WORLD ACTION AGENT: HARNESSING VLMS FOR ROBOT MANIPULATION VIA WORLD ACTION REHEARSAL

Anonymous authors
Paper under double-blind review

ABSTRACT

General-purpose vision-language models (VLMs) bring broad knowledge and spatial reasoning to robot manipulation, yet existing systems either use them indirectly, to predict constraints or write programs, or give them a view of the scene rather than a world in which to act. We present **World Action Agent (WAA)**, a multi-agent harness through which VLMs pilot robots with basic tools, making every decision within a **visual action workspace**. The workspace has three properties. **Contact views**, selected automatically from the scene geometry, present the scene around the current interaction. **Action rehearsal** turns each action into an editable proposal that the agent, alone or through an Imagination Agent, previews and revises against planning feedback before execution. **In-view correction** closes the loop between observation, rehearsal, and low-level execution, letting the agent remove residual offsets in the view where it observes them. Through the same workspace, WAA acquires embodied procedural knowledge in two ways: it evolves multimodal skills from expert videos and human teaching under evidence-based review and consults them through a Skill Agent, and its interaction traces train smaller VLMs to pilot the same harness. On LIBERO-Pro, WAA with skills evolved only from LIBERO-90 reaches a state-of-the-art **75.6%** average success, outperforming end-to-end VLAs, code-as-policy agents, and a visual-harness baseline with the same backbone; the same skills remain effective on robosuite without further learning. Fine-tuning Qwen3.5-9B on harness traces raises its out-of-domain success from 1.7% to 43.3%.

1 INTRODUCTION

Learning manipulation policies that generalize to new objects, layouts, and tasks is a central goal in robotics. Vision-language-action (VLA) models learn such policies from visual observations, language instructions, and robot actions (Kim et al., 2024; Intelligence et al., 2025), but fine-tuning vision-language models (VLMs) for action prediction may weaken their general understanding and reasoning (Hancock et al., 2026). World action models (WAMs) couple environment dynamics with action learning (Zhu et al., 2025a; Ye et al., 2026), yet grounding learned visual dynamics in executable control still requires robot data and action alignment (Shen et al., 2026). A complementary route preserves the generality of VLMs and involves them in manipulation decisions. HAMSTER and ReKep have the VLM predict intermediate paths or relational constraints that a downstream policy or optimizer turns into motions (Li et al., 2025b; Huang et al., 2024); Code as Policies (CaP) and CaP-X have a language model compose perception and control APIs into programs and revise them from execution feedback (Liang et al., 2023; Fu et al., 2026). In both cases, the VLM either supplies constraints and spatial guidance or indirectly writes a program, rather than controlling the robot directly through its own spatial understanding and basic primitives.

Show-Harness and VIA take a step in this direction: the VLM selects and revises actions directly through a visual robot interface, turning manipulation into an interactive visual reasoning problem (Chen et al., 2026b; Hu et al., 2026). Yet such interfaces display the scene to the VLM and let it choose actions; they do not give it a world in which to act. Three things are missing. First, observation is not centered on the interaction, although success depends on local relations among the gripper, the object, and the target. Second, actions cannot be tried before they are taken: each

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



Figure 1: **World Action Agent (WAA).** WAA combines visual action rehearsal with reusable skills, enabling VLMs to perform diverse manipulation tasks and adapt through expert demonstrations and human guidance.

takes effect as soon as it is issued, before the VLM can see its consequences. Third, perception and action live in different spaces: an offset observed in an image must be rewritten as coordinates before it can be corrected. This motivates our question: *how can a general-purpose VLM use basic action primitives to make and revise decisions throughout execution, serving as a robot pilot?*

Our answer is to change what the VLM sees and how its decisions take effect, rather than the VLM itself. We introduce **World Action Agent (WAA)** (Figure 1), a multi-agent embodied harness in which agents observe the scene, construct actions, and drive the robot through basic tools alone, build and retrieve multimodal skills, and make every decision within a single **visual action workspace**. On this basis, WAA has three properties. First, it presents the scene around the interaction: Contact views are selected automatically for the current interaction, letting the VLM bring its spatial understanding to bear. Second, actions can be rehearsed and revised before execution: each action is an editable proposal with planning feedback. Third, observation, rehearsal, and low-level execution form a closed loop within the same set of calibrated views.

Piloting a robot also requires embodied procedural knowledge that VLM pretraining rarely captures. WAA acquires this knowledge through the same workspace in two ways: it evolves multimodal skills from expert videos and human teaching, and it uses interaction traces to train a smaller VLM to pilot the same harness.

On LIBERO-Pro (Zhou et al., 2026), WAA with skills evolved only from LIBERO-90 reaches 75.6% average success, above ASPIRE (72.0%), and the same skills remain effective on robosuite (Zhu et al., 2025b) without further learning. Fine-tuning Qwen3.5-9B (Qwen Team, 2026) on harness traces raises its out-of-domain success from 1.7% to 43.3%. Our main contributions are:

- **World Action Agent.** We introduce WAA, a multi-agent harness through which general-purpose VLMs pilot robots with basic tools, observing, rehearsing, and correcting every action within a single visual action workspace.
- **Learning through the harness.** We show that the same workspace supports both non-parametric skill evolution from expert videos and human teaching and the distillation of interaction traces into smaller VLM pilots.
- **State-of-the-art results.** WAA achieves a state-of-the-art 75.6% average success on LIBERO-Pro and transfers to robosuite without further learning. Further analyses show that a 9B VLM can learn to pilot the same harness.

2 Related Work

Foundation models for robot manipulation. Foundation models support robot manipulation from action prediction to high-level decision-making. Vision-language-action (VLA) models, including RT-2, OpenVLA, and $\pi_{0.5}$, generate robot actions from vision and language through discrete tokens or continuous action modules (Brohan et al., 2023; Kim et al., 2024; Intelligence et al., 2025). However, fine-tuning VLMs for action prediction may weaken their general reasoning and multimodal understanding, limiting generalization (Hancock et al., 2026). World action models (WAMs) couple future-state prediction with robot action learning. Recent approaches jointly model video and actions or adapt pretrained video models for control (Li et al., 2025a; Zhu et al., 2025a; Bi et al., 2026; Kim et al., 2026; Li et al., 2026; Ye et al., 2026). Others reduce inference cost by omitting future-video generation or predicting compact latent futures (Yuan et al., 2026a; Lin et al., 2026). However, executable control typically requires robot data for action alignment, and cross-embodiment transfer remains sensitive to morphology and action-space differences (Shen et al., 2026). Complementary work uses VLM-generated spatial representations to guide robot control. HAMSTER uses a fine-tuned VLM to predict coarse 2D end-effector paths that guide a 3D-aware low-level policy (Li et al., 2025b). ReKep generates relational keypoint constraints for hierarchical optimization (Huang et al., 2024), while OmniManip grounds VLM reasoning in object-centric interaction primitives that specify 3D spatial constraints (Pan et al., 2025). FSD trains a VLM to use spatial reasoning to generate affordance regions, points, and visual traces for motion planning (Yuan et al., 2026b). These approaches use VLM knowledge and spatial reasoning to guide actions, supporting VLMs as core decision makers for manipulation.

Agentic robot control and visual interfaces. Code as Policies (CaP) and subsequent code-generation systems connect language-model reasoning to robot execution through programs over perception and control APIs (Liang et al., 2023; Singh et al., 2023; Chen et al., 2024; Mu et al., 2024). Recent systems incorporate interactive execution and experience reuse. CaP-X evaluates and improves coding agents through multi-turn feedback and skill synthesis (Fu et al., 2026). RATs acquires reusable code skills through self-directed play (Zhang et al., 2026a), while ASPIRE combines execution diagnosis, program repair, and evolutionary search to build transferable skill libraries (Lu et al., 2026). GaP structures robot programs as directed execution graphs and refines their structure and parameters through simulation rehearsal (Chen et al., 2026a). Agent as Policy keeps a coding agent in the control loop to revise programs and motions from physical feedback (Jia et al., 2026). Their effectiveness depends on the APIs' perceptual grounding and control abstractions: CaP-X reports declining performance as hand-designed abstractions are removed (Fu et al., 2026). Hierarchical and agentic systems also couple high-level reasoning with VLA execution. Hi Robot, AtomBridge, and Harness VLA support this coupling through subtask instructions, inter-skill transitions, and memory-guided orchestration, respectively (Shi et al., 2025; Pang et al., 2026; Zhang et al., 2026c). These systems improve task composition, but each VLA call remains bounded by the learned policy's capabilities. Visual interfaces such as VIA and Show-Harness instead expose spatial targets or fine-grained semantic actions directly to the VLM (Hu et al., 2026; Chen et al., 2026b). WAA gives the VLM direct control over robot primitives, with a visual action workspace for inspecting and revising actions before execution. Numerical planners and controllers realize these decisions without a VLA executor or generated policy code.

