RoboUniverse-10K: Toward the Scaling Era of Agentic Robotics via Real-to-Sim Environments

Jinkun Liu[*,1,2], **Yingze Wang**[*,1,2], **Situ Wang**[*,1,2], **Deyao Hong**[*,1,2], **Mingju Gao**[*,2,3], **Wenyi Li**[*,2], **Yizhe Chi**[*,2], **Youjie Zheng**[1,2], **Kaisen Yang**[1,2], **Xiangyang Wang**[1,2], **Bingxiang He**[1], **Yuxin Zuo**[1], **Boyuan Chen**[1], **Yue Chen**[3], **Cheng Qian**[4], **Tianxing Chen**[5], **Huichi Zhou**[6], **Huaxiu Yao**[7], **Hao Tang**[3], **Hao Zhou**[1], **Ya-Qin Zhang**[1], **Calvin Xiao**[‡,2], **Tiga Altman**[‡,1,2], **Huan-ang Gao**[†,‡,1,2]

[1]**THU**, [2]**Einsia.AI**, [3]**PKU**, [4]**UIUC**, [5]**HKU**, [6]**UCL**, [7]**UNC-Chapel Hill**

[*]Equal Contribution [†]Project Lead [‡]Corresponding Authors



Agentic Robotics, in which foundation models act as coding agents to control robots through executable programs, has emerged as a compelling paradigm for embodied intelligence. Yet unlike software engineering, where massive executable environments power post-training, robotics remains constrained by small, manual benchmarks and in-distribution evaluation. To bridge this gap, we introduce **RoboUniverse-10K**, 10,807 interactive, resettable, and verifiable physics environments built from real-world videos via an automated Real2Sim and quality assurance pipeline. Each runs in an isolated, Harbor-compatible harness that hides privileged states and grading logic to mirror real-world sensing. From the same pipeline, we hold out 500 unseen tasks to establish **RoboUniverse-Verified**, an expert-verified zero-shot benchmark. Evaluating 11 closed- and open-source models reveals a huge gap: frontier models reach 85.8% success, while open-source baselines lag at 2.6% to 10.0%, bottlenecked by low-level grasping, deformable-object manipulation, and post-contact physical control. Finally, we collect about 17K multi-turn trajectories totaling 0.79B tokens with GPT-6 Astra and perform full-parameter supervised fine-tuning on Qwen3.6-35B-A3B, yielding **RoboUniverse-35B-A3B**. Our model achieves substantial zero-shot gains on RoboUniverse-Verified, lifting the success rate from 0.8% to 21.2%, while transferring effectively to LIBERO-Max, RoboCasa, RoboTwin 2.0, and a real physical Franka robot. We open-source the RL-ready environments and model to lay the foundation for large-scale reinforcement learning in Agentic Robotics.

Homepage: https://lab.einsia.ai/robo-universe
Correspondence: gha24@mails.tsinghua.edu.cn
Date: November 23, 2026

1 Introduction

Recently, *Agentic Robotics*—a paradigm in which frontier foundation models act as *Coding Agents* to control robots—has emerged rapidly. Rather than relying solely on end-to-end visuomotor mappings, these agents write and execute code that invokes perception tools, planning routines, and low-level control interfaces in a closed-loop manner [1–5], already surpassing end-to-end Vision-Language-Action (VLA) policies on several mainstream leaderboards [6–8]. Looking to the trajectory of software engineering (SWE), the rapid capability leap of coding agents has been fundamentally driven by *large-scale, executable, and automatically verifiable* post-training environments [9–14]. By mining real-world pull requests to construct hundreds of thousands of interactive repositories, the SWE community unlocked large-scale trajectory distillation and reinforcement learning with verifiable rewards (RLVR) [15, 16], yielding substantial gains for open-source models on benchmarks such as SWE-bench [17]. We posit that the cornerstone of general-purpose Embodied Intelligence is generalization, and the root of generalization lies in the scale and diversity of training environments.

Yet Agentic Robotics has not truly entered its scaling era, primarily due to two fundamental bottlenecks. *First, the bottleneck of environment scale and construction cost.* Existing robotic simulation benchmarks—such

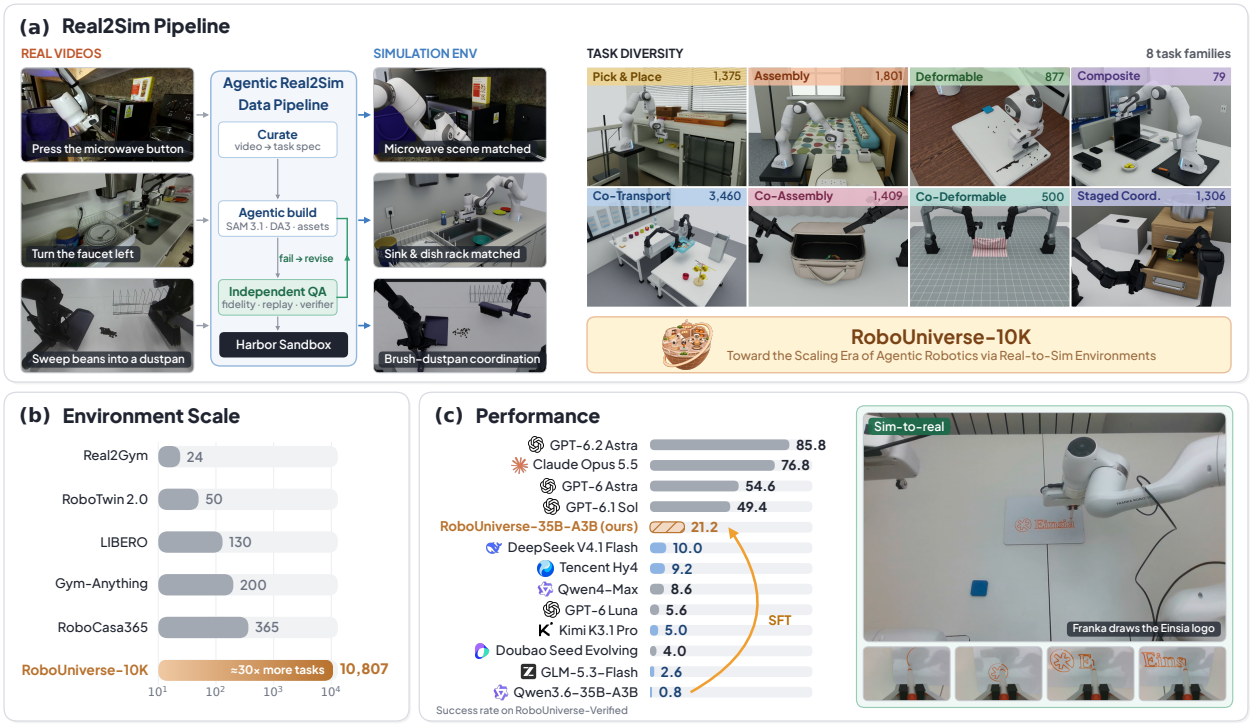

Figure 1 Overview of RoboUniverse-10K. (a) An agentic Real2Sim pipeline turns real robot videos into verifiable simulation environments spanning eight task families. (b) RoboUniverse-10K-10K is far larger than existing robot simulation benchmarks. (c) SFT on RoboUniverse-10K lifts Qwen3.6-35B-A3B from 0.8% to 21.2% on RoboUniverse-Verified, and the resulting model transfers zero-shot to a real Franka arm.

as LIBERO, RoboCasa, RoboTwin, and RoboDojo [18–22]—rely heavily on manual scene authoring and handcrafted success criteria, keeping their task counts limited to tens or a few hundred. Even recent efforts that leverage LLMs or real-world videos to synthesize simulation tasks (e.g., Real2Gym and Gym-Anything for Robotics) [23–30] remain modest in scale. Unlike software engineering tasks that can be mined directly from existing code repositories, each robotic task requires constructing a physically plausible interactive scene and authoring rigorous verification logic, incurring a far higher construction cost. *Second, the lack of generalization caused by "train on test."* Constrained by limited environment scale, prior works [31–34] typically train and evaluate agents within the same task settings and environment distributions. Training and testing in homogeneous environments not only risks overfitting, but also fails to faithfully demonstrate an agent's true generalization to unseen physical scenes and novel manipulation tasks.

To address these challenges, we introduce **RoboUniverse-10K**, a suite of 10,807 interactive, resettable, and automatically verifiable Isaac Sim [35] environments across 4,132 single-arm Franka and 6,675 dual-arm AgileX tasks for Agentic Robotics post-training and evaluation. Inspired by mining SWE tasks from real-world pull requests, we convert diverse *real-world robot manipulation videos* into distinct concrete tasks via hierarchical deduplication and normalization. Specifically, an *end-to-end Real2Sim construction and quality assurance (QA) pipeline* equips a coding agent with SAM 3.1 [36], DA3 [37], a curated asset library, and physics-modeling skills to reconstruct scenes, author reference solutions, and write verifiers, which an independent QA agent audits for physical fidelity, reset consistency, and verifier robustness against reward hacking. Meanwhile, a *Harbor-compatible isolated harness* separates the code-execution sandbox from the simulator and verifier, restricting interaction to public observations and control APIs while hiding ground-truth states and grading logic to mirror real-world constraints and support parallel rollouts.

From the same pipeline, we hold out 500 unseen, expert-verified tasks (200 single-arm and 300 dual-arm) across all eight task families to establish the zero-shot benchmark **RoboUniverse-Verified**, on which we systematically evaluate 11 leading closed- and open-source models. Our evaluation reveals two core findings. First, a sharp capability gap persists: frontier closed-source models such as GPT-6.2 Astra and Claude Opus

5.5 reach 85.8% and 76.8% overall success, whereas open-source baselines lag at 2.6% to 10.0% and struggle sharply on dual-arm coordination. Second, failures stem predominantly from physical execution rather than task comprehension. Weaker models are bottlenecked by low-level grasping, dropping to 0% to 4% success on grasp-required tasks. Even frontier models struggle with deformable-object manipulation, where cloth tasks barely benefit from capability gains, and roughly 80% of their failures occur after contact, pointing to fine-grained post-contact physical control as the primary remaining bottleneck.

To validate the utility of RoboUniverse-10K as a scalable post-training infrastructure, we conduct supervised post-training and generalization experiments. Using GPT-6 Astra [38], we collect about 17K multi-turn interaction trajectories across the training environments, totaling 0.79B tokens, and perform full-parameter supervised fine-tuning (SFT) on the open-source Qwen3.6-35B-A3B model, yielding **RoboUniverse-35B-A3B**. SFT yields substantial zero-shot gains on RoboUniverse-Verified, lifting the overall success rate from 0.8% to 21.2% (single-arm from 2.0% to 40.5% and dual-arm from 0.0% to 8.3%). Furthermore, zero-shot evaluations on LIBERO-MAX, RoboCasa, and RoboTwin 2.0 in the Embodied First Exam [7] format, alongside a sim-to-real experiment on a physical Franka robot, confirm that the acquired capabilities transfer effectively beyond our environments.

In summary, our main contributions are threefold:

- **RoboUniverse-10K:** We present an automated Real2Sim construction and agent-based quality assurance pipeline, and open-source 10,807 Harbor-compatible, isolated, RL-ready simulation environments, overcoming the scale bottleneck limitation of prior benchmarks.
- **RoboUniverse-Verified Benchmark:** We curate an expert-verified zero-shot test suite of 500 tasks, benchmark 11 closed- and open-source models, and pinpoint low-level grasping, deformable-object manipulation, and post-contact physical control as the primary bottlenecks of current embodied coding agents.
- **RoboUniverse-35B-A3B Model:** We train and open-source an SFT model on RoboUniverse-10K, lifting zero-shot success on RoboUniverse-Verified from 0.8% to 21.2% and validating the effectiveness of our environments in cross-task and sim-to-real transfer.

2 The RoboUniverse-10K Collection

2.1 Environment Design and Execution Harness

RoboUniverse-10K provides interactive, resettable, and automatically verifiable physics environments for post-training and evaluating embodied coding agents. Each environment comprises four components: a natural-language task instruction, an Isaac Sim [35] physical scene, a robot observation and control interface, and a task-specific verifier. Together, they form a closed loop supporting fast resets and parallel rollouts for both large-scale trajectory collection (SFT) and online reinforcement learning (RL).

Agent Harness and Sandbox Tools. We package each task in a format compatible with Harbor [39], the containerized evaluation framework used in Terminal-Bench [40], connecting the agent's code-execution sandbox to the simulation backend via a unified harness (Figure 2). Upon initialization, the runtime loads the scene and provides the agent with the task instruction and API documentation. Inside the sandbox terminal, the agent writes and runs Python programs that interact with the simulator exclusively through five exposed tools (Table 1). The simulation backend executes joint actuation and physical contact at the specified control frequency and returns updated observations. Simulation time advances only during `step` and `step_chunk` calls, remaining paused during observation reads, code authoring, and model inference.

Execution–Verification Isolation. To mirror real-world robotic sensing constraints, the agent sandbox is strictly isolated from the simulation backend and grading logic. Ground-truth object states, reference solutions, and verifiers remain hidden on the environment side. Each episode permits a single attempt bounded by action-step and wall-clock budgets; the agent cannot reset the scene or query intermediate scores.

