EMBODIED SKILL BENCH: BENCHMARKING HOW WELL AGENTS WITH EMBODIED SKILLS PERFORM ON PHYSICAL TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents are increasingly used as high-level controllers for robots, drawing on a growing body of VLA policies, perception models, and procedural know-how from earlier robot runs. These modules are hard for agents to reuse because they differ in interfaces, embodiment assumptions, and conditions of use, and how much they actually help on physical tasks has not been measured. In this paper, we first package them as embodied skills, which extend agent skills by making embodiment assumptions and conditions of use explicit, and build a repository of 250 action, perception, and experience skills from 1,052 existing resources. On this basis we introduce EMBODIEDSKILLBENCH, which runs an agent in a closed observation-action loop and records success, actions, tokens, and wall-clock time. It supports three modes: native control, skill-only execution, and Agent-Directed Reuse (ADR), a mode we propose in which the agent keeps its native controls and decides for each skill whether to invoke, modify, or ignore it. Across four agents on 100 perturbed LIBERO-Pro tasks, we find that skills help an agent only as far as they exceed what it can already do. Skill-only execution lifts GPT-5.6-Sol from 11% to 63% and Kimi-K3 and Claude-Opus-5 by about 20 points, while cutting their token usage by 58% to 85%. Yet all four agents then end up between 61% and 64%, and GPT-6-Astra falls from 92% under native control to 61%. This suggests that as agents grow stronger, the role of embodied skills shifts from supplying missing capability to reducing cost: GPT-6-Astra already solves most tasks through native control, so restricting it to skills removes more than it adds. Under ADR, which lets it keep native control and call skills selectively, GPT-6-Astra recovers to 85%, within 7 points of native control, while using 14.1% fewer tokens and 11.9% less time. With Jev as a skill choosing motion directions and gripper actions, it reaches 81% with 50% fewer tokens and 57% less time.

1 INTRODUCTION

Large language model agents are becoming a high-level control layer for robots. Rather than relying on a single policy to map an instruction directly to robot movements, an agent can break a task into steps, call specialized modules, and revise its plan after observing the result. This lets the language model handle task-level reasoning while robot-specific modules handle perception and movement. SayCan selects pretrained skills according to what is feasible in the scene (Brohan et al., 2023b), while Code as Policies generates programs that combine perception and control (Liang et al., 2023). Recent agent harnesses connects an agent to its tools and environment, and they add planning, memory, or recovery around frozen vision-language-action (VLA) policies (Zhang et al., 2026b; Ding et al., 2026). These policies translate images and language instructions into robot actions without being retrained by the agent. Collectively, these developments have produced a growing ecosystem of reusable embodied resources, including native controllers, VLA models, visual perception tools, and practical guidance distilled from earlier robot runs. These resources provide knowledge that a general-purpose agent need when working on physical tasks

Recent work has explored reusable agent skills, embodied agent frameworks, and benchmarks for physical task execution. SkillsBench evaluates the utility of reusable agent skills, but covers only digital tasks, leaving the physical constraints and execution feedback of embodied skill use unex-

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.

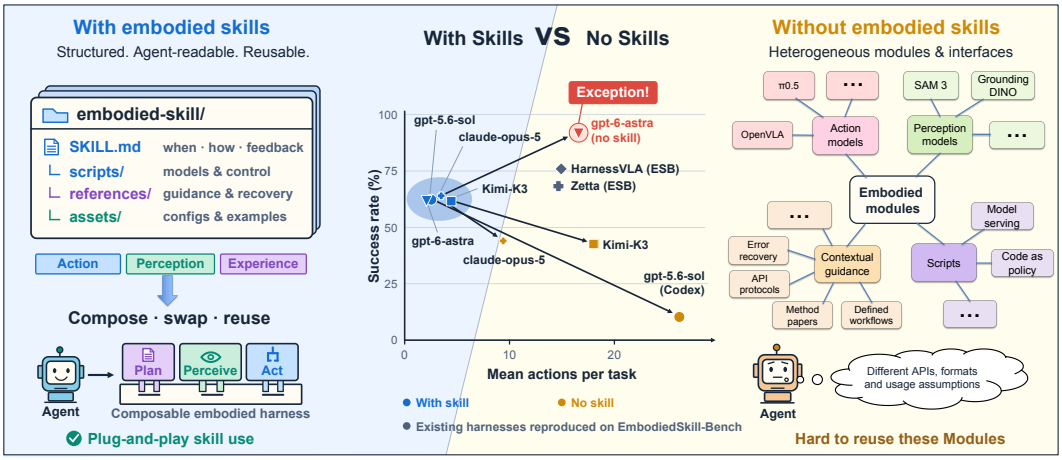


Figure 1: **EmbodiedSkillBench: Standardizing embodied modules and evaluating skill use. Left:** Embodied skills package action, perception, and experience resources in a common, agent-readable format for composition and reuse. **Right:** Without this structure, agents must assemble workflows from heterogeneous modules and interfaces. **Center:** Arrows connect each agent's with-skill and no-skill results. Skills reduce mean action counts for all four agents and improve success for gpt-5.6-sol, Kimi-K3, and claude-opus-5; gpt-6-astra achieves higher success without skills.

amined. (Li et al., 2026) HarnessVLA and Zetta manually integrate a fixed set of modules into a method-specific harness, rather than making these modules independently reusable across agents and workflows. (Zhang et al., 2026b; Ding et al., 2026) CaP-X and LIBERO-Pro evaluate agents or policies under specified control configurations, but configuration-level scores alone do not isolate how much these various and different modules contribute to an agent's performance. (Fu et al., 2026; Zhou et al., 2025)

As Figure 1 illustrates, the central challenge is these heterogeneous embodied modules are hard to reused by agents. Differences in APIs, data formats, embodiment assumptions, and execution conditions require an agent to determine how to invoke each module, when it applies, and how to interpret its feedback. Without a common way to expose these requirements, modules remain tied to individual integrations. Consequently, how effectively agents can reuse the available resources remains unclear. Answering this question requires a common representation that preserves module-specific physical constraints while supporting composition and reuse.

To address these gaps, we propose EMBODIEDSKILLBENCH, a benchmark for evaluating how effectively and efficiently agents use reusable embodied modules to solve physical tasks. EMBODIEDSKILLBENCH supports controlled comparisons across agents and skill configurations through a shared multi-turn interaction protocol, allowing agents to invoke modules and revise their plans based on physical feedback. To support this evaluation, we formulate embodied skills as a common representation that extends agent skills (Zhang et al., 2025a; Xu & Yan, 2026) with explicit embodiment assumptions, interface requirements, applicability conditions, and guidance for interpreting execution feedback. Using an autonomous agent workflow, we construct an Embodied Skill Repository of 250 skills from resources spanning four VLA model families with five model profiles, three perception models, eight programmed control interfaces, and procedural knowledge. We check the generated skills through scripts against the common protocol for skill component completeness and the availability of their declared model and control interfaces. The repository covers three complementary roles: action skills execute robot behaviors, perception skills interpret observations, and experience skills guide planning and recovery. By comparing agents with and without skills on the same tasks and linking skill-use traces to task success and resource consumption, EMBODIEDSKILLBENCH measures the benefits and costs of embodied skill use.

We evaluate four agents on 100 LIBERO-Pro task cases spanning language, object, task, and swap perturbations, comparing performance with and without skills and examining different skill compo-

sitions. Embodied skills improve success by 19-52 points for GPT-5.6-Sol, Kimi-K3, and Claude-Opus-5 while reducing token usage by up to 85%. Skill composition also matters: five configurations evaluated with GPT-5.6-Sol yield success rates ranging from 36% to 77%. In particular, GPT-6-Astra achieves 92% with native controls but only 61% with skills, suggesting that many capabilities supplied by existing embodied modules have been trained and internalized in it. Motivated by this finding, we propose Agent-Directed Reuse, which allows agents to inspect, adapt, combine, or bypass skills while retaining native controls. This raises GPT-6-Astra's success to 85%, recovering 24 percentage points over the with-skill setting while reducing tokens by 14.1% and runtime by 11.9% relative to native control setting, with a seven-point success tradeoff. We further integrate Jev as a waypoint-execution skill, allowing GPT-6-Astra to retain planning and recovery while delegating local motion decisions. This configuration achieves 81% success and reduces mean runtime from 20.52 to 8.74 minutes relative to native control.

Our contributions are:

- We formulate embodied skills as a common abstraction for heterogeneous embodied modules and establish an Embodied Skill Repository of 250 skills covering action, perception, and procedural guidance for physical tasks.
- We introduce EMBODIEDSKILLBENCH, a framework for evaluating how well agents with embodied skills perform on physical tasks. It supports multiple agents and flexible skill composition within a multi-turn action-observation loop.
- Our experiments show that embodied skills improve task success and reduce the execution costs across different agents. A further study of GPT-6-astra motivates us to propose a novel interaction paradigm with skills: Agent-Directed Reuse, enhancing efficiency of agent on physical, with a measurable tradeoff in task success.

2 RELATED WORK

Agents for physical control. Language-model agents connect instructions to robot actions through skill selection, code generation, and feedback-driven planning (Brohan et al., 2023b; Liang et al., 2023; Huang et al., 2023b). These approaches differ in the level of control delegated to the agent: some expose complete skills, while others require composing lower-level operations. Multimodal grounding further connects reasoning to physical observations and motion objectives (Driess et al., 2023; Huang et al., 2023a), while CaP-X examines how control abstractions affect coding agents (Fu et al., 2026). EMBODIEDSKILLBENCH focuses on evaluating how well agents perform with these embodied modules on physical tasks.

Embodied skills and harnesses. Generalist robot policies provide reusable language-conditioned control capabilities (Brohan et al., 2023a; 2022; Team et al., 2024; Kim et al., 2025b), but remain vulnerable to perturbations (Zhou et al., 2025). Agent harnesses and skill libraries add memory, recovery, and reusable procedures around robot controllers (Zhang et al., 2026b; Ding et al., 2026; Lu et al., 2026; Zhang et al., 2026a). However, these resources differ in their interfaces and usage assumptions, making their combination a separate challenge. EMBODIEDSKILLBENCH standardizes heterogeneous capabilities as skill packages and compares how different agents use them, with skill provision and the surrounding harness specified explicitly.