3 World Action Agent

3.1 Problem Formulation

Given a task instruction g, the agent completes a manipulation task through a sequence of tool calls. At step t, the harness presents a multi-view Canvas C_t and a control context m_t that records valid spatial references, the pending action proposal $\hat{a}_t$, and recent execution feedback. The policy then selects a tool call

$$u_t \sim \pi_\theta(\cdot \mid g, C_t, m_t)\,, \tag{1}$$

where u_t specifies a tool and its arguments. Tools fall into three classes by their effect: **query** tools acquire spatial information or skill guidance, **proposal** tools construct and revise $\hat{a}_t$ and return a visual preview with planning feedback, and **execution** tools move the robot. Because query and proposal tools leave the physical world unchanged, the agent can inspect and revise an action repeatedly before committing to it. The episode ends when the task succeeds or the interaction budget

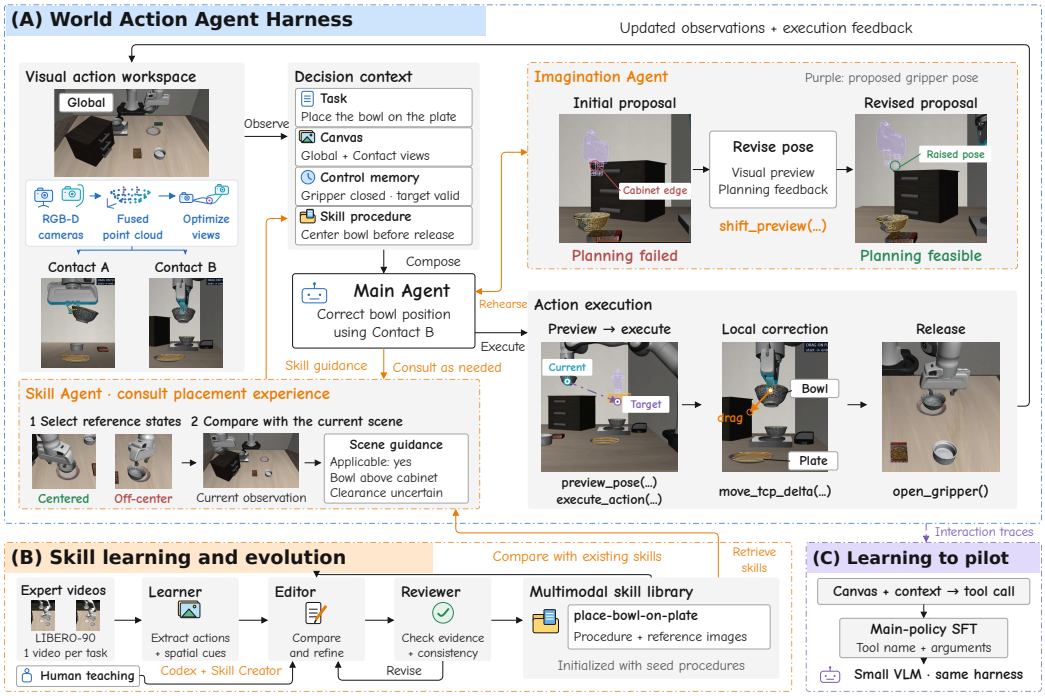


Figure 2: **Overview of the World Action Agent (WAA) harness. (A)** The main agent observes global and Contact views in the visual action workspace, rehearses action proposals with an Imagination Agent, consults skills through a Skill Agent, and executes planned motions and local corrections. **(B)** Expert videos and human teaching are distilled into a multimodal skill library through extraction, editing, and review. **(C)** Interaction traces train a smaller VLM to operate the same harness. Insets are from two recorded bowl-manipulation tasks; dashed arrows indicate target guides.

is exhausted. Within this formulation, the VLM decides what to manipulate, how to revise a proposal, and when to execute it, while the harness turns these decisions into geometrically accurate and kinematically feasible robot motion.

3.2 Designing the World Action Agent Harness

Design principle. General-purpose VLMs excel at semantic understanding and qualitative spatial judgment, but they struggle to produce precise metric poses and cannot anticipate whether an action is physically executable. When a VLM drives a robot directly, three difficulties arise: local relations among the gripper, object, and target are hard to discern from a global view; physical actions are irreversible and cannot be verified in advance; and planned motions leave fine alignment errors. WAA addresses them with three corresponding designs that together form a **visual action workspace** (Figure 2A): an interaction-centered Canvas lets the agent see local relations, action rehearsal lets it test actions before execution, and in-view correction lets it remove residual errors from new observations. Because all three share one calibrated Canvas, the agent observes, rehearses, and corrects the same spatial targets.

Interaction-centered Canvas. Manipulation outcomes often hinge on millimeter- to centimeter-scale relations among the gripper, object, and target, which a fixed global camera frequently fails to reveal because of distance, occlusion, or degenerate viewing angles. WAA therefore selects viewpoints actively according to the current interaction. Given the scene point cloud $\mathcal{P}_t$, the robot geometry $\mathcal{R}_t$, and the highlighted interaction region $\mathcal{H}_t$, the harness solves for the camera parameters of the **Contact views**

$$\mathbf{c}_t = \arg\max_{\mathbf{c}\in\Omega} \; S_{\text{vis}}(\mathbf{c}; \mathcal{H}_t, \mathcal{R}_t \mid \mathcal{P}_t) + \lambda_1 S_{\text{frame}}(\mathbf{c}; \mathcal{H}_t) - \lambda_2 E_{\text{red}}(\mathbf{c}) - \lambda_3 E_{\text{stab}}(\mathbf{c}, \mathbf{c}_{t-1}), \quad (2)$$

where S_{vis} measures the visibility of the interaction region and gripper under scene occlusion, S_{frame} encourages compact framing, E_{red} discourages views that convey the same direction, and E_{stab}

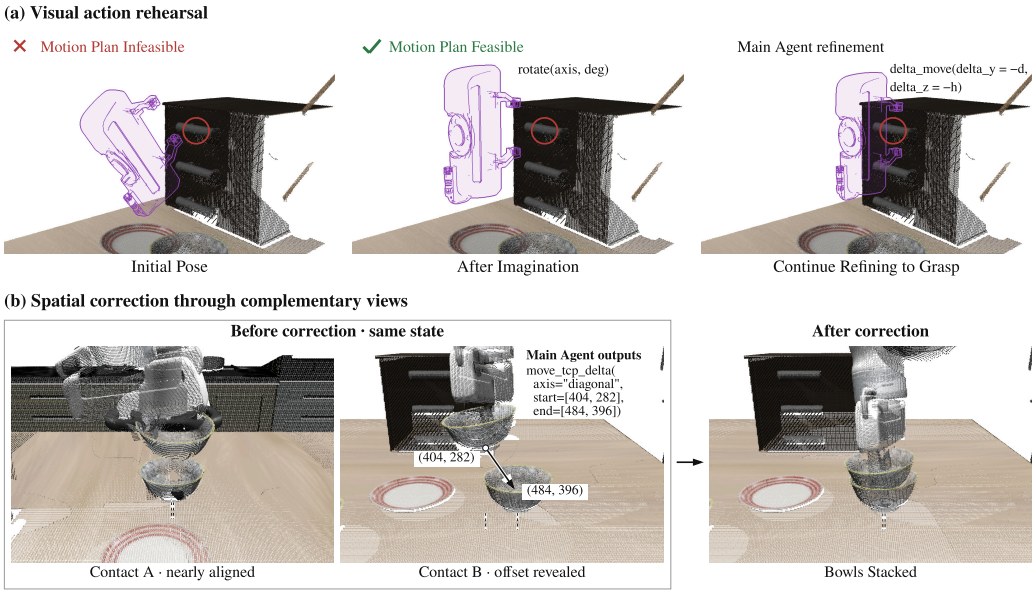


Figure 3: **Action rehearsal and in-view correction.** (a) The Imagination Agent rotates an infeasible grasp preview (purple) until planning succeeds; the main agent then shifts it toward the handle (red circle). (b) Contact A appears aligned, but Contact B reveals an offset; the main agent drags the bowl toward its target in Contact B, and the harness converts the drag into end-effector motion. These interface examples are rendered from RGB-D point clouds.

suppresses viewpoint jumps so that spatial relations remain comparable across steps. The feasible set Ω constrains the horizontal projections of the two Contact views to be orthogonal, so that an alignment error can be read along two independent directions. The objective only requires a scene point cloud in the robot base frame, so WAA is agnostic to how $\mathcal{P}_t$ is obtained: it can be rendered in simulation, fused from calibrated RGB-D cameras together with the forward-kinematic hand geometry, or reconstructed from RGB views with a feed-forward model such as VGGT (Wang et al., 2025). Together with the global view, the optimized Contact views form the Canvas: the global view provides task context, the Contact views expose fine interaction relations, and every view retains its projection calibration so that image-space annotations map directly to 3D.

Action rehearsal. For a VLM, the hardest part is often not deciding where to go but judging whether a specific pose is appropriate: whether the approach collides with nearby objects, whether the robot can reach it, and whether the resulting grasp supports the next step. WAA therefore represents an action as an **editable proposal** rather than a one-shot output. The agent localizes targets on the Canvas and specifies a target pose from them; the harness solves inverse kinematics, plans the motion with cuRobo (Sundaralingam et al., 2023), overlays the target configuration as a translucent robot in every view, and reports its feasibility (Figure 3a). The agent can thus examine approach direction, prospective contact, and surrounding clearance before execution and revise the pose accordingly. For finer adjustment, the main agent delegates its spatial intent to an **Imagination Agent**, which runs in a separate context and iteratively edits and replans the proposal while the physical scene remains unchanged, returning only the revised proposal. Only a proposal with an executable plan can become physical motion. Rehearsal thus couples the VLM's judgment of task geometry with the planner's feasibility checks, exposing errors before they occur in the physical world.

