opus and Fable remain at 10% and 0%, respectively. Adding calibrated depth then improves overall success by 20 percentage points for each agent, bringing Astra, Opus, and Fable to 70%, 30%, and 20%, respectively. For Opus and Fable, these gains occur entirely on easy tasks, where success rises from 20% to 60% and from 0% to 40%. Astra benefits on both subsets, improving from 80% to 100% on easy tasks and from 20% to 40% on hard tasks. We hypothesize that dynamic proprioception helps agents assess the robot's response to executed actions, while calibrated depth reduces uncertainty about the spatial relationship between the gripper and target objects. These signals may therefore support more accurate actions and corrections during manipulation.

Demonstration gains depend on the agent and format. To examine how demonstration availability and format affect long-horizon execution, we evaluate Astra, Opus, and Fable on the ten long-horizon tasks under three conditions. **None** uses the same no-demonstration setting as the primary evaluation. **Video** additionally provides an MP4 video of successful task execution. **Video + Traj.** further provides a successful action trajectory recorded in a different episode of the same task. Direct replay of this trajectory does not solve the evaluation episode, but the trajectory provides action-level cues about task dynamics. All conditions use full observations and high reasoning effort. We report stable stage completion for the easy and hard subsets and across all ten tasks.

Table 3 shows that video demonstrations improve long-horizon execution for Astra and Fable, whose overall stable stage completion rises from 36.0% to 66.2% and from 8.3% to 17.5%, respectively. Opus, however, shows no clear benefit from video alone. For Astra and Fable, video demonstrations yield larger gains on easy tasks than on hard tasks. We hypothesize that videos clarify subgoal ordering and intermediate states, helping agents plan and monitor execution. On harder tasks, however, difficult physical interaction may remain the dominant bottleneck even with this guidance, limiting further gains from video demonstrations.

Adding action trajectories does not consistently produce further gains over video alone. Opus's overall stable stage completion increases from 5.8% to 17.5%. Astra instead declines from 66.2% to 58.0%, mainly on hard tasks. Fable gains little beyond video alone. Since each reference trajectory comes from a different episode of the same task, we hypothesize that it provides useful low-level control cues but may also encourage actions that do not match the current episode. Overall, all three agents improve over the no-demonstration baseline under at least one demonstration condition, but additional action-level information does not reliably yield further improvements.

5 Diagnosing the Strongest Agent

5.1 Localizing Astra's Long-Horizon Advantage

To understand where Astra's advantage comes from, we decompose the five hard long-horizon tasks into 20 physical stages and manually annotate whether each stage is reached, whether it succeeds, and its failure mode. We compare Astra with the other three highest-scoring agents under the same high-reasoning, no-demonstration setting.

We further categorize stages by their physical interaction and failures by where they arise, as illustrated in Figure 2. *Object Manipulation* moves or repositions a movable object, whereas *Mechanism Interaction* changes the state of an articulated mechanism. For failed stages, *Local Execution* denotes failure of the current physical operation itself, while *Cross-stage Interference* occurs when actions in one stage disturb, occlude, or undo a physical state established by another stage. To isolate local physical competence, execution success rates are computed after excluding failures attributed to cross-stage interference.

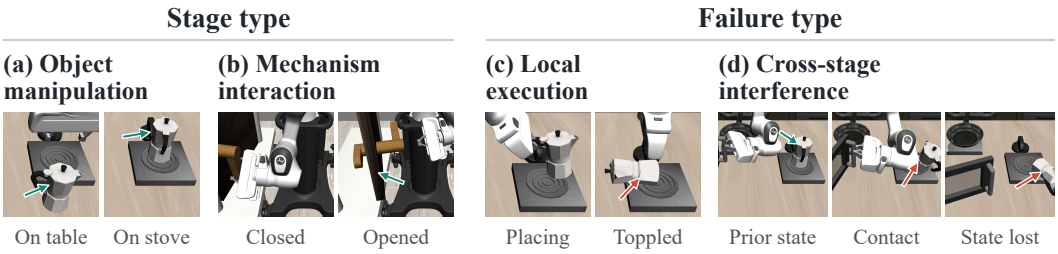

Figure 2: **Stage and failure taxonomy. (a–b)** Stage types: Object Manipulation and Mechanism Interaction; **(c–d)** failure types: Local Execution and Cross-stage Interference.

Table 4: Stage-level diagnosis on the hard long-horizon tasks. *Reached* and *Success* are computed over all stage opportunities. Object Manipulation and Mechanism Interaction report execution success rates after excluding failures attributed to cross-stage interference. Failure modes are percentages of reached-but-failed stages and may overlap.

	Stage Progress (%)		Execution SR (%)		Failure Mode (%)	
Agent	Reached	Success	Object Manipulation	Mechanism Interaction	Local Execution	Cross-stage Interference
GPT-6 Astra	**65.0**	**36.7**	**66.7**	**94.1**	**23.5**	52.9
GPT-5.6 Sol	43.3	13.3	**66.7**	22.2	61.1	27.8
Claude Opus 5	41.7	5.0	28.6	7.1	81.8	0.0
Claude Fable 5.1	26.7	6.7	20.0	37.5	83.3	0.0

Table 4 shows that Astra's advantage appears at both task progression and physical execution. It reaches 65.0% of all stages and successfully completes 36.7%, substantially exceeding the other agents on both measures. The largest execution gap appears in mechanism interaction: Astra reaches a 94.1% execution success rate, while the other agents range from 7.1% to 37.5%. By contrast, Astra and Sol attain the same 66.7% rate on object manipulation. The failure annotations reinforce this distinction. Local execution accounts for 61.1–83.3% of reached-stage failures for the weaker agents, but only 23.5% for Astra. These results therefore localize Astra's long-horizon advantage primarily to stronger mechanism-level physical execution.

5.2 Sustained Contact During Mechanism Interaction

We next examine what Astra does differently during mechanism interaction. We focus on opening operations in which the gripper must remain physically engaged with a handle while the articulated mechanism moves. We quantify this behavior using the *sustained-contact ratio*:

$$N_{\text{sustained}} = \sum_{t=1}^{N_{\text{exec}}} \mathbf{1}[\ell_t \geq 5], \qquad \text{SCR} = \frac{N_{\text{sustained}}}{N_{\text{exec}}} \times 100\%, \tag{3}$$

where ℓ_t is the length of the gripper–handle contact run containing control step t, with $\ell_t = 0$ when no contact occurs.

Figure 3 shows a substantial difference between Astra and the other agents. Astra reaches a mean SCR of 20.63%, whereas the other agents range from 4.23% to 7.17%. The same measure also tracks local opening success. Among operations with zero SCR, none of 19 succeed. The success rate rises to 7/19 for SCR between 0 and 10%, to 10/15 for SCR between 10 and 25%, and to 11/11 when SCR exceeds 25%. Using the original 64 opening operations rather than the visualization bins, SCR has a Pearson correlation of $r = 0.711$ with local opening success. Together, these results associate Astra's higher opening success with more consistent sustained physical engagement.