Benchmarks for physical tasks. RLBench, CALVIN, and LIBERO evaluate manipulation, long-horizon control, and knowledge transfer (James et al., 2020; Mees et al., 2022; Liu et al., 2023). LIBERO-Pro, VLABench, and EmbodiedBench further probe robustness, language understanding, and embodied reasoning (Zhou et al., 2025; Zhang et al., 2025b; Yang et al., 2025). These suites offer complementary tests of what a policy or agent can accomplish across tasks and changing conditions. SkillsBench and SkillRet evaluate skill utility and retrieval in digital tasks (Li et al., 2026; Kang et al., 2026). Physical execution additionally requires compatible module interfaces and feedback that supports verification or recovery. Retrieval quality alone therefore does not establish that the selected capabilities can be used effectively. EMBODIEDSKILLBENCH makes skill provision explicit in physical tasks, linking module use to task success and execution effort. This enables comparison of the benefits and costs of supplied capabilities under matched task conditions.

3 FORMULATION

We define an embodied skill as a discoverable, progressively loadable, and reusable agent skill supporting interaction with a physical environment.

We represent a package as:

$$s = \langle m, I, S, R, A \rangle \tag{1}$$

where m is metadata specifying the skill name and capability description; I contains instructions for skill use, feedback interpretation, and recovery; S contains scripts for control function and invocation of action and perception models; R contains method descriptions, model-use documentation, and procedural experience; and A contains assets such as model descriptors, configurations, interface schemas, and calibration data. Metadata and instructions are recorded in SKILL.md; the optional S, R, and A components are organized in scripts, references, and assets, respectively, following the open Agent Skills format (Xu & Yan, 2026).

Embodied module types. An embodied module is a source resource that contributes executable capability, perceptual evidence, or procedural knowledge for interaction with a physical environment. We distinguish three types by their resource form. **Model modules** include action and perception models. **Code modules** implement scripts for control function, observation processing, model invocation, and runtime adaptation. **Text modules** comprise context recording methods, interface documentation, procedural experience, and guidance for planning, verification, and recovery. This classification describes the source resources, while the operational role of an assembled skill depends on how those resources are combined and used.

Mapping modules to components. S contains Control and perception wrappers or interface declarations; R method references, repository documentation, and experience material; A model descriptors and supporting configurations. Model weights and serving runtimes remain external dependencies rather than being copied into each package. From these resources, the authoring agent derives I: applicability conditions, invocation steps, feedback interpretation, and bounded recovery or stopping rules. It then writes m to summarize the skill's purpose and trigger conditions. For example, a grasping skill combines a control interface in S, the corresponding model profile and input schema in A, and model-use or recovery guidance in R; I explains how to attempt and verify the grasp, while m makes this capability discoverable. The mapping is many-to-many: one module can support multiple skills, and a skill can combine resources from several sources.

4 EMBODIEDSKILLBENCH

EMBODIEDSKILLBENCH evaluates how effectively agents use embodied skills on physical tasks. It brings together the Embodied Skill Repository, a configurable agent-environment interface, and a trace-based evaluator. Figure 2 summarizes how embodied modules are assembled into skills, work in the embodied environment, and evaluated through multi-turn physical interaction.

The repository contains 250 embodied skill packages: 133 action skills (53.2%) for VLA-based and programmed control, 20 perception skills (8.0%) for localization, segmentation, pointing, and grasp proposals, and 97 experience skills (38.8%) providing guidance for planning, verification, and recovery.

Benchmark runs combine an agent and a selected skill set with a physical task under a shared observation-action protocol. The current evaluation uses 100 LIBERO-PRO task cases spanning multiple manipulation domains and perturbations. Task success, accepted actions, recorded tokens, and elapsed time jointly characterize the effectiveness and execution effort of each configuration.

4.1 SKILL CONSTRUCTION

The construction workflow in Figure 2 comprises module collection, component structuring, and agent-assisted skill assembly. We describe the source inventory and the concrete package scopes in the current Embodied Skill Repository.

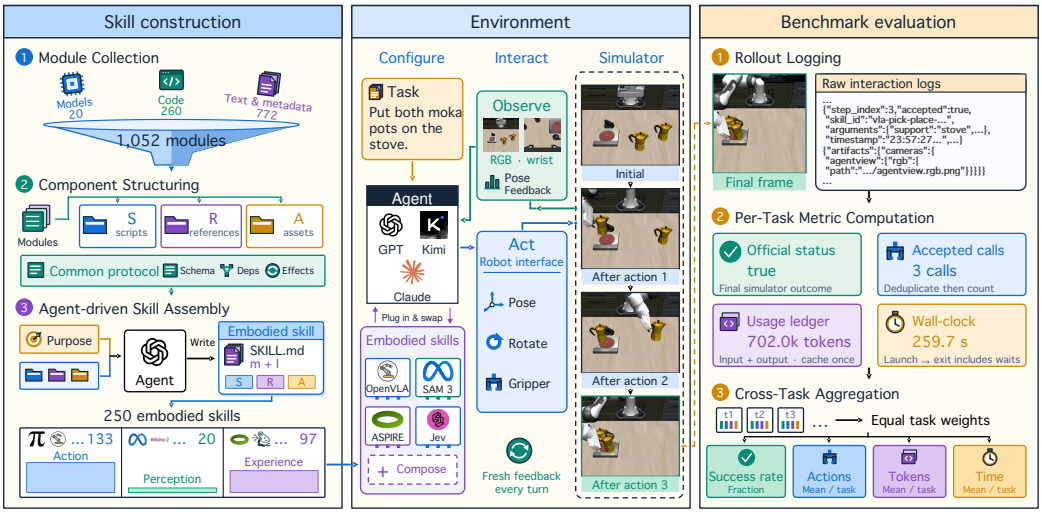


Figure 2: **EMBODIEDSKILLBENCH overview:** Skill construction collects heterogeneous modules, structures them under a common protocol, and uses an agent to assemble reusable embodied skills. The environment lets agents compose selected skills into task-specific harnesses and interact with a simulator through a multi-turn observation-action loop. Benchmark evaluation logs the resulting interactions, computes per-task success, actions, tokens, and time, and aggregates them across tasks.

Module collection. The corpus contains 1,052 deduplicated resource records: 227 control resources, 33 perception-interface records, 20 model descriptors, 417 experience materials, 23 publication references, 13 repository references, and 319 supporting assets. We create a card for each record, containing its type, functional description, and source. The main repository's VLA interfaces cover four model families through five profiles: two $\pi_{0.5}$ checkpoints (RLinf and Madokalif), OpenVLA-OFT, SmolVLA, and X-VLA (Black et al., 2025; Kim et al., 2025a; Shukor et al., 2025; Zheng et al., 2026). Perception resources support SAM3, Molmo2, and Contact-GraspNet (Carion et al., 2026; Clark et al., 2026; Sundermeyer et al., 2021). The 20 descriptor records include configuration and source variants. Experience sources include model-use instructions, manipulation and recovery procedures, and task-state verification guidance.

Component structuring. Applying the mapping in Section 3 groups the inventory into 260 script resources (S), 453 reference resources (R), and 339 assets (A). A common access protocol exposes resource reading and native-call preparation while retaining input schemas, dependencies, coordinate conventions, model profiles, and execution limits. For action-skill assembly, we retain eight task scopes—grasp, place-in, place-on, place-next-to, open, close, toggle, and push—across five VLA profiles and three budgets (base, brief, extended). Two concrete $\pi_{0.5}$ grasping interfaces and three general-subgoal interfaces also contribute to the VLA package. Six programmed interfaces for end-effector and gripper control, along with two mobile-base interfaces, provide the remaining action scopes. For perception, two general SAM3 interfaces and 16 queries specialized by target, view, resolution, and confidence are kept distinct, alongside one Molmo2 pointing interface and one Contact-GraspNet grasp-proposal interface. These perception skills use public observations without changing simulator state.

Agent-assisted skill assembly. Given an operational purpose, an authoring agent selects compatible resources from the structured inventory and reads their interface contracts and supporting material. It writes instructions I specifying applicability, invocation steps, feedback interpretation, verification, and bounded recovery or stopping rules, and metadata m describing the capability and its trigger conditions. The assembly plan links these instructions to the selected S, R, and A components. Materialization writes SKILL.md and the associated resource directories, records dependencies, and preserves source links. For skills providing procedural guidance, the instructions define read-only decision procedures and identify the action and perception capabilities required to carry them out; these dependencies must be selected separately.

4.2 Environment

As shown in the middle panel of Figure 2, a run is configured with a task description, an agent, and a composable set of embodied skills. Through the shared observation interface (Observe), the agent reads scene and wrist RGB images, robot state, and execution feedback, then uses available skills or native robot controls to act through the robot interface (Act). Actions may specify end-effector motion, rotation, or gripper control, or invoke an available VLA skill. After each accepted action, the simulator returns fresh observations to support verification and replanning, forming a multi-turn loop until task termination or budget exhaustion.

4.3 Evaluator

Evaluation has three stages. Rollout Logging records multi-turn interactions, models' input and output histories and usage records, timestamps, and official simulator feedback. Per-Task Metric Computation derives task success, action counts, language-model tokens, and elapsed time from these records. Success follows the official final task status; actions count accepted state-changing invocations after deduplication; tokens sum recorded input and output usage, counting cached input once; elapsed time spans agent launch to exit, including model calls, tools, and waiting. Cross-Task Aggregation reports official task success rate and mean actions, tokens, and elapsed time over 100Clean on LIBERO-PROand each metric uses its corresponding task-level records. Together, these measures capture task effectiveness and execution effort across configurations.

5 Experiments and Analysis

Our experiments focus on three research questions:

RQ1 (Effectiveness) Do embodied skills improve agents' performance on physical tasks?

RQ2 (Composition) How do different embodied skill sets selected from skill repository affect agents' performance on physical tasks?

RQ3 (Complementarity) Can state-of-the-art models still benefit from embodied skills?

5.1 Experimental setup

Agents and harnesses. We evaluate gpt-5.6-sol and gpt-6-astra using Codex CLI v0.148.0. We also evaluate Kimi-K3 through Kimi Code CLI v0.40.0 and claude-opus-5 through Claude Code v2.1.186. RQ1 compares gpt-5.6-sol, Kimi-K3, and claude-opus-5; RQ2 compares five skill configurations with gpt-5.6-sol; RQ3 separately studies gpt-6-astra.