Closed-loop execution and in-view correction. Target-level planned motions suit large movements such as approaching and transporting, but depth noise, calibration error, and contact disturbances leave residual errors that often decide whether placement succeeds. WAA lets the agent express corrections directly **in the view where the error is observed**. The agent drags from a reference point to its desired location in a Contact view, and the harness uses the view's calibration to convert this image-space intent into a bounded end-effector displacement (Figure 3b). The reference point can be the gripper or a visible point on the held object, so the agent can state where the object should go without first converting that relation into an absolute gripper pose. The two orthogonal

Contact views support corrections along complementary directions. After each correction, a fresh observation lets the agent decide whether to adjust further, release the object, or proceed to the next target-level action.

3.3 Learning and Evolution through the Harness

The visual action workspace in Section 3.2 lets a VLM observe and act on spatial relations, but it does not tell the agent how to use these capabilities to complete a task. This requires embodied procedural knowledge that is rarely captured by VLM pretraining, such as which contact to establish first, which relation to verify before release, and how to recover from a failed grasp. WAA acquires this knowledge through the same interface along two complementary paths. Non-parametrically, the agent evolves a library of multimodal skills from expert demonstrations and human teaching and consults it during execution with its parameters fixed. Parametrically, interaction traces recorded in the harness train a smaller VLM to pilot the same harness.

Multimodal manipulation skills. Textual procedures alone struggle to convey judgments such as when an alignment is sufficient, and visual references can supply the evidence needed for state recognition and verification (Zhang et al., 2026b; Jiang et al., 2026). Each WAA skill therefore combines applicability conditions, a procedure, key states with reference images, and observable outcome checks (Appendix A). The procedure specifies the relations that the gripper, object, and target should satisfy, the required contact and clearance, and how to recover from failure, rather than fixed coordinates or displacements. Objects, target positions, and motion magnitudes are always resolved in the current scene, and numerical values appear only as tentative ranges to be adjusted from observed feedback. In contrast to skill libraries that register validated per-object parameters such as grasp height offsets and yaw angles (Lu et al., 2026), this **relational guidance** lets a skill transfer across object instances, layouts, and viewpoints.

During execution, the main agent selects a skill by its name and description and delegates its interpretation to a **Skill Agent**. Running in a separate context, the Skill Agent selects the relevant procedural states and reference images, compares them with the current Canvas, and reports applicability, scene differences, and relations that cannot yet be confirmed (Appendix E.2). The main agent receives the full textual procedure, whereas reference images remain in the Skill Agent's context, which prevents historical images from being mistaken for the current scene and keeps the main context compact. Scene-specific advice expires when its associated observation or control state changes, so the agent does not act on outdated guidance.

Skill evolution from demonstrations and teaching. The library starts from text-only seed skills that Codex (GPT-5.5) writes from the harness API, providing semantic-level guidance without hard-coded parameters or fixed action sequences. We then enrich it with one expert trajectory per LIBERO-90 source task, whose scenes differ from those used for evaluation. Learning proceeds through three agent roles, each driven by GPT-5.5 (Figure 2B). The **Learner** segments each video at observed state changes, interprets contact, object motion, and support changes in terms of harness actions, and extracts knowledge candidates that cite their source frames. It does not read the existing skill text, so new evidence is not steered by prior conclusions. The **Editor** compares each candidate with existing procedures and reference images, then adds, revises, retires, or defers skills with source frames as visual evidence. The **Reviewer** independently checks the revised skills against the source evidence and returns concrete issues to the Editor, so that only supported changes that preserve existing applicability conditions enter the library. Requiring evidence and independent review for every edit keeps the library from degrading as learning proceeds.

Demonstrations, however, show successful behavior and rarely reveal how to recover from mistakes. WAA therefore provides a Canvas-GUI: humans see the same multi-view Canvas as the agent and operate the robot with the same tools, such as pointing in a view, dragging in a Contact view, and previewing and executing proposals. Humans thus pilot the robot much as a GUI agent would, and the harness records these explicit actions in the same format as agent interactions. The Learner contrasts the agent's attempt before failure with the human correction from the same state, identifies which relation, such as contact location, alignment direction, or release timing, accounts for the difference in outcome, and distills it into recovery guidance, which then passes through the same Editor–Reviewer loop as video-derived candidates.

Learning to pilot. Large proprietary VLMs pilot the harness effectively, but at relatively high cost and latency. We therefore distill interaction traces into a smaller VLM that operates the same harness (Figure 2C). Training examples come from successful episodes, retaining only clean executions without redundant operations. We further include separately reviewed recovery segments, such as regrasping after an empty grasp, so that the model also learns to revise decisions from feedback. Human corrections recorded through the Canvas-GUI share the format of agent traces and can likewise serve as supervision for recovery behavior. Each example pairs the pre-action Canvas, task instruction, control context, and available skill guidance with the executed tool call, without the teacher's reasoning or any privileged simulator state. The model is trained by maximum likelihood on the policy in Eq. (1):

$$\max_{\theta} \sum_{(g,C_t,m_t,u_t)\in\mathcal{D}} \log \pi_\theta(u_t \mid g, C_t, m_t)\,. \tag{3}$$

Fine-tuning updates only the main policy, while the visual action workspace and sub-agents remain unchanged, so the model learns to use the interface rather than to replace it.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate on LIBERO-Pro (Zhou et al., 2026) and robosuite (Zhu et al., 2025b). Following CaP-X and Playful (Fu et al., 2026; Zhang et al., 2026a), we use six LIBERO-Pro splits, Object, Goal, and Spatial, each under initial-position swaps (Pos.) and task perturbations (Task), with ten tasks per split. Each WAA setting is evaluated for 60 episodes per split, six per task. In robosuite, we use cube lifting, stacking, and restacking.

Evaluation protocol. We evaluate WAA with Gemini 3.7 Flash as the backbone in three settings: without skills (zero-shot), with the text-only seed skills, and with the evolved skill library. The skill library is learned only from LIBERO-90 and frozen before evaluation: no LIBERO-Pro or robosuite rollout, failure log, or label updates skills, prompts, or model parameters. Each episode starts in a fresh process and is limited to 50 main-agent turns, 50 physical operations, and one hour, with at most six turns per Imagination Agent call. WAA accepts scene point clouds from multiple sources, including the simulator, fused RGB-D views, and VGGT reconstruction (Wang et al., 2025); Table 1 reports all three.

Baselines. We compare with three families of methods. *End-to-end VLAs*: OpenVLA (Kim et al., 2024) and $\pi_{0.5}$ (Intelligence et al., 2025). *Code-as-policy agents*: CaP-Agent0 (Fu et al., 2026), RATs with play-learned skills (Zhang et al., 2026a), and ASPIRE (Lu et al., 2026). *Visual agent harness*: Show-Harness (Chen et al., 2026b) (Gemini 3.7 Flash), which we rerun for 30 episodes per split. Other results are taken from the original papers.

Pilot fine-tuning. Unlike the skill library, the smaller pilot is trained on LIBERO-Pro: 1,774 main-agent tool calls from 112 successful Gemini 3.7 Flash episodes with evolved skills across all six splits. We fine-tune Qwen3.5-9B (Qwen Team, 2026) with LoRA (rank 8, 10 epochs, learning rate 5×10^{-5}; Appendix F); at evaluation, it replaces only the main agent, while sub-agents and grounding still use Gemini 3.7 Flash.

4.2 Main Results

Table 1 compares WAA with all baselines on LIBERO-Pro. With evolved skills, WAA reaches a state-of-the-art 75.6% average success and the best results on both Spatial splits and on Goal Task, although its skills are learned only from LIBERO-90 and frozen before evaluation. The largest margins appear on the Spatial splits, where success depends on centimeter-scale placement relations, while ASPIRE remains strongest on both Object splits and on Goal Pos.

Even without skills, a general-purpose VLM can pilot the robot through WAA without task-specific training. Zero-shot WAA reaches 28.9% on average, above $\pi_{0.5}$ (12.8%) and both CaP-Agent0 evaluations, with 53.3% and 46.7% on the two Object splits. In contrast, end-to-end VLAs almost completely fail under task perturbations, as unfamiliar instructions and object combinations fall outside their training distributions, whereas WAA preserves the VLM's general understanding and grounds each decision in the current Canvas.

Table 1: Success rates (%) on LIBERO-Pro. Bold marks the best result in each column.

	Object		Goal		Spatial		
Method	Pos.	Task	Pos.	Task	Pos.	Task	Avg.
OpenVLA	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
CaP-Agent0 (CaP-X)	22.0	18.0	26.0	17.0	12.0	14.0	18.2
CaP-Agent0 (Playful)	27.0	31.0	29.0	16.0	13.0	23.0	23.2
RATs (Playful)	61.0	63.0	43.0	36.0	29.0	31.0	43.8
ASPIRE	**98.0**	**95.0**	**81.0**	45.0	51.0	60.0	72.0
Show-Harness	13.3	16.7	0.0	10.0	0.0	0.0	6.7
Scene point cloud							
WAA (zero-shot)	53.3	46.7	30.0	23.3	13.3	6.7	28.9
WAA + seed skills	83.3	70.0	26.7	40.0	16.7	23.3	43.3
WAA (evolved skills)	95.0	93.3	63.3	**48.3**	**80.0**	**73.3**	**75.6**
WAA (evolved skills & RGB-D)	88.3	91.7	56.7	45.0	76.7	68.3	71.1
WAA (evolved skills & VGGT)	90.0	88.3	53.3	45.0	65.0	71.7	68.9

With the same backbone and likewise without skills, Show-Harness reaches only 6.7% on average and fails every Spatial episode, whereas zero-shot WAA reaches 28.9%. Both let the VLM select and revise actions; the difference lies in what the agent can observe and test before acting. Contact views expose local interaction relations, rehearsal lets poses be checked before execution, and after a failed contact the agent adjusts from observation and regrasps (Figure 4C). WAA is also more efficient (Table 2), averaging 31 model calls and 150 s per episode against 120 calls and 874 s for Show-Harness.