5.3 Remaining Failures: Cross-stage Interference and Geometry

The preceding analyses show that Astra has largely reduced the local execution failures that dominate weaker agents, particularly during mechanism interaction. Its remaining errors therefore show a different failure pattern: successfully completing one physical interaction does not guarantee that the resulting scene remains suitable for subsequent stages.

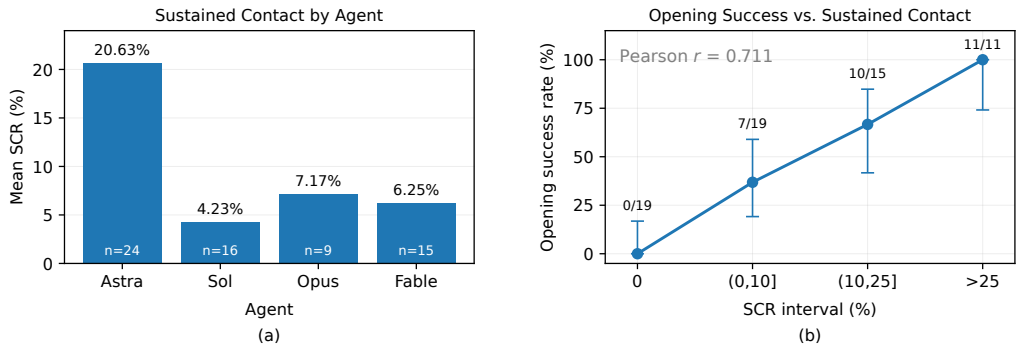

Figure 3: **Sustained contact during mechanism interaction.** **(a)** Mean sustained-contact ratio (SCR) across agents for entered opening operations. **(b)** Local opening success across SCR intervals. Error bars in (b) show 95% Wilson confidence intervals. SCR intervals are used only for visualization. Pearson correlation is computed over the original 64 opening operations.

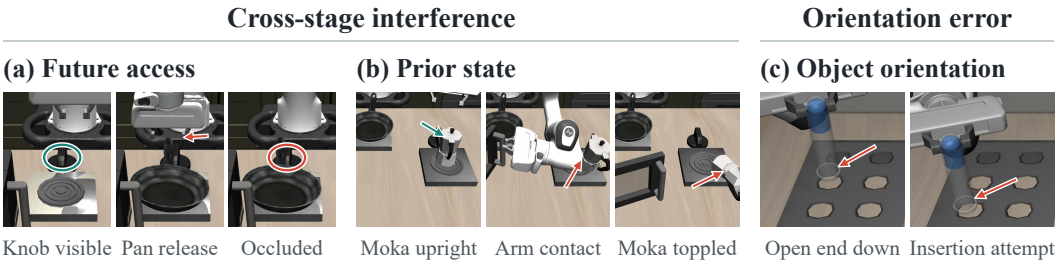

Figure 4: **Residual failure modes of Astra.** (a) A placement blocks access required by a later stage. (b) A subsequent action disturbs previously established state. (c) The intended operation is executed with an invalid object orientation.

Figure 4 illustrates two recurring forms of failure. First, actions interfere across stages. A reasonable placement can occlude or block a later interaction, while a subsequent motion can disturb an object or mechanism whose state was already correct. Second, Astra can select the correct object and operation while realizing it with invalid geometry. For example, it may attempt tube insertion with the tube in the wrong orientation.

These failures indicate that stronger local manipulation alone is not sufficient for reliable long-horizon control. Astra can execute individual physical interactions effectively, yet still struggles to preserve task-relevant state across stages and to realize actions with task-valid geometry.

6 Conclusion

We introduced LIBERO-Agent, a benchmark for evaluating general-purpose agents through autonomous observation selection and native robot control. Its perception-focused, short-horizon, and long-horizon tasks provide complementary tests of embodied competence. GPT-6 Astra achieves the highest Performance Score of 45.0/100 among seven evaluated agents. It excels at target identification and easy short-horizon manipulation, but remains less reliable on harder short-horizon and long-horizon tasks. More broadly, our benchmark reveals a common limitation across agents: correct target identification does not consistently translate into reliable physical execution. Our ablations show that richer observations improve manipulation, while demonstration benefits vary by agent and format. Further analysis shows that Astra's advantage is strongest in mechanism interaction and is associated with sustained physical contact. As local execution improves, its remaining failures shift toward cross-stage interference and geometric errors, highlighting anticipation of downstream consequences and preservation of previously achieved states as key remaining challenges.

AI USE STATEMENT

We used generative AI tools for writing refinement, language editing, and assistance in organizing and presenting experimental analyses. All reported results, quantitative analyses, and scientific claims were independently checked by the authors against the underlying experimental records. We did not use generative AI tools to generate experimental results.

ETHICS STATEMENT

This work evaluates general-purpose agents in simulated robot manipulation environments and does not involve human subjects, user data collection, or deployment of physical robots in real-world settings. The benchmark focuses on measuring agent capabilities under controlled simulation conditions. We do not identify any additional ethical concerns beyond responsible evaluation and reporting of model capabilities.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the benchmark protocol, simulation interface, task construction, evaluation metrics, and experimental settings in the main paper and appendix. The appendix includes task specifications, success semantics, episode conditions, demonstration formats, agent harness information, and additional per-task results. The supplementary material further provides the complete rollout-level records. These details are provided to facilitate reproduction of the reported evaluations and analysis.

REFERENCES

Alibaba Cloud. Alibaba unveils Qwen3.8-Max: Its largest and most capable flagship model to date. https://www.alibabacloud.com/en/press-room/alibaba-unveils-qwen3-8-max, 2026. Accessed September 24, 2026.

Anthropic. Introducing Claude Fable 5.1 and Claude Mythos 5.1. https://www.anthropic.com/claude-fable-and-mythos-5-1, 2026a. Accessed September 24, 2026.

Anthropic. Introducing Claude Opus 5. https://www.anthropic.com/news/claude-opus-5, 2026b. Accessed September 24, 2026.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, Weijie Wan, Baijun Chen, Songling Liu, Haowen Yan, et al. Robodojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, et al. Workarena: How capable are web agents at solving common knowledge work tasks? *arXiv preprint arXiv:2403.07718*, 2024.

Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Dantong Niu, Fei-Fei Li, et al. Cap-x: A framework for benchmarking and improving coding agents for robot manipulation. *arXiv preprint arXiv:2603.22435*, 2026.

