ROBO-HARNESS K1: HARNESSING ROBOT-USE AGENTS VIA PERCEPTION AUGMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Foundation vision-language models (VLMs) recognize objects, interpret instructions, and reason about spatial relations. A central question for robotic manipulation is how to fully translate this capability into physical actions. Adding a learned action head, as in vision-language-action (VLA) models, requires large-scale demonstrations, may degrade the inherent VL understanding of VLMs, and generalizes poorly. Recent results of Astra (GPT-6) as policy show the potential of robot-use agents (RUAs), which keep the VLM intact as a direct controller. However, this route is slow, expensive, and difficult for weaker models. We step back and ask: what is fundamentally missing between a capable VLM and a working manipulation policy? We find that the answer is accessible perceptual toolkits for 3D spatial understanding and measurement. We introduce ROBO-HARNESS K1, which exposes *perception as tools for harness*: the agent queries calibrated depth, inspects visual anchors with persistent object identities, evaluates grasp hypotheses, and makes actions and motions from the returned evidence. ROBO-HARNESS K1 enables the VLM to understand depth and space without invasive depth-encoder retraining, so the agent can reason about 3D object relations instead of guessing. On LIBERO-PRO, Gemini 3.7 Flash with K1 reaches 77.8%, surpassing GPT-6 Astra's 61.1% with an RGB-only harness, and K1 further augments Astra to 88.9%. Without target fine-tuning, Gemini with K1 transfers to three RoboSuite arms (90.0% average on shared tasks) and dual-arm RoboTwin tasks. For training RUA models, K1 provides better sample efficiency and generalization than VLAs and vanilla RUA. A Qwen3.5-9B model fine-tuned on only 107 episodes reaches 44.2% on new-state generalization versus 30.2% for OpenVLA, and 13.9% on new-task generalization versus 0.0% for OpenVLA. These results suggest that training RUAs with a proper perceptual harness outperforms VLAs, unlocking VLMs' great potential for robotic manipulation.

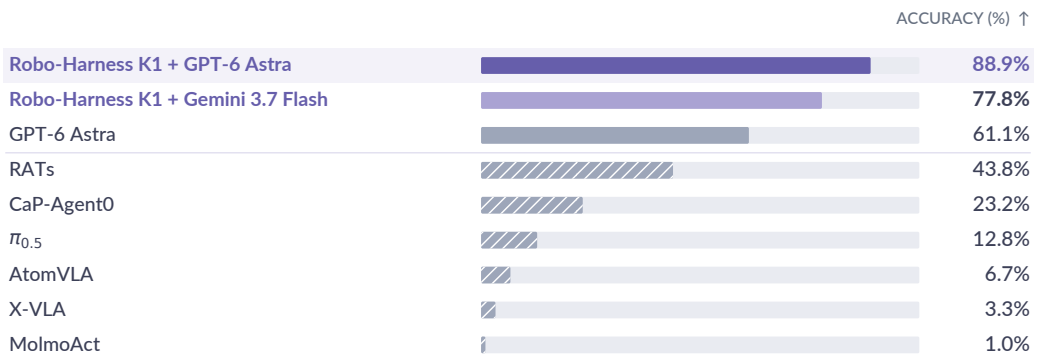


Figure 1: **ROBO-HARNESS K1 with Gemini 3.7 Flash surpasses GPT-6 Astra on LIBERO-PRO tasks. Astra with ROBO-HARNESS K1 obtains 27.8% performance gains.** Published baselines are from Table 1 (Zhang et al., 2026a; Sun et al., 2026). Evaluation subsets, trial counts, and comparison scope are detailed in Appendix C.3.

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

1 INTRODUCTION

Foundation vision-language models (VLMs) already possess strong visual-language understanding: recognizing objects, interpreting instructions, and reasoning about spatial relations (Bai et al., 2025; Brohan et al., 2023). A central question for robotic manipulation is how to maximally translate this capability into physical actions.

Vision-language-action (VLA) models add learned action heads to the VLM backbone (Kim et al., 2025; Physical Intelligence et al., 2025). This enables embodiment adaptation but may degrade the inherent VL understanding of VLMs: knowledge insulation studies show that naive motor supervision impairs semantic transfer (Driess et al., 2025), and vision-representation analyses confirm that action adaptation damages original visual features (Kachaev et al., 2025). Language-aligned supervision reduces this mismatch (Hancock et al., 2025), but a fundamental tension remains between action specialization and VL preservation. VLAs also generalize poorly without large demonstration coverage: LIBERO-PRO exposes failures consistent with trajectory memorization when objects, layouts, or instructions change (Zhou et al., 2025), despite the broader generalization shown by heterogeneous co-training (Physical Intelligence et al., 2025).

Recently, the emergence of Astra (GPT-6) as policy shows that the robot-use agent (RUA), by keeping the VLM intact as a direct controller, has great potential (Robocurve, 2026a; Hu et al., 2026; Chen et al., 2026; Robocurve, 2026b). Show-Harness further highlights the importance of semantically accessible action units (Chen et al., 2026). However, this route is slow, expensive, and strongly dependent on frontier-model capability. For weaker VLMs, an RGB-only interface asks the agent to infer metric geometry from pixels, a task whose difficulty far exceeds what spatial reasoning from RGB alone can handle.

We step back and ask: *what is fundamentally missing between a capable VLM and a working manipulation policy?* We find that the answer is **accessible perceptual toolkits for 3D spatial understanding and measurement**. VLMs understand what objects are and how they relate semantically, but manipulation requires calibrated depth, object-relative distances, and gripper-aligned geometry that RGB does not provide. One could add depth as a new input modality, but this requires architectural changes and additional alignment training (Chen et al., 2025a), an invasive and expensive path. Instead, we make perception available as callable tools. The agent can request the missing spatial evidence through tool calls, just as a language agent queries external APIs, so it reasons about 3D relations instead of guessing from RGB alone.

We introduce ROBO-HARNESS K1, a task-agnostic harness built around *perception as tools*. Each detected region receives a persistent identity anchor that survives across observations. Visual reference lines overlay calibrated measurements onto the VLM's existing RGB views, making 3D object relations, orientations, and distances visible without a depth encoder. The agent can query spatial measurements, inspect projected grasp hypotheses, and select generic motions from the returned evidence. The K1 harness supplies calibrated spatial evidence, not task-specific `pick`/`place` primitives or hidden object poses. The VLM remains responsible for choosing the target, the motion, and the recovery strategy.

With ROBO-HARNESS K1, Gemini 3.7 Flash reaches 77.8% on matched LIBERO-PRO cases, surpassing GPT-6 Astra's 61.1% with the RGB harness, and K1 augments Astra to 88.9%. Gemini also transfers without any target-task fine-tuning to three RoboSuite arms and to RoboTwin's dual-arm tasks in SAPIEN. On RoboTwin Hard, where visual and environmental perturbations cause trained VLAs to degrade sharply (RDT: $47.6 \rightarrow 19.2\%$; π_0: $64.0 \rightarrow 25.2\%$), zero-shot K1 reaches 28.0% with only a 4-point drop from Easy, surpassing RDT and π_0 without any target demonstrations. It shows great transferability, generalization and robustness of ROBO-HARNESS K1. Beyond zero-shot deployment, ROBO-HARNESS K1 tool-call traces naturally align with next-token prediction, enabling a new training paradigm: Qwen3.5-9B fine-tuned on only 107 teacher episodes outperforms all action-policy baselines across evaluation splits and is the only method to generalize on unseen tasks where every VLA baseline scores 0.0%.

Our contributions are:

- We identify accessible perceptual evidence as the critical missing piece between VLM competence and manipulation capability, and propose perception as tools: a non-invasive harness that augments

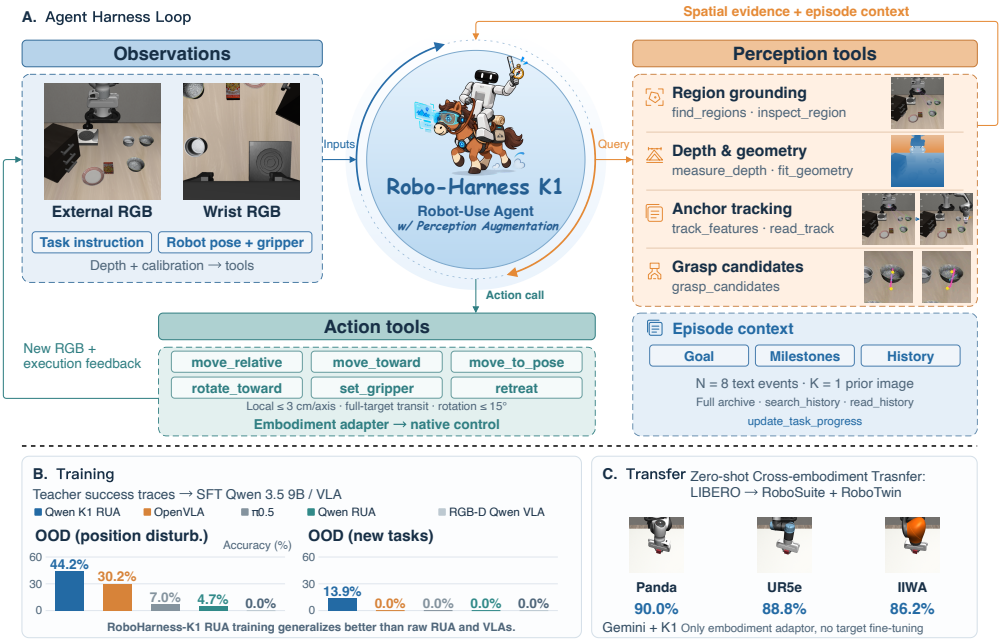


Figure 2: **Robo-Harness K1: perception-augmented Robot-Use Agents. (A) Agent harness loop.** The VLM receives RGB views, the instruction, and robot state. It queries region grounding, calibrated geometry, anchor tracking, and grasp hypotheses as needed, while depth and calibration stay behind the tool interface. **(B) Training.** Qwen3.5-9B learns from tool-call traces via SFT, and generalizes well. Bars show epoch-5 accuracy on new-state and held-out-task evaluation. Qwen K1 RUA uses our harness, Qwen RUA is the RGB-only direct-action baseline. All methods share source episodes but differ in inputs and tools. **(C) Transfer.** Gemini with K1 transfers from LIBERO to RoboSuite (Panda, UR5e, IIWA) without target fine-tuning, using explicit embodiment adapters and the same Panda gripper.

the VLM's existing interface with calibrated depth, visual anchors, and grasp hypotheses, enabling the agent to understand 3D space instead of guessing from RGB.

- We show that K1 boosts both Gemini and Astra without weight updates. Gemini with K1 surpasses RGB-only Astra and transfers zero-shot to RoboSuite and dual-arm RoboTwin tasks across embodiments and environments.
- We find that training RUAs with K1 provides better sample efficiency and generalization than VLAs, with a compact student outperforming action-policy baselines from only 107 episodes.

2 Related Work

VLM models for robotic policy. The field has explored several routes to turn VLM knowledge into robot policies. The most direct approach represents actions as language tokens, also known as vision-language-action (VLA) models. RT-2 showed that co-training a VLM on web and robot data yields a policy that can follow novel instructions, establishing that VLM representations transfer to motor control (Brohan et al., 2023). OpenVLA scaled this recipe into an open-source action-token policy (Kim et al., 2025), and $\pi_{0.5}$ extended it through heterogeneous co-training with a continuous action expert, achieving broader task coverage (Physical Intelligence et al., 2025). Yet a growing body of evidence reveals a cost: action adaptation can degrade the VL representations inherited from pretraining. Knowledge insulation studies show that naive motor supervision impairs semantic transfer (Driess et al., 2025), and visual-representation analyses confirm that action fine-tuning damages original features (Kachaev et al., 2025). Language-aligned supervision reduces this degradation (Hancock et al., 2025), but a fundamental tension between action specialization and VL preservation remains. Generalization further depends on demonstration coverage: LIBERO-PRO tests show that VLA policies tend to replay memorized trajectories rather than adapt to changed

objects and relations (Zhou et al., 2025). In contrast, we investigate a complementary direction: keeping the VLM intact as a direct controller and supplying the spatial evidence it lacks through tool calls, avoiding the representation damage of action heads.

Agentic robot systems. Early agentic systems decompose manipulation into high-level planning and pre-built primitives: SayCan grounds language in skill affordances (Ahn et al., 2022), Code as Policies generates executable programs (Liang et al., 2023; Fu et al., 2026; Zhang et al., 2026a), VoxPoser builds spatial value maps (Huang et al., 2023a), and Inner Monologue adds environment feedback for replanning (Huang et al., 2023b; Fu et al., 2026; Zhang et al., 2026a). A more radical departure, which Isola terms the *robot-use agent* (RUA) (Isola, 2026), treats the VLM itself as a direct controller that sends actuator commands and receives sensor observations through the same agentic loop. VIA, Show-Harness, and Inspect Robots instantiate this paradigm by exposing pose commands, visual interaction, or semantic motion units rather than fixed skill APIs (Hu et al., 2026; Chen et al., 2026; Robocurve, 2026a). GUAVA and Show-Harness further show that such harness designs generalize across tasks and can be distilled into compact policies (Liu et al., 2026; Chen et al., 2026). Our focus is what perceptual evidence the RUA receives within this loop: persistent visual anchors, calibrated measurements, and projected grasp hypotheses that resolve spatial uncertainty before the agent commits to a motion. RoboHarness and Harness-VLA (Huang et al., 2026; Zhang et al., 2026b) operate at a different level, orchestrating heterogeneous task-specific policies through memory-driven planning. In contrast, ROBO-HARNESS K1 augments the evidence that a single RUA model uses for its own decisions, addressing the per-step perception gap rather than the policy-selection problem.