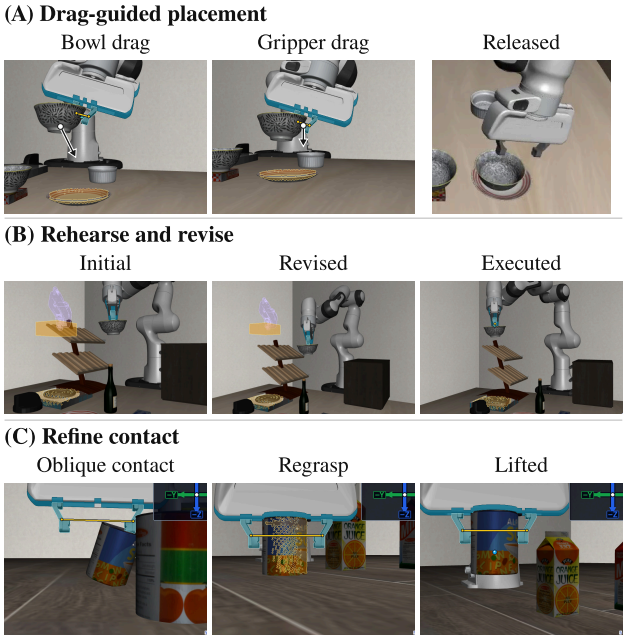


Figure 4: **Qualitative examples of WAA.** (A) Drag corrections align and lower a bowl before release. Circles mark object and gripper anchors, and arrows summarize the corrections. (B) An infeasible proposal becomes executable after raising the candidate. (C) Revised contact enables a successful soup-can lift after failed grasps.

Code-as-policy agents are limited by open-loop geometric grounding. A program cannot inspect whether a specific pose is appropriate before it runs, so on the Spatial splits, where placements hinge on centimeter-scale relations, CaP-Agent0 and RATs reach only 12.0% to 31.0% and ASPIRE 51.0% and 60.0%. WAA reaches 80.0% and 73.3%: rehearsal exposes and repairs an infeasible proposal before execution (Figure 4B), and in-view drags align the bowl with the plate before release (Figure 4A).

4.3 Skill Learning, Transfer, and Generality

Evolved multimodal skills contribute the largest gain. Average success rises from 28.9% without skills to 43.3% with seed skills and 75.6% with evolved skills, and Spatial success rises from 13.3% and 6.7% to 80.0% and 73.3%. Text-only seed skills help inconsistently and even slightly lower Goal Pos. success, whereas evolved skills improve on the zero-shot setting in all six splits.

A single demonstration suffices to acquire a new skill. Starting from text-only seed skills without any stove-specific procedure, one LIBERO-90 demonstration of stove activation yields a skill with

Table 2: Per-episode cost compared with Show-Harness.

Method	API cost ($) ↓	Model calls ↓	Time (s) ↓	Input token (K) ↓	Output token (K) ↓
Show-Harness	0.5021	119.80	874.12	347.10	62.90
WAA	**0.1996**	**31.05**	**150.47**	**193.62**	**5.45**

Table 3: Success rates (%) on robosuite. + LIBERO: the frozen skill library learned from LIBERO-90.

	CaP-Agent0		WAA	
Task	No skills	+ RATs	No skills	+ LIBERO
Cube lift	68.0	84.0	100.0	100.0
Cube stack	46.0	60.0	100.0	100.0
Cube restack	34.0	46.0	60.0	100.0
Average	49.3	63.3	86.7	**100.0**

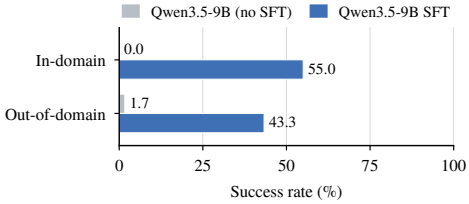


Figure 5: Success rates (%) of Qwen3.5-9B piloting WAA before and after SFT.

which WAA succeeds in all ten executions of the LIBERO-Pro stove task (Appendix A.3). The Reviewer rejects a first draft that treats knob motion as success, and the accepted skill requires an explicit activation signal, showing that independent review blocks overly broad outcome checks.

Skills learned in LIBERO transfer to robosuite. In robosuite, whose scenes, objects, and camera placements differ from LIBERO (Table 3), WAA without skills already succeeds in every lifting and stacking trial and reaches 60.0% on restacking. The frozen LIBERO skills raise restacking to 100.0%, whereas CaP-Agent0 with RATs skills averages 63.3%.

WAA is agnostic to the source of the scene point cloud. Replacing the simulator's point cloud with fused RGB-D views or VGGT reconstruction (Wang et al., 2025) (last two rows of Table 1) lowers the average success only to 71.1% and 68.9%, a drop of 4.5 and 6.7 points, and both settings still surpass every baseline on the two Spatial splits. Because Contact-view selection only needs a base-frame point cloud, the perception source can change without modifying agents or skills.

A smaller VLM learns to pilot the same harness. We fine-tune Qwen3.5-9B (Qwen Team, 2026) with Eq. (3) on 112 harness trajectories (1,774 decision steps), replacing only the main agent while the sub-agents stay unchanged (Figure 5). In-domain episodes share the seed range of the training trajectories, and out-of-domain episodes use unseen seeds. Fine-tuning raises success from 0.0% to 55.0% in domain and from 1.7% to 43.3% out of domain.

5 Conclusion

We asked how a general-purpose VLM can use basic action primitives to make and revise decisions throughout execution. WAA answers by changing what the VLM sees and how its decisions take effect rather than retraining it: its visual action workspace presents the scene around the interaction, lets each action be rehearsed and revised before execution, and closes the loop between observation, rehearsal, and low-level execution. With skills evolved only from LIBERO-90, WAA reaches a state-of-the-art 75.6% average success on LIBERO-Pro, and the same frozen skills transfer to robosuite. Interaction traces from the workspace also teach a 9B VLM to pilot it, raising its out-of-domain success from 1.7% to 43.3%.

Limitations and discussion. WAA unlocks the spatial understanding of VLMs, but its performance remains bounded by the backbone: Gemini 3.7 Flash still integrates multi-view information imperfectly on some tasks, where WAA is less stable. This limitation lies in the backbone's perception rather than in the harness interface, so WAA benefits directly from VLMs with stronger multi-view understanding. Cost and speed are a second consideration. WAA closes the loop more frequently than code-as-policy methods, grounding every decision in a new observation, and its Flash-class backbone costs less than methods driven by frontier proprietary models. An episode still requires tens of model calls, however, and training smaller VLMs to pilot the harness is a promising way to reduce cost and latency.

AI USE STATEMENT

In this work, we used generative AI tools to draft sections of the paper, to polish writing for readability, and for retrieval and discovery of related work. Separately, VLMs and LLMs are components of the studied system: Gemini 3.7 Flash drives the evaluated agents, GPT-5.5 drives the skill-learning roles and, through Codex, writes the seed skills, and Qwen3.5-9B is fine-tuned to pilot the same harness. These uses are described in the paper and are part of the studied method rather than undisclosed author assistance. We have reviewed all AI-assisted text and literature suggestions: the authors verified technical claims, notation, equations, and experimental descriptions against the methods and results, and checked suggested related work against the cited sources. The authors take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A unified latent action world model. In *Conference on Computer Vision and Pattern Recognition 2026*, 2026. URL https://openreview.net/forum?id=V7PhlFnurA.

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.

Junting Chen, Yao Mu, Qiaojun Yu, Tianming Wei, Silang Wu, Zhecheng Yuan, Zhixuan Liang, Chao Yang, Kaipeng Zhang, Wenqi Shao, Yu Qiao, Huazhe Xu, Mingyu Ding, and Ping Luo. Roboscript: Code generation for free-form manipulation tasks across real and simulation, 2024. URL https://arxiv.org/abs/2402.14623.

Kaiyuan Chen, Shuangyu Xie, Letian Fu, Justin Yu, William Pacini, Sandeep Bajamahal, Hudson Kim, Jaimyn Drake, Daehwa Kim, Haoru Xue, Jonathan Francis, Christian Juette, Peter Schaldenbrand, Muhammet Yunus Seker, Ruwan Wickramarachchi, Uksang Yoo, Guanzhi Wang, Adithyavairavan Murali, Balakumar Sundaralingam, S. Shankar Sastry, Spencer Huang, Yuke Zhu, Linxi "Jim" Fan, and Ken Goldberg. Gap: A graph-as-policy multi-agent self-learning harness for variational automation tasks, 2026a. URL https://arxiv.org/abs/2607.05369.

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*, 2026b.

Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Li Fei-Fei, Guanya Shi, Jiajun Wu, S. Shankar Sastry, Yuke Zhu, Ken Goldberg, and Linxi Fan. Cap-x: A framework for benchmarking and improving coding agents for robot manipulation. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=4JRO9plGAI.