Skill configurations. In the **Selected** with-skill setting, gpt-5.6-sol have access to all 250 embodied skills and it is required to review the repository and select skills for each task individually. The resulting task-conditioned skill sets comprise action, perception, and experience skills. The gpt-5.6-sol, Kimi-K3, and claude-opus-5 settings receive the same selected skill set for each task, enabling comparison under common skill provision. For gpt-5.6-sol, we also evaluate **BM25** lexical retrieval (Robertson & Zaragoza, 2009) and **Embedding** retrieval using BGE-small-en-v1.5. Both use the same 250 candidates and 481 public task-and-capability queries, taking the top-ranked skill per query and deduplicating within each task. They provide 4.23 and 3.82 skills per task on average, respectively. We additionally evaluate **HarnessVLA (ESB)** and **Zetta (ESB)**, our skill-based reconstructions of the original methods (Zhang et al., 2026b; Ding et al., 2026). These configurations compose their action modules and procedural guidance into fixed skill sets for gpt-5.6-sol. In the **no-skill** setting, agents have no embodied skills available and interact through EmbodiedSkillBench observation and robot-control interfaces.

Tasks and Metrics. We evaluate on Clean100, comprising 25 base tasks from LIBERO-Pro crossed with language, object, task, and swap perturbations (Zhou et al., 2025). The 100 task cases span LIBERO-10, Goal, Object, and Spatial (24, 28, 24, and 24 cases), covering multi-stage manipulation, varying goals, object identities, and spatial relations (Liu et al., 2023). Success rate is the primary metric; mean actions, tokens, and elapsed time measure execution effort(Section 4.3).

Table 1: Skill effects on Clean100 (seed 26). L10, Goal, Object, and Spatial contain 24, 28, 24, and 24 tasks; Overall pools all 100. SR is success rate (%); Act., Tok., and Time are mean agent-level actions, recorded language-model tokens (millions), and elapsed minutes, including failed tasks. Boldface marks the better value within each model pair (both for ties). Annotations below with-skill values show relative changes against no skill: green ↑ denotes SR gains, green ↓ reductions in Act., Tok., or Time, red ↓ SR declines, and gray 0.0% no change. A dash (-) indicates an undefined relative change when the no-skill baseline is zero.

		L10				Goal				Object				Spatial				Overall			
Model	Skill	SR	Act.	Tok.	Time	SR	Act.	Tok.	Time	SR	Act.	Tok.	Time	SR	Act.	Tok.	Time	SR	Act.	Tok.	Time
gpt-5.6-sol	no	0.0	30.04	6.004	11.11	14.3	21.07	4.710	10.82	4.2	28.00	5.692	12.89	25.0	25.88	5.115	13.43	11.0	26.04	5.354	12.01
	with	**41.7**	**4.83**	**1.846**	**8.79**	**67.9**	**1.61**	**0.455**	**2.92**	**79.2**	**1.25**	**0.442**	**2.81**	**62.5**	**1.96**	**0.528**	**3.32**	**63.0**	**2.38**	**0.803**	**4.40**
		-	↓83.9%	↓69.3%	↓20.9%	↑375.0%	↓92.4%	↓90.3%	↓73.0%	↑1800.0%	↓95.5%	↓92.2%	↓78.2%	↑150.0%	↓92.4%	↓89.7%	↓75.2%	↑472.7%	↓90.9%	↓85.0%	↓63.4%
Kimi-K3	no	4.2	21.25	3.471	69.63	46.4	15.54	2.545	48.41	**75.0**	16.42	2.176	30.21	45.8	18.92	3.132	52.44	43.0	17.93	2.819	50.10
	with	**41.7**	**5.21**	**1.541**	**19.30**	**60.7**	**5.68**	**1.348**	**17.20**	**75.0**	**1.21**	**0.335**	**4.98**	**70.8**	**5.17**	**1.490**	**19.65**	**62.0**	**4.37**	**1.185**	**15.36**
		↑900.0%	↓75.5%	↓55.6%	↓72.3%	↑30.8%	↓63.4%	↓47.0%	↓64.5%	0.0%	↓92.6%	↓84.6%	↓83.5%	↑54.5%	↓72.7%	↓52.4%	↓62.5%	↑44.2%	↓75.6%	↓58.0%	↓69.3%
claude-opus-5	no	8.3	9.21	3.356	25.13	46.4	7.25	2.295	20.03	70.8	11.75	3.196	18.05	50.0	9.71	2.526	16.74	44.0	9.39	2.821	19.99
	with	**41.7**	**5.29**	**1.249**	**12.46**	**60.7**	**4.25**	**0.536**	**6.54**	**83.3**	**1.21**	**0.192**	**2.54**	**70.8**	**3.04**	**0.667**	**7.71**	**64.0**	**3.48**	**0.656**	**7.28**
		↑400.0%	↓42.5%	↓62.8%	↓50.4%	↑30.8%	↓41.4%	↓76.6%	↓67.3%	↑17.6%	↓89.7%	↓94.0%	↓85.9%	↑41.7%	↓68.7%	↓73.6%	↓54.0%	↑45.5%	↓62.9%	↓76.7%	↓63.6%

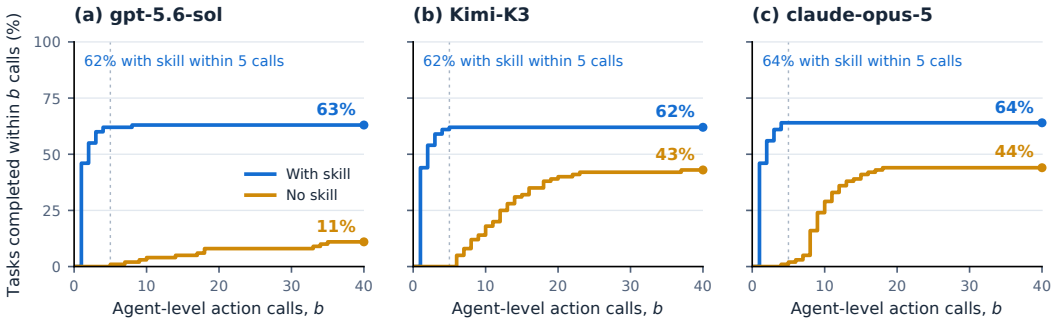


Figure 3: Empirical completion profiles for (a) gpt-5.6-sol, (b) Kimi-K3, and (c) claude-opus-5. Each curve shows the percentage of all 100 tasks completed within b agent-level action calls, on a linear horizontal axis. The dashed line marks five calls.

5.2 RQ1: Effectiveness of Embodied Skills

Task success and completion within few calls. The three panels in Figure 3 show that embodied skills improve both final task success and completion within a small number of agent-level action calls. Embodied skills increase success by 52 points for gpt-5.6-sol, 19 for Kimi-K3, and 20 for claude-opus-5. Mean action calls per task decrease by 90.9%, 75.6%, and 62.9%, respectively. Thus, the gains concern not only whether agents finish a task, but how much high-level interaction is needed to do. Their occurrence across three agents and multiple task domains supports embodied skills as reusable capabilities and guidance, rather than solutions restricted to one agent or task type.

Domain-specific strengths under shared skill provision. As illustrated in Table 1, with the same selected skill sets, the three agents achieve similar overall success (62-64%) but retain distinct domain strengths: gpt-5.6-sol leads on Goal, claude-opus-5 on Object, and Kimi-K3 and claude-opus-5 tie on Spatial. All three reach only 41.7% on L10, indicating a shared challenge with multi-stage tasks. Common skill provision therefore does not eliminate model-specific differences in task interpretation and skill coordination.

5.3 RQ2: Composition of Embodied Skills

We evaluate five skill configurations with gpt-5.6-sol on the same 100 tasks: task-conditioned **Selected** sets, **BM25** lexical retrieval, **Embedding** retrieval, and fixed **HarnessVLA (ESB)** and **Zetta (ESB)** compositions. Embedding ranks candidates by cosine similarity between normalized BGE-small-en-v1.5 representations of public queries and skill documents. Figure 4 shows that Embedding

Table 2: Five skill configurations on Clean100 (seed 26), each evaluated on all 100 tasks. Success rates (SR, %) are grouped by domain (24/28/24/24 tasks) and perturbation (25 tasks each). Actions, tokens, and time are per-task means, including failures. Boldface and underlining indicate the best and second best values in each column, respectively. All settings use gpt-5.6-sol with Codex; (ESB) denotes our skill-based reconstruction.

	Domain SR (%)				Perturbation SR (%)				Overall			
Setting	L10	Goal	Object	Spatial	Lang.	Object	Task	Swap	SR (%)	Actions	Tokens (k)	Time (min)
Selected	41.7	67.9	79.2	62.5	**92.0**	**84.0**	40.0	36.0	63.0	**2.38**	**803.2**	**4.40**
BM25	4.2	46.4	33.3	58.3	68.0	44.0	8.0	24.0	36.0	8.09	2,731.6	9.76
Embedding	25.0	39.3	45.8	58.3	72.0	64.0	8.0	24.0	42.0	12.73	2,918.7	11.03
HarnessVLA (ESB)	**50.0**	**78.6**	**100.0**	**79.2**	**92.0**	56.0	**80.0**	**80.0**	**77.0**	14.70	2,113.8	14.12
Zetta (ESB)	33.3	**78.6**	91.7	70.8	72.0	60.0	68.0	76.0	69.0	14.51	2,399.7	16.66

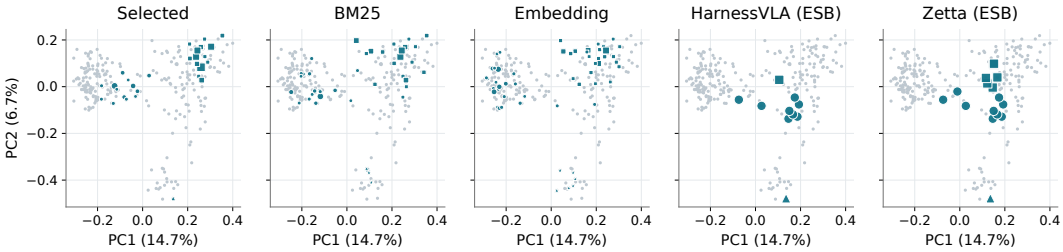


Figure 4: Skill distributions under five provision settings. A shared PCA projects 384-dimensional BGE skill-document embeddings. Gray points show the reference skill documents; teal points show selected skills, with area increasing with the number of tasks selecting each skill. Circles, triangles, and squares denote action, perception, and experience skills. The panels contain 28, 41, 54, 10, and 17 distinct selected skills, respectively, across 100 tasks. The projection retains 21.5% of embedding variance and illustrates document semantics rather than behavioral equivalence.

selects 54 distinct skills across the task suite, compared with 41 for BM25 and 28 for Selected; the two reconstructed harnesses use fixed sets of ten and seventeen skills.