3 METHOD

3.1 PERCEPTION AS TOOLS

The core idea of ROBO-HARNESS K1 is to make spatial evidence accessible through the VLM's existing image-and-language interface. Instead of asking the agent to infer metric structure from RGB alone or training it to consume a new depth modality, the harness exposes four complementary tool families (Figure 3): *region grounding* identifies visible entities, *depth and geometry* measures their spatial relations, *anchor tracking* maintains these references across actions, and *grasp candidates* translates observed geometry into possible hand poses.

Formulation. We model the agent-harness interaction as a Markov decision process (MDP) $\langle \mathcal{S}, \mathcal{U}, \mathcal{H}, g \rangle$, where $\mathcal{S}$ is the set of environment states, $\mathcal{U} = \mathcal{U}_{\text{reason}} \cup \mathcal{U}_{\text{act}}$ is the tool-call action space (partitioned into reasoning and physical actions), $\mathcal{H}$ is the frozen harness that serves as the transition function (executing tool calls and returning observations), and g is the language instruction that remains fixed throughout an episode. Let $s_t \in \mathcal{S}$ denote the environment state at decision step t and o_t the observation (RGB views, TCP pose, gripper state) derived from s_t. The agent, parameterized by π_θ (a VLM with parameters θ), maintains a running context c that accumulates tool receipts across steps, and selects the next tool call, a tool identifier $u \in \mathcal{U}$ together with its arguments a (e.g., pixel coordinates, region identifiers, or motion parameters), conditioned on the instruction, the current observation, and this context:

$$(u, a) \sim \pi_\theta(\cdot \mid g, o_t, c). \tag{1}$$

The key structural property is that the action space splits into two disjoint subsets, giving rise to two types of transitions. *Reasoning steps* (e.g., `find_regions`, `measure_depth`) query the harness for perceptual evidence without altering the environment state:

$$u \in \mathcal{U}_{\text{reason}}: \quad r = \mathcal{H}_{\text{reason}}(u, a; s_t), \qquad s_{t+1} = s_t, \qquad c \leftarrow \mathcal{M}(c, r). \tag{2}$$

The harness returns a receipt r (measurements, region identities, grasp hypotheses, or memory retrievals), and the context manager $\mathcal{M}$ appends it to the running context c. The agent may issue multiple reasoning steps in sequence to gather sufficient evidence.

Action steps (e.g., `move_relative`, `move_to_pose`, `set_gripper`) commit a physical command that advances the environment:

$$u \in \mathcal{U}_{\text{act}}: \quad (r, o_{t+1}) = \mathcal{H}_{\text{act}}(u, a; s_t), \qquad s_{t+1} \neq s_t, \qquad c \leftarrow \mathcal{M}(c, r, o_{t+1}). \tag{3}$$

Here o_{t+1} is the new observation (RGB views, TCP pose, gripper state) captured after execution, and r reports achieved motion and residual error. Only action steps increment the decision index t and

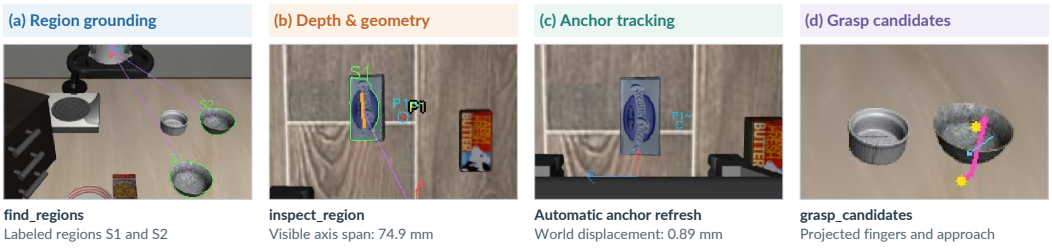


Figure 3: **Four perceptual tool families through the VLM's existing RGB interface.** Recorded image crops show (a) labeled regions, (b) measured surface geometry, (c) a persistent anchor after camera motion, and (d) a projected grasp hypothesis. The complete tool schemas, overlay conventions, and source calls appear in Appendix A.5.

produce a state transition. The episode terminates when the agent calls `done` or reaches a step budget. The harness sensor input o_t includes RGB and metric depth from two kinds of cameras (external and wrists), camera calibration, TCP pose, and gripper opening. The agent receives original-view RGB and tool receipts, not dense depth arrays or raw point clouds. The interface was frozen before the reported evaluation (Appendix C).

We now describe each tool family. Tool-specific details, overlay conventions, and extended visual examples appear in Appendix A.5.

First is the **Perception tools**, we have four families of perception tools as follows.

(i) Region grounding. Before any spatial measurement, the agent and the harness must agree on a target entity. `find_regions` accepts a text query and uses SAM 3 (Carion et al., 2025) to return candidate masks with persistent identifiers overlaid on the original image. The agent selects among candidates by visual inspection. `inspect_region` refines a region through an agent-specified bounding box, and `check_region_view` projects measured surface points into the other camera with a depth-consistency check, enabling cross-view verification without an additional sensor (Figure 3a).

(ii) Depth and geometry. Once a target is grounded, `measure_depth` converts a selected pixel (u, v) into a metric surface location via calibrated back-projection:

$$x^w = T_c^w \left[D^c(u,v)(K^c)^{-1}(u,v,1)^\top ; 1 \right], \tag{4}$$

where D^c is measured depth, K^c the intrinsic matrix, and T_c^w the camera-to-world transform. `fit_geometry` estimates a plane or principal direction from valid depth samples over a region. `region_relation` reports relative displacements between regions or between a region and the robot TCP. The harness overlays point labels, principal axes, and TCP-to-surface reference lines onto the RGB view (Figure 3b), making metric relations directly visible to the agent.

(iii) Anchor tracking. Observations should remain useful beyond the call that produced them. TAPNext++ (Jung et al., 2026) tracks image-point correspondences across frames, and depth measurements lift these into world coordinates, separating apparent motion from the wrist camera from actual surface displacement. Each anchor carries a persistent identifier and a validity flag: when correspondence becomes uncertain, the live measurement is invalidated but the historical reference is retained for the agent's memory (Figure 3c).

(iv) Grasp candidates. A surface location alone does not specify a hand orientation. Given a selected region, `grasp_candidates` uses GraspGen (Murali et al., 2025) to propose parallel-jaw poses from its measured point cloud. An adapter maps the provider's hand geometry to calibrated robot finger-pad coordinates. The agent receives candidate TCP positions, orientations, pregrasp waypoints, and schematic projections onto the RGB view (Figure 3d), and it selects and execute one candidate.

Control and feedback. Motion tools use explicit coordinate conventions. `move_relative` bounds each axis to $\delta_{\text{axis}} = 0.03$ m per step:

$$|\Delta x_i| \le \delta_{\text{axis}}, \quad i \in \{x, y, z\}, \tag{5}$$

mimicking delta move action chunks. `move_to_pose` and `rotate_toward` maintain absolute targets across bounded steps. `move_toward` makes a direct move to a target within one call,

stopping on tolerance, stall, or execution budget rather than the local per-axis limit. This creates an adaptive control granularity: the agent uses fine-grained bounded steps when precise alignment or visual feedback is needed, and switches to direct moves for longer transits to a known target, reducing model calls, keeping the context history clean, and accelerating execution. Every action reports achieved motion and residual error. Anchor co-motion supplies evidence of grasp success. Appendix A details the full tool interface, coordinate conventions, and gripper semantics.

Memory. The memory contains the instruction, current evidence, the last N text exchanges, and images from the previous K decision ($K = 0$ means the agent can only see current state's image). If not mentioned otherwise, we set N=8, K=1. Complete exchanges are archived and accessible through tools `search_history` and `read_history`. Each call includes a concise decision note like chain-of-thoughts, recording evidence, intent, and expected outcome, enabling faithful student supervision from the information actually available at each step.

3.2 From Tool Interaction to Policy Learning

The K1 harness enables frontier VLMs to manipulate without any weight update. A natural next question is: *can smaller, open-weight models acquire the same capability through training?* Frontier agents like Gemini and Astra succeed through massive pretraining, but their spatial understanding manifests as tool calls, a format that aligns directly with the VLM's native next-token prediction objective. A student can therefore learn to predict the next tool call, including the observation, the requested measurement, and the evidence gathered before each motion, through standard next-token prediction rather than a foreign action output through a separate head. We hypothesize that this alignment makes general spatial reasoning easier for compact models to master than raw action regression, preserves existing VLM capabilities, and gives perception-augmented RUAs better policy learning efficiency and generalization potential than VLAs.

Tool-use supervision. We supervise one executable assistant tool call per decision. Filtering removes execution errors, invalid calls, and rejected completions. With context c_j and serialized target y_j, the student minimizes

$$\mathcal{L}_{\text{agent}} = -\sum_{j} \sum_{k \in y_j} \log \pi_\theta(y_{j,k} \mid c_j, y_{j,<k}). \tag{6}$$

We adapt Qwen3.5-9B (Qwen Team, 2026) using language-side LoRA (Hu et al., 2022) with the vision encoder frozen. At test time it invokes the same frozen harness tools. We use the Gemini evaluation as a common source for both tool-use and action-prediction supervision, excluding held-out conditions. The shared pool contains 107 successful episodes (Appendix B).

4 Experiments

The experiments aim to verify three research questions: (i) Does the the harness of Robo-Harness K1 improve frozen-model deployment? (ii) Does training RUAs with K1 provide better sample efficiency and generalization than VLAs? (iii) Do K1's spatial conventions transfer to other robot arms?

4.1 Experimental Setup

Benchmarks. All experiments use LIBERO-PRO (Zhou et al., 2025) for both frontier evaluation and student training. Cross-environment transfer uses RoboSuite (Zhang et al., 2026a) (single-arm and dual-arm tasks with Panda, UR5e, and IIWA) and RoboTwin (Chen et al., 2025b) (bimanual tasks under Easy and Hard conditions). Task accuracy is the percentage of episodes satisfying the native simulator success criterion. Trial counts, seeds, budgets, and replay checks appear in the appendix. **Frontier evaluation.** We evaluate Gemini 3.7 Flash and GPT-6 Astra on Spatial, Object, and Goal suites in LIBERO-PRO, each with position-swap and task perturbations. A smaller paired comparison matches tasks and initial states across two GPT-6 Astra interfaces (RGB-only versus K1). The RGB-only Inspect Robots agent (Robocurve, 2026a) serves as the original RUA harness baseline. Appendix C specifies the manifests and configurations. **Student training and evaluation.** We use 107 successful Gemini episodes as training data for all student methods. Split A evaluates exact training configurations (in-domain), B evaluates new initial states of the same conditions (out-of-distribution (OOD), with position disturbance), and C evaluates task conditions held out from fine-tuning (OOD

Table 1: **LIBERO-PRO task accuracy (%).** Position denotes the position-swap suite. MolmoAct, NORA, X-VLA, and AtomVLA are from Sun et al. (2026), Table IV; other VLA and CaP rows are from RATs (Zhang et al., 2026a). Averages use the six displayed conditions.

	Object		Goal		Spatial		Average
Method	Position	Task	Position	Task	Position	Task	
VLA Methods							
OpenVLA	0.0	0.0	0.0	0.0	0.0	0.0	0.0
π_0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$\pi_{0.5}$	17.0	1.0	38.0	0.0	20.0	1.0	12.8
MolmoAct	6.0	0.0	0.0	0.0	0.0	0.0	1.0
NORA	0.0	0.0	0.0	0.0	0.0	0.0	0.0
X-VLA	2.0	8.0	1.0	9.0	0.0	0.0	3.3
AtomVLA	10.0	0.0	2.0	11.0	16.0	1.0	6.7
Code as Policy Methods							
CaP-Agent0	27.0	31.0	29.0	16.0	13.0	23.0	23.2
RATs	61.0	63.0	43.0	36.0	29.0	31.0	43.8
RUA Methods							
ROBO-HARNESS K1 + Gemini	**80.0**	**93.3**	**70.0**	**56.7**	**73.3**	**90.0**	**77.2**

with new tasks). Baselines include Qwen3.5-9B action-regression (continuous action chunks), $\pi_{0.5}$ (official flow-matching architecture), OpenVLA-7B (discretized action tokens from RGB), and a vanilla RUA that trains Qwen3.5-9B to emit JSON actions from a single unannotated RGB image as the same in Robocurve (2026b) (Appendix B.3). All methods share the same episodes, train for five epochs, and save checkpoints at epochs 1, 3, and 5. **Zero-shot transfer.** We deploy the frozen Gemini agent on RoboSuite and RoboTwin without any target-task fine-tuning. For RoboSuite, the adapter changes embodiment-specific control and sensing while retaining the generic K1 interface. For RoboTwin, we evaluate ten bimanual tasks under Easy and Hard initial conditions. Appendices D and D.2 detail adaptation boundaries and task coverage.

4.2 PERCEPTION TOOLS UNLOCK MANIPULATION CAPABILITY

K1 achieves strong accuracy across manipulation categories. Table 1 reports 77.2% overall accuracy: Object 86.7%, Spatial 81.7%, and Goal 63.3%. Object and Spatial benefit most from K1's perception tools, as these categories primarily require accurate target localization and spatial reasoning for approach and placement. Goal tasks involve articulated or multi-stage contact sequences, where execution difficulty extends beyond spatial perception.