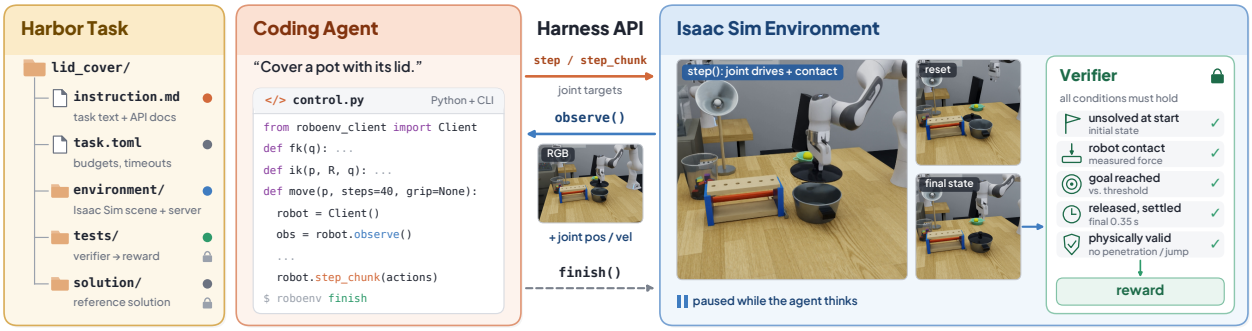

Figure 2 A RoboUniverse-10K environment. Each Harbor task packages the instruction, configuration, simulation environment, verifier, and reference solution. The coding agent acts only through the harness API, and a hidden verifier checks the final simulator state to return a binary reward.

When the agent calls `finish()` or exhausts its budget, the independent verifier inspects the execution logs and final simulator state to assign a terminal binary reward. By withholding success flags, progress signals, and privileged states, the harness requires the agent to judge physical outcomes and termination timing solely from sensor observations.

Table 1 Tools exposed by the sandbox to the coding agent.

Tool	Input	Output & Function	Advances Sim
`describe()`	—	Returns action space, observation specs, control period, and budgets.	No
`observe()`	—	Returns camera images, joint positions/velocities, sim time, and step count.	No
`step(action)`	Single action	Executes one control step and returns updated observations.	Yes
`step_chunk(actions)`	Action sequence	Executes a sequence of steps and returns the final observation.	Yes
`finish()`	—	Ends the episode and submits logs to the independent verifier.	No

2.2 Dataset Composition and Task Taxonomy

Data Sources and Scale. RoboUniverse-10K comprises 10,807 distinct simulation environments, split into 10,307 training tasks and 500 held-out zero-shot evaluation tasks (RoboUniverse-Verified). DROID [41] is the primary source of demonstrations for the single-arm Franka tasks. We also reconstruct tasks from demonstrations in BridgeData V2 [42], BC-Z [43], RT-1 [44], Berkeley Fanuc [45], Berkeley Cable Routing [46], RoboCook [47], Berkeley Autolab UR5 [48], Columbia PushT [49], RoboTurk [50], and CMU Stretch [51]. The dual-arm AgileX tasks draw on bimanual teleoperation datasets including RoboDesk, RoboCOIN [52], GM100 [53], RoboDojo [22], ViFailback [54], MM-30 [55], and RoboMIND2 [56]. Detailed task counts across embodiments and categories are shown in Figure 3.

Hierarchical Deduplication and Normalization. Multi-source robotics datasets often exhibit heavy redundancy and inconsistent task granularity. To reduce overlap while preserving task diversity, we organize demonstrations into a five-level hierarchy: *Embodiment* → *Taxonomy* → *Concrete Task* → *Layout* → *Trajectory*. We normalize task granularity at the *Concrete Task* level, treating a demonstration as distinct only when it introduces a different interaction type, a substantially different scene context, or a meaningful change in the manipulated objects. Demonstrations differing solely in camera viewpoint, surface appearance, minor spatial arrangements, initial object poses, or execution trajectories are treated as duplicates, with a single representative retained for each concrete task.

Task Taxonomy. Based on end-effector contact mechanics and bimanual coordination patterns, we organize the collection into two embodiment tracks covering eight task families, as shown in Figure 3. Single-arm Franka tasks comprise four families. *Pick & Place* involves rigid-body relocation and 3D reorientation, while *Assembly & Articulation* covers constrained assembly and sustained-contact interaction with mechanisms such

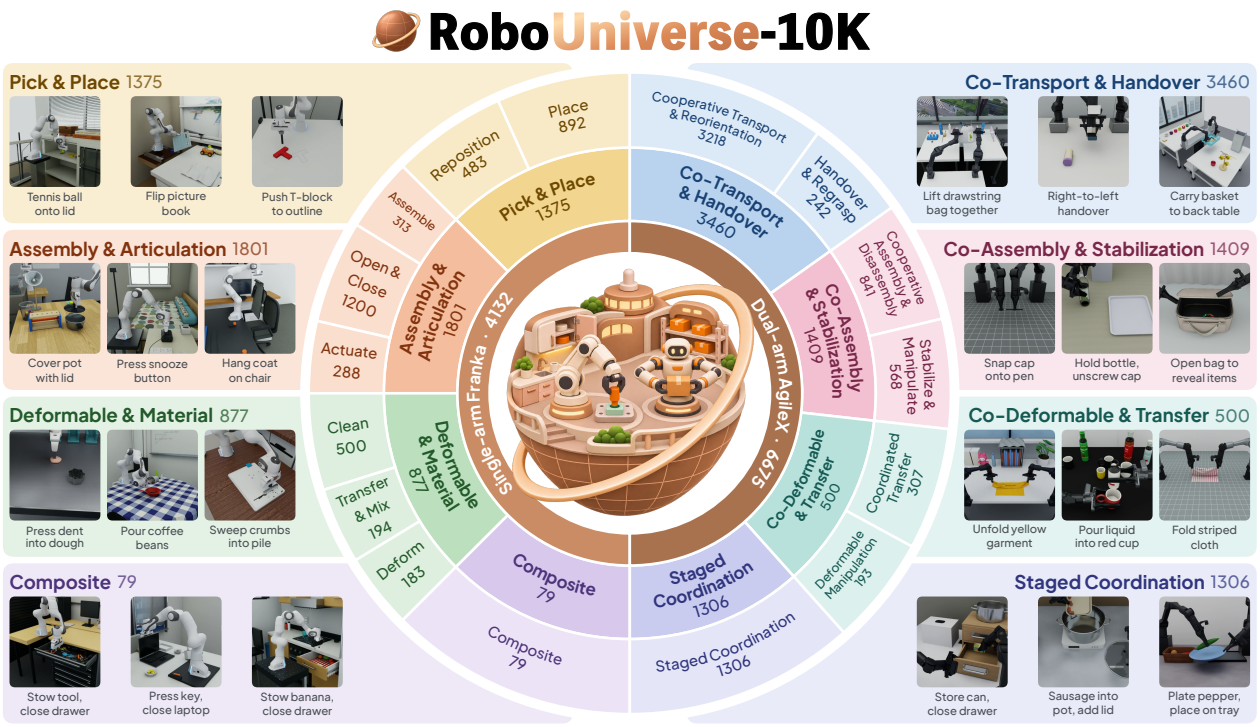

Figure 3 Task taxonomy of RoboUniverse-10K. The 4,132 single-arm Franka tasks and 6,675 dual-arm AgileX tasks span eight task families. Numbers are task counts; images show example tasks.

as drawers, doors, and buttons. *Deformable & Material* addresses shaping non-rigid objects like cloth and dough alongside wiping, pouring, and sweeping, and *Composite* chains multiple primitives within a single episode. Dual-arm AgileX tasks likewise span four families. *Co-Transport & Handover* covers joint lifting, relocation, and inter-arm passing or regrasping, whereas *Co-Assembly & Stabilization* focuses on bimanual assembly and asymmetric stabilization and manipulation. *Co-Deformable & Transfer* involves coordinated folding, flattening, and container-stabilized pouring, and *Staged Coordination* captures asynchronous, causally ordered multi-stage workflows.

2.3 Real-to-Sim Data and Quality Assurance Pipeline

RoboUniverse-10K transforms real-world robot manipulation videos into executable, verifiable RL environments through an automated Real2Sim construction and quality-assurance (QA) pipeline.

Data Curation and Task Specification. We first filter real-world manipulation videos to retain episodes with complete executions and clear goals, removing duplicate or near-duplicate scenes while maintaining broad coverage across task families. A coding agent then inspects each video to identify the manipulated objects, initial state, and desired outcome, formulating a natural-language task instruction. Each specification preserves the core objective and necessary constraints of the real demonstration while permitting any valid strategy to complete the task.

Harness-Assisted Environment Construction. We equip the coding agent with a unified Real2Sim harness integrating SAM 3.1 [36] segmentation, DA3 [37] depth estimation, a pre-built asset library, and Isaac Sim. Using segmentation masks and depth maps to recover object contours and spatial layouts, the agent selects robots, object meshes, textures, and lighting from the asset library—supplementing geometry via external retrieval or procedural code when needed. To ensure physical fidelity, we provide physics-modeling skills that guide the agent in configuring collision geometry, mass and inertia, friction, joint limits, and drive parameters, as well as verifying support, contact, and release dynamics. Supported by these tools, the agent authors the scene, reference control program, and verifier, iterating on rendered-versus-source visual comparisons and

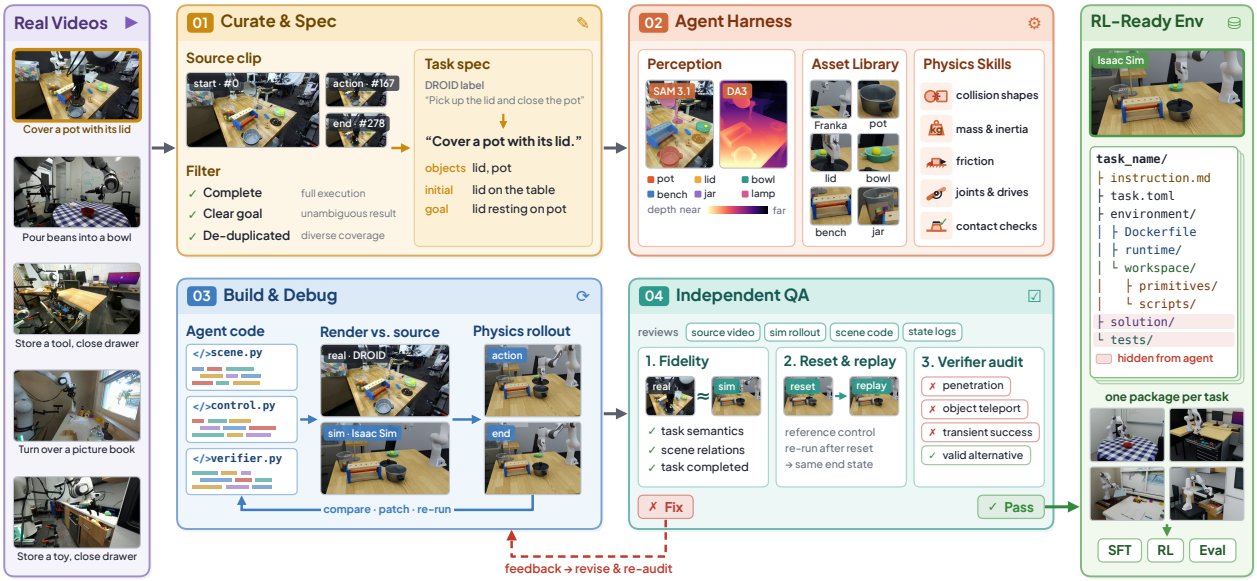

Figure 4 Real-to-Sim Data and Quality Assurance Pipeline. From each real video, a coding agent writes a task specification and builds the scene, reference controller, and verifier with perception tools, an asset library, and physics skills. An independent QA agent checks fidelity, reset and replay, and the verifier, and sends failing environments back for revision. Accepted environments are packaged as Harbor tasks.

physics-execution feedback until the robot solves the task through genuine contact, friction, and dynamics rather than scripted motion.

Independent Agent QA. Each generated environment is audited by a separate QA agent against the source video, simulation rollout, scene code, and state logs. The audit enforces three checks: (1) *fidelity*, verifying that task semantics, key spatial relations, and physical completion match the source video; (2) *reset and replay*, confirming that the reference controller reliably reproduces the task after environment reset; and (3) *verifier audit*, testing failure cases and valid strategy variants to eliminate false positives and false negatives—specifically guarding against object penetration, teleportation, and transient success. Environments that fail any check are returned to the builder agent with diagnostic feedback for revision and re-audit; passing environments are packaged into isolated Harbor tasks.

3 The RoboUniverse-Verified Benchmark

3.1 Experimental Setup

Tasks and Models. We evaluate on the 500 zero-shot evaluation tasks of RoboUniverse-Verified, comprising 200 single-arm Franka tasks and 300 dual-arm AgileX tasks that together cover all eight task families in Figure 3. The single-arm tasks comprise 45 Pick & Place, 68 Assembly & Articulation, 68 Deformable & Material, and 19 Composite tasks; the dual-arm tasks comprise 93 Co-Transport & Handover, 99 Co-Assembly & Stabilization, 78 Co-Deformable & Transfer, and 30 Staged Coordination tasks. We evaluate seven closed-source and four open-source models, listed in Table 2. The GPT-series models run in their native Codex agent harness, and all other models run in Claude Code; the reasoning effort of each model is indicated in the table.

Evaluation Protocol. Each task permits a single attempt: the agent can neither reset the scene nor query intermediate scores. The control-step budget varies by task from 250 to 4,000 steps, with a median of 3,000. The evaluation metric is the task success rate as determined by the independent verifier.

Table 2 Main results on RoboUniverse-Verified. Success rate (%) of each model with its coding-agent harness. Badges give the reasoning effort; *Overall* covers all 500 tasks; the best result in each column is in **bold**.