Table 2 shows that Embedding reaches 42% success, compared with 36% for BM25, 63% for Selected, 77% for HarnessVLA (ESB), and 69% for Zetta (ESB). Its six-point advantage over BM25 accompanies more action calls (12.73 versus 8.09) and longer execution (11.03 versus 9.76 minutes). Broader retrieval across the repository therefore does not imply more efficient task completion. Embedding supplies no action skill on seven tasks, compared with twenty for BM25; all seven end without a physical action or task success. All they have access are native cotrollers. The wine-bottle case in Appendix B.2 illustrates this distinction between relevant guidance and executable capability.

The domain results also rule out a uniform ranking of the retrieval methods. Embedding exceeds BM25 on L10 (25.0% versus 4.2%) and Object (45.8% versus 33.3%), falls below it on Goal (39.3% versus 46.4%), and matches it on Spatial (58.3%). Across perturbations, it scores higher on language and object changes but matches BM25's 8% on task changes and 24% on swap. In contrast, HarnessVLA (ESB) reaches 80% on both task changes and swap, while Selected leads on object perturbations at 84%. These profiles motivate selection strategies that account for executable coverage and useful combinations of capabilities, alongside semantic relevance.

5.4 RQ3: Complementarity with state-of-the-art models

Internalized competence and execution cost. Following the gains across three agents in RQ1, we separately examine whether embodied skills also complement GPT-6-astra. Table 3 shows that GPT-6-astra achieves 92% success with no skill, substantially exceeding the 61% of the with-skill setting and outperforming it in every domain. This gap suggests that GPT-6-astra may already have internalized some of the knowledge and capabilities represented by embodied modules, allowing it to solve these tasks through native robot tools. Nevertheless, embodied skills substantially reduce

Table 3: GPT-6-astra settings on Clean100 (seed 26). Boldface and underlining indicate the best and second best values in each column, respectively. Annotations next to Agent-Directed Reuse values show relative changes from no skill setting: red ↓ denotes lower SR, and green ↓ denotes reduced actions, tokens, or time.

	Domain SR (%)				Overall			
Configuration	L10	Goal	Object	Spatial	SR (%)	Actions	Tokens (k)	Time (min)
With skill	37.5	67.9	79.2	58.3	61.0	**2.24**	**563.5**	**3.40**
With Jev skill	58.3	<u>92.9</u>	79.2	<u>91.7</u>	81.0	20.90	<u>1,142.0</u>	<u>8.74</u>
No skill	**70.8**	**100.0**	**100.0**	**95.8**	**92.0**	16.59	2,300.6	20.52
Agent-Directed Reuse	<u>62.5</u>	89.3	<u>95.8</u>	<u>91.7</u>	<u>85.0</u> ↓7.6%	<u>12.87</u> ↓22.4%	1,976.3 ↓14.1%	18.09 ↓11.9%

recorded execution effort: mean actions decrease by 86.50%, tokens by 75.51%, and time by 83.43% per task. These results reveal a tradeoff between success rate and the lower execution cost.

Agent-Directed Reuse. Realizing the contrast above, we propose a new interaction paradigm between agents and embodied skills: can an agent preserve its own task-solving ability while leverage embodied skills to reduce execution effort? We investigate **Agent-Directed Reuse**, where GPT-6-astra is allowed to inspect, use, modify, combine, or ignore these embodied skills completely according to itself. Skill usage nowadays asks agents to follow workflow in the skills, limitating the potential of the powerful model. This configuration reaches 85% success, recovering 24 percentage points over the with-skill setting while remaining seven points below no skill. Relative to no skill, it reduces mean actions, recorded tokens, and elapsed time by 22.4%, 14.1%, and 11.9%, respectively. The result supports Agent-directed interaction paradigm as a promising route to complementarity.

Jev as an embodied skill. Jev offers a specialized decision interface that selects among specified options and returns their probabilities (TypeSafe AI, n.d.). Drawing on the robot-control workflow of EmbodiedJev (FBddcz, n.d.), we package Jev as an embodied skill and evaluate GPT-6-astra with this single skill in our benchmark. We encapsulate Jev as a waypoint-execution skill that selects discrete motion directions and gripper commands for GPT-6-astra-specified subgoals, while GPT-6-astra retains visual grounding, task planning, and recovery. This combination achieves 81% success, 20 percentage points above the original with-skill setting but below Agent-Directed Reuse and no skill. Mean runtime falls to 8.74 minutes despite more agent-level action calls. Success is high on Goal and Spatial but lower on L10, leaving multi-stage tasks challenging.

6 Conclusion

We introduced EmbodiedSkillBench, a benchmark for evaluating how well agents use embodied modules to solve physical tasks. By organizing action, perception, and experience resources into a common skill representation and a reusable repository, EmbodiedSkillBench supports comparisons across agents and felxible skill selection under a shared interaction protocol.

Our results show that embodied skills can improve task success and reduce execution effort, but their value depends on how they complement an agent's existing capabilities. Semantic relevance alone does not ensure that selected skills provide executable, complementary capabilities, while prescribed workflows may constrain agents already capable of effective native control, such as GPT-6-astra. Agent-Directed Reuse and the Jev model integration on GPT-6-astra illustrate how flexible reuse and selective delegation can improve efficiency, although preserving the success rate of native control remains a challenge.

Future work should develop skill selection strategies that jointly consider task requirements, agent capabilities, and compatibility among skills. Physical feedback could guide when agents delegate to a skill, resume direct control, or revise a procedure. Retaining verified revisions across tasks could further support continual adaptation, making embodied skills more responsive to changing tasks and agent capabilities.

Ethics Statement

This work evaluates agents in simulation and does not involve human subjects. Skill packages are provenance-linked to their source artifacts, and the benchmark keeps private simulator state, model weights, and credentials outside agent workspaces. We will document licenses and any restrictions for redistributed code, data, and model checkpoints. Task execution and recovery procedures are intended for controlled simulation; deployment on a physical robot requires separate safety, calibration, and human-oversight validation.

Reproducibility Statement

The supplementary material provides the repository of 250 embodied skills, a machine-readable skill index, the common module protocol, and source-resource snapshots. It includes scripts, prompts, and configuration examples for skill construction and resource access, as well as offline evaluation code for computing and aggregating task success, action counts, token usage, and elapsed time from rollout records. The supplementary material also includes benchmark execution code and environment setup scripts to support reproduction of the experiments. Usage instructions are provided to support reproduction of these procedures.

AI Use Statement

We used large language models to assist with writing and polishing the manuscript, including improving language clarity, reducing redundancy, and refining paragraph structure. We used them to support literature retrieval and discovery, including suggesting search terms and identifying potentially relevant prior work for the related-work discussion. We also used them for research execution: an LLM-based authoring agent inspected structured module descriptions and interface contracts, selected resources for requested capabilities, and generated operational skill instructions and metadata; LLM agents were additionally evaluated as task controllers under the benchmark protocol. The authors take full responsibility for the final content of this paper, including the accuracy and integrity of its text, citations, claims, and conclusions, and for verifying material produced with AI assistance.

References

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Conference on Robot Learning (CoRL)*, pp. 17–40, 2025.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023a.

Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on robot learning*, pp. 287–318. Pmlr, 2023b.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle,

Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. Sam 3: Segment anything with concepts, 2026.

Christopher Clark, Jieyu Zhang, Zixian Ma, Jae Sung Park, Rohun Tripathi, Sangho Lee, Mohammadreza Salehi, Jason Ren, Chris Dongjoo Kim, Yinuo Yang, et al. Molmo2: Open weights and data for vision-language models with video understanding and grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 28652–28668, 2026.

Xin Ding, Liang Mi, Mingzhe Huang, Zixuan Wang, Chao Zhang, Zixu Hao, Fu Chen, Xiangyu Li, Yikai Zheng, Yaoyu Guo, et al. Zetta ζ: An efficient closed-loop embodied harness for self-evolving physical intelligence. *arXiv preprint arXiv:2608.16590*, 2026.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

FBddcz. EmbodiedJev: MuJoCo robot decision workbench with MiniCPM5-2B, Jev and compatible model APIs. Software repository, n.d. URL `https://github.com/FBddcz/embodied-jev`. Accessed September 26, 2026.

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.

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. In *Conference on Robot Learning (CoRL)*, pp. 540–562, 2023a.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner Monologue: Embodied reasoning through planning with language models. In *Conference on Robot Learning (CoRL)*, pp. 1769–1782, 2023b.

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), 2020.

Ryangkyung Kang, Hongcheol Cho, and Youngeun Kim. SkillRet: A large-scale benchmark for skill retrieval in LLM agents. *arXiv preprint arXiv:2605.05726*, 2026.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Robotics: Science and Systems (RSS)*, 2025a.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning (CoRL)*, pp. 2679–2713, 2025b.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, Chujun Tao, Binxu Li, Xuandong Zhao, Hejia Geng, Xiaojun Wu, Junwei Zhou, Xiaokun Chen, Hanwen Xing, Yubo Li, Qunhong Zeng, Di Wang, Yuanli Wang, Roey Ben Chaim, Penghao Jiang, Haotian Shen, Luyang Kong, Xinyi Liu, Runhui Wang, Xuanqing Liu, Jiachen Li, Xin Lan, Yueqian Lin, Wengao Ye, Junwei He, Songlin Li, Yue Zhang, Yipeng Gao, Yijiang Li, Ze Ma, Liqiang Jing, Tianyu Wang, Kaixin Li, Yiqi Xue, Haoran Lyu, Yizhuo He, Yuchen Tian, Shutong Wu, Bowei Wang, Yixuan Gao, Bo Chen, Litong Liu, Sikai Cheng, Jiajun Bao, Shuaicheng Tong, Shuwen Xu, Terry Yue Zhuo, Tinghan Ye, Qi Qi, Miao Li, Longtai Liao, Zelin Tan, Chang Shi, Xilin Tang, Srinath Tankasala, Boqin Yuan, Yaoyao Qian, Jianhong Tu, Chenguang Wang, Yizhou Sun, Wei Wang, Aaron Taylor, Ziyue Yang, Changkun Guan, Zhikang Dong, Xinyu Zhang, Steven Dillmann, Han-chung Lee, and Dawn Song. SkillsBench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 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 *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500, 2023.

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

Runyu Lu, Yubo Wu, Ethan Kou, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, et al. Aspire: Agentic/skills discovery for robotics. *arXiv preprint arXiv:2607.00272*, 2026.

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.

Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. Contact-GraspNet: Efficient 6-DoF grasp generation in cluttered scenes. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13438–13444. IEEE, 2021. doi: 10.1109/ICRA48506.2021.9561877.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

TypeSafe AI. API reference. TypeSafe AI documentation, n.d. Software documentation for Jev; accessed September 25, 2026.

Renjun Xu and Yang Yan. Agent skills for large language models: Architecture, acquisition, security, and the path forward. *arXiv preprint arXiv:2602.12430*, 2026.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, Heng Ji, Huan Zhang, and Tong Zhang. EmbodiedBench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 70576–70631. PMLR, 2025.

Barry Zhang, Keith Lazuka, and Mahesh Murag. Equipping agents for the real world with Agent Skills. Anthropic Engineering, 2025a. Published October 16, 2025; accessed September 25, 2026.

Junyi Zhang, Jiaxin Ge, Hanjun Yoo, Letian Fu, Zihan Yang, Yaowei Liu, Raj Saravanan, Shaofeng Yin, Justin Yu, Dantong Niu, Zirui Wang, Roei Herzig, Ken Goldberg, Yutong Bai, David M. Chan, Ion Stoica, Angjoo Kanazawa, Jiahui Lei, Haiwen Feng, and Trevor Darrell. Playful agentic robot learning. *arXiv preprint arXiv:2606.19419*, 2026a.

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

Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Yi Nie, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, Jiaji Rao, Zhengru Fang, Changxu Wei, Yu Wang, Wenbo Ding, and Chao Yu. Harness VLA: Steering frozen VLAs into reliable manipulation primitives via memory-guided agents. *arXiv preprint arXiv:2607.08448*, 2026b.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Ya-Qin Zhang, Jiangmiao Pang, Jingjing Liu, Tai Wang, and Xianyuan Zhan. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025.

A RQ1: NO-SKILL INTERFACE AND A MULTI-OBJECT CASE STUDY

The following three case studies connect configuration choices to the actions and observations of individual tasks. Each comparison fixes the task ID and environment seed (26), and follows the evaluated configuration's own recorded rollout. Cases are selected for their explanatory trajectories: multi-object progress for RQ1, grounding and phase composition for RQ2, and controller switching and waypoint revision for RQ3. They illustrate mechanisms behind the aggregate results, rather than estimating their frequency. Outcomes come from the official task status. Action indices refer to state-changing agent calls. Tables report cumulative input plus output tokens, counting cached input once, and recorded elapsed time.

A.1 THE NO-SKILL FRAMEWORK

What the agent receives. The agent controls one persistent LIBERO-Pro episode through a typed, task-agnostic robot interface. Its initial message contains the instruction and the external-camera and wrist-camera RGB observations. Subsequent observations contain the current robot state, execution feedback, and official task status. The controller reference specifies coordinate frames, units, gripper semantics, and geometric tolerances. Task decomposition, target identification, grasp selection, and recovery decisions are made by the language model from these observations.

The native interface uses a Franka Panda with a joint-position controller. Cartesian commands pass through a frozen PyRoKi inverse-kinematics (IK) gate; invalid solutions are rejected before execution. Positions and translations use metres in the LIBERO world frame, with $+z$ pointing upward. Full orientations use quaternions ordered (x, y, z, w). Camera calibration and pixel-depth queries provide geometric grounding of agent-selected pixels. They do not assign semantic identities to those pixels. The native RGB-D resolution is 256×256; the interface's higher-resolution view is a deterministic enlargement.

Table 4: No-skill tool surface. Read-only calls leave the physical state unchanged. Each accepted state-changing call returns a fresh observation.

Interface	Function supplied to the agent
observation, end-effector state	Current instruction, RGB, status, end-effector pose, and gripper state.
camera calibration, pixel depth, pixel-to-world projection	Camera intrinsics/extrinsics, metric depth, and calibrated back-projection of a selected pixel.
gripper contact	Identity-free gripper-width/contact evidence; the agent checks the images to assess a grasp.
IK pose check	Read-only feasibility check of a proposed full pose.
relative end-effector motion, absolute end-effector motion, full-pose IK motion	Bounded relative translation, absolute Cartesian motion, and full-pose IK motion.
gripper control	Open or close the gripper while holding the measured arm joints.

Interaction and budgets. Read-only queries may precede an action. The first physical call in a model response ends that round, and the agent consumes its returned RGB before choosing another action in a later response. Task interaction in the no-skill condition is through the registered native tools. The gpt-5.6-sol, Kimi-K3, and claude-opus-5 cases below use Codex CLI, Kimi Code CLI, and Claude Code, respectively.

The following card presents excerpts from the staged no-skill system instructions. Throughout the appendix, interface and package identifiers are displayed as readable labels; prompt wording is otherwise retained.

No-skill system prompt

You control one persistent embodied episode without an Embodied Skill Library. No task-level skill, VLA, SAM3, learned perception, semantic coordinate service, grasp proposal, memory, recipe, shell, Python, raw HTTP, simulator object, or RPent API is available. Use only the benchmark-owned native tools shown by the harness.

The initial user message includes the current agentview and wrist RGB. You may call read-only tools repeatedly. The first state-changing tool in one LLM response ends that physical round and returns a fresh observation with fresh RGB. Consume that result before selecting one action in a later LLM response. A second action from the same response is rejected before simulator execution. Tool/controller/contact feedback is task-agnostic evidence, not object identity, grasp proof, or official success. Stop immediately on official success, termination, or truncation.

action budget exhausted is a terminal resource boundary for the Agent: no further state-changing call can execute, so return a concise final status instead of polling read-only tools.

A.2 Case: placing both moka pots on the stove

Task and supplied skills. The instruction is "put both moka pots on the stove" (LIBERO-10, object perturbation, task 08). Two yellow moka pots begin on the counter, to the right of the stove. Completion requires moving both objects, so a successful first transfer is only an intermediate state. The selected gpt-5.6-sol, Kimi-K3, and claude-opus-5 configurations receive the same four packages. Planning is supplied by esbench-decompose-atomic, with progress tracking in multi-object progress tracking. The action package vla-pick-place-on-openvla-oft-v1 executes a bounded OpenVLA-OFT transfer. The final experience package, support-placement verification, guides placement verification.

Table 5: RQ1 case results on the same two-pot task, seed 26. Time is in minutes; tokens are in thousands.

Configuration	Outcome	Actions	Tokens (k)	Time (min)
gpt-5.6-sol / no skill	Failure	40	7,250.6	13.31
gpt-5.6-sol / with skill	Success	3	702.0	4.33
Kimi-K3 / no skill	Failure	25	4,736.5	87.70
Kimi-K3 / with skill	Success	3	317.5	3.35
claude-opus-5 / no skill	Failure	13	3,036.4	19.46
claude-opus-5 / with skill	Success	3	364.8	4.86

gpt-5.6-sol: a reusable transfer closes the execution gap. Without skills, the agent repeatedly approaches and closes around the pots, then moves toward the stove. The images show that these movements do not establish a retained payload: both pots remain on the counter at action 40, when the action budget ends. With skills, the agent first calls the pick-and-place package with object "yellow moka pot nearest the stove" and support "stove." After this 160-step operation, the first pot is above the stove and official success remains false. The next call targets "yellow moka pot remaining on the counter." Its returned image shows the first pot on the stove and the second held above the counter. A third call continues this remaining-object transfer, reaching official success after 64 further simulator steps. This is a three-call, 384-step rollout: progress tracking distinguishes the completed object from the remaining one, while the skill executes each contact-rich transfer.

Kimi-K3: recovery attempts versus retained multi-object progress. The no-skill rollout topples the first pot by action 5. It then attempts the second pot, which is also lying on the counter by action 18. At the configured 25-action boundary, neither pot is on the stove. In the skill-enabled rollout, Kimi first names the pot "closer to the stove," then names the pot "on the table" in the two subsequent calls. The second transfer finishes on action 3, after 400 simulator steps. The two successful skill-enabled agents therefore follow the same observable structure: target one object, inspect the result, target the remaining object, and continue until official completion.

claude-opus-5: reusing the same transfer across two objects. Without skills, claude-opus-5 makes 13 calls involving end-effector motion and gripper control, but the rollout ends without completing the two-pot task. With the same four packages as the other agents, it first requests the pot "closest to the stove," then the pot "farthest from the stove," and finally the pot "still on the wooden table." The three pick-and-place calls execute 160, 160, and 56 simulator steps, respectively, reaching official success after 376 steps. The with-skill rollout takes 4.86 minutes, compared with 19.46 minutes for the unsuccessful no-skill rollout. Here the same control package is reused with revised object descriptions as the scene changes.

Relation to RQ1. This case illustrates successful skill-enabled execution across all three RQ1 agents: each reuses a language-conditioned transfer while tracking which object remains to be moved. All three finish in three calls with skills, whereas their no-skill rollouts fail on this task. The shared packages therefore support both control execution and multi-object progress tracking. The target-grounding failure in Appendix C.3 examines the complementary situation in which repeated delegation is insufficient. Neither case alone establishes the frequency of these behaviors across tasks.