K1 benefits both weaker and stronger models. On matched cases, Gemini with K1 reaches 77.8%, surpassing the RGB-only Astra result of 61.1% (Figure 1; Table 7). K1 also augments Astra to 88.9%, a gain of 27.8 percentage points. This shows that perception tools and model capability are complementary: a weaker model equipped with K1 can outperform a stronger model without it, and K1 further lifts an already strong model. Appendix B.4 compares the interfaces.

4.3 TRAINING COMPACT RUAS FROM TOOL-CALL TRACES

K1 RUA outperforms all action-prediction baselines. At epoch 5, Qwen K1 RUA reaches 51.2% in-domain (split A) and 44.2% on new initial states (split B), versus OpenVLA's 20.9%/30.2% and $\pi_{0.5}$'s 7.0%/7.0% (Figure 4; Tables 8, 9). Qwen VLA, which replaces tool-call supervision with continuous action regression on the same backbone, remains at 0.0% (Appendix B.3). The RGB-only text-action baseline (Qwen RUA) with the Inspect Robot Harness (Robocurve, 2026b), which removes perception tools and predicts actions in text form, reaches only 7.0% on A and 4.7% on B (Appendix B.4). Invalid actions terminate 35.2% of its evaluations, confirming that perception tools and a structured interaction protocol are both essential beyond the text-action format alone. On held-out task conditions (split C), K1 RUA reaches 13.9% and is the only method to succeed, while all action baselines score 0.0%. On new states, OpenVLA is stronger in Object (62.5% versus 43.8%), whereas K1 RUA leads in Spatial (46.7% versus 6.7%), suggesting that tool-use training particularly strengthens reusable spatial reasoning (Table 9). Note that generalization on B and C does

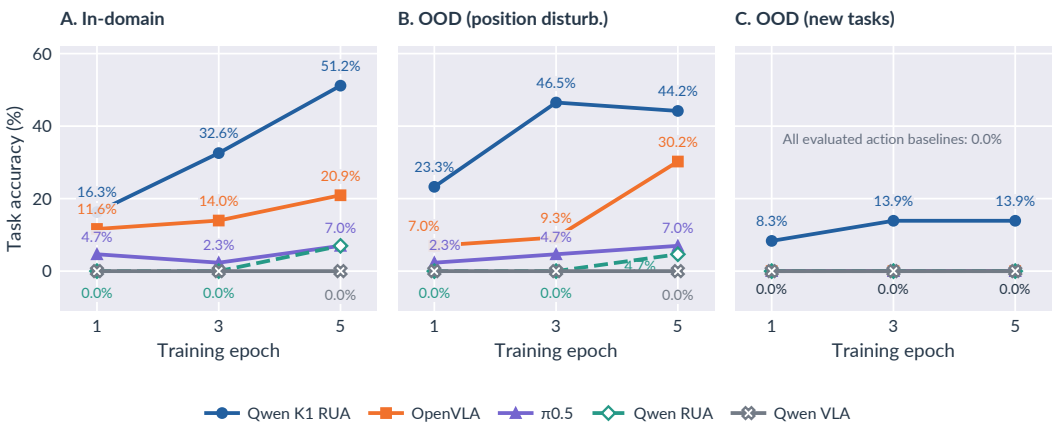


Figure 4: **Task accuracy across training epochs.** A: in-domain tests with exact training configurations. B: OOD new initial states (development set). C: OOD held-out task conditions. Qwen K1 RUA uses our harness. Qwen RUA is the RGB-only text-action baseline. Qwen VLA is the modified VLA with the same Qwen3.5-9B backbone. All methods share the same 107 teacher episodes. Training details are in Appendix B.

Table 2: **RoboSuite zero-shot transfer accuracy (%).** K1 uses Gemini without target fine-tuning. CaP-Agent0 and RATs are published results (Zhang et al., 2026a); – denotes untested tasks.

	References (Zhang et al., 2026a)		K1+ Gemini (ours)		
Task/Methods or Arms	CaP-Agent0	+ RATs skills	Panda	UR5e	IIWA
Cube lifting	68.0	84.0	**100.0**	**100.0**	**100.0**
Cube restacking	34.0	46.0	**100.0**	**100.0**	**100.0**
Cube stacking	46.0	60.0	**100.0**	**100.0**	**100.0**
Nut assembly	0.0	0.0	**60.0**	55.0	45.0
Spill wiping	**100.0**	**100.0**	65.0	–	–
Two-arm handover	**24.0**	20.0	5.0	–	–
Two-arm lifting	10.0	34.0	**75.0**	–	–
Common four-task avg.	37.0	47.5	**90.0**	88.8	86.2
Seven-task avg.	40.3	49.1	**72.1**	–	–

not degrade significantly compared to in-domain performance, despite training on only 107 episodes. Further scaling of demonstration diversity is a natural next step and may yield additional gains.

4.4 Zero-shot transfer across environments and embodiments

K1 transfers across robot arms without fine-tuning. On the four shared tasks, K1 with Gemini achieves 90.0% on Panda, 88.8% on UR5e, and 86.2% on IIWA (Table 2). The small spread supports separating decisions from actuation: the agent reasons about measured geometry, while the adapter controls each arm. UR5e and IIWA retain the Panda gripper to isolate changes in arm kinematics. Cube tasks remain fully successful across arms, while nut assembly accounts for the accuracy differences. This pattern suggests that precise contact remains more sensitive to embodiment than coarse object relocation, despite sharing the same high-level decision interface.

Complex tasks expose coordination limits. Across the full Panda task set, K1 reaches 72.1%, including 65.0% on wiping and 75.0% on two-arm lifting, but only 5.0% on handover. Handover additionally requires coordinating release with a secure receiving grasp, beyond moving two arms to geometric targets. Official-base OpenVLA and $\pi_{0.5}$ achieve 0.0% under the tested transfer mappings (Appendix D).

K1 generalizes to a different simulator and dual-arm embodiment. All evaluated RoboTwin tasks use the AgileX dual-arm embodiment in SAPIEN, extending beyond single-arm LIBERO-PRO. Without RoboTwin fine-tuning, Gemini reaches 32.0% on Easy and 28.0% on Hard, a gap of only 4.0 points (Figure 5). Hard randomizes lighting, background textures, clutter, and table

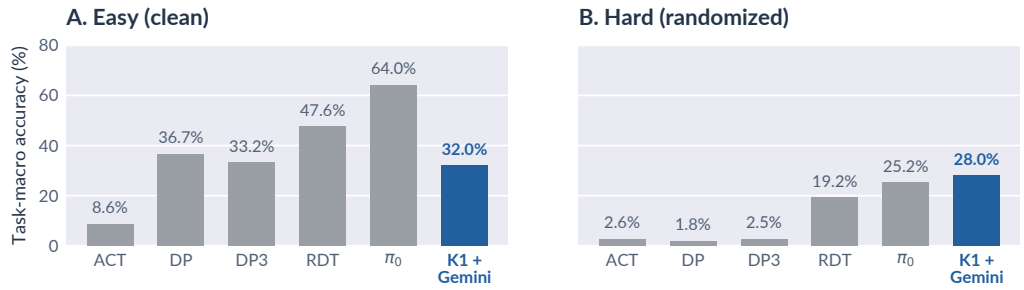


Figure 5: **RoboTwin zero-shot transfer.** Task-macro accuracy under Easy and Hard conditions. Gray: trained-policy references from RoboTwin (Chen et al., 2025b). Blue: K1 + Gemini without target-task fine-tuning. Per-task scores are in Appendix D.2.

height (Chen et al., 2025b). Under these perturbations, the published VLA references decline sharply: RDT drops from 47.6% to 19.2%, and π_0 from 64.0% to 25.2%. K1's smaller drop suggests a robustness advantage from separating semantic interpretation and metric measurement. Once a target is grounded, depth-backed relations provide spatial evidence beyond its changing RGB appearance, and the agent can remeasure after acting. This offers a plausible explanation for K1's competitive Hard accuracy despite lower Easy performance. Appendix D.2 details the transfer protocol.

4.5 Perceptual evidence in practice

The agent actively queries perception tools throughout execution. Across Spatial, Object, and Goal in LIBERO-PRO, 180 Gemini trajectories contain 6,282 model decisions. Motion-tool requests account for 67.5%, perception for 25.5%, and memory/progress for 3.9%. The remaining rounds check completion or reach the output limit (Appendix E.1). Thus the agent allocates substantial interaction to acquiring and maintaining evidence alongside execution. Decision notes accompany tool calls rather than forming separate reasoning-only steps.

Qualitative trace. In the paired trace (Appendix E.3), K1 grounds the bowl, inspects it in the wrist view, closes, checks a small lift, and measures bowl-to-plate alignment. Testing a grasp before transport connects each decision to observed physical progress. The matched RGB agent terminates without native success. This illustrates visual servoing through repeated measurement and action feedback.

Performance is robust across memory configurations. K1 maintains strong accuracy under varied text and image history lengths (Appendix E.2), confirming that the harness itself, rather than a specific context window, drives performance. Across settings, longer text history tends to help the agent maintain behavioral consistency over multi-step episodes, while extending image history beyond one frame can introduce redundant or stale visual context that adds noise. Our default configuration ($K{=}1, N{=}8$) balances these two trends.

5 Conclusion

We study how to fully unlock VLM competence for manipulation in robot-use agent (RUA). Robo-Harness K1 exposes perception as tools: through visual reference lines and persistent spatial anchors, the agent acquires calibrated 3D evidence, reasons about object relations in its existing image-and-text interface instead of guessing from RGB, and executes generic motions from the returned measurements. With the K1 harness, an ordinary Gemini Flash agent surpasses GPT-6 Astra on matched tasks, and K1 further augments Astra itself. For training RUAs, K1 provides better sample efficiency and generalization than VLAs, with a compact student outperforming action-policy baselines from only 107 episodes. These results suggest that training RUAs with a proper perceptual harness outperforms VLAs, unlocking VLMs' great potential for robotic manipulation. Modest held-out accuracy and the simulation-only setting bound the current conclusion. Perceptual evidence acquisition is a promising target for robot post-training, opening a new direction for policy learning that preserves the VLM's native training interface.

AI USE STATEMENT

Generative AI tools were used to assist with setting up and debugging the experimental environment, supporting routine experiment implementation and execution, organizing and formatting experimental results, conducting literature searches, and improving the clarity and presentation of the manuscript.

The core research ideas and methods, the design of the experimental methodology and protocols, and the interpretation, discussion, and validation of the experimental results were developed and performed by the authors without reliance on generative AI.

All AI-assisted code, configurations, text, literature suggestions, and organized results were reviewed and verified by the authors. References were checked against their original sources, and the authors take full responsibility for the final content, claims, results, and artifacts presented in this work.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as I can, not as I say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. URL https://arxiv.org/abs/2204.01691.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL https://arxiv.org/abs/2511.21631.

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

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, et al. SAM 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*, 2025. URL https://arxiv.org/abs/2511.16719.

Pingyi Chen, Yujing Lou, Shen Cao, Jinhui Guo, Lubin Fan, Yue Wu, Lin Yang, Lizhuang Ma, and Jieping Ye. SD-VLM: Spatial measuring and understanding with depth-encoded vision-language models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 33938–33968. Curran Associates, Inc., 2025a. doi: 10.52202/085713-1140. URL https://papers.neurips.cc/paper_files/paper/2025/hash/30bc3a3a44c9d2e3f32e6dd1cd18f552-Abstract-Conference.html.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Qiwei Liang, Zixuan Li, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025b. URL https://arxiv.org/abs/2506.18088v1.

Yanzhe Chen, Zechen Bai, Zhijun Cao, Wenzheng Zeng, Kevin Qinghong Lin, Yiqi Lin, Guoqiang Liang, Kevin Yuchen Ma, Qiming Huang, and Mike Zheng Shou. Show-Harness: Just a VLM agent can play robots. *arXiv preprint arXiv:2609.10522*, 2026. URL https://arxiv.org/abs/2609.10522.

Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z. Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, and Sergey Levine. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *arXiv preprint arXiv:2505.23705*, 2025. URL https://arxiv.org/abs/2505.23705.

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.

Asher J. Hancock, Xindi Wu, Lihan Zha, Olga Russakovsky, and Anirudha Majumdar. Actions as language: Fine-tuning VLMs into VLAs without catastrophic forgetting. *arXiv preprint arXiv:2509.22195*, 2025. URL https://arxiv.org/abs/2509.22195.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Hengyuan Hu, Priya Sundaresan, Jensen Gao, and Dorsa Sadigh. VIA: Visual interface agent for robot control. *arXiv preprint arXiv:2607.11119*, 2026. URL https://arxiv.org/abs/2607.11119.

Jinbang Huang, Yuanzhao Hu, Zhiyuan Li, Ran Qi, Yixin Xiao, Zhanguang Zhang, Mark Coates, Tongtong Cao, and Yingxue Zhang. RoboHarness: Memory-driven orchestration of heterogeneous robot policies for long-horizon planning. *arXiv preprint arXiv:2607.18060*, 2026. URL https://arxiv.org/abs/2607.18060.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3d value maps for robotic manipulation with language models. In *Conference on Robot Learning*, 2023a. URL https://arxiv.org/abs/2307.05973.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782. PMLR, 2023b. URL https://proceedings.mlr.press/v205/huang23c.html.

Phillip Isola. Robot-use agents. https://web.mit.edu/phillipi/www/writing/robot-use-agents.html, 2026. Blog post, September 7, 2026.