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, et al. Maniskill2: A unified benchmark for generalizable manipulation skills. *arXiv preprint arXiv:2302.04659*, 2023.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

Wenqi Huang, Charley Lee, Leonard Tng, and Serena Ge. Deepswe: Measuring frontier coding agents on original, long-horizon engineering tasks. *arXiv preprint arXiv:2607.07946*, 2026.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, 2024.

Huashuo Lei, Wenxuan Song, Huarui Zhang, Jieyuan Pei, Jiayi Chen, Haodong Yan, Han Zhao, Pengxiang Ding, Zhipeng Zhang, Lida Huang, et al. Robomemarena: A comprehensive and challenging robotic memory benchmark. *arXiv preprint arXiv:2605.10921*, 2026.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International conference on robotics and automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn llm agents. *Advances in neural information processing systems*, 37:74325–74362, 2024.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *International Conference on Learning Representations*, volume 2024, pp. 9025–9049, 2024.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

OpenAI. GPT-6 Astra model. https://developers.openai.com/api/docs/models/gpt-6-astra, 2026a. Accessed September 24, 2026.

OpenAI. GPT-5.6 Sol model. https://developers.openai.com/api/docs/models/gpt-5.6-sol, 2026b. Accessed September 24, 2026.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. Androidworld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, volume 2025, pp. 406–441, 2025.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, 2024.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Anyi Xu, B Li, Bangcai Lin, Bing Xue, BingCheng Xian, Bingzheng Xu, Bochao Wu, Bowei Zhang, Boyi Deng, CC Yu, et al. Deepseek-v4. 1-flash: Pushing the limits of kv cache compression. *arXiv preprint arXiv:2609.19969*, 2026.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, et al. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. *arXiv preprint arXiv:2502.09560*, 2025.

Yue Yang, Shuo Cheng, Yu Fang, Homanga Bharadhwaj, Mingyu Ding, Gedas Bertasius, and Daniel Szafir. Lilo-vla: Compositional long-horizon manipulation via linked object-centric policies. *arXiv preprint arXiv:2602.21531*, 2026.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, et al. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11142–11152. IEEE, 2025.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Kevin Lin, Abhiram Maddukuri, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

A AGENT HARNESSES

Table 5 lists the agent harness version used for each evaluated model. Versions are taken from the archived run manifests and session metadata.

Table 5: Agent harnesses and versions used in our experiments.

Model	Agent harness	Version
GPT-6 Astra	Codex	0.154.0
GPT-5.6 Sol	Codex	0.154.0
Claude Opus 5	Claude Code	2.1.139
Claude Fable 5.1	Claude Code	2.1.139
DeepSeek V4.1 Flash	DeepSeek Harness	0.1.5-rc.2
Kimi K3	Kimi Code	0.42.0
Qwen3.8-Max	Qwen Code	0.24.0

B TASK SUITE AND EVALUATION DETAILS

B.1 TASK CONSTRUCTION AND PRIMARY EVALUATION SUITE

Our primary evaluation comprises 30 tasks: ten perception-focused tasks, ten short-horizon manipulation tasks, and ten long-horizon tasks. Tables 6–8 enumerate their instructions, physical requirements, and scoring units. The task collection combines LIBERO scenes and predicates (Liu et al., 2023), RoboMemArena task sequences (Lei et al., 2026), LiLo-VLA task scenes (Yang et al., 2026), VLABench assets and mechanisms (Zhang et al., 2025), and adapted manipulation mechanisms in the shared MuJoCo/robosuite runtime (Zhu et al., 2020). Task adaptation specifies the scene, initial state, public instruction, and evaluator together. The same observation and control interface is used across these sources.

The perception suite varies the information needed to select a target. Most tasks pair identification with pick-and-place or pressing a paired button. T08 instead requires physical probing: its three mugs have the same appearance, and the agent must lift them to distinguish their physical properties. Target identification and completion of the requested physical operation are scored separately.

The short-horizon split groups T11, T13–T15, and T20 as easy tasks, and T12 and T16–T19 as hard tasks. These groups distinguish elementary grasping and actuation from more constrained pushing, articulated motion, orientation control, insertion, and contact coverage. The long-horizon easy subset comprises T23, T24, T26, T27, and T29; the hard subset comprises T21, T22, T25, T28, and T30. Its harder tasks combine goal interpretation, temporary-buffer planning, persistent contact constraints, or coupled mechanism interactions. Each difficulty subset contains five tasks and receives equal task weighting internally.

B.2 TASK-SPECIFIC SUCCESS SEMANTICS

The evaluators combine terminal predicates with physical event histories according to the task instruction. Object placement and ordinary articulation tasks use the corresponding containment, support, or joint predicates. Tasks involving counting, support constraints, or causal dependencies additionally record intermediate events. Full success and partial stage completion are stored separately.

Paired textual and visual goals. T21 and T22 share a scene definition, an initial-state bank, and five terminal predicates. T21 communicates the desired arrangement in text, whereas T22 supplies a goal image. Both accept any execution order that satisfies the five goals at termination. The evaluator records changes in the set of satisfied goals, including regressions when later actions disturb an earlier placement or mechanism state. A goal that was satisfied earlier but is false at termination does not contribute to the automatic terminal-goal count.

Table 6: Perception-focused tasks. The target-selection requirement is distinct from the complete physical outcome. The identifiers T01–T30 are shared across the task catalogs and execution examples.

ID	Primary information	Required physical outcome
T01	Object pose	Place the only upright marker in the collection bin.
T02	Spatial relation	Place the object between the other two objects in the basket.
T03	Color	Place the red object in the collection bin.
T04	Texture	Place the striped object in the collection bin.
T05	Instance marking	Place the object with exactly one square mark in the collection bin.
T06	Shape	Press the button paired with the small ball, among ball, cylinder, and cube candidates.
T07	Object category	Place the cola in the collection bin.
T08	Physical property	Physically sample all three visually identical mugs and place the heaviest in the green bin.
T09	Reflectance	Press the button paired with the most reflective object.
T10	Size	Press the button paired with the largest cylinder.

Table 7: Short-horizon manipulation tasks. Easy and hard refer to the evaluation split. A short-horizon task can contain several physical events, but its reported task success requires the complete operation.

ID	Split	Task	Success requirement
T11	Easy	Pick and place	Alphabet soup is in the basket.
T12	Hard	Push plate	The plate reaches the specified region in front of the stove.
T13	Easy	Press button	The red button is physically actuated.
T14	Easy	Close drawer	The cabinet's top drawer reaches its closed state.
T15	Easy	Open drawer	The cabinet's top drawer reaches its open state.
T16	Hard	Open microwave	The microwave door reaches its open state.
T17	Hard	Turn on stove	The stove knob reaches the activation condition.
T18	Hard	Insert tube	The blue chemistry tube satisfies the target slot's position, vertical alignment, and motion tolerances.
T19	Hard	Wipe spill	Grasp the sponge, cover the spill through physical wiping contact, and release the sponge on the blue rest area.
T20	Easy	Pour into mug	Lift the wine bottle and complete a physical pour event over the mug.