Model & Reasoning effort	Harness	Single-arm				Dual-arm				Overall
		Pick & Place	Assembly & Articulation	Deformable & Material	Composite	Co-Transport & Handover	Co-Assembly & Stabilization	Co-Deformable & Transfer	Staged Coordination	
Closed-source Models										
GPT-6.2 Astra max	Codex	**86.7**	**88.2**	**79.4**	**84.2**	**96.8**	**83.8**	**75.6**	**93.3**	**85.8**
Claude Opus 5.5 max	Claude Code	80.0	80.9	66.2	68.4	87.1	74.7	**75.6**	70.0	76.8
GPT-6 Astra max	Codex	66.7	54.4	42.6	42.1	65.6	47.5	55.1	60.0	54.6
GPT-6.1 Sol max	Codex	53.3	50.0	32.4	47.4	61.3	56.6	39.7	46.7	49.4
GPT-6 Luna max	Codex	26.7	5.9	4.4	10.5	4.3	0.0	3.8	0.0	5.6
Qwen4-Max xhigh	Claude Code	31.1	20.6	4.4	10.5	0.0	5.1	6.4	0.0	8.6
Doubao-Seed-Evolving-1105 high	Claude Code	15.6	10.3	5.9	0.0	1.1	0.0	1.3	0.0	4.0
Open-source Models										
DeepSeek-V4.1-Flash-1017 max	Claude Code	33.3	17.6	8.8	5.3	2.2	7.1	7.7	3.3	10.0
Tencent-Hy4 high	Claude Code	20.0	19.1	14.7	26.3	2.2	3.0	2.6	6.7	9.2
Kimi-K3.1-Pro max	Claude Code	11.1	5.9	4.4	5.3	2.2	2.0	10.3	0.0	5.0
GLM-5.3-Flash max	Claude Code	8.9	5.9	4.4	5.3	0.0	1.0	0.0	0.0	2.6

3.2 Main Results

Overall Performance. Table 2 reports the success rate of each model on each of the eight task families, together with its overall success rate. Performance differs sharply across models: GPT-6.2 Astra and Claude Opus 5.5 achieve overall success rates of 85.8% and 76.8%, GPT-6 Astra and GPT-6.1 Sol reach 54.6% and 49.4%, and none of the remaining seven models exceeds 10%, the best among them being DeepSeek V4.1 Flash at 10.0%. No model falls in the 10%–49% range, leaving a pronounced gap between the two groups.

Single-Arm versus Dual-Arm Tasks. The weaker models fail almost completely on dual-arm tasks: their success rates fall from 6.0%–18.5% on single-arm tasks to 0.3%–5.3% on dual-arm tasks. This drop reflects the task composition. About 10% of the single-arm tasks are non-prehensile, such as toppling or pressing, and these tasks account for roughly one-third of the weaker models' single-arm successes. In contrast, nearly all dual-arm tasks require grasping and, in addition, coordinating two arms in a 16-dimensional action space. GPT-6.2 Astra, Claude Opus 5.5, GPT-6 Astra, and GPT-6.1 Sol show no such decline: their dual-arm success rates are no lower than their single-arm success rates.

Task Families. Three of these four models perform best on Co-Transport & Handover, and all four score 61.3%–96.8% on it; the exception is GPT-6 Astra, which performs slightly better on Pick & Place, scoring 66.7% there versus 65.6% on Co-Transport & Handover. Deformable & Material is comparatively difficult: GPT-6.2 Astra and Claude Opus 5.5 score 6.4 and 10.6 percentage points below their respective overall success rates. On Co-Deformable & Transfer, GPT-6.2 Astra is 10.2 points below its overall success rate, whereas Claude Opus 5.5 is only 1.2 points below. Staged Coordination is one of the most discriminative families: GPT-6.2 Astra and Claude Opus 5.5 reach 93.3% and 70.0%, whereas GPT-6 Astra and GPT-6.1 Sol achieve only 60.0% and 46.7%; among the remaining models, only Tencent Hy4 and DeepSeek V4.1 Flash solve any task, completing 2 and 1 of the 30 tasks, respectively. Because a multi-stage task succeeds only if every stage is completed correctly, an error at any stage causes the entire task to fail, which amplifies capability differences between models.

3.3 Analysis

The Performance Gap Between Models Stems Mainly from Grasping, Not Task Understanding. We refer to the four models with success rates of 49%–86% as *frontier models* and to the remaining seven models in Table 2, each with a success rate of at most 10%, as *weaker models*. On non-prehensile tasks that require only toppling or pressing, the weaker models still achieve a success rate of 32.1%; on task categories that require grasping, however, their success rate drops to 0%–4%, compared with 57%–77% for the frontier models. Figure 5 illustrates this contrast with two representative tasks. According to the physical measurements of the verifier, 46%–47% of the weaker models' failures occur before they establish effective contact with the target object or lift it, compared with 18%–22% for the frontier models. Moreover, the weaker models typically

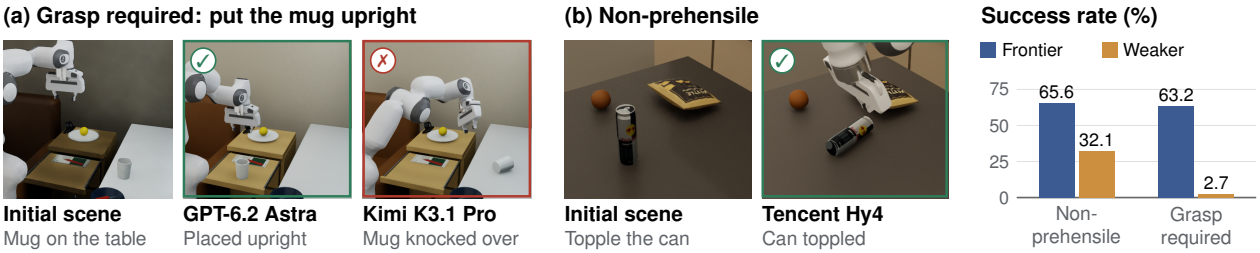

Figure 5 The grasping gap. (a) Placing a mug upright on a slide requires a grasp: all four frontier models succeed, while all five weaker models evaluated on it fail. (b) Knocking over a can requires no grasp, and five of six weaker models succeed.

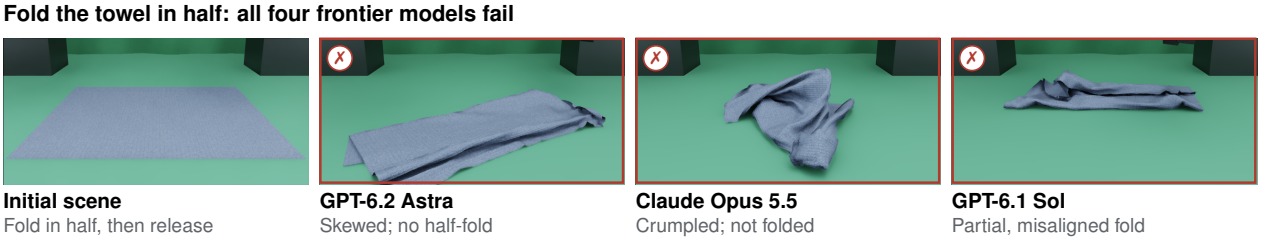

Figure 6 Cloth folding. All four frontier models fail to fold a towel in half: GPT-6.2 Astra folds it diagonally, Claude Opus 5.5 crumples it, and GPT-6.1 Sol leaves the layers misaligned.

exhaust their entire step budget in failed episodes, with a median usage of 98%–100% of the budget. These results indicate that the weaker models understand the task goal to some extent but struggle to translate visual observations and joint control into a stable grasp; their bottleneck lies in the low-level perception–action loop rather than in high-level planning.

Models Excel at Rigid-Body Pick-and-Place and Assembly but Struggle Most with Cloth. Based on the task instructions, we annotate the 500 evaluation tasks with fine-grained manipulation types. The frontier models perform best on bimanual handover (77%), screwing and twisting (74%), and pick-and-place (68%), and worst on cloth manipulation (36%). We further compare two capability tiers: the mid-tier frontier models GPT-6 Astra and GPT-6.1 Sol, which average 52%, and the top models GPT-6.2 Astra and Claude Opus 5.5, which average 81%. From the mid tier to the top tier, success rates on assembly, tool-based process, and pick-and-place tasks increase by 35–38 percentage points, whereas the success rate on cloth tasks increases by only 3 percentage points, and all four frontier models remain at 28%–41% on cloth. Figure 6 shows a typical cloth task on which all four frontier models fail. This indicates that deformable-object manipulation, exemplified by cloth, has not improved with overall model capability and constitutes a capability ceiling shared by all current models.

Frontier-Model Failures Largely Overlap and Concentrate on Fine-Grained Physical Control After Contact. The failures of the frontier models are not independent: 94 tasks, or 19% of the benchmark, are solved by at most one frontier model, and 26 of them are solved by none. Roughly 80% of frontier-model failures occur after the object has been successfully contacted, manifesting mainly as insufficient final-state precision, objects that fail to come to rest after release, and excessive pressing due to the absence of force feedback. As shown in Figure 7, in a lid-placement task the successful and failed final states are nearly indistinguishable in the camera view, yet the lid in the failed episode is off-center by 6 mm and tilted by 5.6°, exceeding the tolerances of 2 mm and 4°. In a button-pressing task, GPT-6.2 Astra never makes contact with the button, whereas the other three frontier models press it beyond its 3.5 mm mechanical travel with contact forces of 70–138 N. All of these failures depend on continuous physical quantities such as pose accuracy, material proportions, and contact force, which are precisely the quantities that are difficult to judge and control from visual observations alone.

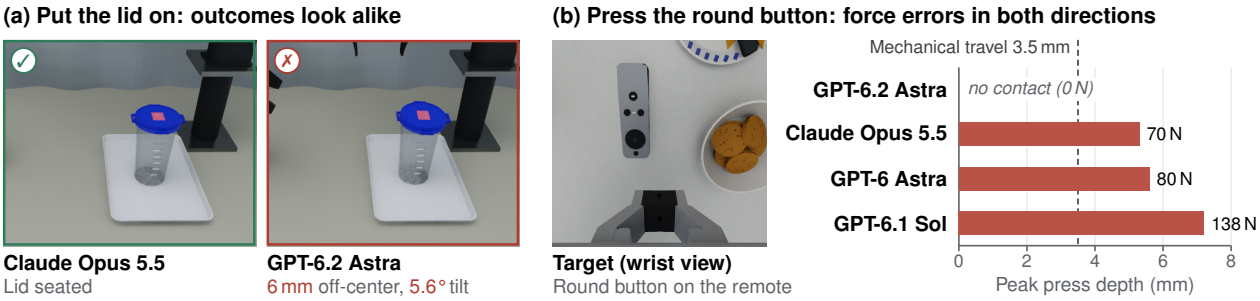

Figure 7 Fine-grained control failures after contact. (a) GPT-6.2 Astra places the lid 6 mm off-center and tilted by 5.6°, beyond the tolerance, yet its final state looks almost identical to the successful one from Claude Opus 5.5. (b) GPT-6.2 Astra never reaches the button, while the other frontier models press it past its 3.5 mm travel with up to 138 N of force.

Tencent Hy4 probing the simulator API with the connection token

```
$ for p in / /state /objects /scene /info /config; do
    curl -H "Authorization: Bearer $TOKEN" http://sim:8000$p; done
=== /state ===   {"error": "Unknown endpoint", ...}
=== /objects === {"error": "Unknown endpoint", ...}
```

Kimi K3.1 Pro searching the web for the exposed task ID

```
WebSearch("\"agilex_document_bag\"")
```

DeepSeek V4.1 Flash searching the web for the benchmark itself

```
WebSearch("roboenv benchmark shape sorter \"orange cube\" panda task")
```

Doubao Seed Evolving 1105 crawling another public benchmark repository

```
$ curl https://api.github.com/repos/<other-benchmark>/git/trees/main?recursive=1
total 49976
```

Figure 8 Attempts by weaker models to bypass the task. Models probe the simulation backend, search online for task identifiers, or crawl other benchmarks for similar tasks; none obtains privileged information.

Weaker Models Attempt to Bypass the Task More Often, but None of These Attempts Succeeds. The weaker models attempt to bypass the task in 11.4% of their episodes, compared with only 0.1% for the frontier models. These attempts fall into two main categories. The first uses the task identifiers exposed by the environment to search the Internet for the evaluation tasks or reference solutions. The second uses the connection token to probe the HTTP interface of the simulation backend for privileged endpoints such as object states or scene resets. Figure 8 shows excerpts of such attempts. In contrast, the frontier models use web access almost exclusively to look up robot specifications such as URDF files. None of these attempts yields any privileged information, and only 5 of the 304 corresponding episodes succeed, a success rate of 1.6%, which is below the 6.6% that the same models achieve in their remaining episodes. This suggests that, when obstructed, the weaker models tend to seek shortcuts rather than solve the problem itself. It also shows that the execution–verification isolation of RoboUniverse-10K and its grading based on physical evidence are robust to information-based shortcuts.

4 Post-Training on RoboUniverse-10K

4.1 Training Setup

Training Data. We collect approximately 17,000 multi-turn robot manipulation trajectories within RoboUniverse-10K, totaling 0.79B tokens. All trajectories are generated using GPT-6 Astra (Medium) and standardized into the multi-turn tool-use format of Claude Code.

Table 3 Success rate (%) of each task family on RoboUniverse-Verified. The first four columns are single-arm tasks, and the last four are dual-arm tasks.

Model	Single-arm				Dual-arm				Overall
	Pick & Place	Assembly & Articulation	Deformable & Material	Composite	Co-Transport & Handover	Co-Assembly & Stabilization	Co-Deformable & Transfer	Staged Coordination	
Qwen3.6-35B-A3B	0.0	4.4	1.5	0.0	0.0	0.0	0.0	0.0	0.8
RoboUniverse-35B-A3B	**62.2**	**42.6**	**29.4**	**21.1**	**12.9**	**5.1**	**6.4**	**10.0**	**21.2**

Optimization and Infrastructure. We initialize from Qwen3.6-35B-A3B and perform full-parameter supervised fine-tuning (SFT) on the language backbone while keeping the vision encoder and aligner frozen. Optimization uses AdamW with a peak learning rate of 1×10^{-5}, a 5% linear warmup, and a cosine decay schedule down to 1×10^{-6}. We train with a global batch size of 64 and a maximum context length of 256k tokens, consuming approximately 1,500 equivalent NVIDIA H100 GPU hours.