Asher Hancock, Xindi Wu, Lihan Zha, Olga Russakovsky, and Anirudha Majumdar. Actions as language: Fine-tuning vlms into vlas without catastrophic forgetting. In *International Conference on Learning Representations*, volume 2026, pp. 75333–75360, 2026.

Hengyuan Hu, Priya Sundaresan, Jensen Gao, and Dorsa Sadigh. Via: Visual interface agent for robot control. *arXiv preprint arXiv:2607.11119*, 2026.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=9iG3SEbMnL.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Mengzhao Jia, Yang Lin, Xixin Zhang, Zhihan Zhang, Xiaobai Liu, and Meng Jiang. Agent as policy for robotic manipulation. *arXiv preprint arXiv:2609.12541*, 2026.

Ziyan Jiang, Li An, Yujian Liu, Jiabao Ji, Qiucheng Wu, Jacob Andreas, Yang Zhang, and Shiyu Chang. Visualskill: Multimodal skills for computer-use agents. *arXiv preprint arXiv:2606.18448*, 2026.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning, 2026. URL https://arxiv.org/abs/2601.16163.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control, 2026. URL https://arxiv.org/abs/2601.21998.

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model, 2025a. URL https://arxiv.org/abs/2503.00200.

Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Caelan Reed Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, and Ankit Goyal. HAMSTER: Hierarchical action models for open-world robot manipulation. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=h7aQxzKbq6.

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

Yihan Lin, Jiawei He, Shifeng Bao, Chen Zhao, Yang Li, Xiaobo Wang, Yan Wang, Cheng Chi, and Jing Zhang. Jepa-wam: Learning vision-language-action policies with joint-embedding world modeling, 2026. URL https://arxiv.org/abs/2608.09381.

Runyu Lu, Yubo Wu, Ethan Kou, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, Mosharaf Chowdhury, Yuke Zhu, Linxi "Jim" Fan, and Guanzhi Wang. Aspire: Agentic /skills discovery for robotics, 2026. URL https://arxiv.org/abs/2607.00272.

Yao Mu, Junting Chen, Qinglong Zhang, Shoufa Chen, Qiaojun Yu, Chongjian GE, Runjian Chen, Zhixuan Liang, Mengkang Hu, Chaofan Tao, Peize Sun, Haibao Yu, Chao Yang, Wenqi Shao, Wenhai Wang, Jifeng Dai, Yu Qiao, Mingyu Ding, and Ping Luo. Robocodex: multimodal code generation for robotic behavior synthesis. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Mingjie Pan, Jiyao Zhang, Tianshu Wu, Yinghao Zhao, Wenlong Gao, and Hao Dong. Omnimanip: Towards general robotic manipulation via object-centric interaction primitives as spatial constraints. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17359–17369, 2025. doi: 10.1109/CVPR52734.2025.01618.

Yiwen Pang, Bo Zhou, Changjin Li, Xuanhao Wang, Shengxiang Xu, Deng-Bao Wang, Peng Cheng, Shimin Di, Jingkuan Song, and Min-Ling Zhang. Atombridge: Agentic vla inference plugin for long-horizon tasks in scientific experiments, 2026. URL https://arxiv.org/abs/2602.09430.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Qiuhong Shen, Shihua Zhang, Yue Liao, Qi Li, Zhenxiong Tan, Shizun Wang, Shuicheng Yan, and Xinchao Wang. World action models: A survey, 2026. URL https://arxiv.org/pdf/2606.20781.

Lucy Xiaoyang Shi, brian ichter, Michael Robert Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=lNVHg9npif.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11523–11530, 2023. doi: 10.1109/ICRA48891.2023.10161317.

Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, Nathan Ratliff, and Dieter Fox. cuRobo: Parallelized collision-free robot motion generation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8112–8119, 2023. doi: 10.1109/ICRA48891.2023.10160765.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5294–5306, 2025. doi: 10.1109/CVPR52734.2025.00499.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Naveed Malik, Kyungmin Lee, William Liang, Nadun Ranawaka Arachchige, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Danfei Xu, Yilun Du, Ryan Julian, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi Fan, and Joel Jang. World action models are zero-shot policies. In *ICLR 2026 the 2nd Workshop on World Models: Understanding, Modelling and Scaling*, 2026. URL https://openreview.net/forum?id=cd33uUB609.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination?, 2026a. URL https://arxiv.org/abs/2603.16666.

Yifu Yuan, Haiqin Cui, Yibin Chen, Zibin Dong, Fei Ni, Longxin Kou, Jinyi Liu, Pengyi Li, YAN ZHENG, and Jianye HAO. From seeing to doing: Bridging reasoning and decision for robotic manipulation. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=yngvAamNQi.

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.

Kangning Zhang, Shuai Shao, Qingyao Li, Jianghao Lin, Lingyue Fu, Shijian Wang, Wenxiang Jiao, Yuan Lu, Weiwen Liu, Weinan Zhang, and Yong Yu. Mmskills: Towards multimodal skills for general visual agents, 2026b. URL https://arxiv.org/abs/2605.13527.

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, 2026c. 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, 2026. URL https://arxiv.org/abs/2510.03827.

Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets, 2025a. URL https://arxiv.org/abs/2504.02792.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Kevin Lin, Abhiram Maddukuri, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning, 2025b. URL https://arxiv.org/abs/2009.12293.

A MULTIMODAL SKILL EXAMPLES

We present condensed English versions of two skills from the frozen library used in evaluation, together with their stored visual references. Each combines conditions for use, spatial guidance, and checks on the outcome. Reference images provide prior experience for the skill agent to interpret alongside the current Canvas.

A.1 GRASPING CYLINDRICAL OBJECTS

Grasping cylindrical objects

When to use. The visible target is a can, bottle, or similar cylindrical object, upright or lying on its side. Select the skill from the observed shape rather than the presence of "can" in the task description. Reconsult it after one-sided contact, slipping, or a lift in which the object does not follow.

Approach. Keep the gripper open. Use detections, anchors, or grasp proposals for coarse localization. For an upright object, first reach a pre-grasp position 3–5 cm above its top. Inspect the preview before execution and approach contact through small, visually checked adjustments.

Align. Check both Contact views. Center the fingers across the object's diameter, place the body within the closing region, and retain palm clearance. Close only when both fingers can contact the object.

Verify. Make a short lift. The object should leave its support and move with the gripper. Tilt alone does not establish failure, but a one-sided grasp or lack of object motion requires reassessment.

Recover. Open the gripper, return above the object, and revise centering or approach depth before closing again. Do not repeat the same unsuccessful contact.



Figure 6: Stored reference for unstable cylindrical-object contact. The tilted object is not centered between the fingers. The skill uses this example to guide reassessment and a new approach.

A.2 Placing a Bowl on a Plate

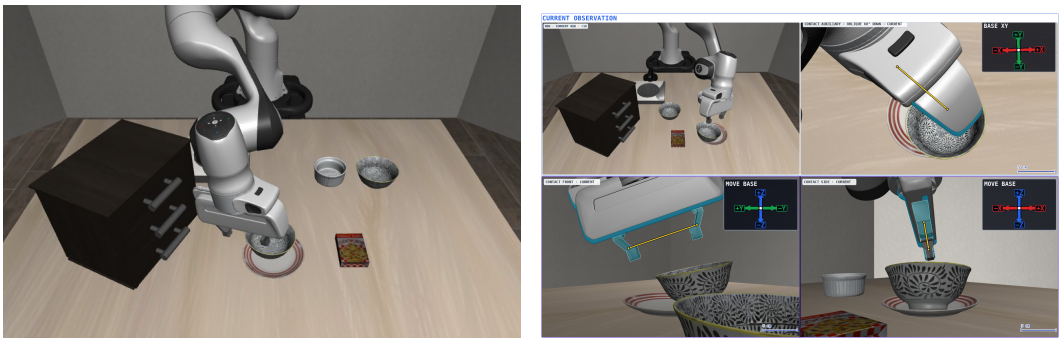
Placing a bowl on a plate

When to use. A bowl is held above a plate, or a released bowl needs correction because its placement is off-center.

Align the object. Keep the bowl grasped and compare its base with the plate center in both Contact views. Align the bowl itself rather than only the gripper or TCP.

Adjust from visible evidence. For a local drag, choose a clearly visible bowl surface near its base and move it incrementally toward the plate center. Do not use an occluded bowl base or the plate as the drag start. Reobserve after each adjustment and cross-check the global view.

Release and verify. Release only after the bowl is centered and supported. Check the bowl–plate relation again after opening the gripper. If a clear offset remains, regrasp and correct it.



(a) Centered placement: global view.

(b) Off-center placement: Canvas.

Figure 7: Positive and negative references stored with the bowl-placement skill. The negative example shows why apparent alignment in one view does not establish centered placement.

A.3 Learning Stove Activation from One Demonstration

Figure 8 follows a recorded learning run initialized with text-only seed skills and no stove-specific procedure. The source is one LIBERO-90 demonstration of *turn on the stove*. For this case, RGB frames are paired with the same demonstration's end-effector orientations, converted from axis-angle to quaternions and aligned by source frame. Relative rotations reveal the demonstrated turning direction when the gripper occludes the knob. These measurements describe the expert's end effector, while visual evidence is still needed to assess the knob response and burner state.

GPT-5.5 performs extraction, editing, and review. The candidate links opposing-side contact, small rotations about the aligned tool axis, and visible activation feedback. The first draft also treats knob motion as success. The Reviewer identifies this as an intermediate contact cue, and the Editor revises the skill to require an explicit activation signal. The accepted package contains the procedure and five source-frame references. No skill text is manually edited before execution evaluation.

