SHOW-HARNESS: VLM AGENTS CAN PLAY ROBOTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Foundation vision–language models (VLMs) exhibit broad intelligence about the world, yet translating this intelligence into robot control remains challenging. We present Show-Harness, an *Embodied Harness* that enables VLMs to **"play" robots** through a compact semantic interface linking intent to action. Show-Harness exposes discrete semantic action units that VLMs can naturally reason over, while embodiment-specific interpreters deterministically ground them into local robot actions, keeping the VLM directly responsible for each step's physical decisions. Through the same interface, Show-Harness demonstrates the feasibility of (1) directly unlocking closed-source frontier VLMs for zero-shot robot control, and (2) adapting small-scale open-source VLMs for low-cost deployment with just a few GPU-hours of fine-tuning. We further develop GUMI, a GUI-based manipulation interface that enables humans and agents to "play" robots and collect demonstrations in the same semantic action space across embodiments. Extensive experiments show that Show-Harness-equipped VLM agents generalize robustly across tasks, embodiments, and environments, outperforming representative agentic and VLA paradigms. These results suggest that the right interface can unlock substantial embodied capability from foundation VLMs, without requiring additional model capacity or costly embodiment-specific pretraining.

1 INTRODUCTION

Foundation vision–language models (VLMs) already encode much of what robot manipulation requires, from recognizing objects and spatial relations to decomposing long-horizon goals (Driess et al., 2023; Yuan et al., 2024; Ji et al., 2025; Team et al., 2025). Yet this knowledge does not readily translate into robot behavior. Vision–language–action (VLA) models pull VLMs toward low-level control by fine-tuning them to predict embodiment-specific continuous actions, collapsing broad semantic knowledge into an opaque pixel-to-actuation mapping that often requires repeated adaptation across tasks, environments, and embodiments (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; O'Neill et al., 2024). Hierarchical and programmatic systems take the opposite route: the VLM produces explicit intermediate abstractions, including subtask-level calls (Ahn et al., 2022; Huang et al., 2022; Shi et al., 2025; Yang et al., 2025b) and programs over hand-designed control APIs (Liang et al., 2023; Huang et al., 2023a; Chen et al., 2024a). Despite their effectiveness, the physical control is mediated by downstream controllers or system-specific mechanisms, weakening the direct link between semantic intent and physical execution.

We argue that bringing foundation model intelligence into the physical world requires a suitable interface: an action space that is semantically meaningful to the VLM yet sufficiently fine-grained for direct physical control. We introduce **Show-Harness**, a model-agnostic *Embodied Harness* that enables VLMs to **"play" robots** through such an interface. As shown in Fig. 1, Show-Harness reformulates robot control into a set of semantic action units. Each unit specifies a single movement step in a given direction or a gripper action, making it directly interpretable to the VLM. An embodiment-specific interpreter then deterministically grounds each unit into a small, bounded robot motion, letting every semantic decision land in the physical world. Around this core, the harness closes the loop: it organizes multi-view observations and proprioception into a perceptual context, supports subtask reasoning and recovery, and returns execution feedback after every unit, turning digital VLMs into *situated agents* that reason and act in a semantic space they natively understand, choosing the direction of every step while the interpreter sets its magnitude.

Show-Harness enables two modes of robot control. First, it seamlessly turns closed-source frontier VLMs into zero-shot robot agents without fine-tuning, outperforming representative agentic har-

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.