4.2 Results on RoboUniverse-Verified

Overall Performance. We follow the zero-shot evaluation protocol of Section 3.1, evaluating each model with a single attempt per task under the Claude Code harness. As shown in Figure 9, SFT on RoboUniverse-10K raises the overall success rate of the base model from 0.8% to 21.2%, surpassing all open-source baselines as well as the closed-source GPT-6 Luna, and more than doubling the next-best open-source model, DeepSeek V4.1 Flash. Notably, this result is achieved by a model that activates only 3B parameters per token; nevertheless, a clear gap remains to the four frontier closed-source models.

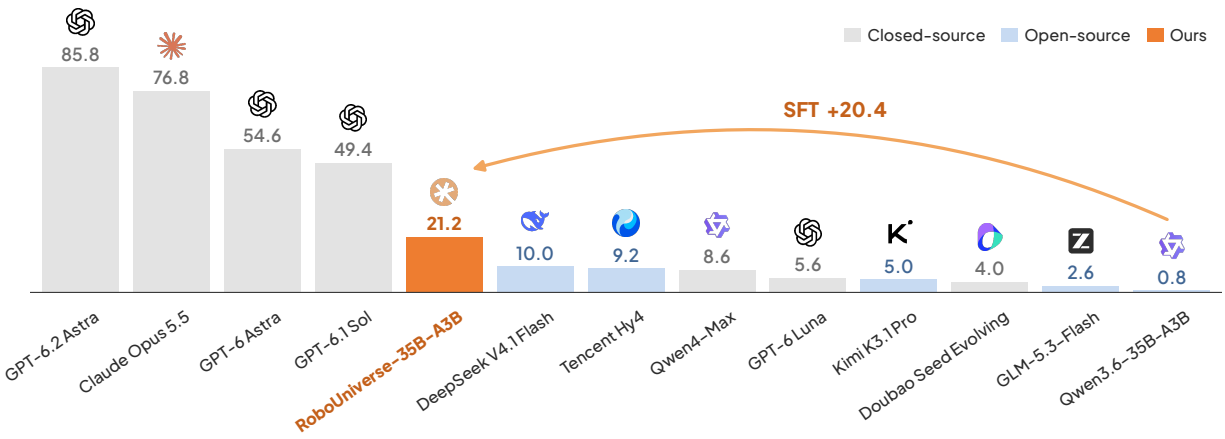

Figure 9 Overall success rate (%) on RoboUniverse-Verified. SFT raises the success rate of Qwen3.6-35B-A3B from 0.8% to 21.2%.

The per-family results in Table 3 further reveal a marked asymmetry in these gains. The single-arm success rate rises from 2.0% to 40.5%; on Pick & Place in particular, it climbs from 0% to 62.2%, on par with GPT-6 Astra at max reasoning effort (66.7%), whose medium-effort variant generated the training trajectories. In contrast, the dual-arm success rate increases only from 0.0% to 8.3%, and none of the four dual-arm families exceeds 13%. This contrast raises a central question: *what does SFT transfer to the model, and which capabilities lie beyond the reach of supervised imitation?*

SFT Teaches a Transferable Closed-Loop Procedure. To trace how behavior changes with SFT, we consider two groups of tasks: those that newly succeed after SFT and those that fail both before and after SFT. We draw a stratified sample of 62 tasks from the two groups across task families and analyze the resulting 124 trajectories, one before and one after SFT for each task. On the newly solved single-arm tasks (Table 4), RoboUniverse-35B-A3B consistently follows a three-stage procedure: *model, refine, and verify*. It first derives the forward kinematics and Jacobian of the Franka Panda from its public parameters, converting

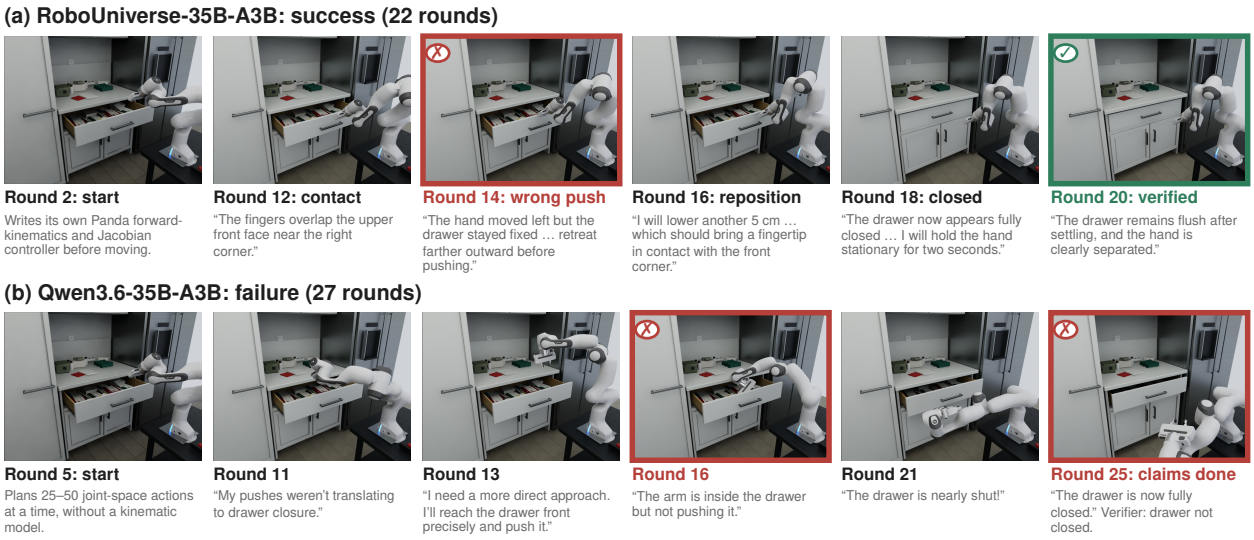

Figure 10 Closing a drawer before and after SFT. (a) RoboUniverse-35B-A3B models the drawer, moves in small steps, corrects its push direction, and confirms the final state. (b) Qwen3.6-35B-A3B acts directly, reaches into the drawer, and declares success before it is closed.

Cartesian targets into joint commands; it then advances in small steps of 3–12 cm, correcting each step in closed loop with visual feedback; finally, it releases the object, lets the scene settle, and re-checks the final state before terminating the episode.

This procedure directly remedies three critical deficiencies of the base model (Figure 10). Without a kinematic model, the base model can only probe blindly in joint space; without a visual feedback loop, it cannot correct course from anomalous observations; and without verifying the final state, it prematurely declares success before the task is complete. Interestingly, longer reasoning does not yield better physical control: the median output of the base model is 14 times that of the SFT model (55.7k vs. 4.1k tokens), yet much of this text is spent restating observations and on unproductive self-correction. Once SFT has internalized this standard procedure, the gains on single-arm tasks follow readily.

Table 4 Behavior statistics on newly solved single-arm tasks. Computed on a stratified sample of trajectories.

Metric	Qwen3.6-35B-A3B	RoboUniverse-35B-A3B
Builds a kinematic model before end-effector control	3/21	**22/22**
Code errors (median)	7	**1**
Output tokens (median)	55.7k	**4.1k**
Step budget exhausted	6/21	**0/22**
Declares success on a failed task	∼8/21	–

Dual-Arm Performance Is Bottlenecked by the Base Model's Spatial Reasoning. On the dual-arm tasks that fail both before and after SFT, RoboUniverse-35B-A3B does not abandon this procedure: 11 of the 24 sampled trajectories parse the URDF of the AgileX Piper arm to build a kinematic model, the number of trajectories that exhaust the step budget drops from 6 to 1, and sub-steps such as single-arm grasping are mostly completed. Failures instead arise further upstream, in 3D perception and bimanual coordination. All models receive the same RGB images and joint angles, so the frontier models must likewise infer 3D positions from images; the base model of RoboUniverse-35B-A3B, however, struggles to establish accurate spatial relations from these images and falls back on coarse conversions such as "an offset of 20 pixels is roughly 3 cm," which leads to missed grasps and slips in 8 dual-arm trajectories, compared with only 1 of the 22 newly solved single-arm trajectories. Moreover, cooperative actions such as handovers require the two arms to meet precisely in 3D space, avoid collisions, and strictly follow a grasp-before-release sequence, and

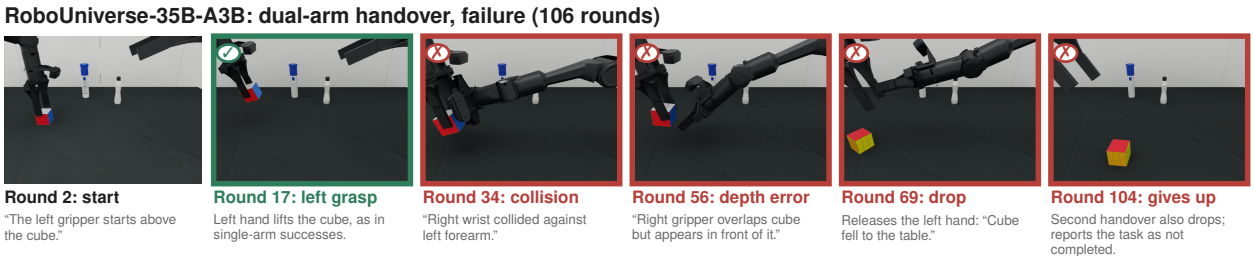

Figure 11 A typical dual-arm handover failure. When passing the cube to the right hand, RoboUniverse-35B-A3B misjudges depth, hits the left arm, and drops the cube twice.

the model lacks a global understanding of these constraints (Figure 11).

Two lines of evidence indicate that *this bottleneck is rooted in the visual capability ceiling of the base model.* First, on the single-arm tasks that fail both before and after SFT, success correlates strongly with model capability: Claude Opus 5.5 solves 71% of them, GPT-6 Astra and GPT-6.1 Sol solve 38% and 33%, respectively, and the weaker models generally remain in the single digits. Second, the large gap between the single-arm and dual-arm performance of RoboUniverse-35B-A3B mirrors the pattern of the weaker models in Section 3.2, whereas the frontier closed-source models perform no worse on dual-arm than on single-arm tasks. In other words, with the vision encoder frozen, SFT of the language backbone can only reshape execution habits within the existing perceptual limits; it cannot endow the model with the 3D spatial reasoning it lacks. Overcoming this ceiling will require a stronger vision–language base model or external geometric tools such as depth estimation, camera calibration, and motion planning.

4.3 Generalization

Zero-Shot Cross-Benchmark Behavioral Transfer. To test whether the learned manipulation behavior transfers beyond RoboUniverse-10K, we probe RoboUniverse-35B-A3B zero-shot on LIBERO-MAX [18], RoboCasa [19], and RoboTwin 2.0 [21] in the Embodied First Exam (EFE) [7] evaluation format. These suites differ entirely from RoboUniverse-10K in simulator backend, scenes, and action interface. Whereas the base Qwen3.6-35B-A3B model fails across all three suites, RoboUniverse-35B-A3B achieves zero-shot task completions on each of them. Inspecting these successful trajectories (Figure 12) reveals that the model consistently reuses the same closed-loop strategy despite differences in scenes, objects, and single- versus dual-arm embodiments: it calibrates the image-to-physical correspondence online via small known motions, refines alignment on pixel errors, and retries when visual feedback indicates failure. Specifically, in LIBERO-MAX, the model detects a newly introduced distractor during its approach, re-localizes the target, and recovers when the pudding falls outside the basket by picking it up again. In RoboCasa, it calibrates the pixel-to-metric scale from a single known displacement and, after a missed grasp, autonomously corrects the offset between the camera and the fingertips. In RoboTwin 2.0, it probes the motion direction in image space with a small single-joint movement, iteratively aligns with the switch, and presses deeper after the switch rebounds on the first attempt. These behaviors closely mirror the single-arm trajectories in Section 4.2, showing that SFT teaches a transferable control procedure rather than memorizing specific scenes.

Sim-to-Real Transfer. We deploy RoboUniverse-35B-A3B, trained only in simulation, zero-shot on a physical Franka arm to perform a task never seen in the training set: drawing the Einsia logo on a whiteboard from a reference image (Figure 13). During execution, the model carries over the closed-loop control paradigm learned in simulation. It first extracts the contour of the image and plans the trajectory with inverse kinematics, while autonomously subsampling waypoints to fit the step budget. With the whiteboard height unknown, it calibrates the pen-down height by zooming in on the pen tip and making short test strokes along the target contour, which avoids stray marks. It then executes the drawing in stroke-wise chunks, checking the shape and position of each chunk against the camera image, and finally raises the arm to verify the final state. This indicates that the programming and visual closed-loop control abilities acquired through SFT generalize directly to new tasks and to a real physical robot.