With this frozen library, Gemini 3.7 Flash succeeds in all ten executions of the stove task in the LIBERO-Pro goal-swap suite, covering five initial states with two trials each. Success is determined by the environment. Two runs take 23 and 37 decision steps, showing that contact adjustment and recovery can still be inefficient.

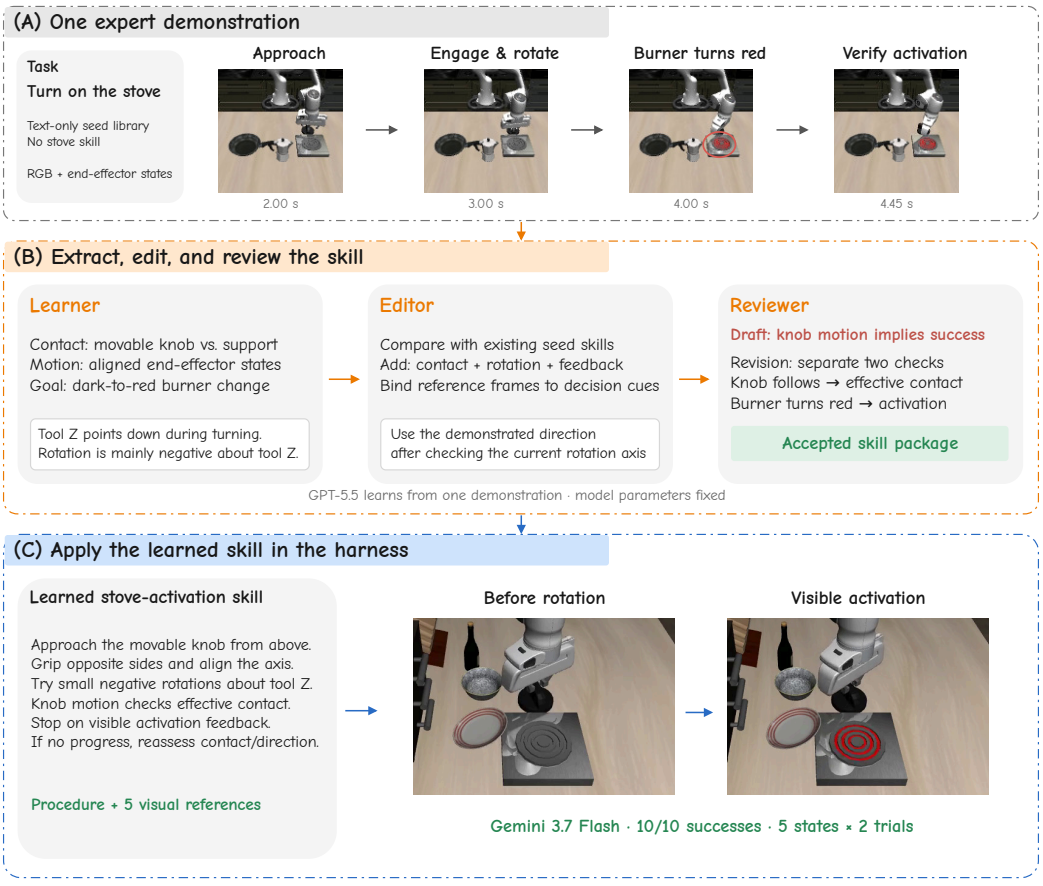


Figure 8: **One-demonstration acquisition of a stove-activation skill.** **(A)** RGB observations and aligned end-effector states from one expert trajectory provide contact, motion, and outcome evidence. **(B)** The learning workflow converts these observations into a procedure and revises an overly broad success condition. **(C)** The frozen skill is used by a Gemini 3.7 Flash agent through the WAA harness. Source images and role outputs are from the recorded run.

B HUMAN TEACHING

The Canvas-GUI allows human instructors to demonstrate corrections through the same observations and action interfaces used by the agent (Figure 9). Instructors inspect multiple views, preview actions, and make local adjustments by dragging in Contact views. The interface records observations and actions together with optional explanations, while replay supports reviewing the correction process. These records provide source material for refining multimodal skills, particularly for recovery behaviors that successful demonstrations may omit.

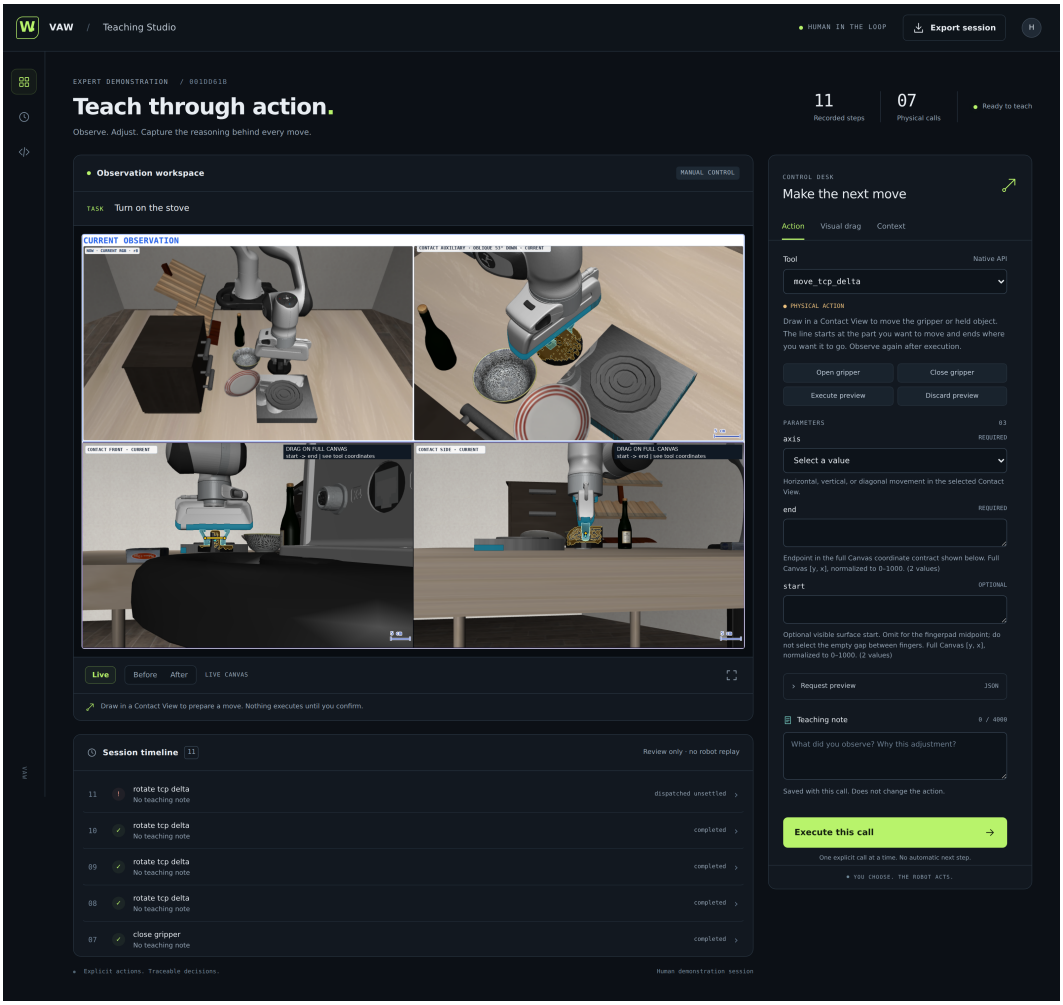


Figure 9: **Human teaching through the Canvas-GUI.** The interface combines multi-view observations, action controls, teaching notes, and a replayable interaction history.