Table 8: Long-horizon task requirements and scoring units. K denotes the number of scored stages or goals. Cooking goals are evaluated at the terminal state without an imposed order.

ID	Split	K	Goals or stages
T21	Hard	5	Prepare the cooking area from text: moka pot on its stove, that stove on, frying pan on its stove, that stove on, and microwave open.
T22	Hard	5	Match the cooking goal image, using the same five physical goals as T21.
T23	Easy	8	Open and close the top, middle, and bottom drawers in order; reopen the occupied drawer; place butter inside it. Final closure is optional.
T24	Easy	4	Lift the tomato-sauce bottle, complete two distinct pours over the frying pan, and place the bottle in the bowl drainer.
T25	Hard	4	Move the left bowl to the empty plate, the middle bowl to the left plate, the right bowl to the middle plate, and the buffered bowl to the right plate. No bowl may touch the tabletop.
T26	Easy	3	Transfer tomato sauce, milk, and orange juice from the first cabinet to the second cabinet.
T27	Easy	4	Open the middle drawer, place cookies inside, place chocolate inside, and close the drawer.
T28	Hard	2	Release the rotating latch and open the door.
T29	Easy	3	Lift the hammer, establish hammer–nail contact, and drive the nail into the block.
T30	Hard	4	Place the first and second counterweights, produce a physical lift through the seesaw, and place the exposed target in the basket.

Ordered search and placement. T23 checks the drawer inspection sequence before the final butter placement. The occupied target drawer is determined from the initial scene. Its eight required stages exclude an optional final closure. T27 instead explicitly requires closure after the two placements, so closure contributes to its four-stage score. The evaluator records ordered progress and checks the required final object arrangement.

Repeated events and persistent constraints. For T24, two pours must be distinct physical events; the pour counter tracks bottle elevation, tilt, and placement relative to the receptacle, and rejects a detected third pour. T25 uses the empty plate as a temporary buffer and records the four ordered transfers. Bowl–table contact is monitored throughout execution, and the final buffer must be empty. T30 requires released counterweights to contact the movable board before the mechanism lifts the target. At completion, the target must be in the basket while both counterweights remain supported at the far end.

Active sensing and contact-dependent operations. T08 uses three visually identical mugs with masses of 0.18, 0.36, and 0.72 kg. Its automatic evaluator requires grasp-and-lift evidence for each mug, followed by placement of the heaviest mug in the green bin, released and clear of the gripper. Mass labels remain evaluator-private. T19 requires the grasped sponge to cover seven spill markers during physical table contact before it is released on the rest area. T29 requires hammer–nail contact before accepting a nail displacement of at least 0.075 m.

Insertion geometry. The automatic tube checker in T18 tests lateral error of at most 0.018 m, a tube-root height relative to the rack root in $[0.032, 0.085]$ m, absolute tube-axis alignment with the vertical of at least 0.97, linear speed of at most 0.08 m/s, and angular speed of at most 0.60 rad/s. Absolute axis alignment measures verticality and is insensitive to end-for-end reversal.

B.3 Episode conditions and outcome adjudication

The primary comparison evaluates GPT-6 Astra, GPT-5.6 Sol, Claude Opus 5, Claude Fable 5.1, DeepSeek V4.1 Flash, Kimi K3, and Qwen3.8-Max. Each configuration contributes three executions of every primary task, for 90 episodes per agent. The simulator seed is fixed at 100, and the task-specific initial state is held fixed across the three executions. These repetitions measure variation in agent execution under the same scene conditions.

Primary runs use high reasoning effort where supported, full observations, and no demonstrations. Each episode has a 1,800-second evaluator wall-clock limit. The recorded prompts specify a ceiling of 1,000,000 accepted OSC sequence submissions, with 1–50 native actions per submission; the wall-clock limit is the operative execution budget. A native action advances one 20 Hz control interval. The agent can write local programs and inspect selected files in its episode workspace. Benchmark observations and evaluator-private records are kept separate in the data interface.

C Task Prompts and Demonstration Context

C.1 Shared episode prompt

Each episode provides the task instruction followed by a shared execution prompt. Figure 5 reproduces the no-demonstration prompt from the successful T23 execution used in Appendix D. Line breaks are added for layout. The reference to reset attempts is retained from the recorded template; the evaluated configuration exposes no reset operation. Task-specific strategy and control programs are supplied by the agent during execution.

Recorded task and shared execution prompt

```
Open and close all drawers in order to check. Put butter into the
drawer that already contains an object.

A LIBERO episode has been prepared for you.

This query has a 1800-second evaluator wall-clock budget shared by
all reset attempts. Exceeding it ends the query as an official
task failure.

1. Call the `start_episode` robot tool exactly once to begin and
   receive the initial observation and `max_agent_steps` budget.
2. Control the robot with the `osc_sequence` robot tool. Its
   `actions` argument is an array of 1 to 50 normalized 7D OSC_POSE
   micro actions in `[dx, dy, dz, rx, ry, rz, gripper]` order.
   Every component must be within [-1, 1]. Each vector executes
   one LIBERO policy interval; translation 1.0 corresponds to
   0.05 m, rotation 1.0 to a 0.5 rad rotation-vector component,
   gripper -1 opens, and +1 closes. One sequence call counts as
   one Agent action, with at most 1000000 accepted calls.
3. Wait for each action to complete, then inspect
   `benchmark_inputs/current_observation/observation.json` and any
   referenced files before issuing another command.
4. When you have completed the task, call the `finish_episode`
   robot tool exactly once. Only finish reports official task
   success.
```

Figure 5: Shared episode instructions, shown with the T23 task text. The prompt describes the interaction contract and points to the current observation. It supplies no drawer-opening or object-transfer program.

For demonstration-conditioned episodes, the following notice is inserted after the task instruction:

> A demonstration from a separate episode of the same task is available at `benchmark_inputs/expert_demo/`. The current scene configuration and object or goal poses may differ.

Both demonstration conditions use this notice. The available files determine whether the reference contains video alone or video together with an action trajectory.

C.2 Representative task instructions

Task instructions state the desired operation and its public constraints. The following examples preserve the recorded task wording:

> **T08, physical perception.** Determine which of the three visually identical mugs is heaviest by physically weighing them, then place the heaviest mug in the green bin.
>
> **T18, precision manipulation.** Pick up the blue chemistry tube and insert it vertically into the rear-center empty slot of the tube rack.
>
> **T25, constrained rearrangement.** Cyclically rearrange the three black bowls from left to right in the head camera. No bowl may touch the tabletop at any time.

T28, coupled articulation. Open the door. It is secured by a rotating latch.

T30, mechanism use. Lift the hidden object and place it in the basket.