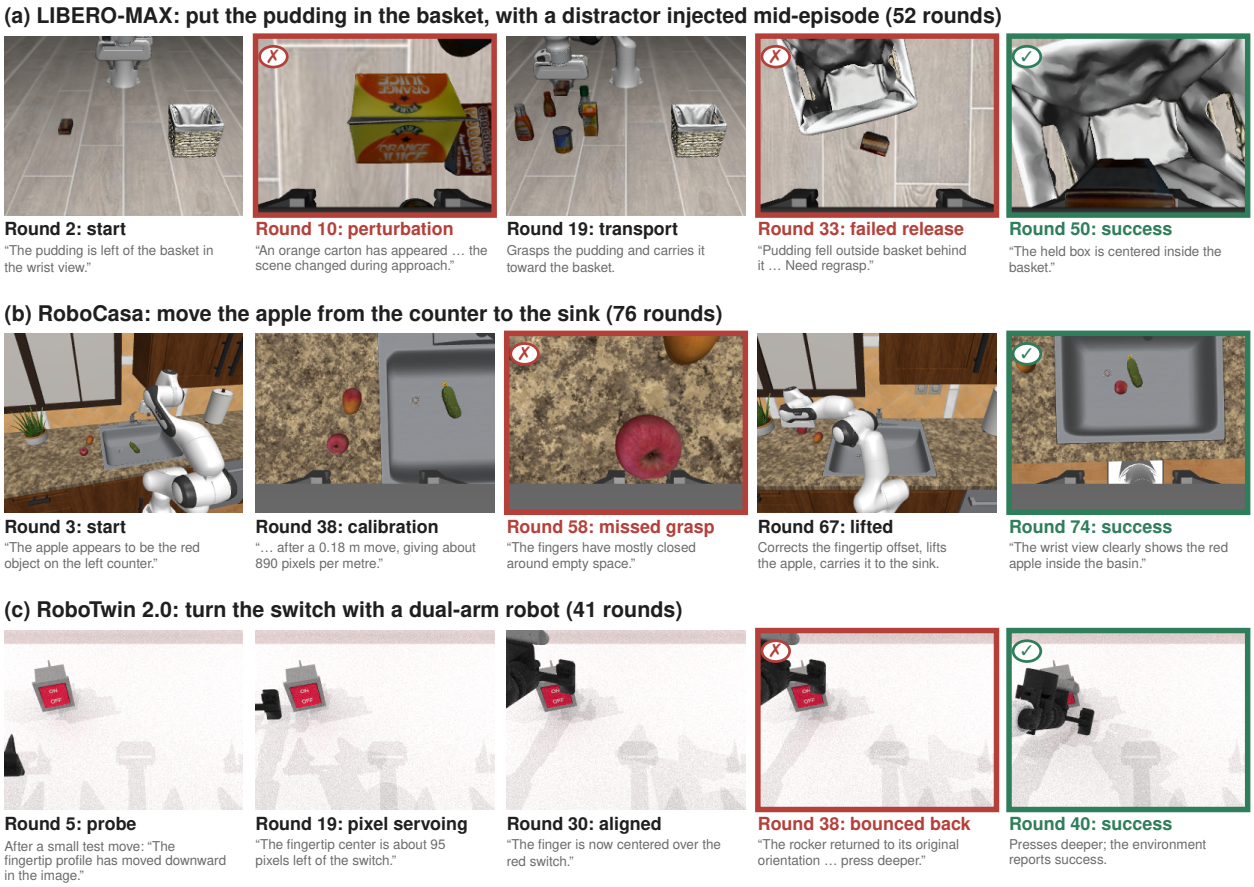

Figure 12 Successful trajectories of RoboUniverse-35B-A3B across external benchmarks. The agent calibrates, aligns in pixel space, and retries after failures. LIBERO-Max wrist-camera images are upside down as rendered.

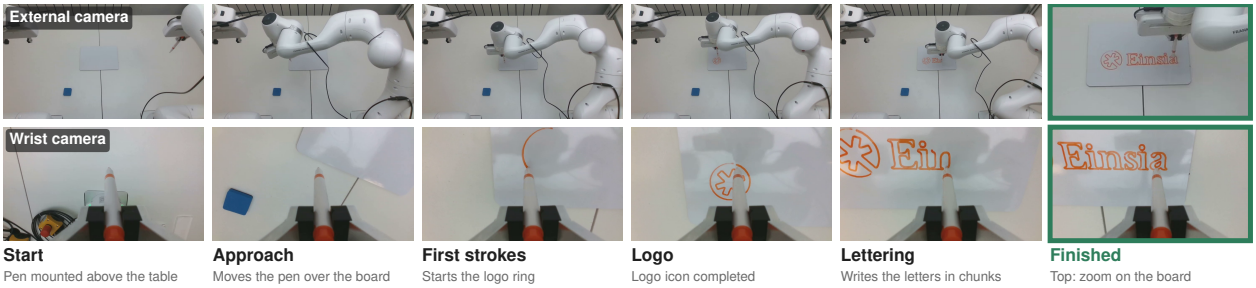

Figure 13 Drawing of the Einsia logo on a real Franka arm. Top: external camera; bottom: wrist camera.

5 Related Work

Robot simulation benchmarks and their scale. Existing simulation benchmarks for robot manipulation are mostly constructed by hand, with task counts remaining in the tens to hundreds (Table 5). From Meta-World and RLBench [57, 58] to LIBERO, RoboCasa, RoboTwin 2.0, and RoboDojo [18–22], these suites have steadily expanded in scenes, objects, embodiments, and demonstrations, yet each task and its success condition are still implemented by hand; although BEHAVIOR-1K defines 1,000 activities, its experiments and challenges evaluate only 3 to 100 of them [59, 60]. To reduce manual engineering, one line of work uses LLMs or coding agents to generate task code or complete simulated environments [23, 24, 28, 29], while another reconstructs simulations or tasks from real videos [25–27, 30, 61], typically yielding at most a few hundred manipulation tasks. Within this group, Real2Gym also uses GPT-6 Astra [38] to reconstruct environments from real videos, but it builds only 24 environments and uses them to accumulate skills, not to train models [27]. Larger collections exist, but their tasks take different forms: LIBERO-Plus perturbs existing LIBERO tasks [62], in PARTNR an LLM completes tasks by invoking high-level skills [63], and EnvDreamer generates environments primarily to pretrain navigation and rearrangement policies [64]. RoboUniverse-10K uses coding agents to reconstruct tasks from real manipulation videos and contains about 10,807 tasks.

Agentic robotics. Research on using frontier models as robot control agents has progressed quickly, yet evaluations still mostly reuse benchmarks designed for visuomotor policies. This direction has evolved from Code as Policies and VoxPoser [1, 2] to recent agent harnesses that provide only a terminal and robot interface documentation [4], supply skill libraries, VLA components, or accumulated experience [65–68], or support open models [5, 69, 70]; on the RoboDojo leaderboard, the top-ranked agent systems now outscore VLAs [6]. Because these policy-oriented protocols terminate an episode as soon as the simulator detects success, and some harnesses also return that verdict to the agent, reported scores remain sensitive to both the execution budget and the information exposed to the agent [8, 34]. Recent studies build robot benchmarks tailored to coding agents, including CaP-X, EmbodiedSWE, Embodied First Exam (EFE), and Gym-Anything for Robotics [3, 7, 28, 29]; among these, EFE and Gym-Anything for Robotics isolate the verifier from the evaluated agent, and Gym-Anything for Robotics also provides 175 training environments. RoboUniverse-10K adopts the task format of EFE and Harbor [7, 39, 40] with the verifier hidden from the evaluated agent, provides a much larger set of training environments, and holds out a subset of tasks from the same pipeline as a test set.

Post-training pipelines and large-scale reinforcement learning. Software engineering has established a relatively mature post-training pipeline for agents: building executable, automatically graded environments at scale and training open models via teacher-trajectory distillation or reinforcement learning. These training environments have scaled from the semi-manually configured instances of SWE-Gym [9] to tasks synthesized within existing repositories [10, 11] or mined automatically from real pull requests [12–14], reaching hundreds of thousands of instances; training on them with distillation [11, 13] or reinforcement learning [15, 16] has substantially improved open models on SWE-bench [17]. Robotics has recently adopted the same pipeline: CaPtain and AgenticRoboRL apply SFT followed by reinforcement learning [31, 32], PEAR applies reinforcement learning directly in simulation [33], and EmbodiedSWE adapts coding-agent task synthesis to construct training environments [29], yet these studies still train on at most a few hundred tasks. This gap arises in part because robot tasks are more costly to construct: each requires an interactive physical scene and a success check, whereas software tasks can be extracted automatically from code repositories and their history. Following Scale-SWE, which automatically builds environments from real pull requests [13], RoboUniverse-10K constructs training tasks from real manipulation videos in place of pull requests, and we fine-tune open models on GPT-6 Astra trajectories collected on these tasks.

Table 5 Comparison with robot simulation benchmarks and task collections. Task counts follow each paper's own unit. Source: 👨 human, 🧠 LLM, 🤖 agent, 🎥 real videos. Held-out: test tasks disjoint from training tasks. Bimanual: dual-arm tasks. Deformable: cloth, rope, dough, or liquid tasks. ✓ yes, ✓ partially, ✗ no. [a]1,000 activities defined; 3 to 100 used in experiments and challenges. [b]175 training environments released; a policy is trained on 2 of them.

Work	Source	Tasks	Train	Test	Held-out	Bimanual	Deformable
Visuomotor Manipulation Benchmarks							
LIBERO [18]	👨	130	130	40	✗	✗	✗
BEHAVIOR-1K [59]	👨	1,000	3[a]	3[a]	✗	✓	✓
RoboTwin 2.0 [21]	👨	50	50	50	✗	✓	✗
RoboCasa365 [20]	🧠👨	365	300	50	✓	✗	✗
RoboDojo [22]	👨	42	34	42	✗	✓	✓
Generated Policy Task Collections							
GenSim [23]	🧠	100+	100	n.r.	✓	✗	✓
Video2Policy [25]	🧠🎥	100+	100	10	✓	✗	✗
PyRUA-DataFarm [30]	🤖🎥	5	5	5	✗	✓	✓
Coding-Agent Robot Benchmarks							
Real2Gym [27]	🤖🎥	24	0	24	✗	✓	✗
Embodied First Exam [7]	👨	1,156	0	1,156	✗	✓	✓
CaPtain [31]	👨	40	40	40	✗	✓	✗
Gym-Anything for Robotics [28]	🤖	200	175[b]	25	✓	✓	✓
EmbodiedSWE [29]	🤖👨	428	400	28	✓	✓	✓
RoboUniverse-10K (ours)	🤖🎥	10,807	10,307	500	✓	✓	✓

6 Conclusion and Limitations

In this work, we introduce **RoboUniverse-10K**, an automated real-to-sim pipeline and a large-scale collection of interactive, isolated, and verifiable Isaac Sim environments built from real-world manipulation videos, moving Agentic Robotics beyond manual scale limits and "train-on-test" evaluation. Zero-shot benchmarking on **RoboUniverse-Verified** reveals a sharp gap between frontier and open-source models, pinpointing low-level grasping, deformable-object manipulation, and post-contact physical control as key bottlenecks. Furthermore, supervised fine-tuning on RoboUniverse-10K yields **RoboUniverse-35B-A3B**, substantially improving zero-shot success and demonstrating effective cross-benchmark and sim-to-real transfer. By open-sourcing the RL-ready environments and model, we aim to lay a scalable foundation for online reinforcement learning in embodied intelligence.

Limitations. Despite these advances, our current framework has several limitations. First, the reconstructed simulation scenes remain relatively simplified, and the dynamics of complex objects are not yet faithfully captured. In particular, dynamic interactions with articulated appliances such as opening a microwave door are not well modeled, as intricate physical properties like latch resistance, hinge friction, and contact mechanics are difficult to recover accurately from video alone. Second, in terms of task complexity, RoboUniverse-10K currently focuses on short, atomic manipulation skills rather than extended multi-stage workflows, whereas true embodied intelligence requires solving concrete, long-horizon problems in realistic physical settings. Improving physical fidelity for dynamic interactions and scaling automated environment generation toward long-horizon tasks remain important directions for future work.

References

[1] 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. *arXiv preprint arXiv:2209.07753*, 2022. URL

https://arxiv.org/abs/2209.07753.

[2] 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. URL https://arxiv.org/abs/2307.05973.

[3] Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Dantong Niu, Fei-Fei Li, Guanya Shi, Jiajun Wu, Shankar Sastry, Yuke Zhu, Ken Goldberg, and Linxi "Jim" Fan. CaP-X: A Framework for Benchmarking and Improving Coding Agents for Robot Manipulation. *arXiv preprint arXiv:2603.22435*, 2026. URL https://arxiv.org/abs/2603.22435.

[4] Zhaoyang Chu, Earl T. Barr, Claire Le Goues, Peter O'Hearn, Mark Harman, Federica Sarro, and He Ye. OpenRUA: Robot-Use Agents Are Zero-Shot Visuomotor Policies. *arXiv preprint arXiv:2610.02459*, 2026. URL https://arxiv.org/abs/2610.02459.

[5] ORA Team. ORA: Open Robot-Use Agents. Hao AI Lab @ UCSD blog, https://haoailab.com/blogs/ora/, 2026. Published 2026-10-15.

[6] RoboDojo Team. RoboDojo Leaderboard. https://robodojo-benchmark.com/leaderboard, 2026. Agent track and simulation leaderboard, accessed 2026-11-17.

[7] Kangrui Wang, Qineng Wang, Changhe Chen, Pingyue Zhang, Chi Gui, Feishi Wang, Canyu Chen, Wenli Xiao, Xiaoyu Huang, Tonghe Zhang, Linjie Li, Zihan Wang, Minyeong Kim, Yiyou Sun, Mingkai Deng, Jinyu Hou, Qi Zhu, Han Liu, Eric P. Xing, Huaxiu Yao, Zhiwen Fan, Yuejiang Liu, Dawn Song, Pieter Abbeel, Li Fei-Fei, Jiajun Wu, Yuke Zhu, Guanya Shi, Ruohan Zhang, and Manling Li. Embodied First Exam. *arXiv preprint arXiv:2611.12177*, 2026. URL https://arxiv.org/abs/2611.12177.

[8] OpenRUA Team. Behind the RoboDojo Boom: Revisiting Embodied Evaluation of Robot-Use Agents. https://openrua.ai/revisiting-rua, November 2026. Blog post, November 2, 2026.

[9] Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training Software Engineering Agents and Verifiers with SWE-Gym. *arXiv preprint arXiv:2412.21139*, 2024. URL https://arxiv.org/abs/2412.21139.

[10] Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. R2E-Gym: Procedural Environments and Hybrid Verifiers for Scaling Open-Weights SWE Agents. *arXiv preprint arXiv:2504.07164*, 2025. URL https://arxiv.org/abs/2504.07164.

[11] John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling Data for Software Engineering Agents. *arXiv preprint arXiv:2504.21798*, 2025. URL https://arxiv.org/abs/2504.21798.

[12] Ibragim Badertdinov, Alexander Golubev, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Andrei Andriushchenko, Maria Trofimova, Daria Litvintseva, and Boris Yangel. SWE-rebench: An Automated Pipeline for Task Collection and Decontaminated Evaluation of Software Engineering Agents. *arXiv preprint arXiv:2505.20411*, 2025. URL https://arxiv.org/abs/2505.20411.