Sebastian Jung, Artem Zholus, Martin Sundermeyer, Carl Doersch, Ross Goroshin, David Joseph Tan, Sarath Chandar, Rudolph Triebel, and Federico Tombari. TAPNext++: What's next for tracking any point (TAP)? *arXiv preprint arXiv:2604.10582*, 2026. URL https://arxiv.org/abs/2604.10582.

Nikita Kachaev, Mikhail Kolosov, Daniil Zelezetsky, Alexey K. Kovalev, and Aleksandr I. Panov. Don't blind your VLA: Aligning visual representations for OOD generalization. *arXiv preprint arXiv:2510.25616*, 2025. URL https://arxiv.org/abs/2510.25616.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as Policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023. URL https://arxiv.org/abs/2209.07753.

Haowen Liu, Xirui Li, Shaoxiong Yao, Peng Shi, Tianyi Zhou, Jia-Bin Huang, Furong Huang, and Jiayuan Mao. Guava: An effective and universal harness for embodied manipulation. *arXiv preprint arXiv:2606.18363*, 2026. URL https://arxiv.org/abs/2606.18363.

Adithyavairavan Murali, Balakumar Sundaralingam, Yu-Wei Chao, Wentao Yuan, Jun Yamada, Mark Carlson, Fabio Ramos, Stan Birchfield, Dieter Fox, and Clemens Eppner. GraspGen: A diffusion-based framework for 6-dof grasping with on-generator training. *arXiv preprint arXiv:2507.13097*, 2025. URL https://arxiv.org/abs/2507.13097.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

Qwen Team. Qwen3.5-9B model card, 2026. URL https://huggingface.co/Qwen/Qwen3.5-9B.

Robocurve. Inspect robots: Open-source evaluation framework for physical AI, 2026a. URL https://github.com/robocurve/inspect-robots. Evaluated agent source pinned to commit 7e4d1b7aee1c0d3cfc3a05a7492b9d12cda666f9.

Robocurve. GPT-6 Astra on robot arms, 2026b. URL https://openai.robocurve.org/gpt-6-astra/.

Xiaoquan Sun, Zetian Xu, Chen Cao, Zonghe Liu, Yihan Sun, Jingrui Pang, Ruijian Zhang, Zhen Yang, Kang Pang, Dingxin He, Mingqi Yuan, and Jiayu Chen. AtomVLA: Scalable post-training for robotic manipulation via predictive latent world models. *arXiv preprint arXiv:2603.08519*, 2026. URL https://arxiv.org/abs/2603.08519v1.

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

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

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. Libero-pro: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025. URL https://arxiv.org/abs/2510.03827.

Appendix

Appendix Contents

A Method and Tool Details **15**
A.1 Tool families and contracts . . . 15
A.2 Segmentation, tracking, and geometry . . . 16
A.3 Grasp proposal and ranking . . . 16
A.4 Control and feedback . . . 16
A.5 Perception-tool illustrations . . . 16

B Training and Baselines **18**
B.1 Data construction and filtering . . . 18
B.2 Architectures and optimization . . . 18
B.3 RGB-to-action baseline . . . 19
B.4 RGB baseline and paired interface comparison . . . 19

C LIBERO-PRO Evaluation Details **20**
C.1 Benchmark protocol and data splits . . . 20
C.2 Execution and scoring . . . 21
C.3 Frontier summary and paired comparison . . . 21
C.4 Full checkpoint results . . . 21

D Transfer Experiments **23**
D.1 RoboSuite: cross-embodiment transfer . . . 23
D.2 RoboTwin: bimanual transfer . . . 25

E Additional Analysis **26**
E.1 Tool-use distribution . . . 26
E.2 Memory-context ablation . . . 27
E.3 Qualitative interaction traces . . . 28

List of Tables

1 LIBERO-PRO task accuracy . . . 7
2 RoboSuite zero-shot transfer accuracy . . . 8
3 Perception and action tool contracts . . . 15
4 Post-training configurations . . . 18
5 Interfaces in the paired GPT comparison . . . 20
6 Training and held-out task inventory . . . 21
7 Paired system comparison on matched cases . . . 22
8 Full checkpoint evaluation results . . . 22
9 Epoch-5 accuracy by task category . . . 22
10 Complete RoboSuite transfer results . . . 24
11 K1 + Gemini per-task results on RoboTwin . . . 25
12 Published RoboTwin baseline accuracies . . . 26
13 Individual tool-request counts . . . 27
14 Memory-context ablation results . . . 28
15 Selected tool calls from the K1 trajectory . . . 29
16 Selected calls from the RoboSuite trajectory . . . 30
17 Selected calls from the RoboTwin trajectory . . . 31

List of Figures

1 Frontier model comparison on LIBERO-PRO . . . 1
2 Robo-Harness K1 framework . . . 3
3 Four perceptual tool families . . . 5
4 Task accuracy across training epochs . . . 8
5 RoboTwin zero-shot transfer . . . 9
6 Perception-tool visual atlas . . . 17
7 Gemini model-decision distribution . . . 26
8 Image and text history ablation . . . 27
9 Recorded paired-case trajectory . . . 29
10 RoboSuite UR5e nut-assembly trajectory . . . 30
11 RoboTwin Hard dual-arm roller trajectory . . . 31

A METHOD AND TOOL DETAILS

A.1 TOOL FAMILIES AND CONTRACTS

Table 3 summarizes the public tool families. Each image measurement specifies camera and frame, and pixel arguments explicitly declare their coordinate convention. Inspection labels are user-proposed names, not independently verified object identities.

Table 3: Tool families in the evaluated harness. Measurements and tool execution do not certify task success.

Tool	Contract and boundary
`find_regions`	Text-prompted SAM 3 masks from a current original RGB image, with RGB-D summaries and stable S identifiers. Multiple candidates require model-side disambiguation.
`inspect_region`	Box-prompted segmentation or explicit refresh of a region. The supplied label does not guide semantic segmentation. Returns visible geometry, not a full-object pose.
`measure_depth`	Back-projects a selected valid visible pixel to world coordinates, with TCP-relative displacement. Creates a trackable point reference, not a grasp target.
`fit_geometry`	Fits local visible geometry in an image region. A fitted plane or unsigned principal direction is not an articulation model.
`region_relation`	Measures region-to-robot or region-to-region displacement and available plane relations.
`check_region_view`	Projects an existing measured region into the other available camera, with a depth consistency check. This is not a newly rendered viewpoint.
`track_features`, `read_track`	Starts and queries image correspondences. Visibility estimates are not proof of persistent object identity. Point/region measurement tools also maintain their own anchors.
`inspect_crop`	Enlarges an existing RGB image region. Does not move a camera or reveal an occluded surface.
`grasp_candidates`	Proposes nominal pad-aligned TCP poses, quaternions, and pregrasp points, with image schematics. Does not execute a grasp or certify feasibility.
`move_relative`	Executes a supplied relative translation in world or gripper coordinates. Each requested component is bounded by δ_{axis}. Optional rotation is a bounded world rotation vector.
`move_to_pose`	Creates or resumes an absolute TCP pose target. Executes one bounded translation/rotation chunk and preserves the gripper command. Explicit new coordinates supersede an old target identifier.
`rotate_toward`, `align_axis`	Moves toward an absolute quaternion or aligns a calibrated gripper axis to a requested direction. Rotation is at most 15° per call, with positional anchoring.
`move_toward`	Sends a full absolute position to native control within a single call. Stops on tolerance, stall, or execution budget. Not a collision-free path planner.
`set_gripper`	Uses 0 for closed and 1 for open while holding pose. Reports settling and measured gap, not attachment.
`retreat`	Takes a bounded step toward an earlier observed pose. Does not assume the old path is still free.
`align_region_references`	Computes a translation hypothesis aligning source/destination visible references, with a model-chosen offset and axes. Relies on current measurement and a stable carry relation.
`search_history`, `read_history`	Searches completed episode calls and retrieves paginated evidence, optionally including archived images. Historical frames remain explicitly labeled.
`update_task_progress`, `remember`	Saves model-authored milestones, uncertainties, or working notes. Never replaces the original task or certifies completion.
`done`	Requests sparse native completion feedback in K1. Failure keeps the episode active within the original remaining budgets.

A.2 SEGMENTATION, TRACKING, AND GEOMETRY

Text search returns at most four confident, nonredundant candidates per query. Older region identities remain stored beyond this per-query limit. The implementation filters small masks and duplicates, removes uncertain depth boundaries, and forms a partial visible point cloud. Camera extrinsics include the current wrist pose. Regions can be refreshed using tracked anchors and current depth, while lost correspondence requires reinspection.

The configured tracker is TAPNext++. World displacement combines image correspondence, current depth, and current camera pose. A track can fail from correspondence drift or from a depth-surface change at the tracked pixel. Stored memory and live validity are separate: a persistent identifier does not guarantee that its old coordinates remain accurate.

A.3 GRASP PROPOSAL AND RANKING

The grasp provider is a GraspGen Franka Panda checkpoint that receives only the observed region cloud. The adapter requests 100 grasp samples and up to 24 candidates, validates their transforms, and maps approach/closing/lateral axes into the calibrated robot frame. The TCP is shifted so that the nominal contact reference aligns with the finger-pad midpoint. Parallel-jaw symmetry selects the equivalent orientation requiring less rotation from the current pose.

For each grasp target p_g with approach direction a_g, the pregrasp point is $p_{\text{pre}} = p_g - 0.07a_g$. The scorer evaluates five interpolated positions along the approach path:

$$p(\lambda) = p_{\text{pre}} + \lambda(p_g - p_{\text{pre}}), \qquad \lambda \in \{0, 0.25, 0.5, 0.75, 1\}. \tag{7}$$

At each position, calibrated sparse gripper probes are projected into existing depth views, yielding free-space, near-surface, and unknown fractions. The internal visibility score is

$$s_{\text{vis}} = f_{\text{free}} - 2f_{\text{near}} - 0.25f_{\text{unknown}}. \tag{8}$$

This heuristic ranks visible-space compatibility. Near-duplicate hypotheses within 1.5 cm and 20° are suppressed, and at most three candidates are presented. The agent sees candidate geometry and opaque identifiers. Backend scores, fractions, and rank labels are hidden from the agent but still determine internal ordering.

A.4 CONTROL AND FEEDBACK

Small-step tools execute up to twelve native servo steps per call. Rotation targets retain a fixed positional anchor, and persistent pose targets preserve the originally requested orientation across calls. These mechanisms reduce local undertracking and accumulated drift without global path planning.

Full-target `move_toward` holds the call's initial orientation and stops on positional tolerance (3 mm), orientation tolerance (2°), stall detection, or a 120-step budget. An optional gripper command applies from the start. The tool reports why it stopped.

The agent-facing feedback reducer removes internal diagnostics, recovery hints, and private grasp scores, retaining actual motion and residuals. Repeated same-frame perception triggers a reminder after three stationary transitions, but all tools remain available.

A.5 PERCEPTION-TOOL ILLUSTRATIONS

Figure 6 expands the four examples in Figure 3 into a complete tool atlas. The main-text panels a–d correspond to atlas panels a, f, h, and l, respectively. All panels use archived JPEG inputs actually presented to the agent, not reconstructed images. Crops do not define new measurement coordinates.

Visual encoding. Green S contours denote segmentation estimates. Orange principal-axis lines describe visible geometry. Purple lines connect TCP and surface references. The gripper's A/C/L axes denote approach, closing, and lateral directions. P labels mark measured or tracked surface points, with a tilde denoting tracking estimates. In panels g–i, the same anchor P1 changes pixel location after the wrist moves while maintaining sub-millimeter estimated world displacement, until the tracker reports the anchor as lost.