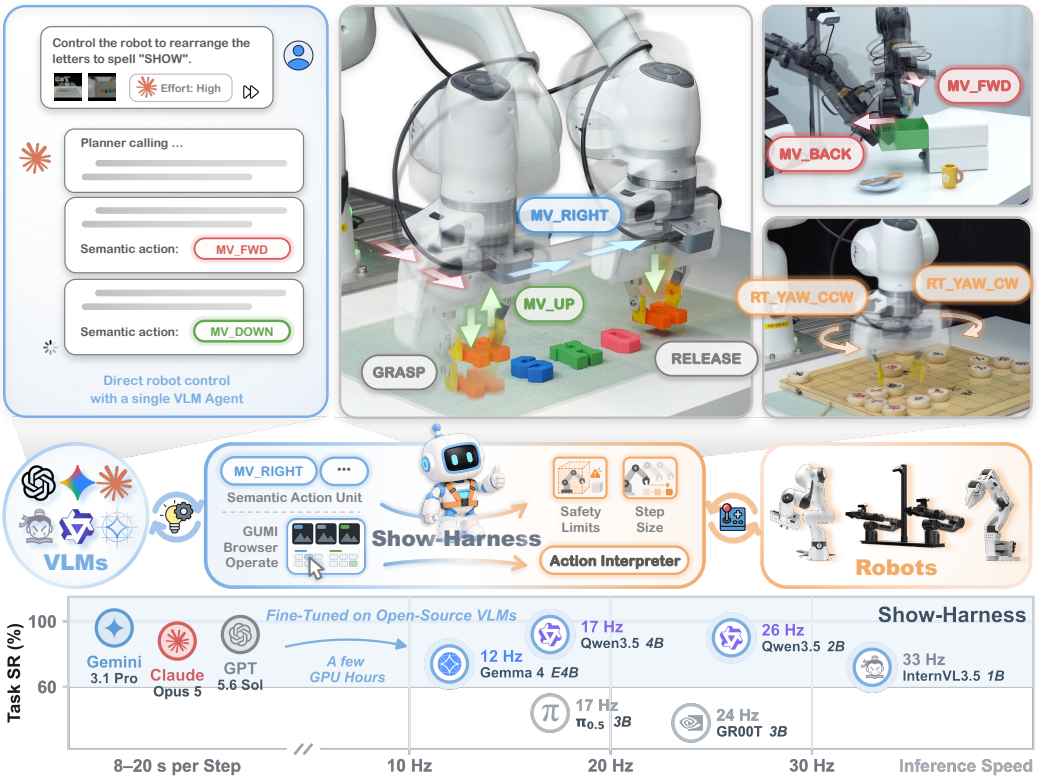


Figure 1: Show-Harness connects foundation VLMs to diverse robot embodiments through one semantic action interface, enabling direct zero-shot deployment with frontier models and capable, efficient control via lightweight adaptation of compact open-source models.

nesses across tasks, embodiments, and environments. Second, it enables lightweight VLMs (e.g., 2B-scale models) to "play" robots in the same action space with just a few GPU-hours of fine-tuning, achieving stronger generalization and sim-to-real transfer than representative VLA paradigms under controlled experiments. Together, these results suggest that a suitable semantic interface can unlock substantial embodied capability from foundation VLMs, providing a scalable path that inherits advances in frontier models while extending such control to smaller, lower-cost models.

Since the action space is interpretable and directly operable by both humans and models, it naturally unifies VLM-based robot agents with GUI-style human control. We further develop a GUI-based manipulation interface (GUMI), enabling cross-embodiment and human–agent demonstration collection without specialized teleoperation hardware. The main contributions of this work are:

- We introduce Show-Harness, a model-agnostic *Embodied Harness* that enables foundation VLMs to directly operate robots through a compact semantic action interface.
- Show-Harness demonstrates the feasibility of directly unlocking closed-source frontier VLMs for zero-shot control, and efficiently adapting small-scale models for low-cost deployment.
- Show-Harness-equipped agents demonstrate strong generalization across tasks, embodiments, and environments, outperforming representative agentic and VLA paradigms. Further studies on physical and semantic adaptability reveal key properties of effective VLM–robot interfaces.

2 Related Work

Foundation models as low-level policies. Vision–language–action (VLA) models repurpose pretrained VLMs to predict low-level robot controls, in the forms of continuous action chunks (Zhao et al., 2023; Chi et al., 2025; Liu et al., 2025; Shukor et al., 2025; Team et al., 2025; Hu et al., 2026b), discretized motor tokens (Brohan et al., 2022; 2023; Kim et al., 2024; Pertsch et al., 2025; Jiang et al., 2026), keyframe or spatial-grid actions in task space (Shridhar et al., 2023; Qu et al.,

2025), or latent action codes learned from robot trajectories and videos (Lee et al., 2024; Ye et al., 2025; Chen et al., 2025b; Zheng et al., 2025; Bu et al., 2025; Bauer et al., 2025; Mu et al., 2026; Kang et al., 2026). Recent extensions couple action prediction with future visual dynamics in unified video–action or world-action models (Li et al., 2025; Zhu et al., 2025; Ye et al., 2026; Li et al., 2026c; Pan et al., 2026; Yang et al., 2026a). This route, however, trades semantics for control. Refitting the model to embodiment-specific motor signals collapses its broad pretrained knowledge into an opaque sensorimotor mapping and usually requires fresh robot trajectories for new task families or embodiments (Zhong et al., 2025; Karcini et al., 2026).