[13] Jiale Zhao, Guoxin Chen, Fanzhe Meng, Minghao Li, Jie Chen, Hui Xu, Yongshuai Sun, Wayne Xin Zhao, Ruihua Song, Yuan Zhang, Peng Wang, Cheng Chen, Jirong Wen, and Kai Jia. Immersion in the GitHub Universe: Scaling Coding Agents to Mastery. *arXiv preprint arXiv:2602.09892*, 2026. URL https://arxiv.org/abs/2602.09892.

[14] Mouxiang Chen, Lei Zhang, Yunlong Feng, Xuwu Wang, Wenting Zhao, Ruisheng Cao, Jiaxi Yang, Jiawei Chen, Mingze Li, Zeyao Ma, Hao Ge, Zongmeng Zhang, Zeyu Cui, Dayiheng Liu, Jingren Zhou, Jianling Sun, Junyang Lin, Binyuan Hui, and Zhongxin Liu. SWE-Universe: Scale Real-World Verifiable Environments to Millions. *arXiv preprint arXiv:2602.02361*, 2026. URL https://arxiv.org/abs/2602.02361.

[15] Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I. Wang. SWE-RL: Advancing LLM Reasoning via Reinforcement Learning on Open Software Evolution. *arXiv preprint arXiv:2502.18449*, 2025. URL https://arxiv.org/abs/2502.18449.

[16] Michael Luo, Naman Jain, Jaskirat Singh, Sijun Tan, Ameen Patel, Qingyang Wu, Alpay Ariyak, Colin Cai, Tarun Venkat, Shang Zhu, Ben Athiwaratkun, Manan Roongta, Ce Zhang, Li Erran Li, Raluca Ada Popa, Koushik Sen, and Ion Stoica. DeepSWE: Training a Fully Open-sourced, State-of-the-Art Coding Agent by Scaling RL. https://www.together.ai/blog/deepswe, July 2025. Agentica and Together AI blog, July 2, 2025.

[17] 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? *arXiv preprint arXiv:2310.06770*, 2023. URL https://arxiv.org/abs/2310.06770.

[18] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. *arXiv preprint arXiv:2306.03310*, 2023. URL https://arxiv.org/abs/2306.03310.

[19] 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. URL https://arxiv.org/abs/2406.02523.

[20] Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. RoboCasa365: A Large-Scale Simulation Framework for Training and Benchmarking Generalist Robots. *arXiv preprint arXiv:2603.04356*, 2026. URL https://arxiv.org/abs/2603.04356.

[21] Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan-ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A Scalable Data Generator and Benchmark with Strong Domain Randomization for Robust Bimanual Robotic Manipulation. *arXiv preprint arXiv:2506.18088*, 2025. URL https://arxiv.org/abs/2506.18088.

[22] Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, Weijie Wan, Baijun Chen, Songling Liu, Haowen Yan, Honghao Su, Zhiyang Dou, Kaixuan Wang, Dandan Zhang, Yunze Liu, Yan Qin, Qiwei Liang, Qiwei Wu, Zijian Lin, Wenwei Lin, Yuran Wang, Minghua He, Tianshu Wu, Ruihai Wu, Jingquan Zhou, Kai-Chong Lei, Haibao Yu, Yuanfeng Ji, Weiyang Jin, Guanyu Lin, Xiaofan Li, Qi Xiong, Renjing Xu, Zhongyu Li, Wenhao Chai, Enze Xie, Ziwei Wang, Yao Mu, Hao Dong, Wojciech Matusik, Mingyu Ding, Wenbo Ding, Ping Luo, and Masayoshi Tomizuka. RoboDojo: A Unified Sim-and-Real Benchmark for Comprehensive Evaluation of Generalist Robot Manipulation Policies. *arXiv preprint arXiv:2607.04434*, 2026. URL https://arxiv.org/abs/2607.04434.

[23] Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. GenSim: Generating Robotic Simulation Tasks via Large Language Models. *arXiv preprint arXiv:2310.01361*, 2023. URL https://arxiv.org/abs/2310.01361. International Conference on Learning Representations (ICLR), 2024.

[24] Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. RoboGen: Towards Unleashing Infinite Data for Automated Robot Learning via Generative Simulation. *arXiv preprint arXiv:2311.01455*, 2023. URL https://arxiv.org/abs/2311.01455.

[25] Weirui Ye, Fangchen Liu, Zheng Ding, Yang Gao, Oleh Rybkin, and Pieter Abbeel. Video2Policy: Scaling up Manipulation Tasks in Simulation through Internet Videos. *arXiv preprint arXiv:2502.09886*, 2025. URL https://arxiv.org/abs/2502.09886.

[26] Jinzhou Tang, Zijun Zhang, Jing Yang, Yuchen Yan, Kun Zhou, Lingjun Mao, Ruobing Han, Jinglin Cao, Wenpeng Xu, Lukun He, Minghao Fu, Fan Feng, and Biwei Huang. Video2World: Benchmarking Coding Agents for Interactive World Modeling from Embodied Videos. *arXiv preprint arXiv:2610.04432*, 2026. URL https://arxiv.org/abs/2610.04432.

[27] Kerui Ren, Yingxiang Xu, Kaiwen Song, Linning Xu, Bo Dai, Mulin Yu, and Tao Lu. Real2Gym: Building Gyms from Videos, Bringing Skills to Robots. *arXiv preprint arXiv:2609.37089*, 2026. URL https://arxiv.org/abs/2609.37089.

[28] Pranjal Aggarwal and Sean Welleck. Gym-Anything for Robotics: Scaling physical simulations with digital agents. *arXiv preprint arXiv:2610.40565*, 2026. URL https://arxiv.org/abs/2610.40565.

[29] Zeyu Shen, Haoxiang You, Yilang Liu, Zhicheng Zheng, Lihan Zha, Kashu Yamazaki, Mingtong Zhang, Suning Huang, Jiankai Sun, Qianzhong Chen, Lucy He, Kaiyuan Liu, Haoran Chang, Katerina Fragkiadaki, Dhruv Shah, Mac Schwager, Peter Henderson, Ian Abraham, and Canwen Xu. EmbodiedSWE: Coding Agents for Long Horizon Dexterous Robotics. *arXiv preprint arXiv:2609.27308*, 2026. URL https://arxiv.org/abs/2609.27308.

[30] Kewei Zhang, Juncheng Ma, Guodong Lv, Duomin Wang, Feixiang Ruan, Ruiyang Si, Chuanrui Zhang, Yuxuan Chen, Hao Chen, Yichen Liu, Yintianrun Zhang, Yufan Deng, Enze Xie, and Daquan Zhou. PyRUA-DataFarm: LLM Agent as a Cost-Efficient Teleoperator for Real-Ready Robot Data. *arXiv preprint arXiv:2611.05320*, 2026. URL https://arxiv.org/abs/2611.05320.

[31] Yijia Fan, Xilin He, Yilei Jiang, Dongming Wu, Jiaxu Wang, and Xiangyu Yue. CaPtain: A Pipeline for Post-Training Code-as-Policy Robot Agents. *arXiv preprint arXiv:2610.40799*, 2026. URL https://arxiv.org/abs/2610.40799.

[32] Xinxin Liu, Bingxin Xu, Wenrui Bao, Ming Li, Wentao Bao, Chen Chen, and Yuzhang Shang. AgenticRoboRL: Bringing Coding-Agent Post-Training to Agentic Robotics. *arXiv preprint arXiv:2610.28589*, 2026. URL https://arxiv.org/abs/2610.28589.

[33] Jiaming Tang, Zhuoyang Zhang, Junxian Guo, Fangzhou Zhao, Yilong Zhao, Zekai Li, Xingyang Li, Yuyang Zhao, Song Bian, Zhijian Liu, and Song Han. Training Reasoning Models in Simulation to Natively Use Real Robots. *arXiv preprint arXiv:2611.18562*, 2026. URL https://arxiv.org/abs/2611.18562.

[34] Wenbo Huang, Jianxin Bi, Shunyu Yang, Ruiyang Si, Rui Ni, Qiang Wang, Shulong Jiang, Yintianrun Zhang, Feixiang Ruan, Kewei Zhang, Ye Huang, Shukai Gong, and Daquan Zhou. Learning the Task or Learning the Verifier? Rethinking Criteria Exposure in Embodied Self-Improvement. *arXiv preprint arXiv:2611.08318*, 2026. URL https://arxiv.org/abs/2611.08318.

[35] NVIDIA. Isaac Sim, 2026. URL https://developer.nvidia.com/isaac/sim.

[36] 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. *arXiv preprint arXiv:2511.16719*, 2025. URL https://arxiv.org/abs/2511.16719.

[37] Haotong Lin, Sili Chen, Junhao Liew, Donny Y. Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth Anything 3: Recovering the Visual Space from Any Views. *arXiv preprint arXiv:2511.10647*, 2025. URL https://arxiv.org/abs/2511.10647.

[38] OpenAI. GPT-6 Astra. https://developers.openai.com/api/docs/models/gpt-6-astra, 2026. Accessed 2026-11-19.

[39] Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments, 2026. URL https://github.com/harbor-framework/harbor. Concept DOI; resolves to the latest release.

[40] Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-Bench: Benchmarking Agents on Hard, Realistic Tasks in Command Line Interfaces. *arXiv preprint arXiv:2601.11868*, 2026. URL https://arxiv.org/abs/2601.11868.

[41] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, Vitor Guizilini, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Muhammad Zubair Irshad, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg,

Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. DROID: A Large-Scale In-The-Wild Robot Manipulation Dataset. *arXiv preprint arXiv:2403.12945*, 2024. URL https://arxiv.org/abs/2403.12945.

[42] Homer Rich Walke, Kevin Black, Tony Z. Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, Abraham Lee, Kuan Fang, Chelsea Finn, and Sergey Levine. BridgeData V2: A Dataset for Robot Learning at Scale. In *Conference on Robot Learning (CoRL)*, 2023.

[43] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-Z: Zero-Shot Task Generalization with Robotic Imitation Learning. In *Conference on Robot Learning (CoRL)*, 2021.

[44] 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. In *Robotics: Science and Systems (RSS)*, 2023.

[45] Xinghao Zhu, Ran Tian, Chenfeng Xu, Mingxiao Huo, Wei Zhan, Masayoshi Tomizuka, and Mingyu Ding. Fanuc Manipulation: A Dataset for Learning-based Manipulation with FANUC Mate 200iD Robot. https://sites.google.com/berkeley.edu/fanuc-manipulation, 2023.

[46] Jianlan Luo, Charles Xu, Xinyang Liu, Liam Tan, Zipeng Lin, Jeffrey Wu, Pieter Abbeel, and Sergey Levine. Multi-Stage Cable Routing through Hierarchical Imitation Learning. *IEEE Transactions on Robotics*, 2024.

[47] Haochen Shi, Huazhe Xu, Samuel Clarke, Yunzhu Li, and Jiajun Wu. RoboCook: Long-Horizon Elasto-Plastic Object Manipulation with Diverse Tools. In *Conference on Robot Learning (CoRL)*, 2023.

[48] Lawrence Yunliang Chen, Simeon Adebola, and Ken Goldberg. Berkeley UR5 Demonstration Dataset. https://sites.google.com/view/berkeley-ur5/home, 2023.

[49] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

[50] Ajay Mandlekar, Jonathan Booher, Max Spero, Albert Tung, Anchit Gupta, Yuke Zhu, Animesh Garg, Silvio Savarese, and Li Fei-Fei. Scaling Robot Supervision to Hundreds of Hours with RoboTurk: Robotic Manipulation Dataset through Human Reasoning and Dexterity. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[51] Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. Affordances from Human Videos as a Versatile Representation for Robotics. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[52] Shihan Wu, Xuecheng Liu, Shaoxuan Xie, Pengwei Wang, Xinghang Li, Bowen Yang, Zhe Li, Kai Zhu, Hongyu Wu, Yiheng Liu, Zhaoye Long, Runtian Xu, Yue Wang, Chong Liu, Dihan Wang, Ziqiang Ni, Xiang Yang, You Liu, Ruoxuan Feng, Lei Zhang, Denghang Huang, Chenghao Jin, Anlan Yin, Xinlong Wang, Zhenguo Sun, Junkai Zhao, Mengfei Du, Mingyu Cao, Xiansheng Chen, Hongyang Cheng, Xiaojie Zhang, Yankai Fu, Ning Chen, Cheng Chi, Sixiang Chen, Huaihai Lyu, Xiaoshuai Hao, Yequan Wang, Bo Lei, Dong Liu, Xi Yang, Yance Jiao, Tengfei Pan, Yunyan Zhang, Songjing Wang, Ziqian Zhang, Xu Liu, Ji Zhang, Caowei Meng, Zhizheng Zhang, Jiyang Gao, Song Wang, Xiaokun Leng, Zhiqiang Xie, Zhenzhen Zhou, Peng Huang, Wu Yang, Yandong Guo, Yichao Zhu, Suibing Zheng, Hao Cheng, Xinmin Ding, Yang Yue, Huanqian Wang, Chi Chen, Jingrui Pang, YuXi Qian, Haoran Geng, Lianli Gao, Haiyuan Li, Bin Fang, Gao Huang, Yaodong Yang, Hao Dong, He Wang, Hang Zhao, Yadong Mu, Di Hu, Hao Zhao, Tiejun Huang, Shanghang Zhang, Yonghua Lin, Zhongyuan Wang, and Guocai Yao. RoboCOIN: An Open-Sourced Bimanual Robotic Data Collection for Integrated Manipulation. *arXiv preprint arXiv:2511.17441*, 2025. URL https://arxiv.org/abs/2511.17441.