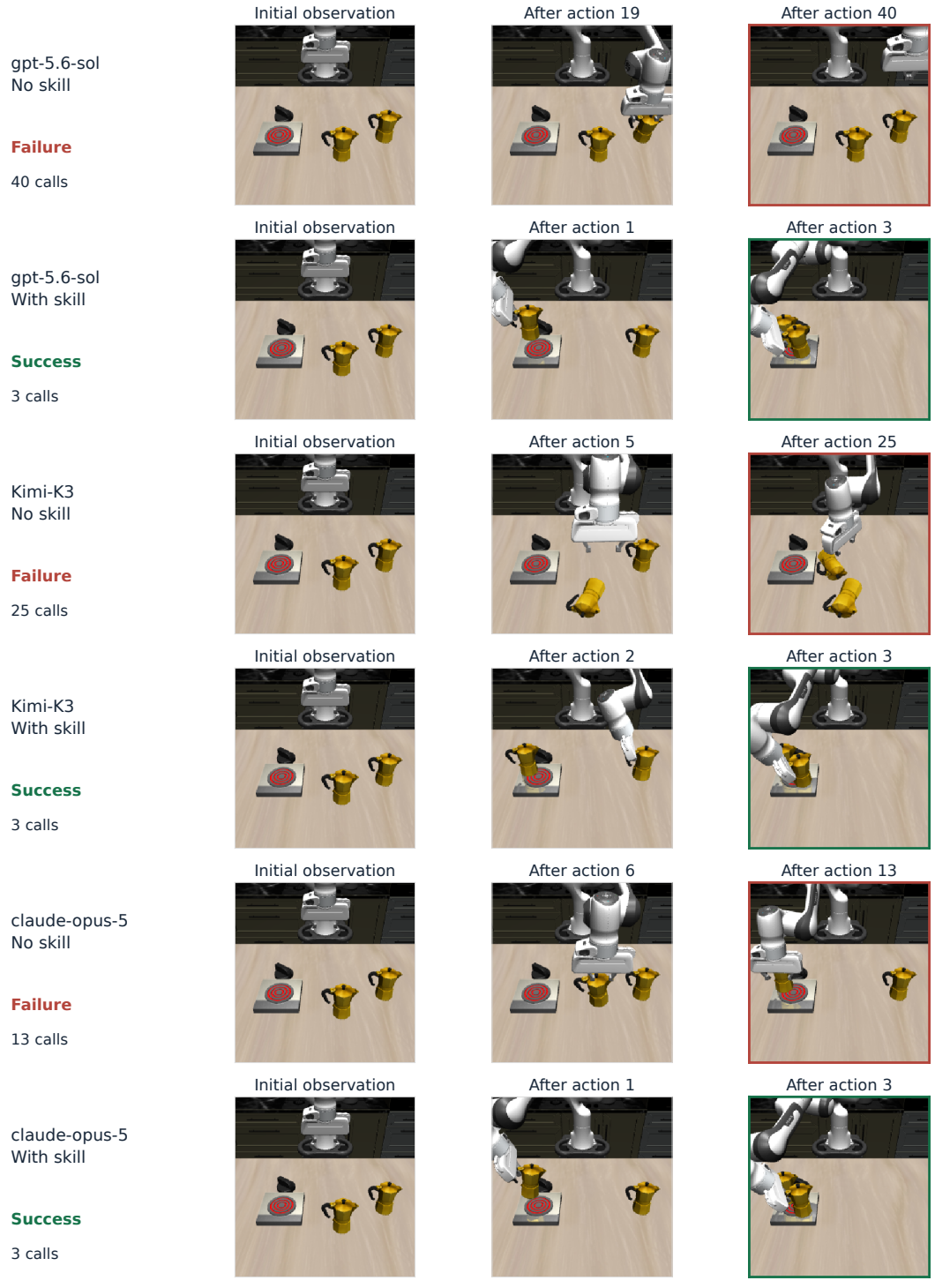


Figure 5: **RQ1: both moka pots must reach the stove.** Each row follows one recorded rollout through its initial observation, a selected intermediate observation, and its final observation. gpt-5.6-sol, Kimi-K3, and claude-opus-5 fail without skills but complete both transfers in three calls with the same supplied packages. Action numbers identify the actual recorded calls; columns are not synchronized in physical time. Borders and outcome labels mark official success or failure.

B RQ2: SKILL SELECTION AND EMBODIED HARNESS COMPOSITION

B.1 HARNESSVLA (ESB) AND ZETTA (ESB)

Common execution substrate. HarnessVLA (ESB) and Zetta (ESB) reconstruct the original control workflows (Zhang et al., 2026b; Ding et al., 2026) as compositions of action, perception, and experience packages. Each package has an agent-readable skill instruction file; executable packages additionally provide a local execution contract. The agent reads a selected package, invokes it through the workspace launcher, and receives a new observation. An experience package supplies guidance for choosing and sequencing these operations. Both reconstructions install a fixed package set across tasks and use gpt-5.6-sol with a 40-action limit.

Table 6: Mapping harness components to embodied skills. HarnessVLA (ESB) installs ten packages; Zetta (ESB) installs seventeen. Experience packages are read-only.

Component	Embodied skill implementation
Free-space motion (both)	move to, move pose, rotate wrist, rotate pitch.
Gripper transitions (both)	set gripper, release.
Local VLA contact (both)	pi0 pick, pi0 doubled; Zetta (ESB) additionally installs VLA object grasp.
Current-scene perception (both)	SAM3 object localization, invoked as a separate read-only query.
HarnessVLA (ESB) planning	HarnessVLA (ESB) memory-guided workflow, with a global rule reference and task-memory schema.
Zetta (ESB) online orchestration	Zetta (ESB) closed-loop evolution workflow, Zetta (ESB) runtime critic, Zetta (ESB) Role1 recovery review.
Zetta (ESB) campaign references	failure clustering, causal diagnosis, critic recovery proposal, gate protocol.

HarnessVLA (ESB): strategy retrieval, live grounding, and local retry. The memory-guided workflow packages abstract subgoal ordering, primitive operating ranges, and failure/repair rules. Its fixed ESB references specify, for example, free-space staging before a VLA contact call, checking gripper and image evidence before transport, and accounting for the offset between a grasped object and the end effector. Object identities, image locations, and geometric targets are bound to the current observation. The agent uses analytic skills for approach, lift, transport, descent, and retreat; the frozen VLA handles a bounded local contact phase. After a failure, the workflow directs the agent to inspect the last primitive, change a relevant factor, and retry that phase while preserving completed progress. The original method's VLA action abstraction is represented by the concrete pi0 pick and pi0 doubled packages.

Zetta (ESB): critic proposals under agent authority. The Zetta (ESB) composition adds a read-only temporal critic and a recovery-review role to the action substrate. The critic contract specifies a feature, comparison operator, threshold, dwell count, cooldown, and activation conditions. A fired critic proposes a bounded recovery. The agent, acting as Role1, checks the proposal and selects the action skills that execute an accepted recovery. The fixed ESB rule reference includes retention-loss and stalled-progress patterns, with re-grounding/re-grasping and local re-engagement procedures. Current observations determine their applicability; absolute object poses are not supplied as a motion oracle. The four campaign-role packages specify clustering, diagnosis, proposal, and gating contracts. The evaluated episodes use the fixed online composition; these campaign roles remain references during execution. This separates the packaged runtime workflow from an offline process for revising its rule set.

The following two excerpts state the operational division of labour and proposal authority in the staged system instructions; method names are displayed with the ESB suffix:

HarnessVLA (ESB): division of labour

Follow the HarnessVLA (ESB) division of labor. Use analytic move to/move pose for free-space approach, lift, transport, descent, and retreat; use rotate wrist/rotate pitch for bounded orientation changes; use set gripper/release for explicit gripper transitions; and reserve one localized pi0 pick or pi0 doubled call for the contact-rich phase it names after semantic and geometric staging are ready.

Zetta (ESB): critic and recovery authority

Run the Zetta (ESB) closed loop. Evaluate every installed critic rule once per physical action using Zetta (ESB) runtime critic semantics: feature path, operator, threshold, dwell, cooldown, and activation conditions. A critic is proposal-only. When a rule fires it rejects the current action and submits a bounded recovery proposal; apply Zetta (ESB) Role1 recovery review to accept or reject that proposal before anything executes, and only then let the installed action skills carry out an accepted recovery.

B.2 Case: placing a wine bottle on the drawer top

Task and comparison. The instruction is "Put the wine bottle on the top of the drawer" (LIBERO-Goal, swap perturbation, task 02). A bottle and a bowl are visible beside the cabinet; its dark top surface must be distinguished from the slatted drawer face. Table 7 compares all five RQ2 configurations on this task. In the Selected setting, gpt-5.6-sol receives three skills: esbench-decompose-atomic, vla-pick-place-on-openvla-oft-v1, and support-placement verification. BM25 retrieves X-VLA opening, VLA placement, and the same support-verification experience. Embedding retrieves three experience skills: narrow-container alignment, rim-collision recovery, and support-placement verification. The two embodied harnesses receive the fixed compositions in Table 6.

Table 7: RQ2 case results on the wine-bottle task, seed 26. All rows use gpt-5.6-sol.

Configuration	Outcome	Actions	Tokens (k)	Time (min)
Selected	Failure	2	592.8	3.75
BM25	Failure	16	3,989.0	15.42
Embedding	Failure	0	72.1	0.56
HarnessVLA (ESB)	Success	8	839.6	5.99
Zetta (ESB)	Success	9	1,334.9	7.01

Selected: correct language does not ensure correct contact. The agent invokes the OpenVLA-OFT package twice with the wine bottle as the object and the drawer top as the support, using 20 chunks per call. In the returned images, the policy manipulates the neighbouring bowl while the bottle remains upright on the table. The rollout does not reach official success, and the agent's final report explicitly identifies the wrong-object grasp. This trace distinguishes an agent issuing the correct object name from a delegated policy physically acting on that object. Repeating the high-level request does not resolve this mismatch in the rollout.

BM25: an executable placement skill can still lack a useful recovery path. The retrieved placement package does grasp the bottle. However, it carries the bottle toward the slatted cabinet face instead of establishing it on the dark top. The agent repeatedly revises the destination language: calls 1–6 use "top of drawer," calls 7–8 use "drawer," and later calls specify the "flat black top surface" or "black tabletop of drawer unit." All sixteen physical calls invoke VLA placement; the retrieved X-VLA opening skill is not used. The final image still shows the bottle at the slatted face and official success remains false. This case therefore exposes a utilization problem beyond selecting any exe-

cutable action: the available and chosen operations must support the geometric correction demanded by the observed failure.

Embedding: relevant guidance without an executable action. The three retrieved packages provide read-only guidance but no action interface. The agent inspects the observation and identifies that no robot-changing skill is installed. It makes no physical action calls and reports failure, consistent with the official outcome. The run ends after 0.56 minutes with 72.1k tokens; the final observation is the unchanged initial scene. This case illustrates a failure of executable coverage before manipulation begins, distinct from the repeated unsuccessful placement calls in BM25. A short rollout here reflects an inability to act, not efficient task completion.

HarnessVLA (ESB): ground, stage, and complete a local grasp. The agent first issues two SAM3 queries, one for the wine bottle and one for the drawer's top surface. Action 1 moves the open gripper above the bottle. Action 2 delegates only "grasp and lift the wine bottle directly below the gripper" to pi0 pick, for twelve chunks. Its feedback reports an unmet kinematic postcondition, with the gripper still about 0.080 m open. The agent closes the gripper explicitly at action 3 and lifts at actions 4–5; the images then show the bottle moving with the gripper. Actions 6–7 transport it above the cabinet and action 8 lowers it onto the top, where official success triggers. The frozen VLA supplies one contact phase, while the agent's selected motion and gripper skills finish the manipulation.