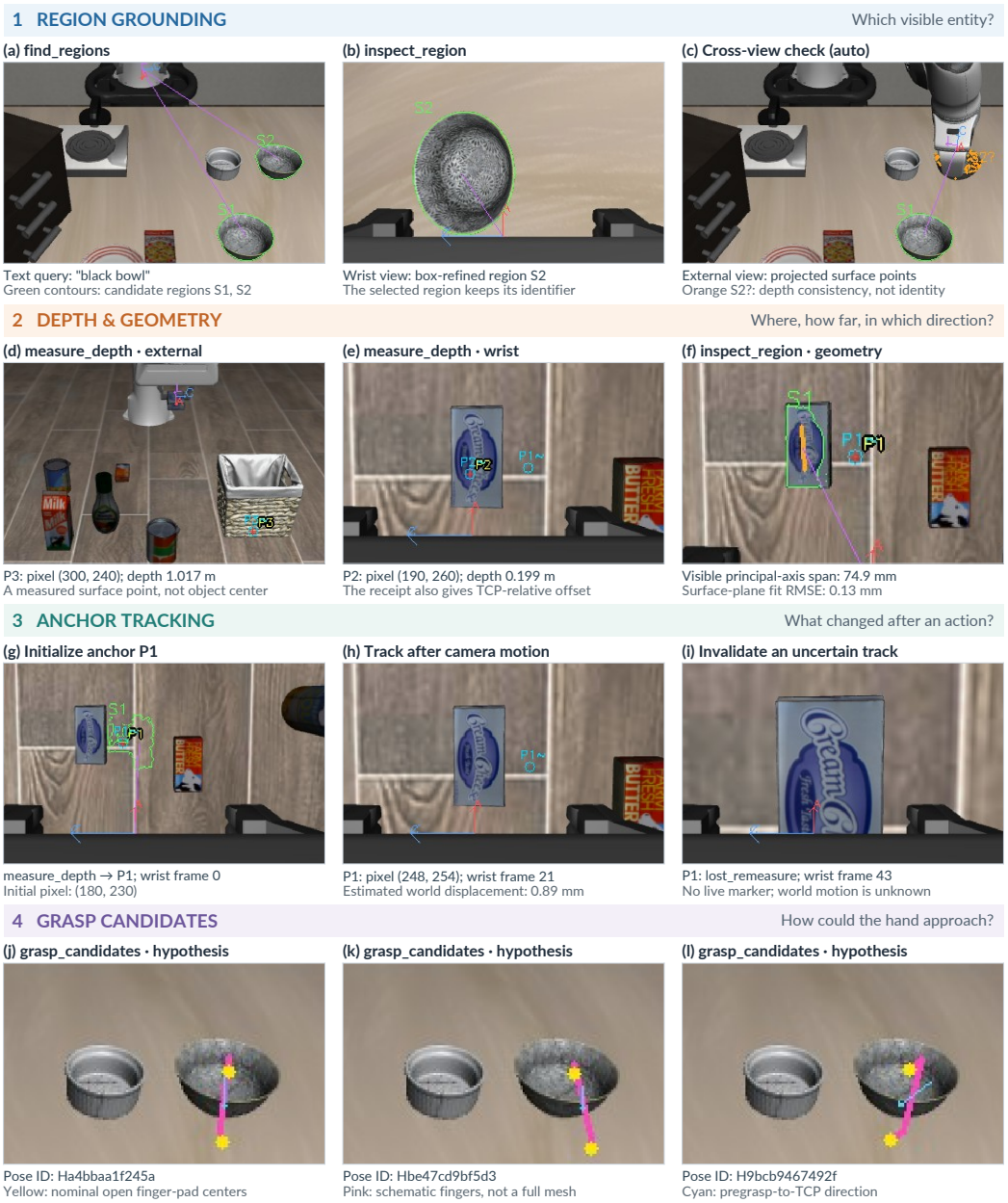


Figure 6: **Perception as tools, illustrated by recorded model-visible feedback.** Each row corresponds to one tool family. (a–c) Text grounding, box refinement, and depth-consistent projection into the other existing view. (d–f) Metric points and visible-surface geometry, with selected receipt values below the images. (g–i) The same wrist-camera anchor changes pixel location but has only 0.89 mm estimated world displacement, before becoming invalid. These are automatic updates, not additional tracking calls. (j–l) Three grasp hypotheses from one tool response, shown as enlarged crops rather than executed hand poses. Green contours denote regions; orange lines, principal axes; purple lines, TCP–surface references. No views or scene geometry are synthesized.

Panel f reports the region's visible principal-axis span and surface-plane residual from the same `inspect_region` call. Panels c and g–i illustrate automatic cross-view and anchor maintenance, not explicit tracking calls. The grasp panels show nominal open finger-pad centers (yellow), simplified finger lines (pink), and pregrasp-to-TCP directions (cyan). These are hypotheses from one response, not executed grasps.

B Training and Baselines

B.1 Data construction and filtering

The training pool contains replay-verified native successes. Of 139 successful Gemini episodes, 32 belong to withheld conditions, leaving 107 episodes from 43 conditions. Acceptance uses native success rather than the additional release diagnostic. The native-action dataset contains 40,808 control frames. Eight-step windows sampled at stride four give 10,241 training windows for chunk-prediction models, which execute four actions before observing again. Original OpenVLA trains on every native frame and executes one action at a time.

The tool dataset uses original pre-action requests, tool schemas, and model-visible images. It excludes twelve execution-error targets, sixteen responses with missing or multiple calls, and 54 rejected completions, retaining 2,482 targets (96.8% of 2,564 decisions). Supervision includes executable calls and public short decision notes. Private provider reasoning and signatures are removed. The `<THINK>` delimiters are ordinary text rather than added tokenizer special tokens.

Filtering a target does not rewrite history. A later successful correction may contain an earlier rejected `done` in its input because that rejection was available to the teacher. This preserves the causal input boundary. Depth and action supervision are regenerated through verified native replay.

B.2 Architectures and optimization

Table 4: Adaptation recipe. All arms use seed 17, batch size 1 with accumulation 8, and save checkpoints at epochs 1/3/5.

Arm	Trainable components	Learning rate	Examples/epoch
Qwen K1 RUA	Language LoRA $r = 16$, $\alpha = 32$, dropout .05	10^{-4}	2,482
Qwen RUA	Same language LoRA, native text output head	10^{-4}	1,670
Qwen VLA (RGB-only / RGB-D)	Same LoRA plus numeric-state/action head, optional depth encoder	10^{-4}	10,241
$\pi_{0.5}$	Language LoRA $r = 16$, $\alpha = 32$, plus full action expert/projections	2.5×10^{-5}	10,241
OpenVLA	All-linear LoRA $r = 32$, $\alpha = 16$, dropout .05	5×10^{-4}	40,808

Qwen K1 RUA. The base is Qwen3.5-9B, with 43,278,336 trainable language-adapter parameters and a frozen vision encoder. Cross-entropy loss applies only to assistant target tokens, using chat-template prefix matching to locate the input/target boundary. Training rejects contexts beyond the model's supported capacity. Inference uses deterministic decoding, a 1,024-token generation cap, and the same harness observations as the teacher.

Qwen VLA. The last text-template hidden state is concatenated with a trainable eight-dimensional proprioception embedding and a 256-dimensional depth feature (zeros for RGB-only). A two-layer bounded multilayer perceptron predicts 8×7 normalized native actions. The loss is masked absolute error over valid action frames. The numeric state encoder is $8 \to 128 \to 256$ and contains measured XYZ, quaternion, and jaw opening. The RGB-D encoder receives current/previous depth and validity channels, resized to 32×32. These are deliberately simple shared-backbone baselines, not a reproduction of a mature pretrained VLA action architecture.

Original OpenVLA. We use original OpenVLA-7B weights rather than a benchmark-fine-tuned checkpoint. Its input is one current external RGB image and the task. The action tokenizer discretizes

normalized components into 256 bins and predicts seven action tokens plus an end token. Statistics are fitted only on the 107 training trajectories. Translation/rotation use the training quantiles, and gripper openness is explicitly converted to the simulator's open/close convention. At inference, one native action executes before the next observation. All-linear LoRA includes vision-side linear layers, yielding 110,828,288 trainable parameters.

$\pi_{0.5}$. The base is the official general $\pi_{0.5}$ checkpoint, not a LIBERO-specific fine-tune. The implementation follows the OpenPI codebase. It retains the official flow-matching architecture and image augmentation, with language LoRA and full training of the action expert and projections. There are 449,710,112 trainable parameters. It receives current external/wrist RGB and discretized robot state in the text prompt. Native seven-dimensional actions are padded to the architecture's 32-dimensional representation, with loss applied only to valid physical dimensions and timesteps. The eight-step horizon is sampled with ten flow steps, and four actions execute before replanning. Normalization statistics come only from the training pool.

Implementation checks. Forward/backward tests verify finite gradients and loss propagation to trainable adapters. Checkpoint round trips preserve parameter identity. Native replays validate action-frame alignment and gripper conventions.

Training took approximately 26.6 hours (Qwen agent), 5.7 hours ($\pi_{0.5}$), and 19.1 hours (OpenVLA). Runs share resources but differ in sequence length, trained modules, and update counts.

B.3 RGB-TO-ACTION BASELINE

This baseline starts from the same Qwen3.5-9B base checkpoint (not the trained agentic adapter) and receives only the task instruction and one current external RGB image. No wrist view, proprioception, tool history, depth, overlay, or teacher reasoning is provided.

We retain all 1,670 teacher decisions that produce physical execution: local motion, target-transit, and gripper commands. Labels encode the commanded action at each boundary, not the observed displacement after contact. The output is a JSON action with fields for mode, translation, rotation, and gripper state. Three modes are supported: `step` (bounded local motion), `target` (full-target transit), and `gripper` (open/close). The executor reads robot state internally but does not expose it to the VLM.

Training uses the same language LoRA settings as the agentic arm, with frozen vision encoder and cross-entropy loss on assistant action tokens only. Five epochs produce 1,045 optimizer updates. Inference is greedy with no constrained decoding. Invalid actions terminate as policy failures.

All epochs 1/3/5 are evaluated on the same A43/B43/C36 manifest. Epoch 5 produces 74 budget terminations, 43 invalid actions, and five native successes. Format robustness is a material part of this baseline's failure, especially at epoch 5, and should not be conflated with purely geometric failure. The complete counts appear in Table 8.

B.4 RGB BASELINE AND PAIRED INTERFACE COMPARISON

We reuse the Inspect Robots agent and tool implementation (Robocurve, 2026a). The LIBERO adapter supplies external and wrist RGB, measured end-effector position/orientation, and normalized gripper opening, without depth measurements.

Its tools are `move_to`, `done`, and `give_up`. The agent specifies world XYZ and optional roll/pitch/yaw relative to the reset orientation. The adapter converts this to the native rotational command using

$$R_{\text{target}} = R_z(\text{yaw})R_y(\text{pitch})R_x(\text{roll})R_{\text{reset}}. \tag{9}$$

The tool interpolates the requested pose into smaller targets. After the sequence, the adapter holds the terminal target for twelve native frames to accommodate the synchronous simulator.

The paired systems share task, initial state, camera placement, image resolution, native horizon, maximum calls, and evaluator. Table 5 lists the differences in perception, memory, motion, and completion feedback. The comparison evaluates K1 as a complete interface. A matched Gemini RGB-only evaluation is not included.

Table 5: Interface differences in the same-GPT comparison.

Property	Original RGB interface	K1
Scene information	Two RGB views	Same two views plus depth-backed measurements
Text history	Original complete history	Last eight exchanges plus retrieval
Image history	Current and previous decision	Current and previous decision, optional retrieved images
Motion	Original interpolated pose tool	Local corrections, persistent poses, full-target transit
Perception	Model interprets RGB	Explicit region, depth, tracking, grasp tools
Progress memory	Dialogue	Dialogue plus model-maintained milestones
Completion	`done` ends episode	`done` returns sparse native check and permits correction

C LIBERO-PRO Evaluation Details

C.1 Benchmark protocol and data splits

LIBERO evaluation and student training use a fixed harness configuration. Transfer retains the perception-and-action interface with environment-specific sensing and control adapters. Episode records and a companion CSV specify the 122 student evaluation cases.

The 180-episode teacher evaluation and both 18-episode GPT evaluations use fixed task manifests, an unchanged harness, and replay-verified native success. Benchmark conditions overlap with harness development. Student post-training holds out twelve task conditions from fine-tuning.

The full evaluation comprises ten tasks in each of Spatial/Object/Goal, two variants (swap and task), and three initial states, giving 180 trials and 30 per reported column. The paired GPT comparison uses task indices 0–2 and state 0 in all six suites, giving 18 trials and six per category. Gemini's paired result is taken from the full evaluation. Published LIBERO-PRO rows in RATs Table 1 instead use 100 trials per column and 600 overall. They are external references, not reruns.

The MolmoAct, NORA, X-VLA, and AtomVLA rows in Table 1 come from Sun et al. (2026), Table IV. We convert its success fractions to percentages and select only Position and Task perturbations for Object, Goal, and Spatial. Their averages are recomputed over these six columns, excluding LIBERO-10 and the Obj/Sem perturbations; they are not the source paper's full-benchmark averages. These rows are published references, not matched reruns.

The Gemini full evaluation yields 139/180 (77.2%) native successes, with category totals of 52/60 Object, 38/60 Goal, and 49/60 Spatial. The paired subset yields 11/18 for GPT RGB, 14/18 for Gemini K1, and 16/18 for GPT K1.

The LIBERO-PRO benchmark contains 50 stored initial states per task. We use explicit state-file indices, not an assumption that an arbitrary pseudorandom seed generates the same scene. The teacher evaluation uses 0, 1, and 2. Student B uses 3. Student C uses 40, 41, and 42.

Student split inventory. Table 6 lists all fine-tuning-covered and held-out conditions by suite. A task condition includes the suite variant: the same task index in `swap` and `task` is not assumed to identify the same instruction. A and B each contain 15 Spatial, 16 Object, and 12 Goal conditions. C contains four per category, evaluated at three states each.

A uses a successful training demonstration for each condition, prioritizing state 0: 33 cases use state 0, nine use state 1, and one uses state 2. Five other conditions have no successful teacher demonstration and are excluded from A/B/C: Goal swap 3 and 5, Goal task 0 and 1, and Spatial swap 4.

C tests new task conditions, including new object relations within shared scenes and manipulation skills. Both swap and task perturbations are represented in fine-tuning.

Table 6: Task-index inventory. A uses one actually trained initial state for each listed condition. B uses the same condition at state 3. C uses all three states 40–42.

Suite	Trained condition indices (A/B)	Held-out (C)
goal_swap	0, 1, 4, 7, 8, 9	2, 6
goal_task	3, 4, 5, 7, 8, 9	2, 6
object_swap	2, 3, 4, 5, 6, 7, 8, 9	0, 1
object_task	2, 3, 4, 5, 6, 7, 8, 9	0, 1
spatial_swap	1, 2, 5, 6, 7, 8, 9	0, 3
spatial_task	1, 2, 3, 4, 6, 7, 8, 9	0, 5

C.2 Execution and scoring

Every local case has 1,200 native control steps at 20 Hz. Agentic cases additionally have a 400-call limit. All controller holds, gripper settling, and transit steps consume this budget. Observation, tool computation, and API waiting do not advance simulation time.