The weighing prompt requires physical sampling but does not identify the heaviest mug. The rearrangement prompt communicates the support constraint without naming the empty plate as a buffer. The seesaw prompt leaves the mechanism-use strategy to the agent and, when available, its demonstration.

C.3 VIDEO AND ACTION-TRAJECTORY BUNDLES

Demonstrations come from separate successful executions of the same task. Their public representation is specified in Table 9. The video contains synchronized head and wrist RGB views displayed side by side at 20 frames per second. Each view also has a contact sheet with at most 12 uniformly sampled frames, retaining the sequence endpoints. These sheets provide a directly viewable overview; the agent can inspect the video further using its own tools.

Table 9: Contents of the demonstration bundle. These columns describe historical reference data; the live observation profile is held fixed across the context comparison.

Reference content	None	Video	Video + Traj.
Head/wrist RGB video	—	Yes	Yes
RGB contact sheets	—	Yes	Yes
Ordered native OSC action vectors	—	—	Yes
Reference depth, calibration, or robot-state stream	—	—	—
Private object states or checker progress	—	—	—

Video + Traj. adds `trajectory.jsonl`. Each row pairs a zero-based `action_index` with an `action` object. The object contains a schema identifier and a `normalized_vector_7d` field. Its seven components follow the online OSC convention: three translation components, three rotation-vector components, and one gripper command. All values lie in $[-1, 1]$, with one action vector per policy interval. The trajectory contains issued actions; it does not include a numerical stream of measured end-effector poses.

Bundle validation checks the task instruction, file inventory, action dimensions and ranges, contiguous action indices, and consistency of the declared frame and action counts. The agent receives a manifest, the RGB video, two contact sheets, and, for Video + Traj., the action file. Current scene observations remain distinct from historical demonstration content, allowing the agent to adapt its execution to the observed scene.

D PROGRAMS CONSTRUCTED DURING SUCCESSFUL EXECUTIONS

We examine two successful GPT-6 Astra episodes under high reasoning effort, full observations, and no demonstrations: drawer inspection and placement (T23), and physical weighing (T08). The examples connect visible program construction and tool use to recorded observations, executed actions, and final task outcomes. Code excerpts preserve the recorded computations, with line wrapping and omission of unrelated display operations. The programs were created during these episodes rather than supplied as task-level primitives.

D.1 COMPUTING SPATIAL COORDINATES FROM SELECTED IMAGE LOCATIONS

At the beginning of T23, Astra inspects the scene and samples selected pixels from the head-camera depth array. It loads the camera intrinsics and camera-to-robot-base transform, back-projects each selected pixel, and prints its three-dimensional coordinate. Figure 6 shows two returned surface points. The agent subsequently issues an approach command with an end-effector target at $(0.63, 0.34, 0.273)$ m before moving closer.

The pixel locations are selected by the agent. Their back-projected coordinates are measurements of visible surfaces, from which the agent chooses an end-effector target with an approach offset. This trace shows how publicly available depth and calibration enter local computation before control. Object poses and grasp targets are not supplied by the evaluator.

T23: agent-written depth back-projection

```
import json, numpy as np
p = 'benchmark_inputs/current_observation/'
o = json.load(open(p + 'observation.json'))
c = o['cameras']['head']
d = np.load(p + c['depth']['metric_file'])
K = np.array(c['intrinsic_matrix_3x3'])
T = np.array(c['matrix_T_robot_base_from_camera_opencv_4x4'])
for v in [124,128,132,140,144,148,152,156]:
    for u in [214,217,220]:
        a = T @ np.r_[
            np.linalg.inv(K) @ np.array([u,v,1]) * d[v,u], 1]
        print((u,v), a[:3].round(3))
```

Selected tool output

```
(217, 128) [0.642 0.428 0.272]
(220, 128) [0.643 0.442 0.273]
```

Figure 6: Numerical geometry constructed from public observations in a successful drawer-search episode. The excerpt and output come from the recorded tool call. Coordinates are expressed in the robot-base frame and measured in metres. The agent selects the pixels and subsequent approach targets.

D.2 CONSTRUCTING AND REUSING A POSE-FEEDBACK HELPER

In the same T23 episode, Astra creates a local Python helper, `robot_pose.py`. Each invocation reads the current end-effector pose, computes position and rotation errors relative to command-line targets, clips the resulting normalized control values, and prints a JSON action sequence. Figure 7 shows the helper's action-generation branch and its subsequent connection to the MCP tool. The rotation error uses the relative rotation $R_{\text{target}} R_{\text{current}}^{-1}$ in rotation-vector form.

The helper repeats one computed action within a submitted batch. Feedback is refreshed between batches, when the agent invokes the helper again on the newly published observation. In the initial orientation adjustment, six consecutive batches execute 60 control steps while retaining the same target orientation. The measured angular error falls from $121.88°$ to $0.76°$. Table 10 reports this and three subsequent same-orientation correction sequences from the episode. Positions can change while the orientation target is held fixed.

The evaluator records all eight required T23 stages as completed. The episode uses 57 accepted control submissions and 895 native control steps; the agent also performs the optional final drawer closure. The first drawer opening is recorded at control step 205, and the required butter placement at step 732. These events connect the programming example to a complete successful task execution.

T23: action-generation branch of `robot_pose.py`

```
import json, sys, numpy as np
from scipy.spatial.transform import Rotation as R
p = 'benchmark_inputs/current_observation/'
o = json.load(open(p + 'observation.json'))
s = o['state']
pose = s['eef_pose_robot_base_xyzw_7d']
if len(sys.argv) > 1:
    target = np.array([float(x) for x in sys.argv[1:4]])
    q = np.array([float(x) for x in sys.argv[4:8]])
    g = float(sys.argv[8])
    n = int(sys.argv[9])
    dp = target - pose[:3]
    dr = (R.from_quat(q) *
          R.from_quat(pose[3:]).inv()).as_rotvec()
    a = np.r_[np.clip(dp*5,-1,1), np.clip(dr*.8,-1,1), g]
    print(json.dumps({'actions': [a.tolist() for _ in range(n)]}))
```

Recorded invocation and action submission

```
const r = await tools.exec_command({
  cmd: "python /tmp/robot_pose.py .63 .34 .273 " +
       "0 .70710678 .70710678 0 -1 10",
  max_output_tokens: 3000
});
text(await tools.mcp__libero__osc_sequence(JSON.parse(r.output)));
```

Execution receipt, selected fields

```
"command_completed": true, "control_steps": 10,
"termination_reason": "sequence_executed",
"observation_id": "obs_000002"
```

Figure 7: Astra constructs a pose-feedback helper and passes its numerical output to native OSC control. The helper is generated in the session. Its action-generation branch is shown with whitespace expanded; the read-only diagnostic branch is omitted. The shell-command string is split for layout. A receipt establishes execution of the sequence; task completion is checked separately.