Zetta (ESB): feedback-guided phase composition. Zetta (ESB) also grounds the bottle and cabinet using two SAM3 queries, stages above the bottle, and makes one twelve-chunk pi0 pick call. After the same unmet grasp postcondition, it explicitly closes the gripper, lifts, transports with clearance, and lowers. Its ninth action is release, which triggers official success. This successful trace demonstrates the installed workflow's use of local feedback and explicit phase transitions. The observed correction is the gripper-and-lift continuation following the contact call; the outcome itself does not establish that a temporal critic intervention caused the success.

Relation to RQ2. Figure 6 distinguishes three requirements for effective skill use: supplying an executable capability, directing it at the intended object and destination, and composing operations that can correct an unsuccessful attempt. Embedding lacks the first requirement; Selected and BM25 have action skills but do not complete the placement. The two reconstructed harnesses use eight and nine calls and finish in 5.99 and 7.01 minutes. Their successful rollouts are longer than the two-call Selected failure and the zero-call Embedding failure, but shorter than BM25's 15.42-minute sequence of unsuccessful placement calls. Thus, low execution cost alone does not establish effective skill use. These five trajectories illustrate configuration-level differences in capability provision and use.

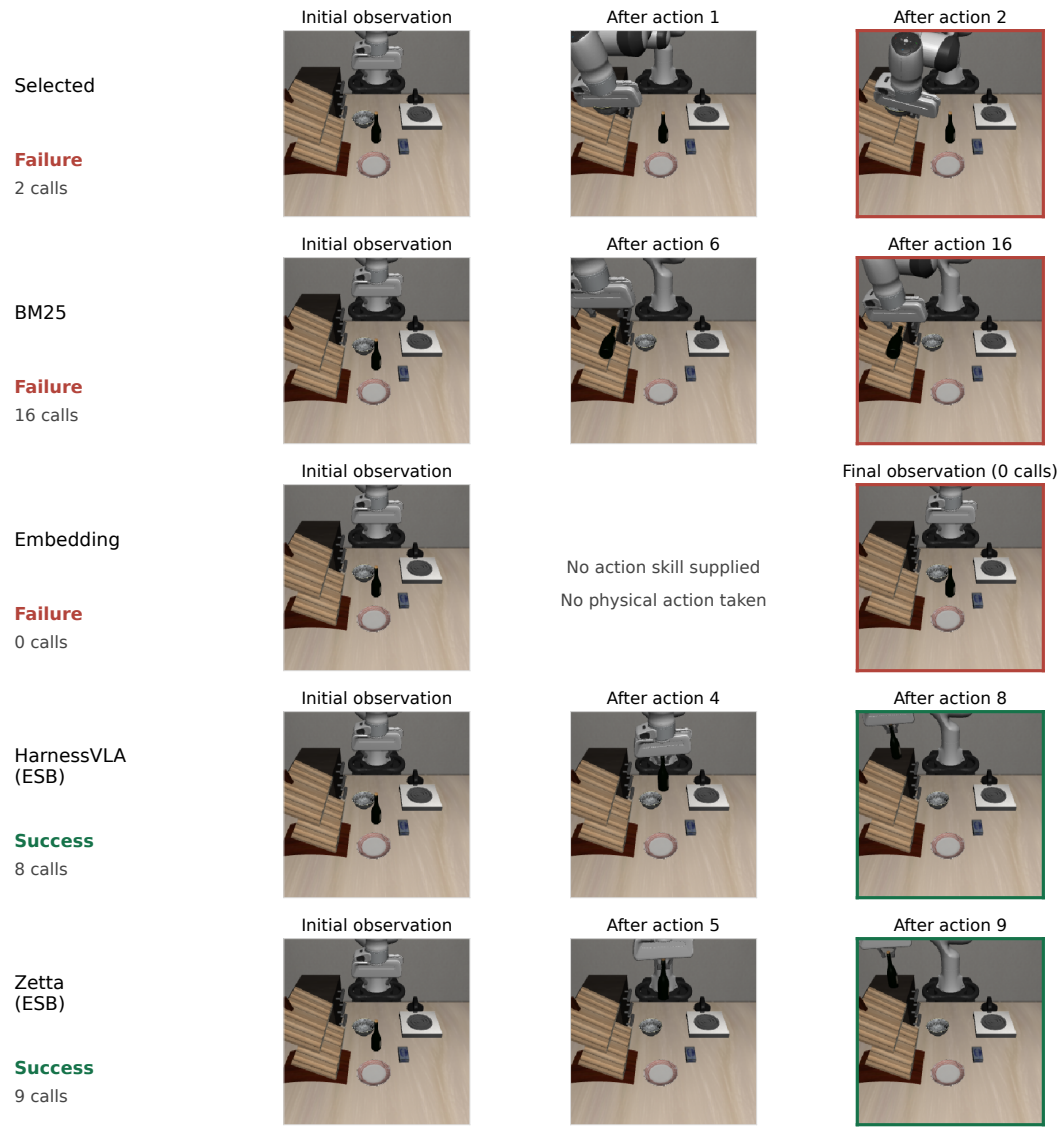


Figure 6: **RQ2: wine bottle onto the drawer top.** The Selected agent leaves the bottle on the table while manipulating the bowl. BM25 moves the bottle to the slatted face. Embedding receives only experience skills and takes no physical action; its initial and final panels show the same recorded observation. HarnessVLA (ESB) and Zetta (ESB) lift the bottle explicitly and transport it to the top surface. Intermediate panels show selected contact or lift states where actions occur, and the last panels show the final observations with official outcome labels. All images are recorded external-camera observations.

C RQ3: Agent-Directed Reuse and the Jev skill

C.1 Agent-Directed Reuse of selected resources

Resource access and control. The agent receives task-conditioned packages as editable files under the embodied-resource directory, together with native robot and geometry tools. Experience packages supply advice; executable packages provide scripts and metadata. Calling the resource launcher executes a workspace script and submits its single emitted tool call. Editing the script or its template changes that submitted call. The agent may also directly invoke the registered VLA, perception, or native robot interface. Provider weights, registered schemas, and global budgets remain fixed. This design lets the agent choose the granularity of delegation: a complete bounded VLA operation, a component from a package, or native motion planned from the current image. The action–observation boundary remains one state-changing call per model response.

The archived system prompt is reproduced below. It grants authority over both resource selection and continuation after feedback:

Agent-directed Reuse: system prompt

You are the decision-maker for this embodied task. You can solve it through your own reasoning and may outperform workflows described in the embodied resource materials.

You have the baseline robot and geometry tools, shell, Python, file reading/editing, and optional embodied resources in the embodied-resource directory. These are ordinary editable workspace materials. You may inspect, use, modify, combine, or ignore them. Their workflow recommendations are advisory; this agent-led approach takes precedence. You decide how to achieve reliable task completion while reducing unnecessary execution time and computation.

The resource directory in the task message describes the available packages. Experience packages contain advice only. Executable packages contain local scripts and metadata: the resource launcher executes a workspace script and submits its one emitted tool call. Editing its code or template changes the call actually executed. You may also call the robot, VLA, or perception tools directly. Provider weights and the global tool schemas/budgets remain fixed.

Shared interaction rules: use the public observations and registered robot tools. At most one state-changing call is allowed per model response; consume its returned fresh RGB before choosing another action. Read-only calls may be grouped. The initial attached observation is already acknowledged. Stop on official success, termination, truncation, or exhausted action budget. Task success is determined by official status. Do not access simulator internals, other episodes, or services outside the registered interface; do not edit benchmark control files or observations. Report the official outcome accurately.

C.2 Wrapping Jev as a waypoint execution skill

Package and division of labour. The Jev follow waypoint package connects GPT-6-astra's visual planning to Jev's structured option selection (TypeSafe AI, n.d.), using the waypoint controller from EmbodiedJev (FBddcz, n.d.). Its skill instruction file describes the handoff and links to a native-planner reference, which defines how to propose a visually grounded short stage. A persistent runtime exposes two tools: read-only observation and state-changing waypoint execution. GPT-6 receives this single embodied skill and chooses the task stages, target pixels, clearances, orientations, gripper goals, and revisions. Jev executes the local motion choices inside each stage. This condition delegates robot execution through the waypoint skill; the selected-resource condition above additionally supports switching to independent native robot tools. The Jev waypoint controller is packaged as a standalone, experiment-specific embodied skill and is evaluated independently of the public 250-skill repository.

The wrapper preserves the waypoint controller's target resolution, seven-channel decision queries, action conversion, and replanning boundaries. The persistent session pauses when the controller

requests a new visual plan, exposes that request to Codex, and resumes after the next skill call. The installed developer instruction is:

Jev waypoint skill: developer instruction

> Use the installed Jev follow waypoint skill for this episode. Read its instructions and native planner reference, call observe, then plan and call Jev follow waypoint until the runtime returns terminal status. Consume each returned observation before the next action. Do not infer task success yourself.

Grounding and plan schema. At each planning boundary GPT-6 sees the current external and wrist RGB, robot proprioception, its previous plan, and recent movement. The 384×384 planning images have a pixel grid and a projected tool-centre marker. A plan contains exactly seven fields: stage, intent, visual evidence, target, rotation delta, gripper, and maximum motor steps. The target can be a selected camera pixel plus a world-frame offset, a relative translation from the current tool centre, or a world coordinate grounded in an earlier observation. Pixel targets are back-projected through the corresponding depth and calibration. Relative translations and pixel-offset components are bounded by ± 0.20 m; rotation increments are bounded by ± 0.5 rad per axis. The skill receives the plan together with the observation's checkpoint identifier.

Jev's local decision. For each local decision, the wrapper supplies the resolved target, current tool pose, signed position and orientation residuals, the stage's gripper goal, and recent motion. It asks Jev to choose one option for each of six motion channels (x, y, z, r_x, r_y, r_z) and the gripper. Motion options are {negative, hold, positive}; gripper options are {open, hold, close}. The motion query asks for movement toward the target, with a 0.005 m translational tolerance and 0.025 rad rotational tolerance. Jev returns a chosen option and probabilities. The executor uses the returned choice; probabilities are recorded, and do not replace the selected direction.

For motion channel i, let e_i be its current residual and let $d_i \in \{-1, 0, 1\}$ encode Jev's choice. The normalized command is

$$u_i = d_i \min\left(0.5, \max\left(0.01, \frac{|e_i|}{s_i r}\right)\right), \qquad s_i = \begin{cases} 0.05 \text{ m}, & i \in \{x, y, z\}, \\ 0.5 \text{ rad}, & i \in \{r_x, r_y, r_z\}, \end{cases} \tag{2}$$

where $r = 5$ is the usual action-repeat length. The operational-space pose controller runs at 20 Hz; the gripper channel maps open to -1 and close to $+1$, while hold preserves its previous command. A skill call contains up to eight such decisions. It yields for a fresh visual plan when the waypoint is reached, the requested stage length expires, or three consecutive motor actions each move the tool centre less than 1 mm. Official task termination also ends execution. A repeated request with the same checkpoint and identical plan returns its saved result, preserving exactly-once physical execution across the handoff.