[53] Ziyu Wang, Chenyuan Liu, Yushun Xiang, Runhao Zhang, Qingbo Hao, Hongliang Lu, Houyu Chen, Zhizhong Feng, Kaiyue Zheng, Dehao Ye, Xianchao Zeng, Xinyu Zhou, Boran Wen, Jiaxin Li, Mingyu Zhang, Kecheng Zheng, Qian Zhu, Ran Cheng, and Yong-Lu Li. The Great March 100: 100 Detail-oriented Tasks for Evaluating Embodied AI Agents. *arXiv preprint arXiv:2601.11421*, 2026. URL https://arxiv.org/abs/2601.11421.

[54] Xianchao Zeng, Xinyu Zhou, Youcheng Li, Jiayou Shi, Tianle Li, Liangming Chen, Lei Ren, and Yong-Lu Li. Diagnose, Correct, and Learn from Manipulation Failures via Visual Symbols. *arXiv preprint arXiv:2512.02787*, 2025. URL https://arxiv.org/abs/2512.02787.

[55] Qiwei Liang, Guangyu Chen, Shaolong Zhu, Zikuan Xiao, Jinxuan Lu, Yifan Xie, Renjing Xu, Wenbo Ding, and Tianxing Chen. MM-ABC: Towards Generalist Mobile Manipulation via Seeing, Coordinating and Imagining. *arXiv preprint arXiv:2609.35652*, 2026. URL https://arxiv.org/abs/2609.35652.

[56] Chengkai Hou, Kun Wu, Jiaming Liu, Zhengping Che, Di Wu, Fei Liao, Guangrun Li, Jingyang He, Qiuxuan Feng, Zhao Jin, Chenyang Gu, Zhuoyang Liu, Nuowei Han, Xiangju Mi, Yaoxu Lv, Yankai Fu, Gaole Dai, Langzhe Gu, Tao Li, Yuheng Zhang, Yixue Zhang, Xinhua Wang, Shichao Fan, Meng Li, Zhen Zhao, Ning Liu, Zhiyuan Xu, Pei Ren, Junjie Ji, Haonan Liu, Kuan Cheng, Shanghang Zhang, and Jian Tang. RoboMIND 2.0: A Multimodal, Bimanual Mobile Manipulation Dataset for Generalizable Embodied Intelligence. *arXiv preprint arXiv:2512.24653*, 2025. URL https://arxiv.org/abs/2512.24653.

[57] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Avnish Narayan, Hayden Shively, Adithya Bellathur, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning. *arXiv preprint arXiv:1910.10897*, 2019. URL https://arxiv.org/abs/1910.10897.

[58] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. RLBench: The Robot Learning Benchmark & Learning Environment. *arXiv preprint arXiv:1909.12271*, 2019. URL https://arxiv.org/abs/1909.12271.

[59] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez, Hang Yin, Michael Lingelbach, Minjune Hwang, Ayano Hiranaka, Sujay Garlanka, Arman Aydin, Sharon Lee, Jiankai Sun, Mona Anvari, Manasi Sharma, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Yunzhu Li, Silvio Savarese, Hyowon Gweon, C. Karen Liu, Jiajun Wu, and Li Fei-Fei. BEHAVIOR-1K: A Human-Centered, Embodied AI Benchmark with 1,000 Everyday Activities and Realistic Simulation. *arXiv preprint arXiv:2403.09227*, 2024. URL https://arxiv.org/abs/2403.09227.

[60] BEHAVIOR Team. BEHAVIOR Challenge. https://behavior.stanford.edu/challenge/, 2026. Accessed 2026-11-18.

[61] Guanxiong Chen, Qianjun Xia, Jiawei Peng, Heng Zhang, Pengyu Jing, Bole Ma, Justin Qian, Yixian Cheng, Ziyi Jiao, Bingyang Zhou, Yiduo Qu, Luoxin Ye, Kaifeng Zhang, Kunyi Wang, Weijia Zeng, Yunuo Chen, Pengzhi Yang, Ziqiu Zeng, Siyuan Luo, Huamin Wang, Chao Liu, Alan Yuille, Fan Shi, Changxi Zheng, Yunzhu Li, Chenfanfu Jiang, and Peter Yichen Chen. Agentic Real2Sim: Physics-based World Modeling with Vision-Language Agents. *arXiv preprint arXiv:2607.19190*, 2026. URL https://arxiv.org/abs/2607.19190.

[62] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: In-depth Robustness Analysis of Vision-Language-Action Models. *arXiv preprint arXiv:2510.13626*, 2025. URL https://arxiv.org/abs/2510.13626.

[63] Matthew Chang, Gunjan Chhablani, Alexander Clegg, Mikael Dallaire Cote, Ruta Desai, Michal Hlavac, Vladimir Karashchuk, Jacob Krantz, Roozbeh Mottaghi, Priyam Parashar, Siddharth Patki, Ishita Prasad, Xavier Puig, Akshara Rai, Ram Ramrakhya, Daniel Tran, Joanne Truong, John M. Turner, Eric Undersander, and Tsung-Yen Yang. PARTNR: A Benchmark for Planning and Reasoning in Embodied Multi-agent Tasks. *arXiv preprint arXiv:2411.00081*, 2024. URL https://arxiv.org/abs/2411.00081.

[64] Kabir Swain, Sijie Han, and Antonio Torralba. EnvDreamer: Large-Scale Multimodal-to-Environment Generation for Embodied AI. *arXiv preprint arXiv:2610.04301*, 2026. URL https://arxiv.org/abs/2610.04301.

[65] Runyu Lu, Yubo Wu, Ethan Kou, Haotian Lin, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, Mosharaf Chowdhury, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Guanzhi Wang. ASPIRE: Agentic /Skills Discovery for Robotics. *arXiv preprint arXiv:2607.00272*, 2026. URL https://arxiv.org/abs/2607.00272.

[66] 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*, 2026. URL https://arxiv.org/abs/2607.08448.

[67] Meng-Hao Guo, Zhe-Han Mo, Jia-Jun Wang, Yi Zhang, Kejin Wang, Yi-Xuan Deng, Jia-Peng Zhang, Yongming Rao, and Shi-Min Hu. Transferring the intelligence of vlms to robotic control. *arXiv preprint arXiv:2609.22966*, 2026.

[68] Wenyi Li, Yizhe Chi, Youjie Zheng, Chunyu Liu, Xiaoqiu Wang, Mingju Gao, Zheng Jiang, Deyao Hong, Calvin

Xiao, Kaisen Yang, and Qinhuai Na. EmbodiedEvo: Self-Evolving Coding Agent For Physical Intelligence. *arXiv preprint arXiv:2610.18867*, 2026. URL https://arxiv.org/abs/2610.18867.

[69] Meng-Hao Guo, Yi Zhang, Zhe-Han Mo, Han-Zheng Wang, Jia-Jun Wang, Xin-Sheng Chen, Jiaxi Song, Yi-Xuan Deng, Jia-Ning Li, Xuanyu Chu, Hengrui Zhang, Huan-ang Gao, Yongming Rao, and Shi-Min Hu. A glimpse of training vlms as foundation models for robotic manipulation. *arXiv preprint*, 2026.

[70] Jiarui Yuan, Yuxin Zuo, Xiaoyu Huang, Wenbo Zhang, Kwanyoung Park, Junlin Yang, Tianxiang Zhang, Ruoqu Chen, Yutao Ouyang, Bingxiang He, Xingtai Lv, Hongjin Wu, Jiajun Liu, Jiahao Zhao, Tianwei Luo, Haozhan Li, Cheng Qian, Jiaxin Ge, Huan-ang Gao, Kaisen Yang, Jiaqi Deng, Qixin Xu, Siqiao Huang, Bowen Zhou, Mengdi Xu, Zhiyuan Liu, Pieter Abbeel, and Ning Ding. Democratizing Robot-Use Agents: A Unified Harness with Open Models. *arXiv preprint arXiv:2611.01560*, 2026. URL https://arxiv.org/abs/2611.01560.

Appendix

A Environment and Collection Details

A.1 Harness API

The runtime resets the scene once before the agent receives the instruction. The agent has one attempt, with access to a terminal and the public Python client. The interface exposes neither privileged object poses nor the verifier. Model reasoning consumes wall-clock time but does not advance physics.

Public call	What the agent gets or requests	Physics
`describe()`	Action names, units, limits and gripper mapping; camera and timing specifications; step, chunk and wall-clock limits.	No advance
`observe()`	Current camera image references and measured joint positions and velocities.	No advance
`save_images(dir)`	Downloads the current RGB observations as PNG images and returns paths by camera.	No advance
`step(action)`	Applies the target for one control interval by default. The optional repeat argument holds it for more intervals; reaching the target is not guaranteed.	One interval per repetition
`step_chunk(actions)`	Executes an ordered sequence within the chunk and remaining step limits. Returns the last observation, executed step count, and episode status.	One interval per executed action
`finish()`	Ends the attempt without settling steps. A separate verifier assigns the terminal reward (0 or 1).	No advance

The client accepts joint targets rather than Cartesian end-effector poses and provides no inverse-kinematics service. The table below shows the settings for two representative tasks: one single-arm Franka task and one dual-arm AgileX task. Read the specification for each task before issuing commands.

Example setting	Franka Panda (bottle-cap task)	AgileX (dual-arm example)
Action vector	7 arm angles (rad) + 2 finger displacements (m); the two finger displacements sum to the jaw opening.	Per arm: 6 angles + 2 finger displacements; left then right, 16 channels total. The two fingers have opposite opening directions.
Control / physics	20 Hz / 240 Hz (12 substeps).	15 Hz / 240 Hz (16 shared substeps for both arms).
RGB views	Fixed external and wrist.	External camera, left wrist camera, and right wrist camera.

A.2 Released environments by source dataset

We use demonstration videos and keyframes to reconstruct the scene layout and the interaction between the robot and the objects. Where available, task annotations clarify the intended goal, and recorded robot trajectories serve as motion references for the simulation controller. For RoboDojo, we use the official demonstration videos released by the RoboDojo team. For each task retained after deduplication, we reconstruct the scene in Isaac Sim and implement a controller that performs the demonstrated interaction with a simulated Franka or AgileX robot. Each released environment can be reset and controlled independently, and its verifier evaluates whether the robot achieves the task goal. The tables below count these environments by source.

Franka (single-arm)

Source dataset	Environments
DROID	2,591
BridgeData V2	465
BC-Z	432
RT-1	302
Berkeley Fanuc	105
RoboTurk	35
Columbia PushT	41
Berkeley Cable Routing	63
RoboCook	54
Berkeley Autolab UR5	42
CMU Stretch	2
Total	**4,132**

AgileX (dual-arm)

Source dataset	Environments
RoboDesk	3,292
RoboCOIN	1,758
GM100	1,038
RoboDojo	184
ViFailback	179
MM-30	158
RoboMIND2	66
Total	**6,675**

A.3 Quality assurance: independent review and repair

An independent QA agent reviews a copy of the builder's environment. It compares the source demonstration with the simulated run, focusing on what the robot acts on and how the object responds. For a pick-and-place task, for example, the reviewer uses the video and recorded object state to check that the object moves with the gripper after contact and settles on the support after release. It then resets the scene and replays the saved control commands to check that they reproduce the outcome.

The reviewer also checks the verifier against the written instruction. It considers both a plausible failure that should be rejected and a different way to achieve the stated goal that should earn credit. When it identifies a defect, the reviewer repairs the scene, controller, or verifier and reruns the case that exposed the problem. It then repeats the reference run and reset replay with the revised verifier and records the evidence and any unresolved issues. Appendix C.3 gives excerpts from the review prompt.

B Benchmark Evaluation Details

B.1 Test-Set Construction

Source and Split. The 500 tasks in RoboUniverse-Verified are produced by the same Real2Sim construction and QA pipeline as the training tasks (Section 2.3). They are held out as a zero-shot evaluation set and are never used to collect training trajectories. Of these, 200 are single-arm Franka tasks reconstructed from DROID episodes, and 300 are dual-arm AgileX tasks reconstructed from multi-source bimanual teleoperation data. Candidate tasks are drawn from environments that pass automated QA by stratified sampling over embodiments and task families, so that the test set covers all eight task families while preserving the diversity of data sources and scenes as far as possible.

Expert Verification. All candidate tasks are manually verified by 10 human experts in embodied AI. With reference to the source video, task instruction, simulation replay, and verifier code, the experts check each task against four criteria: (1) the instruction is clear, unambiguous, and consistent with the task goal in the source video; (2) the scene layout, object geometry, and physical parameters (e.g., mass, friction, and joint limits) are plausible; (3) the reference controller reliably completes the task after scene reset; and (4) the verifier rejects typical failure outcomes while accepting valid alternative solutions, so that it produces no misjudgments. Tasks that fail any check are returned for revision and re-verification, and tasks that cannot be fixed are removed from the candidate pool. All 500 retained tasks pass every check, which ensures that each task is solvable and reliably graded.

Task-Family Assignment. The source data of the single-arm tasks carry 20 manipulation subcategory labels. Because a source label does not always match the capability that a task tests (e.g., a `stack_unstack` task may require insertion, and a `pick_place` task may also close a drawer), each task is assigned individually, based on its instruction, to one of nine fine categories: place, reposition, assemble, open_close, actuate, clean, transfer_mix, deform, and composite. These map to the four single-arm families as follows: *Pick & Place* comprises place and reposition; *Assembly & Articulation* comprises assemble, open_close, and actuate; *Deformable & Material* comprises clean, transfer_mix, and deform; and *Composite* comprises composite. Table 6 shows the resulting composition of each family by source subcategory; 11 of the 20 subcategories are split across families. Since the source data of the dual-arm tasks lack unified subcategory labels, we first use a language model to pre-classify these tasks from their instructions, following the definitions of the four dual-arm task families in Section 2: tasks whose main difficulty lies in deformable objects or pouring are assigned to *Co-Deformable & Transfer*; tasks comprising two or more subgoals of different types with sequential dependencies are assigned to *Staged Coordination*; tasks dominated by assembly, insertion, cap twisting, or one arm stabilizing an object while the other manipulates it are assigned to *Co-Assembly & Stabilization*; and the remaining transport and handover tasks are assigned to *Co-Transport & Handover*. These four families correspond to seven fine categories: *Co-Transport & Handover* comprises cooperative transport and reorientation (36 tasks) and handover and regrasping (57); *Co-Assembly & Stabilization* comprises cooperative assembly and disassembly (32) and stabilize-and-manipulate (67); *Co-Deformable & Transfer* comprises coordinated transfer (33) and deformable manipulation (45); and *Staged Coordination* comprises staged coordination (30). The experts then review and confirm the family of every single-arm and dual-arm task individually, based on the primary manipulation capability that the task assesses.