Table 10: Repeated orientation corrections in the successful T23 episode. Each sequence uses fresh state before successive control batches and retains a common target orientation. Angular error is measured from the recorded end-effector quaternion to the agent-selected target.

Sequence	Control batches	Control steps	Initial error	Final error
1	6	60	121.88°	0.76°
2	4	42	89.68°	2.26°
3	5	61	87.99°	0.14°
4	4	46	90.42°	0.61°

D.3 Acquiring physical evidence through active probing

T08 provides a complementary example in which the necessary evidence is obtained through interaction. Astra successively grasps and lifts the three visually identical mugs. For each mug it executes a batch of 20 upward commands followed by 20 commands with zero pose increments and a closing gripper command. It then reads the current state and proprioception and inspects an RGB image. Figure 8 shows this repeated action pattern and the corresponding force readings.

The sensor-frame z components after the three lift-and-hold batches are -8.64, -12.19, and -6.86 N. The middle mug produces the largest load magnitude, and the agent subsequently places it in the green bin. The private evaluator confirms that all three mugs were physically sampled and that the heaviest was placed, released, and left clear of the gripper. The episode completes all four required events in 28 accepted control submissions and 614 native control steps.

These are raw end-effector sensor readings, which include the gripper load and are expressed in the sensor frame. The trace demonstrates repeated physical probing followed by successful target selection; it does not contain an explicit calibrated mass-estimation program. Here, zero pose increments create a hold interval for sensing within the manipulation sequence.

T08: recorded lift-and-hold action pattern

```
text(await tools.mcp__libero__osc_sequence({actions: [
  ...Array.from({length:20}, () => [0,0,0.6,0,0,0,1]),
  ...Array.from({length:20}, () => [0,0,0,0,0,0,1])
]}));
```

Python executed immediately after the action

```
import json
x = json.load(open(
    'benchmark_inputs/current_observation/observation.json'))
print(x['state'])
print(x['proprioception'])
```

Observed values after each lift-and-hold batch

Probe	Sensor F_z (N)	EEF height (m)	Jaw width (m)
Left mug	-8.635	0.207	0.0613
Middle mug	-12.195	0.180	0.0622
Right mug	-6.861	0.204	0.0616

Final evaluation: all three mugs sampled; heaviest mug in the green bin, released and clear of the gripper; 4/4 events completed.

Figure 8: Active sensing in a successful weighing episode. The same lift-and-hold command pattern is issued for each mug, followed by a state and proprioception read. Values are rounded from the returned observations. Mug locations refer to the initial head-camera view; height is in the robot-base frame and force is in the sensor frame.

E PER-TASK RESULTS ACROSS THREE ROLLOUTS

In this section, we record the per task rollout results. Binary entries give full-task successes out of three rollouts. Long-horizon entries give the sum of three normalized stage-completion fractions divided by three; K denotes the number of stages. The supplementary material provides the complete rollout-level evaluation records.

Table 11: Perception: full-task successes out of three rollouts.

Agent	T01 Upright marker	T02 Between objects	T03 Red object	T04 Striped object	T05 Marked instance	T06 Ball shape	T07 Cola drink	T08 Heaviest mug	T09 Highest reflectance	T10 Largest cylinder
GPT-6 Astra	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3
GPT-5.6 Sol	1/3	0/3	2/3	1/3	2/3	3/3	1/3	1/3	1/3	2/3
Claude Opus 5	3/3	2/3	3/3	3/3	3/3	3/3	3/3	2/3	3/3	3/3
Claude Fable 5.1	2/3	2/3	1/3	3/3	3/3	3/3	3/3	2/3	3/3	3/3
DeepSeek V4.1 Flash	1/3	1/3	1/3	1/3	0/3	1/3	0/3	1/3	1/3	2/3
Kimi K3	1/3	0/3	0/3	2/3	0/3	3/3	1/3	1/3	1/3	2/3
Qwen3.8-Max	1/3	2/3	2/3	2/3	2/3	3/3	1/3	1/3	3/3	3/3

Table 12: Short-horizon manipulation: full-task successes out of three rollouts.

Agent	T11 Pick soup	T12 Push plate	T13 Press button	T14 Close drawer	T15 Open drawer	T16 Open microwave	T17 Turn on stove	T18 Insert tube	T19 Wipe spill	T20 Pour wine
GPT-6 Astra	3/3	3/3	3/3	3/3	3/3	3/3	2/3	0/3	2/3	3/3
GPT-5.6 Sol	3/3	0/3	2/3	3/3	0/3	0/3	2/3	0/3	0/3	2/3
Claude Opus 5	3/3	1/3	3/3	2/3	3/3	0/3	1/3	0/3	1/3	2/3
Claude Fable 5.1	3/3	0/3	3/3	2/3	1/3	0/3	1/3	0/3	1/3	2/3
DeepSeek V4.1 Flash	1/3	0/3	1/3	0/3	0/3	0/3	1/3	0/3	1/3	1/3
Kimi K3	2/3	0/3	1/3	2/3	1/3	0/3	1/3	0/3	1/3	1/3
Qwen3.8-Max	2/3	0/3	3/3	2/3	2/3	0/3	0/3	0/3	1/3	1/3

Table 13: Long-horizon manipulation: summed normalized stage-completion scores divided by three.

Agent	T21 Cooking (text)	T22 Cooking (image)	T23 Drawer search	T24 Pour twice	T25 Rotate bowls	T26 Cabinet transfer	T27 Drawer storage	T28 Unlatch door	T29 Hammer nail	T30 Use seesaw
K	5	5	8	4	4	3	4	2	3	4
GPT-6 Astra	1.80/3	0.80/3	3.00/3	2.00/3	0.00/3	3.00/3	1.00/3	2.50/3	2.00/3	1.00/3
GPT-5.6 Sol	0.80/3	0.20/3	0.25/3	2.00/3	0.75/3	1.67/3	1.00/3	0.00/3	0.67/3	0.00/3
Claude Opus 5	0.00/3	0.40/3	0.00/3	1.50/3	0.00/3	0.33/3	0.00/3	0.50/3	1.00/3	0.00/3
Claude Fable 5.1	0.20/3	0.20/3	0.00/3	1.50/3	0.00/3	0.00/3	0.00/3	1.00/3	2.33/3	0.00/3
DeepSeek V4.1 Flash	0.00/3	0.80/3	0.00/3	0.75/3	0.00/3	1.33/3	0.00/3	0.00/3	1.00/3	0.00/3
Kimi K3	0.00/3	0.80/3	1.00/3	0.75/3	0.00/3	0.00/3	0.50/3	0.00/3	0.33/3	0.00/3
Qwen3.8-Max	0.80/3	0.40/3	0.62/3	2.00/3	0.00/3	0.67/3	0.00/3	0.00/3	0.00/3	0.00/3

Table 14: Observation ablation: task successes out of three rollouts (L1–L3).