Native simulator success is the primary metric. The evaluator replays recorded commands from the same official initial state, verifies maximum TCP deviation no greater than 10^{-6} m, and checks the final success outcome. All completed policy outcomes are retained; infrastructure failures are logged separately and retried.

The release diagnostic holds the arm and opens the gripper for 20 native frames after replay, requiring success on the last ten frames. For Gemini, 128 native successes remain successful, eleven lose success, and three native failures become successful, yielding 131/180 (72.8%) post-release accuracy versus 139/180 (77.2%) native accuracy. Both GPT groups retain their native successes. We report this separately because release can change task completion in either direction.

Student evaluation uses shared GPU inference with parallel simulator workers. We report accuracy and call counts rather than wall-clock latency.

C.3 Frontier summary and paired comparison

Figure 1 combines the six nonzero published baseline averages in Table 1 with three local matched-subset results. RATs, CaP-Agent0, and $\pi_{0.5}$ reach 43.8%, 23.2%, and 12.8%, respectively, using RATs (Zhang et al., 2026a), Table 1. AtomVLA, X-VLA, and MolmoAct reach 6.7%, 3.3%, and 1.0%, using Sun et al. (2026), Table IV. All six averages use the displayed Object, Goal, and Spatial position/task conditions; zero-average baselines are omitted from the figure. The three remaining bars use the same 18 task–variant–state configurations: GPT-6 Astra with the RGB-only Inspect Robots interface achieves 11/18 (61.1%), Gemini with K1 achieves 14/18 (77.8%), and GPT-6 Astra with K1 achieves 16/18 (88.9%). Table 7 retains the category-level breakdown.

A budget-limited, category-balanced subset. API budget constraints limit Astra evaluation to task indices 0, 1, and 2 at initial state 0 in every category–variant combination. Spatial, Object, and Goal each contribute six cases, and swap and task perturbations each contribute nine. This deterministic subset gives the six strata equal weight, matching the full Gemini evaluation's category balance.

Agreement with the full evaluation. Gemini with K1 achieves 139/180 (77.2%) on the complete evaluation and 14/18 (77.8%) on the matched subset, a difference of 0.56 percentage points before rounding. This agreement provides a useful consistency check for the subset. Because it is contained in the full evaluation, the two estimates are dependent. Descriptive 95% Wilson intervals are 70.6–82.7% and 54.8–91.0%, respectively, under a trial-independence approximation.

Comparison scope. The three local systems share task identities and initial states. Published baselines use their respective protocols and trial counts. Paired Astra outcomes and the small-sample significance test are reported in Section C.4.

C.4 Full checkpoint results

Table 7: **Paired system comparison: task accuracy (%).** Tasks and initial states are matched. Each system is evaluated on six cases per category, eighteen in total. The GPT comparison changes perception, memory, control interface, and completion feedback together.

System	Spatial	Object	Goal	Average
RGB + GPT-6 Astra	33.3	83.3	66.7	61.1
K1 + Gemini 3.7 Flash	66.7	**100.0**	66.7	77.8
K1 + GPT-6 Astra	**83.3**	**100.0**	**83.3**	**88.9**

Table 8: Full checkpoint ledger. Incomplete groups are marked *, and unrun groups are –.

Model	Epoch	A	B	C
Qwen K1 RUA	1	7/43 (16.3%)	10/43 (23.3%)	3/36 (8.3%)
Qwen K1 RUA	3	14/43 (32.6%)	**20/43 (46.5%)**	**5/36 (13.9%)**
Qwen K1 RUA	5	**22/43 (51.2%)**	19/43 (44.2%)	**5/36 (13.9%)**
Qwen RUA	1	0/43 (0.0%)	0/43 (0.0%)	0/36 (0.0%)
Qwen RUA	3	0/43 (0.0%)	0/43 (0.0%)	0/36 (0.0%)
Qwen RUA	5	3/43 (7.0%)	2/43 (4.7%)	0/36 (0.0%)
OpenVLA-7B	1	5/43 (11.6%)	3/43 (7.0%)	0/36 (0.0%)
OpenVLA-7B	3	6/43 (14.0%)	4/43 (9.3%)	0/36 (0.0%)
OpenVLA-7B	5	9/43 (20.9%)	13/43 (30.2%)	0/36 (0.0%)
$\pi_{0.5}$	1	2/43 (4.7%)	1/43 (2.3%)	0/36 (0.0%)
$\pi_{0.5}$	3	1/43 (2.3%)	2/43 (4.7%)	0/36 (0.0%)
$\pi_{0.5}$	5	3/43 (7.0%)	3/43 (7.0%)	0/36 (0.0%)
Qwen VLA (RGB-only)	1	0/6 (0.0%)*	0/43 (0.0%)	–
Qwen VLA (RGB-only)	3	0/43 (0.0%)	0/43 (0.0%)	0/35 (0.0%)*
Qwen VLA (RGB-only)	5	0/39 (0.0%)*	0/43 (0.0%)	–
Qwen VLA (RGB-D)	1	–	–	–
Qwen VLA (RGB-D)	3	–	–	–
Qwen VLA (RGB-D)	5	0/43 (0.0%)	0/43 (0.0%)	0/36 (0.0%)

We evaluate Qwen VLA under both RGB-only and RGB-D inputs to test whether depth access improves continuous action regression. RGB-only evaluation stopped after consistently unsuccessful results, leaving the marked A/C groups incomplete. Unrun cases are excluded from denominators. Qwen VLA (RGB-D) at epoch 5 completes all 122 cases with zero success, confirming that neither input variant produces a viable policy under our recipe. In Figure 4, Qwen K1 RUA, Qwen RUA, OpenVLA, and $\pi_{0.5}$ have complete epoch 1/3/5 evaluations. The main text reports Qwen VLA using the RGB-D variant at epoch 5.

Table 9: Epoch-5 success by task category. Denominators: 15/16/12 for A and B; 12 per category for C.

Model	Group	Spatial	Object	Goal
Qwen K1 RUA	A	**6/15 (40.0%)**	**11/16 (68.8%)**	**5/12 (41.7%)**
Qwen K1 RUA	B	**7/15 (46.7%)**	7/16 (43.8%)	**5/12 (41.7%)**
Qwen K1 RUA	C	**3/12 (25.0%)**	**1/12 (8.3%)**	**1/12 (8.3%)**
Qwen RUA	A	0/15 (0.0%)	1/16 (6.2%)	2/12 (16.7%)
Qwen RUA	B	0/15 (0.0%)	1/16 (6.2%)	1/12 (8.3%)
Qwen RUA	C	0/12 (0.0%)	0/12 (0.0%)	0/12 (0.0%)
OpenVLA-7B	A	0/15 (0.0%)	7/16 (43.8%)	2/12 (16.7%)
OpenVLA-7B	B	1/15 (6.7%)	**10/16 (62.5%)**	2/12 (16.7%)
OpenVLA-7B	C	0/12 (0.0%)	0/12 (0.0%)	0/12 (0.0%)
$\pi_{0.5}$	A	0/15 (0.0%)	2/16 (12.5%)	1/12 (8.3%)
$\pi_{0.5}$	B	1/15 (6.7%)	1/16 (6.2%)	1/12 (8.3%)
$\pi_{0.5}$	C	0/12 (0.0%)	0/12 (0.0%)	0/12 (0.0%)

The epoch-5 agent's five C successes cover four task conditions: Goal task 6/state 40, Object swap 0/state 42, Spatial swap 0/states 40 and 41, and Spatial task 5/state 42.

The paired GPT result has six K1-only successes and one RGB-only success. Both solve ten cases and both fail one. The exact two-sided McNemar test gives $2\sum_{i=0}^{1}\binom{7}{i}2^{-7} = 0.125$. The observed improvement does not reach the 0.05 significance threshold in this small paired evaluation.

D TRANSFER EXPERIMENTS

D.1 ROBOSUITE: CROSS-EMBODIMENT TRANSFER

Setup. The RoboSuite adapter retains the K1 runtime and frozen Gemini 3.7 Flash model. The adapter handles sensing and control without target-task demonstrations or fine-tuning.

We reuse RATs/CaP task construction and native success checks for cube lifting, restacking, stacking, square-nut assembly, spill wiping, two-arm handover, and two-arm lifting. Panda covers all seven. UR5e and IIWA cover the first four. All non-wiping arms use a PandaGripper, including UR5e and IIWA, retaining the gripper morphology used by the grasp provider. Wiping uses the task's attached sponge, without a grasp/open-close interface. Panda reset orientations follow the task wrappers. Replacement arms use their declared joint indices rather than Panda-specific addresses. Both LIBERO and these tasks use MuJoCo/robosuite components, so cross-environment does not mean cross-physics-engine transfer.

UR5e and IIWA wiping and dual-arm configurations were outside the validated adaptation scope, not found infeasible. Extending coverage requires checking dual-arm base placement, reset poses, shared reachability, and controller/camera configuration; wiping additionally requires validating sponge attachment and contact alignment. These combinations are therefore marked untested rather than failed in Table 2.

The adapter supplies calibrated 384×384 RGB-D from the original task scene camera and native wrist camera(s). The dual-arm setup uses one scene view and two wrist views, rather than the two-view single-arm interface. Its scene-camera pose is explicitly configured by the adapter. No task-object poses, instance segmentation masks, or shaped rewards are supplied to the agent. Robot kinematics and robot-owned collision geometry are available for embodiment calibration. The completion request retains sparse Boolean native feedback.

Cartesian commands execute in world-frame operational-space control. The generic `move_effectors` interface accepts poses for multiple arms and advances them simultaneously for up to 120 native steps. Unspecified arms hold position, and each arm returns its execution residual. The wiping configuration hides tools that require an actuated gripper.

Trials and budgets. Each method/task/embodiment combination uses reset seeds 271828–271832 and four rollouts per configuration, giving 20 trials. Sampling seeds are $1729 + 4c + r$, where c indexes configurations and r indexes repeats. Reset states are shared across methods, and all completed policy outcomes are included. Repeats share the same initial scene.

All methods use a 20 Hz native controller. Cube and nut tasks have 1,500 native steps, wiping 4,000, and dual-arm tasks 5,000. Gemini has at most 400 decisions, text history $N = 8$, image history $K = 1$, and local translation bound .03 m per axis. These budgets differ from the LIBERO learning evaluation and from RATs. Perception and model waiting do not advance physics.

The evaluation includes 740 replay-verified trials: Gemini 300, OpenVLA 260, and $\pi_{0.5}$ 180. The planned 80 $\pi_{0.5}$–IIWA trials are unevaluated because a compatible native action mapping is unavailable. Replay verification requires matching reset observations and native success, unchanged commands, and TCP error below 10^{-6} m. Infrastructure failures are excluded from policy scoring.

VLA baseline mappings. OpenVLA uses official base weights with the fixed `bridge_orig` unnormalization statistics. It sees the current scene RGB only. Source metric Cartesian translations and Euler increments are mapped to world-frame OSC. One action holds for four native steps, giving a 5 Hz decision rate. No target-embodiment retargeting or action statistics tuning is applied.

The official $\pi_{0.5}$ base uses a 50-action horizon with ten flow steps and executes a ten-step prefix at 20 Hz. Inputs are scene/wrist RGB and measured joints plus gripper state. Official Franka or UR5e normalization assets map delta-joint outputs to absolute targets. IIWA requires a separate joint-space

mapping and is unevaluated. Both base VLAs use single-arm tasks only. Figure 4 instead evaluates LIBERO-fine-tuned models.

Results. RATs Table 3 reports 50 trials per RoboSuite task: CaP-Agent0 has 34, 17, 23, 0, 50, 12, and 5 successes. Adding RATs skills gives 42, 23, 30, 0, 50, 10, and 17. These total 141/350 and 172/350 in Table 2 order. The published reference uses ten trials per randomization versus our four, and differs in reset states, cameras, models, controllers, instructions, and budgets.

The common four-task mean uses cube lifting, restacking, stacking, and nut assembly only. K1 totals are Panda 72/80, UR5e 71/80, and IIWA 69/80. The seven-task Panda total is 101/140, including wiping 13/20, handover 1/20, and two-arm lifting 15/20. An overall rate across different coverage sets should not be used as a robot-to-robot comparison. Table 10 provides every tested combination.

Table 10: Complete transfer ledger. Each tested task has five reset configurations and four rollouts per configuration. Official-base VLA rows are not the LIBERO-fine-tuned models in Figure 4.