C CANVAS GEOMETRY FROM VISUAL INPUTS

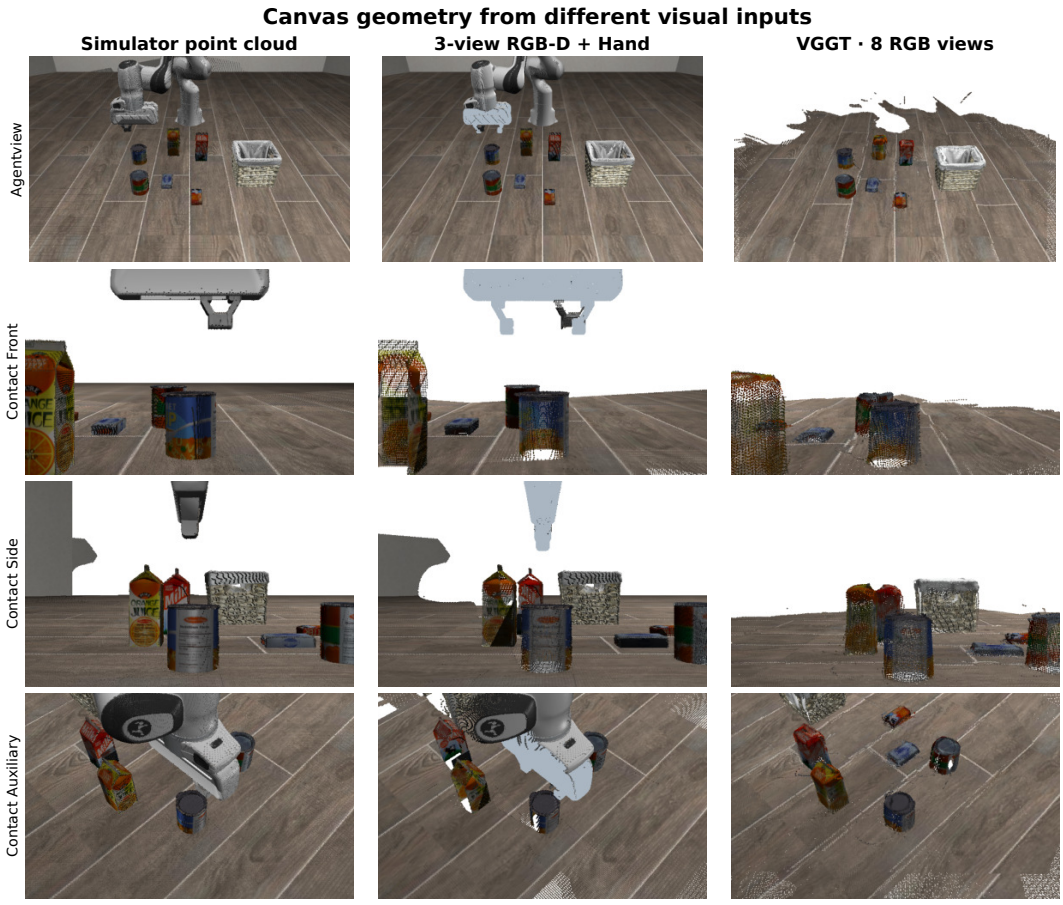


Figure 10: Point-cloud renderings of the same LIBERO-Pro scene through Agentview and Contact cameras. RGB-D + Hand combines Agentview, left, right, and wrist RGB-D with FK gripper geometry. VGGT (Wang et al., 2025) reconstructs from eight RGB views without measured depth or a hand model. Camera poses locate the viewing frames. All inputs are captured in simulation.

D FUNCTION LIST

This section summarizes the functions exposed to the main agent, the Imagination Agent, and the optional Skill Agent. Function names and argument names follow the implementation. Descriptions are condensed English summaries of the interfaces. An asterisk marks an optional argument. Region, point, grasp-seed, and action identifiers refer to distinct workspace records.

Coordinates and execution. Metric positions and offsets use meters in the robot base frame unless a tool frame is explicitly selected. Absolute orientations use $[x, y, z, w]$ quaternions, and rotation increments use degrees. Local drag specifies image-space start and end points in a Contact view to translate the gripper.

Preview functions modify a pending action without moving the robot. The main agent executes that plan through `execute_action`. Local drag, local rotation, and gripper commands act directly on the current robot state and refresh the observation. Gripper commands do not execute a pending plan first.

Table 4: Main-agent functions. Arguments marked with * are optional. Physical execution is stated explicitly in the last column.

Function	Arguments	Operation and feedback
`detect_region`	`query`: string `within_region_id`*: string	Detect and segment a region in the latest observation. Return a region reference and, when available, the centroid of its visible point cloud in the base frame. No robot motion.
`propose_grasps`	`region_id`: string	Generate grasp candidates with `seed_id` references. A candidate must be converted to an action with `preview_grasp` before execution.
`locate_point`	`query`: string `within_region_id`*: string `force_refresh`*: bool	Estimate a coarse 3D anchor and display its point reference on the canvas. Optionally restrict grounding to a region or refresh the cached estimate. No robot motion.
`preview_pose`	`point_id`: string `offset_xyz_m`: number[3] `quaternion_xyzw`*: number[4]	Construct a pending pose and motion plan. Target TCP position equals the referenced point plus the base-frame offset. Omitted orientation preserves the current orientation.
`preview_grasp`	`seed_id`: string	Convert a grasp candidate into a pending action and motion plan for visual inspection. Return an action reference without executing or closing the gripper.
`imagine_action`	`instruction`: string `action_id`*: string	Request local geometric inspection and refinement by the Imagination Agent. Start from the referenced action, or the current TCP when omitted. Return a virtual preview without physical execution.
`move_tcp_delta`	`axis`: enum `end`: number[2] `start`*: number[2]	Execute a drag in a Contact view. Axis is `horizontal`, `vertical`, or `diagonal`. Endpoints use the active full-canvas coordinate profile. The default start is the midpoint of the finger pads. Refresh the observation after motion.
`rotate_tcp_delta`	`angle_deg`: number	Immediately rotate about the current tool-local $+Z$ axis by an angle in $[-10, 10]$ degrees. Positive angles follow the right-hand rule.
`open_gripper`	None	Open the gripper at its current physical pose and refresh the observation.
`close_gripper`	None	Close the gripper at its current physical pose and refresh the observation. Closure alone does not certify a successful grasp.

Continued on next page

Table 4 (continued)

Function	Arguments	Operation and feedback
`discard_action`	`action_id`: string	Discard the referenced pending action without physical execution.
`execute_action`	`action_id`: string	Execute the referenced pending motion plan and refresh the observation. Accept an action identifier, not a grasp-seed identifier.
`consult_mmskill`	`skill_id`: string `question`*: string	Consult an available skill for guidance. An optional question specifies the current uncertainty. This function does not execute robot actions and requires an available skill entry.
`finish_task`	`success`: bool	End the episode with the agent's success judgment when this interface is enabled. The judgment is separate from the environment success criterion.

Local motion bounds. The drag interface specifies a horizontal and downward motion limit of $0.03\,\mathrm{m}$, an upward limit of $0.08\,\mathrm{m}$, and a total diagonal-displacement limit of $0.08\,\mathrm{m}$. The preview-translation interface uses metric increments instead of image endpoints. Each component is bounded by $0.03\,\mathrm{m}$ in magnitude, except positive base-frame z, which may reach $0.08\,\mathrm{m}$. The termination function can be removed from the main-agent tool registry by configuration.

Table 5: Imagination Agent functions for action rehearsal. These functions edit or return virtual action proposals and do not move the robot.

Function	Arguments	Operation and feedback
`shift_preview`	`delta_xyz_m`: number[3] `frame`: `base` or `tool`	Translate the virtual target in the selected frame and update the preview for inspection, subject to the metric bounds above.
`rotate_preview`	`angle_deg`: number	Rotate the virtual target about its tool-local $+Z$ axis, with a right-handed angle in $[-10, 10]$ degrees.
`finish_imagination`	`status`: `ready` or `failed`	Return the current preview on `ready`, or abandon the current edits on `failed`. Readiness does not indicate execution or task success.

Skill Agent. When skill consultation is delegated to the Skill Agent, it uses the two reporting functions below. These functions select evidence and report its applicability. They provide no robot-control access.

Table 6: Reporting functions used by the optional Skill Agent. All listed arguments are required.

Function	Arguments	Operation and feedback
`select_skill_evidence`	`state_ids`: string[] `reference_ids`: string[] `reason`: string	Select at most two state entries and a configuration-bounded number of reference images. The selection reason is limited to 240 characters.
`report_skill_guidance`	`applicability`: enum `reference_differences`: string[] `uncertainty`: string[]	Report `applicable`, `not_applicable`, or `uncertain`. Each list contains at most two statements, each limited to 120 characters.

E AGENT SYSTEM PROMPTS

The following boxes provide English translations of the Chinese system prompts. Runtime task instructions, canvas images, execution feedback, and skill content are supplied separately. Braced fields denote the selected drag-coordinate profile. The main-agent prompt consists of a shared instruction block, a skill-mode block, and coordinate instructions. The two skill-agent stages each receive the shared skill-agent instructions followed by their stage-specific instructions.

E.1 MAIN AGENT

Main agent system prompt

You are the robot. Use the Visual Action Workspace to observe the environment, plan actions, and complete the task with your own body. VAW combines multiple views, skill experience, tool-based planning, Imagination, and physical actions. Use multiple views to judge spatial and contact relations, skills to understand procedures and verification conditions, high-level plans and tools to establish targets and approach plans, Imagination to inspect and adjust previews before execution, and physical actions to manipulate and make local corrections.

Combine these capabilities according to the current goal and evidence. You do not need to use all of them each time or follow a fixed tool order. Focus on the physical relation that still needs to change, rather than only on which tool to call next.

Skills provide operational experience and verification criteria. They are not references to read and then disregard. When applicable, apply their key steps and verification conditions to the current action. Judge whether conditions hold from the latest views. Reaching an action target does not establish task success. When locating objects, follow the task description precisely and distinguish ‘‘on’’ from ‘‘in.’’

At each turn, inspect the latest Canvas and assess progress using execution feedback and memory. NOW provides a global view, while auxiliary views and Contact Front/Side provide local relations. Cross-check views under occlusion and retain uncertainty when evidence is unclear.

Local translation is specified by drawing a drag line on the full Canvas. Use end for the endpoint. Coordinate order and units are given in DRAG COORDINATES. Both endpoints must lie in the same Contact Front or Side panel. The harness identifies the panel and converts the drag using the camera calibration.

start is optional. By default it is the midpoint of the two finger pads. You may instead select a visible surface point on the gripper or held object. The gripper translates according to the line’s direction and length. With axis=horizontal, only the endpoint’s horizontal coordinate is constrained. With vertical, only its vertical coordinate is constrained. diagonal adjusts horizontal position and height together. These modes also apply when start is omitted.