Test-Set Statistics. Table 7 summarizes basic statistics of the test set. Task instructions have a median length of about 40 English words. They specify the target objects, the desired final state, and completion conditions such as releasing the object and letting it come to rest, but do not prescribe how the robot should move. The verifier of each task consists of several named binary conditions (e.g., `lid_open`, `keychain_outside_bag`, and `grippers_released`), each of which corresponds to a physical measurement and its threshold; a task is judged successful only when all conditions are satisfied simultaneously. These conditions typically cover three aspects: effective interaction with the target object, the target final state, and coming to rest after release.

Table 6 Composition of the single-arm task families by source subcategory. Counts are numbers of test tasks. Subcategories marked with † are split across families because each task is assigned individually according to its instruction.

Task family	Source subcategories (count)	Total
Pick & Place	pick_place† (16), push_topple† (11), stack_unstack† (5), push_pull† (4), rotate_twist† (2), sweep† (2), door_hinge_or_open_close† (2), other† (2), pour_transfer† (1)	45
Assembly & Articulation	press_switch† (20), stack_unstack† (13), lid_cover (12), door_hinge_or_open_close† (8), hang_unhang (7), rotate_twist† (3), drawer_slide† (2), pick_place† (2), insert_remove (1)	68
Deformable & Material	wipe_clean (20), stir_mix (14), dough_deformation (8), fold_spread (7), pour_transfer† (7), cable_routing (6), pick_place† (3), sweep† (2), other† (1)	68
Composite	multi_step (5), door_hinge_or_open_close† (5), drawer_slide† (2), press_switch† (2), pick_place† (2), push_topple† (1), push_pull† (1), other† (1)	19
Total		200

Table 7 Statistics of the RoboUniverse-Verified test set. Step budgets and verifier conditions are per task; verifier statistics cover the 357 tasks whose verifiers list their conditions explicitly.

	Single-arm (Franka)	Dual-arm (AgileX)	All
Instruction length (words, median [range])	37 [16–65]	40 [19–65]	39 [16–65]
Control-step budget (median [range])	3,000 [300–4,000]	3,000 [250–3,000]	3,000 [250–4,000]
Tasks with the 3,000-step budget	139 (70%)	232 (77%)	371 (74%)
Verifier conditions per task (median [IQR])		6 [5–8]	

C Prompts

C.1 Rollout agent

Franka | Harbor instruction.md

Press the round silver button on top of the tall silver-and-black drink machine between the toaster oven and the coffee maker. Push the button down, then lift the hand away so it springs back up.

Operating this environment

The robot scene is already reset. Control it using the installed Python client or CLI.
You may write and run your own Python control programs. Simulation advances only when you send actions; time spent reasoning does not advance physics.

```
roboenv describe
roboenv observe --images /tmp/robot-view
```

Open the returned PNG paths with your image-viewing tool. Observations include measured joint positions/velocities. Read the action names, units, limits and control interval before acting. Do not assume a particular joint count, frequency or gripper convention.

```
from roboenv_client import Client
robot = Client()
spec = robot.describe()
obs = robot.observe()
print(robot.save_images("/tmp/robot-view", obs))
# Supply a finite vector in spec["action"] order; choose it using your own controller.
# result = robot.step(action, repeat=10)
# result = robot.step_chunk(actions)  # bounded chunk; returns its final observation
# obs = result["observation"]
# print(robot.save_images("/tmp/robot-view", obs))
```

CLI actions: roboenv step --action '[...]' --repeat 10, or
roboenv step-chunk --file actions.json (a JSON array of action vectors).
Each action lasts exactly one declared control interval. Chunking supplies commands, not an automatic motion planner. Check feedback between chunks. A rejected chunk executes no actions; a runtime error may leave a completed prefix and ends the episode. Use one controller at a time. The client attaches episode/step IDs and retries with an identical request ID to prevent duplicate physical actions.

There is one attempt, no agent-accessible reset, and a finite step/time budget shown by describe. When you have finished, run roboenv finish (or simply end your turn). Finishing does not add settling steps: explicitly execute any needed final observation intervals. The independent verifier grades recorded physical evidence. No score or reference solution is available during the attempt. Source video/production files and private simulator state are not provided to the agent.

AgileX | Harbor instruction.md

Turn both blue bins upside down and place one over each end of the golden metal rack, leaving the middle position empty. Let go so that both bins remain supported by the rack.

Operating this environment

The robot scene is already reset. Control it using the installed Python client or CLI.
You may write and run your own Python control programs. Simulation advances only when
you send actions; time spent reasoning does not advance physics.

```
roboenv describe
roboenv observe --images /tmp/robot-view
```

Open the returned PNG paths with your image-viewing tool. Observations include measured
joint positions/velocities. Read the action names, units, limits and control interval
before acting. Do not assume a particular joint count, frequency or gripper convention.

```
from roboenv_client import Client
robot = Client()
spec = robot.describe()
obs = robot.observe()
print(robot.save_images("/tmp/robot-view", obs))
# Supply a finite vector in spec["action"] order; choose it using your own controller.
# result = robot.step(action, repeat=10)
# result = robot.step_chunk(actions)  # bounded chunk; returns its final observation
# obs = result["observation"]
# print(robot.save_images("/tmp/robot-view", obs))
```

CLI actions: roboenv step --action '[...]' --repeat 10, or
roboenv step-chunk --file actions.json (a JSON array of action vectors).
Each action lasts exactly one declared control interval. Chunking supplies commands,
not an automatic motion planner. Check feedback between chunks. A rejected chunk
executes no actions; a runtime error may leave a completed prefix and ends the episode.
Use one controller at a time. The client attaches episode/step IDs and retries with
an identical request ID to prevent duplicate physical actions.

There is one attempt, no agent-accessible reset, and a finite step/time budget shown by
describe. When you have finished, run roboenv finish (or simply end your turn).
Finishing does not add settling steps: explicitly execute any needed final observation
intervals. The independent verifier grades recorded physical evidence. No score or
reference solution is available during the attempt. Source video/production files and
private simulator state are not provided to the agent.

C.2 Builder agent

Excerpt 1: Reconstruction and camera setup

Watch the supplied video and keyframes. Build a detailed, realistic Isaac Sim scene at the visual quality of the shared RoboDojo reference, and make the robot perform the same task through physical interaction. Preserve important appearance, object relationships and task meaning; exact coordinates, dimensions and timing are unnecessary. Choose the assets, geometry, materials, controller and feasible motion yourself. Reproduce the interaction shown, whether it involves grasping, pushing, an articulated mechanism, tools or material transfer.

Read AGENTS.md, delivery_template/README.md and the workspace embody-sim and embody-perception skills. Initialize the author interface files before implementing your scene. Build the scene behind the operable backend from the start; keep reference control separate and route both demonstration and reset replay through env.step(action). Use the prepared runtime and SAM3.1/DA3 tools when useful; relative depth is not metric depth. Start from your assigned video and runtime documentation. Do not read previous reconstruction demos or other agents' implementations.

Build the complete settled scene and both required RGB cameras before tuning the motion. Read the embody-sim appearance reference and open its shared quality reference; use them for asset access, native rendering and geometric/surface detail. The source video determines object identity, framing, layout, materials and lighting. Choose a view from the source and compare the whole rendered image with that view, resolving the largest differences in silhouettes, proportions and prominent context before spending the run on control. Save this comparison as output/scene_design.png. A short reachability check is useful during layout; keep the scene complete when adapting it for feasible interaction.

Excerpt 2: Contact and robot control

Model physical objects as geometry with appropriate collision and material properties. Movable objects are dynamic; genuinely fixed furniture and structural supports may be static colliders. Solid background props are physical obstacles too. Collision shapes must agree with visible surfaces, including the robot and fingers. Represent task-relevant cavities, openings, walls, handles and thin parts so contact occurs on the visible surface: a visible opening must not be blocked by a solid convex hull, and a visible wall must not allow passage. Simplified colliders are fine when they preserve support, clearance and interaction. Purely visual details such as printed labels need no separate collider. Do not make functional gaps implausibly large, shrink colliders away from visible surfaces, or disable collisions to make the task easier. Fix geometry, grasp, approach or control when objects collide incorrectly.

Use arm/finger joint drives or efforts; let contact, friction and gravity move task objects. Setup/reset poses before the operation are fine. During the task, do not attach or weld objects to the hand, rewrite object poses, drive the target directly, apply task-completing external forces, disable object dynamics, or play back robot joint states frame by frame. Passive mechanical joints and physically justified anchored supports are fine. Indicators or erased marks should respond to actual physical interaction. Motion must avoid solid obstacles; normal contact and small solver compliance are acceptable, but visible or sustained interpenetration, tunneling and hovering are defects to fix, not successful manipulation. Releasing a grasp must leave the object supported in a physically plausible state.

Excerpt 3: Instruction and success conditions

Keep the main instruction and verifier aligned around ordinary task completion. Describe what should visibly change and, when relevant, whether to let go, put a tool down safely, withdraw the hand or leave an object supported. Do not add finishing requirements that the task does not need. Use a few natural sentences, not a numerical checklist or a prescribed motion sequence. The verifier should accept the reasonable range of outcomes that satisfy those sentences: meaningful movement, the intended functional relation and a plausible finished state. Do not invent an exact displacement, destination, orientation, object count or hold duration just because your reference run happens to achieve it. Any task-essential constraint must be clear in the instruction; a precise count, distance or time belongs there only if the source goal actually depends on it.

C.3 QA agent

Excerpt 1: Instruction and verifier alignment

Compare the instruction and implemented success decision in both directions. Follow the actual return path in verify.py/ evaluator_adapter.py and their helpers: a printed diagnostic is not necessarily a success condition. Enumerate each operative predicate and classify it as task semantics, physical validity, evidence integrity, or numerical tolerance. In quality/alignment.md, map every task-semantic predicate to its instruction clause, and map every promised task outcome to the measurement that checks it. Include the code location, gap if any, and resolution. Account for conjunctions/ disjunctions and short-circuit branches, not just names in a JSON report.

The instruction must let a solver understand acceptable completion without reading the verifier or reference controller. Keep the main instruction short, in everyday language, with few numbers. Explain the target, meaningful change or functional relation, and task-relevant finishing state. Necessary release, placement, support or withdrawal must be clear when actually required. Do not add irrelevant finishing actions or require the reference controller's exact route, distance, orientation, duration or final joint/gripper posture. Keep numerical tolerances out of the short main instruction; when the frozen author assignment requires a verifier appendix, give their exact implemented meaning and values there. If a reasonable completion of the instruction would fail, remove the unsupported restriction or repair its tolerance; clarify a genuinely necessary goal in ordinary language. Do not weaken genuine physics or evidence checks. Conversely, an outcome promised by the instruction must be measured rather than assumed from a controller phase or elapsed time.

Excerpt 2: Risks and regression cases

Prioritize risks that can change this task's verdict. Inspect correct object identity, nontrivial progress from an unsolved reset, required relation and final support, brief success followed by failure, completion flags/proximity/controller-waypoint substitutes, and physical shortcuts. Check geometry/coordinate conventions and whether contact or a causal interaction chain is measured where needed; indirect effects through a valid tool or passive mechanism may be legitimate. Review missing/NaN/inconsistent evidence and timestamp coverage, reset/episode contamination, and whether the evaluator consumes the current run's host-generated evidence rather than solver-provided success text. For a deployed API, inspect relevant action validation, duplicate requests, reset/finish behavior and public/private separation. An AST/source hash check alone does not establish physical validity or adversarial isolation.

Reproduce important suspected failures with small, targeted tests, then repair them. Choose a practical set based on the actual predicates and access model: no action or pre-solved start, wrong object, partial/transient completion, incomplete/ nonfinite evidence, and a legitimate alternate completion are useful candidates. Record why a category is not applicable rather than creating meaningless fixtures. Save runnable tests and their exact commands/results in quality/. Label edited traces and CPU fixtures as logic regression evidence; they do not demonstrate a reachable physical exploit or successful robot run. A valid alternative should exercise a real allowance (another goal-satisfying end state or irrelevant final posture), not just relabel the original passing record. Do not force every test to return failure: missing evidence may correctly be inconclusive, and legitimate variations should succeed.

Excerpt 3: Repairs and replay validation

Implement minimal repairs in this task's instruction, verifier, evidence export or backend as needed. Preserve intended task and useful tolerances. For each confirmed issue, record before/after behavior and its regression. If fixing an API/ framework flaw requires shared-infrastructure changes, mark that boundary and retain needs_revision or blocked until resolved; do not silently change shared services. Do not describe the result as hack-proof. Describe the tested threat model, fixed issues and remaining risks.

After repairs, validate the final revision under its own source hashes and paths. Run its frozen template's static/interface checks, a real reference execution and a separate reset replay of saved controls through the same backend.step. Run the final verifier on both. Keep source binding/metadata consistent with the final revision; do not disable audits or rewrite old verdicts to hide source changes. Use the same version for tests and final evidence. Preserve the original operation, USD, controls/states, video and README deliverables; replace stale output evidence with the repaired revision's actual runs, retaining useful before/after evidence in quality/. For the external-wrist contract, reference and reset runs each retain both real camera streams. Inspect videos for completion and obvious physical regressions. Simulator smoke tests and interface passes are not task success. If a meaningful repair cannot be verified, deliver runnable work and the honest outstanding gap rather than accepted.