Agent	Obs.	T11 Pick soup	T12 Push plate	T13 Press button	T14 Close drawer	T15 Open drawer	T16 Open microwave	T17 Turn on stove	T18 Insert tube	T19 Wipe spill	T20 Pour wine
Astra	L1	2/3	2/3	1/3	2/3	3/3	3/3	2/3	0/3	1/3	3/3
	L2	3/3	3/3	3/3	2/3	3/3	2/3	1/3	0/3	2/3	3/3
	L3	3/3	3/3	3/3	3/3	3/3	3/3	2/3	0/3	2/3	3/3
Opus	L1	1/3	0/3	3/3	2/3	0/3	0/3	0/3	0/3	0/3	0/3
	L2	0/3	1/3	3/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3
	L3	3/3	1/3	3/3	2/3	3/3	0/3	1/3	0/3	1/3	2/3
Fable	L1	1/3	0/3	1/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3
	L2	1/3	0/3	2/3	1/3	0/3	0/3	1/3	0/3	0/3	0/3
	L3	3/3	0/3	3/3	2/3	1/3	0/3	1/3	0/3	1/3	2/3

Table 15: Context ablation: summed normalized stage-completion scores divided by three.

Agent	Context	T21 Cooking (text)	T22 Cooking (image)	T23 Drawer search	T24 Pour twice	T25 Rotate bowls	T26 Cabinet transfer	T27 Drawer storage	T28 Unlatch door	T29 Hammer nail	T30 Use seesaw
Astra	None	1.80/3	0.80/3	3.00/3	2.00/3	0.00/3	3.00/3	1.00/3	2.50/3	2.00/3	1.00/3
	Video	1.60/3	1.60/3	3.00/3	3.00/3	3.00/3	2.67/3	3.00/3	2.50/3	3.00/3	2.25/3
	Video + Traj.	2.40/3	1.40/3	3.00/3	2.50/3	2.00/3	3.00/3	3.00/3	1.50/3	3.00/3	2.00/3
Opus	None	0.00/3	0.40/3	0.00/3	1.50/3	0.00/3	0.33/3	0.00/3	0.50/3	1.00/3	0.00/3
	Video	0.40/3	0.00/3	0.12/3	1.50/3	0.75/3	1.33/3	0.50/3	0.00/3	2.00/3	0.00/3
	Video + Traj.	0.20/3	0.20/3	1.00/3	1.50/3	0.00/3	1.00/3	0.75/3	0.00/3	2.67/3	0.00/3
Fable	None	0.20/3	0.20/3	0.00/3	1.50/3	0.00/3	0.00/3	0.00/3	1.00/3	2.33/3	0.00/3
	Video	0.20/3	0.00/3	1.38/3	1.50/3	1.00/3	0.33/3	0.50/3	0.00/3	3.00/3	0.25/3
	Video + Traj.	0.00/3	0.00/3	1.25/3	2.00/3	1.25/3	0.67/3	0.75/3	0.00/3	2.33/3	0.75/3

F Qualitative Rollout Visualizations

We visualize successful and unsuccessful executions from the perception, short-horizon manipulation, and long-horizon suites. Each example shows synchronized head-camera and wrist-camera views at selected control steps. The perception examples share the same task, while the manipulation and long-horizon examples illustrate different tasks within their respective suites.

F.1 Perception

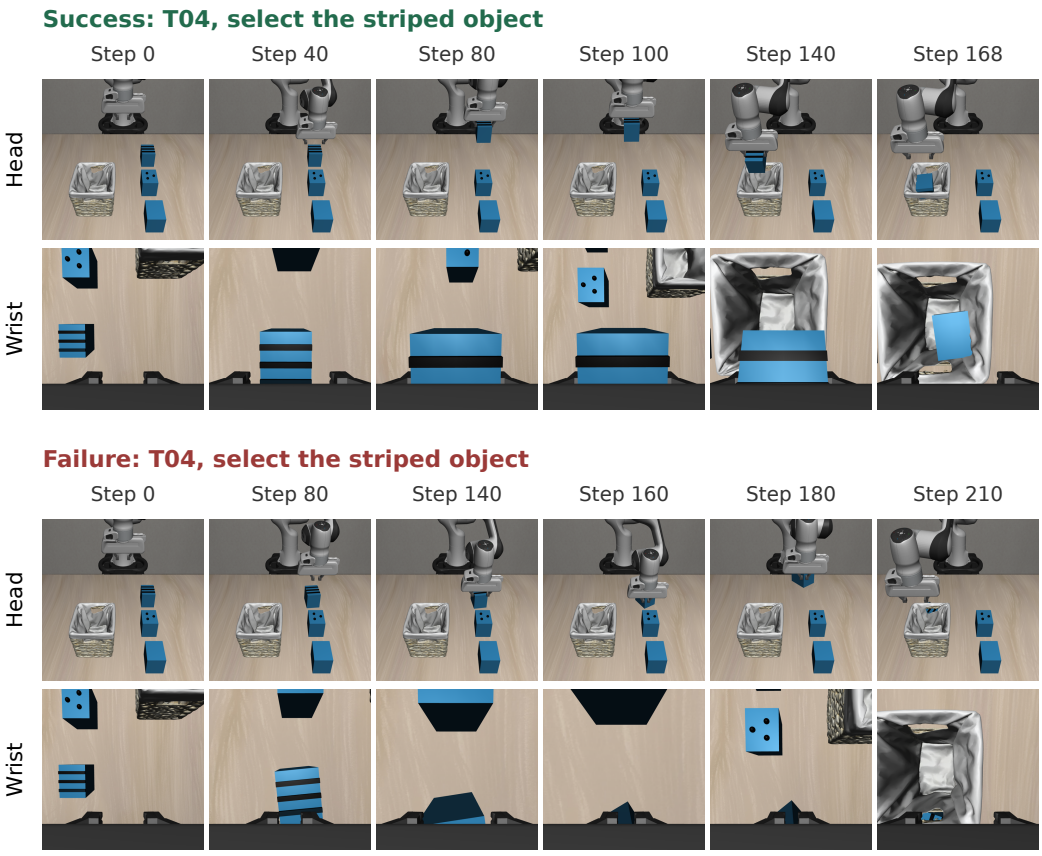

Figure 9: **Perception rollouts on T04.** The instruction is to place the striped object in the collection bin. The successful execution transfers the striped object to the bin. In the unsuccessful execution, the robot also transports the striped object toward the bin, but its final placement does not satisfy the task's success check. The wrist view exposes the texture cues and the objects near the gripper, while the head view shows their positions relative to the bin.