Method	Robot	Task	Successes / trials	Accuracy (%)
K1 + Gemini	Panda	Cube lifting	20/20	100.0
K1 + Gemini	Panda	Cube restacking	20/20	100.0
K1 + Gemini	Panda	Cube stacking	20/20	100.0
K1 + Gemini	Panda	Nut assembly	12/20	60.0
K1 + Gemini	Panda	Spill wiping	13/20	65.0
K1 + Gemini	Panda	Two-arm handover	1/20	5.0
K1 + Gemini	Panda	Two-arm lifting	15/20	75.0
K1 + Gemini	Panda	Tested-task total	101/140	72.1
K1 + Gemini	UR5e	Cube lifting	20/20	100.0
K1 + Gemini	UR5e	Cube restacking	20/20	100.0
K1 + Gemini	UR5e	Cube stacking	20/20	100.0
K1 + Gemini	UR5e	Nut assembly	11/20	55.0
K1 + Gemini	UR5e	Tested-task total	71/80	88.8
K1 + Gemini	IIWA	Cube lifting	20/20	100.0
K1 + Gemini	IIWA	Cube restacking	20/20	100.0
K1 + Gemini	IIWA	Cube stacking	20/20	100.0
K1 + Gemini	IIWA	Nut assembly	9/20	45.0
K1 + Gemini	IIWA	Tested-task total	69/80	86.2
OpenVLA base	Panda	Cube lifting	0/20	0.0
OpenVLA base	Panda	Cube restacking	0/20	0.0
OpenVLA base	Panda	Cube stacking	0/20	0.0
OpenVLA base	Panda	Nut assembly	0/20	0.0
OpenVLA base	Panda	Spill wiping	0/20	0.0
OpenVLA base	Panda	Tested-task total	0/100	0.0
OpenVLA base	UR5e	Cube lifting	0/20	0.0
OpenVLA base	UR5e	Cube restacking	0/20	0.0
OpenVLA base	UR5e	Cube stacking	0/20	0.0
OpenVLA base	UR5e	Nut assembly	0/20	0.0
OpenVLA base	UR5e	Tested-task total	0/80	0.0
OpenVLA base	IIWA	Cube lifting	0/20	0.0
OpenVLA base	IIWA	Cube restacking	0/20	0.0
OpenVLA base	IIWA	Cube stacking	0/20	0.0
OpenVLA base	IIWA	Nut assembly	0/20	0.0
OpenVLA base	IIWA	Tested-task total	0/80	0.0
$\pi_{0.5}$ base	Panda	Cube lifting	0/20	0.0
$\pi_{0.5}$ base	Panda	Cube restacking	0/20	0.0
$\pi_{0.5}$ base	Panda	Cube stacking	0/20	0.0
$\pi_{0.5}$ base	Panda	Nut assembly	0/20	0.0
$\pi_{0.5}$ base	Panda	Spill wiping	0/20	0.0
$\pi_{0.5}$ base	Panda	Tested-task total	0/100	0.0
$\pi_{0.5}$ base	UR5e	Cube lifting	0/20	0.0
$\pi_{0.5}$ base	UR5e	Cube restacking	0/20	0.0
$\pi_{0.5}$ base	UR5e	Cube stacking	0/20	0.0

Method	Robot	Task	Successes / trials	Accuracy (%)
$\pi_{0.5}$ base	UR5e	Nut assembly	0/20	0.0
$\pi_{0.5}$ base	UR5e	Tested-task total	0/80	0.0
$\pi_{0.5}$ base	IIWA	Paused	–	–

D.2 RoboTwin: bimanual transfer

Setting and transfer scope. RoboTwin uses Aloha AgileX dual arms in SAPIEN. Easy denotes `demo_clean`; Hard denotes `demo_randomized`, which varies clutter, illumination, textures, and table height (Chen et al., 2025b). K1 supplies native head and two wrist RGB-D views, robot state, perceptual tools, and generic single- or dual-arm commands. Zero-shot denotes deployment without target-task model updates. The interface adapter was tested on RoboTwin tasks.

Task and episode selection. We evaluate ten tasks selected after preliminary testing, excluding Open Laptop, Pick Dual Bottles, and Place Object Basket from the published table. Each task and condition has ten planned episodes. The retained set combines 57 states prevalidated by the native expert and 125 direct resets, using ascending seeds starting at 10000. Expert validation qualifies initial states only; the agent receives no expert actions or demonstrations.

Of 200 planned episodes, 18 fail native initialization: four Easy and fourteen Hard. Excluding these before policy scoring leaves 96 Easy and 86 Hard episodes. Reset-state, RGB-D, calibration, and scene-layout checks identify no further exclusions. All retained policy failures and execution errors count toward accuracy. The reported set uses the original initialization procedure throughout.

Execution and aggregation. The model-call limit is 400, with text history $N = 8$, image history $K = 1$, and a local translation limit of 3 cm per axis. Table 11 lists task-specific native action limits, which count simulator action requests rather than physics substeps or model calls. We use native task success and retain unsuccessful policy outcomes without best-of resampling. For condition c, the plotted score is $A_c = \frac{1}{10}\sum_t s_{tc}/n_{tc}$, where s_{tc} and n_{tc} are successes and retained episodes for task t. This task-macro average gives Easy 32.0% and Hard 28.0%. Episode-micro averages are 31/96 (32.3%) and 25/86 (29.1%), or 56/182 (30.8%) pooled. Unequal valid episode counts explain the difference.

Table 11: **K1 + Gemini on RoboTwin.** Scores are successes/retained episodes followed by accuracy. Excluded counts are initialization failures in Easy/Hard order.

Task	Easy	Hard	Excluded	Action limit
Grab Roller	8/10 (80.0%)	9/10 (90.0%)	0/0	400
Handover Mic	2/10 (20.0%)	0/10 (0.0%)	0/0	600
Lift Pot	2/10 (20.0%)	2/10 (20.0%)	0/0	400
Move Can Pot	4/8 (50.0%)	5/8 (62.5%)	2/2	400
Place Dual Shoes	0/8 (0.0%)	0/6 (0.0%)	2/4	600
Place Phone Stand	2/10 (20.0%)	0/8 (0.0%)	0/2	400
Put Bottles Dustbin	0/10 (0.0%)	0/8 (0.0%)	0/2	1700
Put Object Cabinet	4/10 (40.0%)	0/8 (0.0%)	0/2	700
Stack Blocks Two	6/10 (60.0%)	7/8 (87.5%)	0/2	800
Stack Bowls Two	3/10 (30.0%)	2/10 (20.0%)	0/0	900
Episode-micro total	31/96 (32.3%)	25/86 (29.1%)	4/14	–
Task-macro average	32.0%	28.0%	–	–

Trained-policy references. ACT, DP, DP3, RDT, and π_0 are published results from RoboTwin (Chen et al., 2025b), Table 6, averaged over our ten task names. The source trains on 50 clean expert demonstrations per task and reports 100-rollout evaluation. Initial states, observations, control interfaces, and inference budgets differ from ours. Here π_0 is distinct from the $\pi_{0.5}$ used in our other experiments.

Interpretation. K1 loses 4.0 percentage points from Easy to Hard, compared with 28.4 for RDT and 38.8 for π_0 on the same task subset. This pattern suggests that measured spatial relations can help an RUA tolerate changes in appearance. The experiment combines lighting with other perturbations, so their individual effects remain unresolved. Strong roller and block-stacking results coexist with

Table 12: **Published RoboTwin task accuracies (%).** Each cell is Easy/Hard. Averages are recomputed over the ten displayed tasks from RoboTwin (Chen et al., 2025b), Table 6.

Task	ACT	DP	DP3	RDT	π_0
Grab Roller	66/6	**98**/1	77/1	74/43	96/**80**
Handover Mic	9/0	53/0	93/7	98/**41**	**100**/13
Lift Pot	7/2	37/0	**85**/0	82/19	84/**36**
Move Can Pot	0/0	42/0	28/0	47/23	**74/32**
Place Dual Shoes	0/0	7/0	1/0	4/**4**	**15**/0
Place Phone Stand	0/0	17/0	8/1	15/6	**35/7**
Put Bottles Dustbin	0/0	23/0	0/0	21/4	**54/13**
Put Object Cabinet	4/18	50/17	40/16	30/**30**	**69**/29
Stack Blocks Two	0/0	6/0	0/0	32/**1**	**40/1**
Stack Bowls Two	0/0	34/0	0/0	**73**/21	**73/41**
Task-macro average	8.6/2.6	36.7/1.8	33.2/2.5	47.6/19.2	**64.0/25.2**

failures on shoes and bottle disposal, indicating substantial task-dependent headroom. The differing reset procedures and sample counts should be considered when comparing systems.

E Additional Analysis

E.1 Tool-use distribution

The aggregate in Section 4.5 covers the complete 180-episode Gemini evaluation on LIBERO-PRO: Spatial, Object, and Goal, each with ten task indices, two variants, and three initial states. This gives 30 base task indices and 60 task–variant conditions, with 139 successful and 41 unsuccessful trajectories. RoboSuite, RoboTwin, Astra, and student rollouts are excluded from this count.

All 6,282 model responses are included, averaging 34.9 per episode. Of these, 6,229 contain one tool request and 53 reach the output-token limit without producing a tool request. Figure 7 partitions all responses by function. Motion accounts for 4,238 requests (67.5%), perception for 1,604 (25.5%), memory/progress for 245 (3.9%), and completion checks for 142 (2.3%). The remaining 0.8% are output-limit responses. These counts reflect model decisions, not native simulation steps. A motion request may execute multiple native steps, while perception and memory requests leave physics paused.

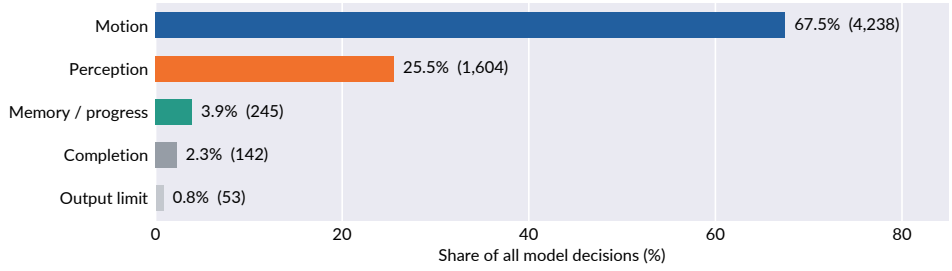


Figure 7: **How Gemini allocates its model decisions.** Shares and counts over 6,282 responses in 180 LIBERO-PRO trajectories, including unsuccessful episodes. Each response occupies exactly one category. Output limit denotes responses without a tool call.

Perception and memory/progress together contribute 1,849 non-motion decisions (29.4%). Of the perception requests, 1,036 occur after the initial simulation frame, across 159 trajectories, confirming that tool use continues after initial observation. We distinguish these observable information-gathering steps from internal model reasoning. Public decision-note metadata records 6,164 format-valid notes and 65 absent or malformed note formats among tool-bearing responses. Notes accompany tool selection within the same response and therefore are not additional steps. Private provider reasoning is excluded from this analysis.

Table 13: Tool-request counts for Figure 7, including requests that return errors.

Category	Tool	Requests
Motion	`move_toward`	1,835
	`move_relative`	654
	`move_to_pose`	547
	`rotate_toward`	271
	`align_axis`	48
	`set_gripper`	883
Perception	`measure_depth`	733
	`find_regions`	384
	`inspect_region`	187
	`inspect_crop`	13
	`grasp_candidates`	274
	`align_region_references`	13
Memory/progress	`update_task_progress`	121
	`search_history`	66
	`read_history`	58
Completion	`done`	142

`align_region_references` computes a geometric alignment hypothesis and is counted as perception, not executed motion. Grasp proposals likewise return candidates without moving the arm. Logged tool errors comprise 23 grasp-proposal errors, five region-inspection errors, three progress-update errors, and two history-read errors. These requests remain in the totals.

E.2 Memory-context ablation

What is varied. We test how much recent interaction should be included automatically in the agent's prompt. K controls the number of previous decision rounds whose images accompany the current observation. N controls the number of recent text exchanges. We vary $K \in \{0, 1, 2, 3, 4\}$ with $N = 8$, and $N \in \{1, 2, 4, 8, 16\}$ with $K = 1$. Complete episode history, `search_history`, `read_history`, and model-maintained task progress remain enabled in every condition. In particular, $K = 0$ removes automatically included past images, not the current observation or the ability to retrieve earlier evidence. This is a prompt-context ablation, not an ablation of the entire memory system.

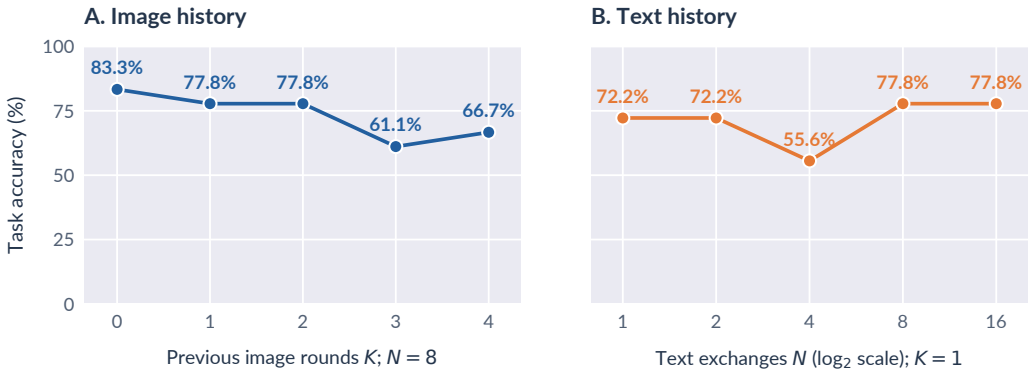


Figure 8: **Longer prompt history does not consistently improve accuracy.** Gemini 3.7 Flash with the frozen K1 harness, evaluated on the same 18 LIBERO-PRO cases per setting. Left: image history K at fixed $N = 8$. Right: text history N at fixed $K = 1$, with a base-2 logarithmic horizontal axis. Point labels show observed accuracy, and lines are visual guides. The protocol and interpretation boundaries are detailed below.

Matched cases and execution. Each setting uses task indices 0–2 from Spatial, Object, and Goal, both `swap` and `task` variants, and initial-state index 0: $3 \times 3 \times 2 = 18$ cases. These are the existing paired-comparison cases, not a new held-out split. Nine parameter settings give 162 unique setting–case evaluations. The two evaluations share the $K = 1, N = 8$ setting, which is counted only once. The experiment retains the frozen harness, perceptual tools, generic motion interface, progress memory, and completion feedback. Agent-side changes are restricted to K, N, and their numeric descriptions in the prompt.