The evaluated Jev configuration uses Codex CLI v0.148.0, GPT-6-astra/medium, Jev 1.13.0, a 40-waypoint budget, 10,000 simulator steps, and a 5,400-second rollout timeout. Consequently, one Jev action in the tables is a waypoint submission, which can contain several local decisions and simulator steps.

C.3 Case: selecting the bowl next to the ramekin

Task and configurations. The instruction is "Pick the akita black bowl next to the ramekin and place it on the plate" (LIBERO-Spatial, task perturbation, task 08). Two visually similar bowls are present. The intended bowl is adjacent to the small ramekin; the other is closer to the plate. Table 8 compares the selected skill workflow, Agent-Directed Reuse, native no-skill control, and the Jev waypoint skill with GPT-6-astra. The selected workflow supplies target-grounding and support-verification guidance, the OpenVLA-OFT pick-and-place package, and explicit release. The reuse configuration makes the corresponding selected resources available with the additional control choices described above.

Table 8: RQ3 case results on the relational bowl task, seed 26. All rows use GPT-6-astra/medium. Jev tokens sum 431.3k GPT-6 tokens and 201.5k Jev tokens; its 14 waypoint calls contain 78 local Jev decisions.

Configuration	Outcome	Actions	Tokens (k)	Time (min)
With skill	Failure	6	1,648.8	9.01
Agent-Directed Reuse	Success	10	736.8	5.79
No skill	Success	23	3,350.8	33.98
With Jev skill	Success	14	632.8	5.42

Selected workflow: recognizing a mismatch does not guarantee correction. The first OpenVLA-OFT call names the bowl "next to the ramekin," but the returned image shows the distractor on the plate. Official success is false. A second wording produces the same result. At action 3 the agent attempts to clear "black bowl on the plate" onto the table, then returns to increasingly specific descriptions of the target: "beside the white ramekin," "closest to the ramekin," and "next to the glazed rim porcelain ramekin." Across all six calls the same pick-and-place package executes 120 chunks (960 simulator steps). The intended bowl remains beside the ramekin. The final agent report correctly identifies the distractor-placement error, yet the selected control sequence has not repaired it.

Agent-Directed Reuse: a bounded trial followed by native recovery. The agent first directly invokes OpenVLA-OFT with a five-chunk budget and the task instruction, then makes a twenty-chunk call. Both return an unsuccessful task status; the images show the bowls still on the table. Instead of issuing another placement prompt, action 3 lifts and opens the gripper under native control. Actions 4–5 position and orient it over the intended bowl and descend toward its rim. Action 6 closes the fingers, action 7 lifts the bowl, and actions 8–10 transport and lower it onto the plate. Official success arrives at action 10. The trace therefore contains two delegated VLA calls followed by eight native calls. It illustrates selective continuation and a change of controller after feedback, without requiring a claim that a local resource file was edited in this rollout.

No skill: direct grounding succeeds with placement corrections. GPT-6 identifies and lifts the intended bowl using native pose and gripper commands. Its first release occurs at action 7, after which the bowl is visibly offset relative to the plate and official success remains false. The agent then performs additional contacts, regrasping, and small pose corrections. Official success occurs at action 23. This rollout establishes that the model can ground the task and repair placement using the native interface. Its longer sequence also gives a concrete comparison for the effort saved by the two delegation designs on this particular task.

Jev skill: revise the waypoint after an empty grasp. The first plan selects external-camera pixel $(332, 191)$ at the intended bowl and adds $0.15\,\mathrm{m}$ of upward clearance. The submitted call is shown below with readable field labels. Its calibrated surface point is approximately $(-0.207, 0.335, 0.914)\,\mathrm{m}$, producing a waypoint at height $1.064\,\mathrm{m}$. In the first local decision, Jev chooses hold for x, positive for y, negative for z, hold for all three rotation channels, and open for the gripper. Equation 2 converts these choices to $(0, 0.5, -0.4352, 0, 0, 0, -1)$.

Recorded first Jev waypoint

Checkpoint	1
Stage	approach target bowl
Intent	Move above the black bowl next to the ramekin
Visual evidence	The target black bowl is at external pixel 332,192 immediately right of the small silver ramekin; the other bowl is lower right
Target	Camera pixel · external view Pixel: [332, 191] · World offset (m): [0, 0, 0.15]
Rotation delta	[0, 0, 0] rad
Gripper	open
Maximum motor steps	8

After aligning, descending, closing, and attempting a lift, the fifth waypoint returns at simulator step 155 with an empty gripper and the bowl still on the table. GPT-6's next plan explicitly notes this state, selects the shifted bowl at pixel $(321, 184)$, and revises the grasp depth. The subsequent descent targets height 0.906 m, about 2.8 cm below the first descent target. Two short closure stages are followed by a lift that visibly carries the bowl. The remaining waypoints transport, align, lower, and settle it onto the plate. Official success occurs at step 386 during waypoint 14. The recovery is a revision made by the visual planner in response to the skill's feedback; Jev continues to perform bounded local tracking within each revised stage.

Table 9: Jev waypoint progression in the RQ3 case. Ranges group consecutive recorded skill calls; simulator-step boundaries refer to the same persistent rollout.

Waypoints	Sim. steps	Observed stage and transition
1–3	0–110	Ground the intended bowl, align, and descend to the first grasp pose.
4–5	110–155	Close and attempt a lift; the returned image shows an empty grasp.
6–7	155–225	Re-ground the shifted bowl and descend to a deeper contact pose.
8–10	225–270	Close, maintain closure, and verify a lift with the bowl retained.
11–12	270–340	Transport the bowl and align it above the plate.
13–14	340–386	Lower and settle; the official success condition becomes true.

Relation to RQ3. The successful reuse, native, and Jev rollouts require 10, 23, and 14 agent-level calls and 598, 935, and 386 simulator steps, respectively. Agent-Directed Reuse takes 5.79 minutes and the Jev skill takes 5.42, compared with 33.98 for native control. These task-level values illustrate two forms of complementarity: choosing when to leave a delegated policy, and keeping high-level recovery in GPT-6 while delegating local tracking. They also explain why call counts alone do not determine runtime: a waypoint call and a VLA rollout contain different amounts of internal execution. The aggregate success/effort tradeoff remains the result of the full Clean100 evaluation; this selected case makes the underlying control decisions visible.

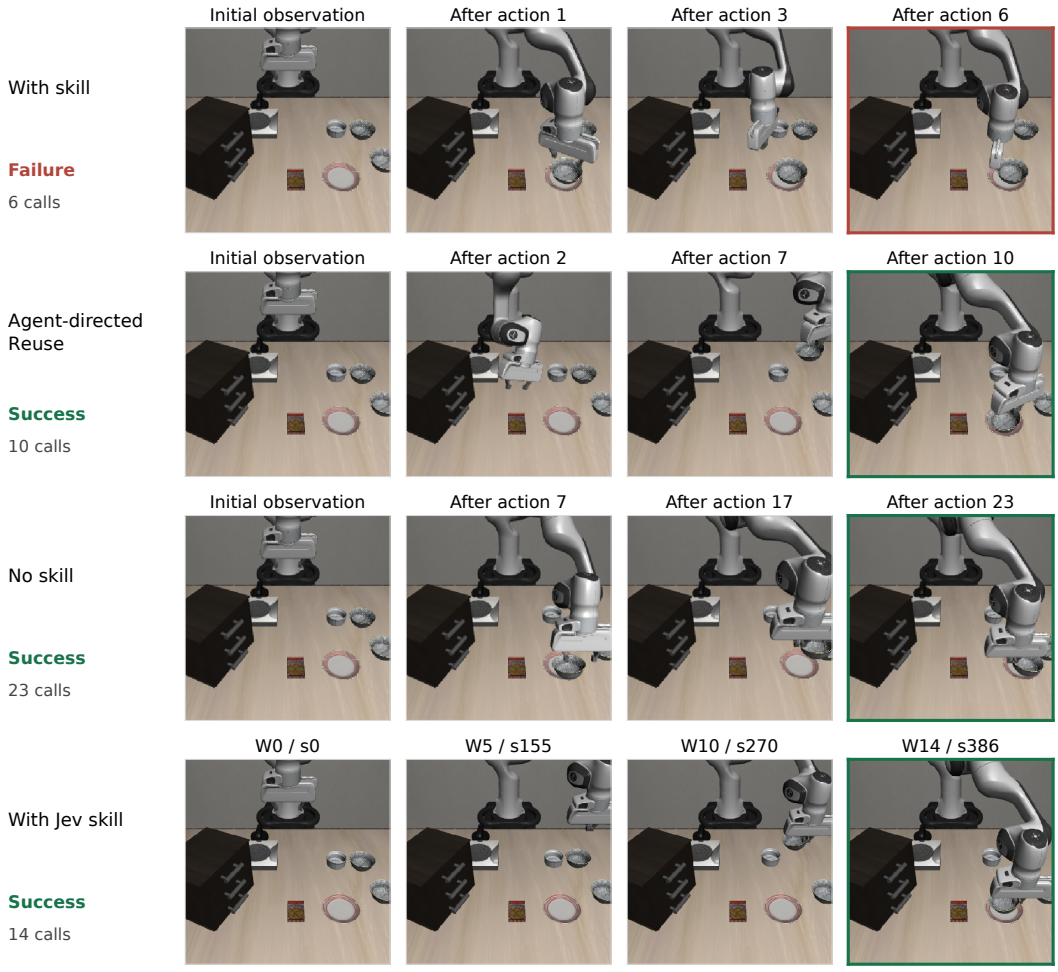


Figure 7: **RQ3: the intended bowl is the one next to the ramekin.** The selected workflow repeatedly places the distractor. Agent-Directed Reuse switches from VLA execution to native control after action 2, lifts the intended bowl by action 7, and succeeds at action 10. Native control requires additional placement corrections. With Jev, GPT-6 observes an empty grasp at waypoint 5, revises the contact, and achieves a retained lift by waypoint 10 before completing placement. Jev panels identify waypoint count (W) and simulator step (s); other rows identify agent action count. Images are the actual saved external-camera frames for each interface.