F.2 SHORT-HORIZON MANIPULATION

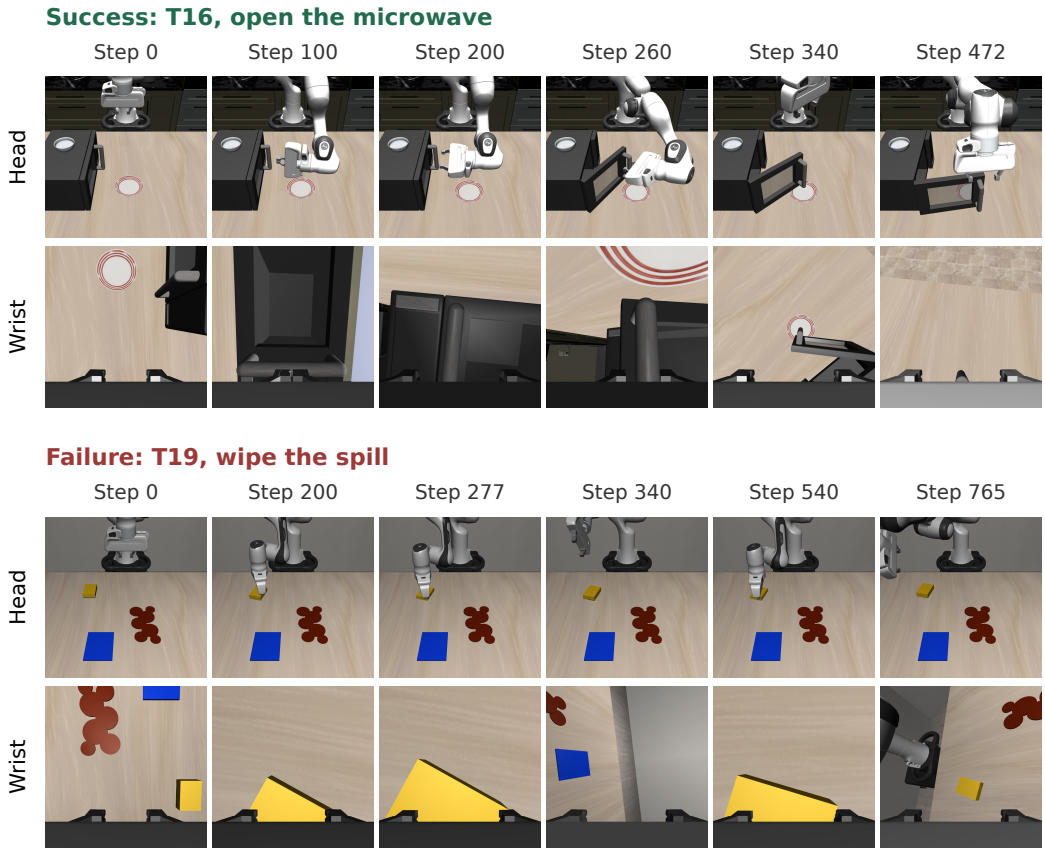

Figure 10: **Short-horizon manipulation rollouts.** *Top:* a successful T16 execution opens the microwave door. The selected frames show approach, gripper reorientation, and door opening. *Bottom:* an unsuccessful T19 execution attempts to grasp the sponge but leaves the spill unwiped when the time budget expires. The brown spill remains visible throughout the selected frames. The paired views show the gripper configuration and its relation to the handle or sponge.

F.3 LONG-HORIZON EXECUTION

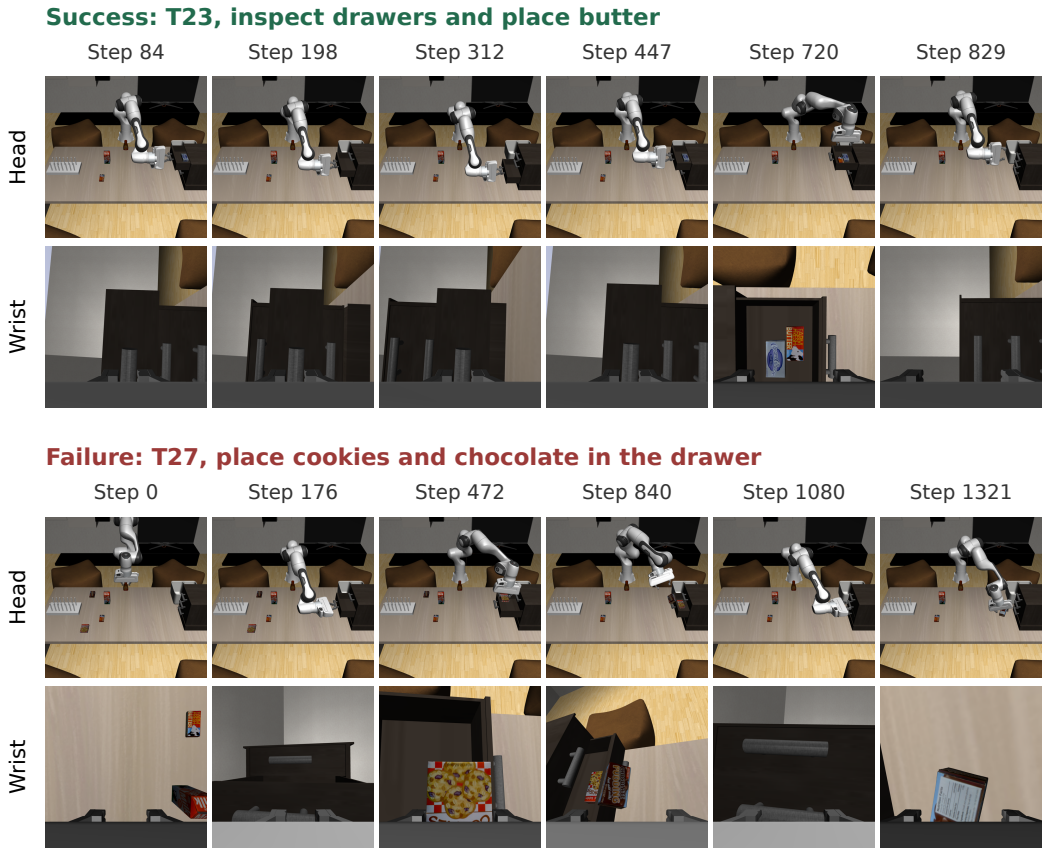

Figure 11: **Long-horizon rollouts.** *Top:* a successful T23 execution opens and closes the drawers to inspect their contents, reopens the occupied drawer, and places butter inside. The selected frames show the top, middle, and bottom drawer openings, the occupied drawer reopened, butter placement, and the final configuration. All eight required stages are completed. *Bottom:* an unsuccessful T27 execution opens the middle drawer and places cookies inside, but does not complete the subsequent chocolate placement before the time budget expires. It completes two of four required stages; the final frame shows the chocolate outside the drawer.