Each episode permits at most 1,200 native simulation steps and 400 model calls. We use the recorded `google/gemini-3.7-flash` endpoint through OpenRouter, temperature 0.2, and a maximum of 1,800 output tokens per response. Backend routing was not explicitly pinned for every request. We verify the K/N configurations and native action replay, with maximum replay TCP discrepancy no greater than 10^{-6} m. Accuracy uses the native simulator criterion, not post-release stability. Infrastructure-interrupted attempts are excluded until a valid run is available. Completed policy failures are retained and are not rerun to select a favorable outcome.

Table 14: **Memory-context ablation outcomes.** Each row uses the same 18 cases. Mean calls include successes and failures. Both evaluations share $K = 1, N = 8$, counted once.

K	N	Successes	Accuracy (%)	Mean calls
0	8	15/18	83.3	28.3
1	8	14/18	77.8	33.0
2	8	14/18	77.8	41.2
3	8	11/18	61.1	47.4
4	8	12/18	66.7	45.9
1	1	13/18	72.2	70.0
1	2	13/18	72.2	72.6
1	4	10/18	55.6	50.7
1	16	14/18	77.8	36.0

Observed effect of image history. At $N = 8$, $K = 0$ reaches 83.3%, $K = 1$ and $K = 2$ reach 77.8%, and $K = 3$ and $K = 4$ reach 61.1% and 66.7%. Mean calls rise from 28.3 at $K = 0$ to 41.2–47.4 at $K = 2$–4. Older images may preserve useful motion cues, but also stale positions, occlusions, and redundant views. These can compete with the current image during grounding. The results favor selective visual context over a longer automatic image history. Retrieval and progress notes remain available at $K = 0$.

Observed effect of text history. At $K = 1$, $N = 1$ and $N = 2$ reach 72.2%, $N = 4$ reaches 55.6%, and $N = 8$ and $N = 16$ reach 77.8%. The shortest windows require 70.0 and 72.6 calls on average, versus 33.0 at $N = 8$ and 36.0 at $N = 16$. With too little immediate text context, earlier attempts and measurements leave the prompt, potentially causing repeated queries or reconstruction of prior decisions. A moderate text window can preserve this continuity without repeatedly retrieving the full record. The non-monotonic curve favors a balance between continuity and relevance.

Practical implication. Text records preserve decisions and milestones, while images provide time-sensitive visual evidence. The two need not use equally long windows. Our main configuration remains $K = 1, N = 8$. Each setting has 18 episodes without repeated evaluation seeds, so a single outcome changes accuracy by 5.6 points. These curves guide context design, not exact tuning.

E.3 Qualitative interaction traces

LIBERO-PRO: pick-and-place with visual servoing.

The instruction is to pick the black bowl next to the ramekin and place it on the plate. K1 reaches native success at frame 322 after 33 model calls and passes post-release replay. The paired RGB episode ends at frame 235 after thirteen decisions with `done`, without native success.

The trace connects visual hypothesis to physical test: after closing, the agent uses the observed lift to update its carrying milestone, then measures the placement relation. The trace includes both failed and successful tool calls. Native success occurs before voluntary release, which is checked separately by the evaluator.

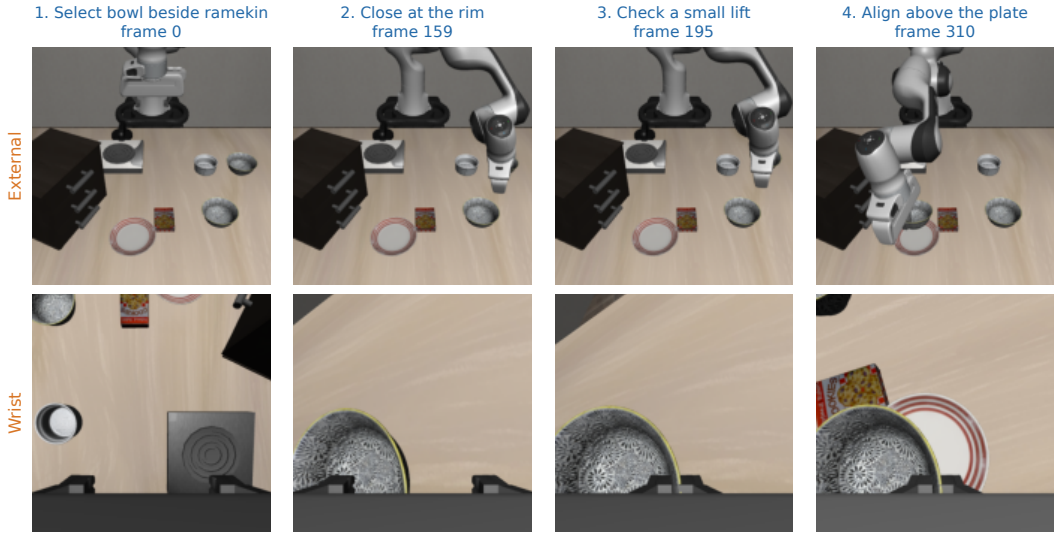


Figure 9: **Recorded LIBERO-PRO trajectory.** GPT-6 Astra with K1 on Spatial swap task 1/state 0. Panels show original external/wrist RGB at four decision frames, without candidate or region overlays. The final panel precedes native termination. Table 15 summarizes the tool evidence.

Table 15: Selected calls from the LIBERO-PRO trace. Skipped calls adjust poses or retrieve history; no resets are omitted.

Call	Operation	Observable role
0	`find_regions`	Search for black bowls. S2 is selected from the existing external view based on the ramekin relation.
2	`grasp_candidates`	Request nominal grasp hypotheses for S2. No motion is executed by this call.
3–9	Transit and rotation	Approach with open jaws, rotate in bounded steps, and begin the persistent pose approach.
10, 17	`inspect_region`	Refresh S2 using a wrist-image bounding box while approaching the rim.
20	`set_gripper`	Close at frame 159 and allow native settling.
21–23	Inspect, lift, update	Reinspect S2, request a 25 mm lift, and update the carry milestone from the resulting observation.
25–28	Find plate and align	Select destination S3 and query a horizontal source–destination reference alignment before transit.
29–32	Local corrections	Reinspect the measured relation and refine the position in bounded steps. Native success ends the rollout at frame 322.

RoboSuite: cross-embodiment nut assembly with regrasping. Gemini with K1 controls a UR5e arm with the Panda gripper to place a square nut onto its peg (reset configuration 0, repeat 0). The episode reaches native success after 35 model calls and 534 control steps, with successful native replay. Figure 10 and Table 16 retain the intermediate regrasp attempts: gripper closure alone does not establish that the nut is being carried.

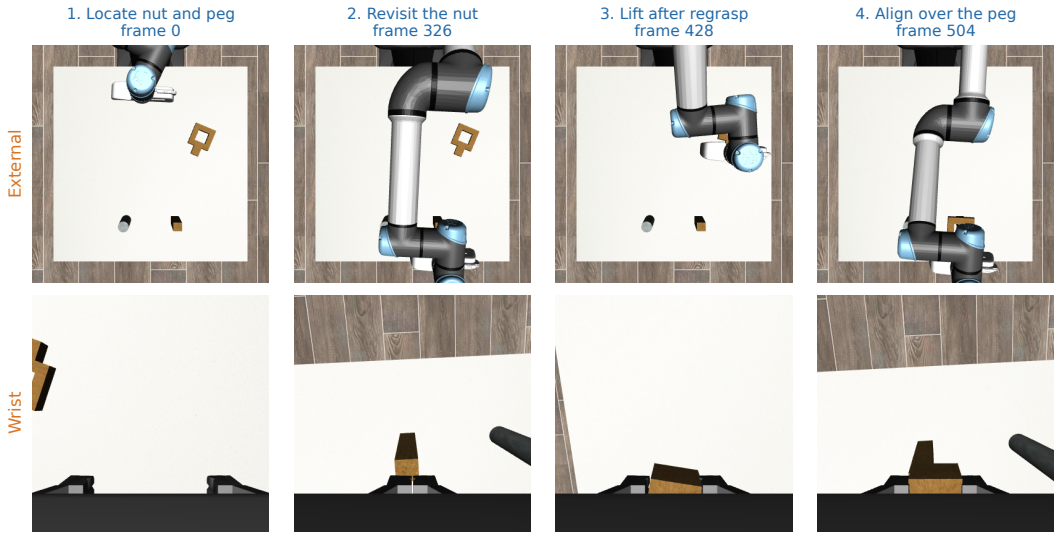


Figure 10: **RoboSuite nut assembly with regrasping.** Recorded external and wrist RGB at four decision frames. The nut remains on the table at frame 326; after the third closure, it is visible near the gripper at frame 428 and above the peg at frame 504. The last panel precedes the final descent and native success at frame 534.

Table 16: Selected calls from the 35-call RoboSuite episode. Omitted calls perform measurements or local motion within the same episode.

Call	Operation	Observable role
0–3	Ground and inspect	Find the nut, refine its handle as S2, measure depth, and request grasp hypotheses.
4–9	Approach and close	Record progress, approach using wrist depth, close at frame 79, and request a 3 cm lift.
10–14	Measure and revisit	Measure the scene and move toward the peg. A new search at frame 171 finds the nut still near its original position.
15–23	Second attempt	Return with open jaws, remeasure, close at frame 235, and repeat lift and transport.
24–29	Reground and regrasp	Find the nut again at frame 326, use wrist depth to refine approach, and close at frame 384.
30–31	Lift and inspect	Lift with closed jaws. The frame-428 wrist image shows the nut near the gripper; a depth query follows.
32–34	Align and descend	Move above the peg, query wrist depth, then descend. Execution stops on native success at frame 534.

The episode illustrates recovery through renewed visual grounding: after unsuccessful transport attempts, the agent returns to the observed nut rather than assuming that a closed gripper implies a completed grasp. The same perception–action loop operates through the UR5e adapter without target-task fine-tuning.

RoboTwin: dual-arm grasping under visual perturbation. In the Hard condition of RoboTwin `grab_roller` (seed 10002), Gemini with K1 follows the instruction "Hold the smooth wooden roller with both arms." The Aloha–AgileX episode succeeds after 12 model calls and 31 native action requests. The sequence includes a rejected completion claim followed by a successful joint lift (Figure 11; Table 17).

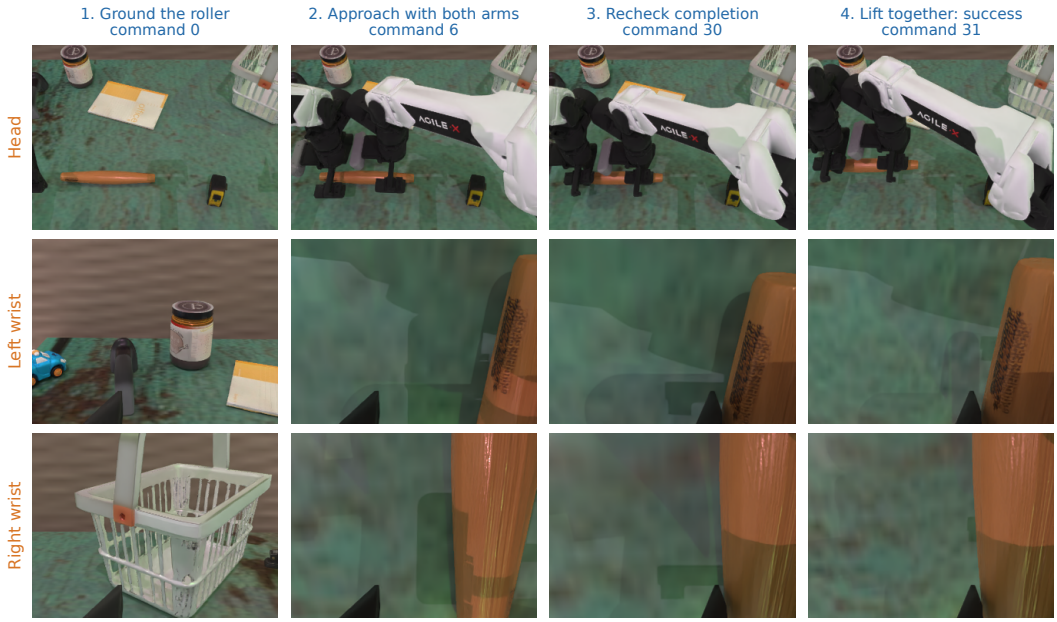


Figure 11: **Dual-arm grasping in a randomized RoboTwin scene.** Recorded head, left-wrist, and right-wrist RGB. Command indices count native action requests, not physics steps or video seconds. The final panel is the recorded terminal observation after both arms lift; it is not an additional model call.

Table 17: Calls from the 12-call RoboTwin episode, including the failed target and rejected completion check.

Call	Operation	Observable role
0–2	Ground and propose	Find roller region S1 in the head view, then request grasp hypotheses separately for the left and right arms.
3–5	`move_effectors`	Approach, lower, and close both grippers using paired arm targets.
6–7	Adjust right grasp	Reopen and reposition the right gripper, then close it while retaining the left grasp command.
8–9	Correct a failed target	A negative left-arm height produces planner failures. The next call restores a positive height and refines the right grasp.
10	`done`	Native success is false at command 30. The completion claim is rejected and the episode remains active.
11	`move_effectors`	Lift both closed grippers by approximately 6 cm in achieved motion. Native success follows at command 31.

This case shows how shared visual grounding supports arm-specific targets and coordinated motion. Sparse completion feedback also separates an agent's verbal claim from physical success, allowing the remaining action to finish the task.