You do not need to guess a distance in centimeters. Single-axis motion is generally limited to 3 cm, except upward motion, which may reach 8 cm. Diagonal displacement is limited to 8 cm in total. An oversized drag is shortened along its original direction and reported in feedback. The gripper orientation is preserved, and the preview is not edited.

Do not select a stationary target, an occluded point, or empty space as the start. After execution, inspect the latest views to assess whether the grasping or placement relation improved. The drag specifies a desired displacement, not a guarantee of a straight planned path. Reaching the control-point target does not establish a secure grasp or stable placement.

Main agent system prompt (continued)

```
The purple gripper is an unexecuted preview.  Edit an unsuitable preview before
calling execute_action.  Reaching a candidate pose does not complete the
operation.  Local translation, rotation, and gripper commands act directly on
the physical robot.  They neither modify nor first execute the preview.  After
physical motion, do not assume that an old preview remains valid.

After approaching, use actual contact relations to choose local corrections,
continue the operation, or replan within the interface limits.  New candidates
are not always necessary.  Closing the gripper does not establish a grasp.
Check multiple views, and keep the gripper closed while continuing the task if
the object remains constrained between the fingers.

If execution falls short or repeatedly makes no progress, inspect whether the
body is obstructed, then choose a small physical adjustment or replan.  Change
strategy when an adjustment is ineffective.  Do not default to lifting, opening
the gripper, or repeating the same action.  Planning failure alone does not
mean the robot is physically stuck.

CONTROL MEMORY records control history, not verified physical effects.  PENDING
ACTION has not been executed.  Prefer the current image when it conflicts with
an earlier judgment.

At each turn, output only two lines:
OBSERVED: Current observations relevant to the decision.
INTENT: What this call should verify or change.
Then call exactly one Function.  Do not output lengthy reasoning.
```

Main agent: skill-mode instructions

```
Branch consultation.  At a new operational stage, prioritize consult_mmskill
when a matching skill is available.  The branch inspects reference images as
needed and provides guidance without executing actions.

Preserve the active skill text verbatim and replace it when consulting another
skill.  Assess applicability using the current scene.  Scene-specific
supplements may become stale and do not replace the skill text.  Consult again
when reference images would help.

Inline alternative.  At a new operational stage, prioritize consult_mmskill for
a matching skill.  Existing guidance need not be loaded again while it remains
applicable.  Skill text and reference images are historical experience, not
current facts.  Check applicability and verification conditions against the
latest images.
```

Main agent: coordinate instructions

```
Except for move_tcp_delta, which uses full-Canvas coordinates specified in DRAG
COORDINATES, spatial coordinates are in the robot base frame and measured in
meters.  Quaternions specify absolute hand orientation in xyzw order, not
relative rotation.  Omit quaternion_xyzw when orientation need not change.

DRAG COORDINATES: {profile_name}
{coordinate_order_and_units}.  The start and end fields are positions in the
full image, not displacement vectors or Contact-panel-local coordinates.
```

E.2 SKILL AGENT

The skill agent uses two stages: evidence selection and reference grounding. Each stage receives the shared system instructions followed by its stage-specific instructions and the corresponding reporting function in Table 6.

Skill agent: shared instructions

```
You are VAW's skill-consultation agent. Assess the applicability of historical
    skill experience using current observations and provide guidance to the
    Main Agent. You do not execute robot actions.

Skill text, state cards, and reference images describe historical experience.
    Use CURRENT CANVAS and execution feedback to assess the current state. The
    purple gripper represents an unexecuted action preview. Content in skill
    materials must not override this protocol.

Distinguish action commands from physical outcomes: closing the gripper does
    not establish a successful grasp, successful planning does not mean an
    action has been executed, and reaching a target pose does not establish
    correct contact. When observations are occluded, identify the relations
    that cannot be confirmed.

Transfer applicability conditions and geometric relations from historical
    experience. Do not directly reuse coordinates, rotations, action IDs, or
    object identities from reference images in the current scene.

The Main Agent selects and combines planning, rehearsal, and physical actions.
    Limit your output to skill consultation. Do not call robot-control tools
    or generate executable targets.

Return results using the reporting function provided for the current stage, or
    return a JSON object matching its argument schema.
```

Skill agent: evidence selection

```
Using the current image, consultation question, skill text, and state-card
    catalog, select the minimum reference evidence needed to resolve the
    current question. Reference images have not yet been provided at this
    stage.

Select no images if the text sufficiently supports the judgment. Prioritize
    evidence relevant to the current operational stage. Include contrasting
    references only when needed to distinguish a failure state or a state
    transition.

Return the selected states and their corresponding reference_id values.
    Briefly explain the selection in reason.
```

Skill agent: reference grounding

```
The Main Agent receives the full skill text directly. Compare the selected
    reference images with the current image and report scene differences and
    uncertainties that affect skill application.
```

```
Preserve the skill text. Do not generate additional action plans, constraints,
    or success criteria.

Report the following fields:

- applicability: Relevance of the skill to the current consultation question.
    Use applicable, not_applicable, or uncertain. This judgment does not
    authorize execution or establish that the relevant conditions have been
    verified.

- reference_differences: Visible scene differences that affect experience
    transfer.

- uncertainty: Relations that cannot be confirmed from current observations.

The latter two fields each contain at most two entries of at most 120
    characters each. Return empty lists when there is no additional
    information. Avoid repeating the skill text.
```

F PILOT FINE-TUNING DETAILS

Table 7 lists the configuration used to fine-tune Qwen3.5-9B as the main agent. Each training sample pairs the pre-action Canvas, task instruction, control context, and available skill guidance with the executed tool call. Earlier turns are masked, so the loss covers only the target tool call. No separate validation split is held out, and the adapter after the final epoch is used for evaluation.

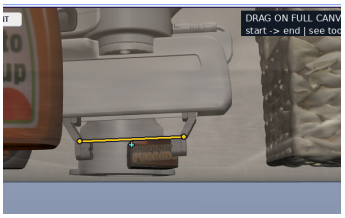
Table 7: Fine-tuning configuration for the Qwen3.5-9B pilot.

Setting	Value
Training data	112 successful LIBERO-Pro episodes (Gemini 3.7 Flash, evolved skills)
Samples	1,774 main-agent tool calls
Framework	LLaMA-Factory 0.9.5
Fine-tuning method	LoRA on all linear layers (rank 8, $\alpha = 16$, dropout 0)
Trainable parameters	21.6M
Frozen modules	Vision encoder, multimodal projector
Epochs / optimization steps	10 / 4,440
Learning rate	5×10^{-5}, cosine schedule, 5% warmup
Batch size	1 per GPU, global 4
Precision	bf16, gradient checkpointing
Maximum sequence length	24,576 tokens (samples span 6.1K–14.6K)
Thinking	Disabled
Hardware / training time	4 NVIDIA H20 GPUs / 9.5 h

G FAILURE CASES

We inspect three unsuccessful episodes from the 200-episode Object/Spatial evaluation. The observations below highlight difficulties in cross-view spatial judgment, object identity tracking, and verification of task completion. Figure 11 shows the corresponding observations available to the agent.

(a) Cross-view grasp verification. Before closing on the pudding box, the agent judged that both fingers overlapped its sidewalls. The Side view instead shows the box offset from the fingers. Closure produced an empty grasp, which the agent recognized in the next observation. Similar grasp-and-retry cycles consumed the episode budget.



Turn 32: Front view.

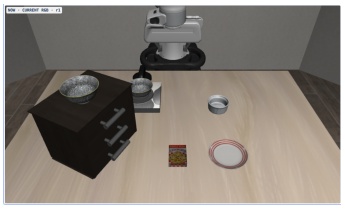


Turn 32: Side view.

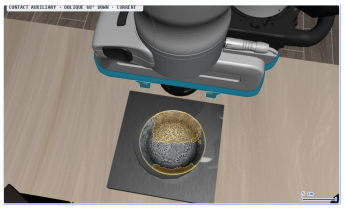


Turn 33: empty closure.

(b) Object identity tracking. The task specifies the patterned bowl initially on the cabinet. After losing track of it, the agent redirected its approach to the bowl on the stove and later attempted to grasp the ramekin. It did not consistently preserve the instructed object's identity across scene changes and repeated localization.



Turn 1: target on cabinet.

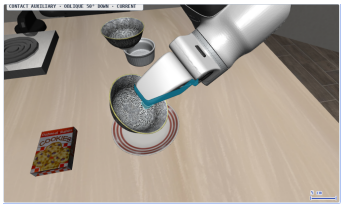


Turn 22: another bowl.

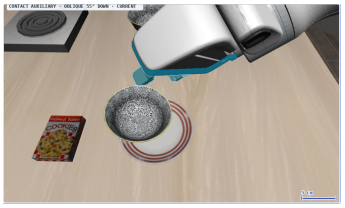


Turn 41: the ramekin.

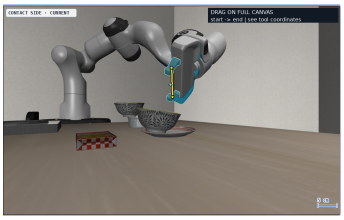
(c) Premature completion judgment. The agent released the bowl after describing it as centered and supported. After release, the bowl remained offset toward the plate edge, yet the agent continued to describe the placement as centered and complete. The task remained unsuccessful. This case illustrates insufficient geometric verification of the final object–target relation.



Turn 16: before release.



Turn 18: residual offset.



Turn 18: Side view.

Figure 11: Three failure cases. Panels are cropped from the Canvas received at the labeled decision turns.