Foundation models as intermediate decision makers. Another route keeps the VLM above low-level control, using pretrained semantic knowledge to decompose instructions and emit subgoals, keypoints, affordance targets, value maps, or spatial constraints (Driess et al., 2023; Yuan et al., 2024; Ji et al., 2025; Huang et al., 2023b; Duan et al., 2024b; Huang et al., 2024). Downstream controllers then handle their physical realization. This preserves semantics but surrenders physics: the model specifies intent without seeing how it is realized (Hu et al., 2026a; Galanti et al., 2026), and each representation is bound to a carefully engineered, system-specific grounding pipeline (Huang et al., 2023b; Duan et al., 2024b; Huang et al., 2024; Liu et al., 2024). Realistic deployment further requires long-horizon planning, closed-loop feedback, and failure recovery (Galanti et al., 2026), pushing these hierarchical designs toward the agentic systems discussed next.

Agentic robot system and embodied interfaces. Agentic systems harness the VLM for closed-loop robot control (Huang et al., 2022; Yang et al., 2025b; Liu et al., 2026; Lima et al., 2026) while also facilitating memory (Rana et al., 2023; Zhang et al., 2026c; Anwar et al., 2025), feedback-driven monitoring and reflection (Huang et al., 2022; Shinn et al., 2023), and failure-aware replanning and recovery (Liu et al., 2023c; Duan et al., 2024a; Yang et al., 2025b). A central design choice is the embodied interface through which the agent's decisions are realized: agents either generate programs over perception and motion APIs (Liang et al., 2023; Singh et al., 2022; Huang et al., 2023a; Vemprala et al., 2024; Chen et al., 2024a; Fu et al., 2026; Zhao et al., 2026), select predefined skills (Ahn et al., 2022; Rana et al., 2023; Ahn et al., 2024; Yuan et al., 2025; Li et al., 2026a; Santos et al., 2026), steer learned policies (e.g., VLAs) with language subgoals (Shi et al., 2025; Hu et al., 2026a; Zhang et al., 2026c; Chen et al., 2026a; Peng et al., 2026), or offload decisions to symbolic planners and numerical optimizers (Liu et al., 2023a; Chen et al., 2024b; 2025c; Yu et al., 2023). The abstraction level of this interface strongly affects performance (Berman et al., 2026; Hu et al., 2026a): agents are reliable with hand-designed high-level primitives such as `place(object, target)`, but degrade as these are replaced by lower-level ones (Fu et al., 2026; Liu et al., 2026). Prevailing systems therefore expose coarse-grained skills or VLAs as callable primitives (Ahn et al., 2022; Shi et al., 2025; Zhang et al., 2026c; Chen et al., 2026a; Peng et al., 2026), leaving the agent to decide *what* while an opaque executor determines *how* (Galanti et al., 2026). Show-Harness instead exposes the *how* through fine-grained semantic action units with deterministic, transparent physical realization, allowing the VLM to remain directly responsible for fine-grained physical decisions.

From digital interfaces to physical manipulation. Show-Harness's discrete semantic action space echoes a broader interface principle in digital and simulated agents: exposing compact, interpretable actions operable by both humans and foundation models. Computer-use and game agents act through mouse, keyboard, or controller inputs (Lin et al., 2025; Qin et al., 2025; Wang et al., 2025; Ouyang et al., 2026; Zhang et al., 2026b), while simulated embodied agents use discrete or skill-level actions (Li et al., 2023; 2024; Yang et al., 2025a; Li et al., 2026b). In navigation, discrete directional primitives are native to simulators and benchmarks (Savva et al., 2019; Krantz et al., 2020), and recent general-purpose agents can drive them competitively (Zhou et al., 2026). Real-world manipulation, however, lacks a comparably simple and broadly operable interface: control is typically embodiment-specific, high-dimensional, and requires fine-grained spatial precision. Show-Harness bridges this gap by deterministically grounding fine-grained semantic actions into physical robot motion, while exposing the same action space to both VLM agents and humans.

3 Show-Harness

3.1 Overview

Show-Harness is an embodied harness designed to translate the intelligence of foundation VLMs into physically grounded robot behavior by placing the VLM inside an iterative perception–reasoning–action loop. Specifically, as illustrated in Fig. 2, to scaffold foundation VLMs for situated

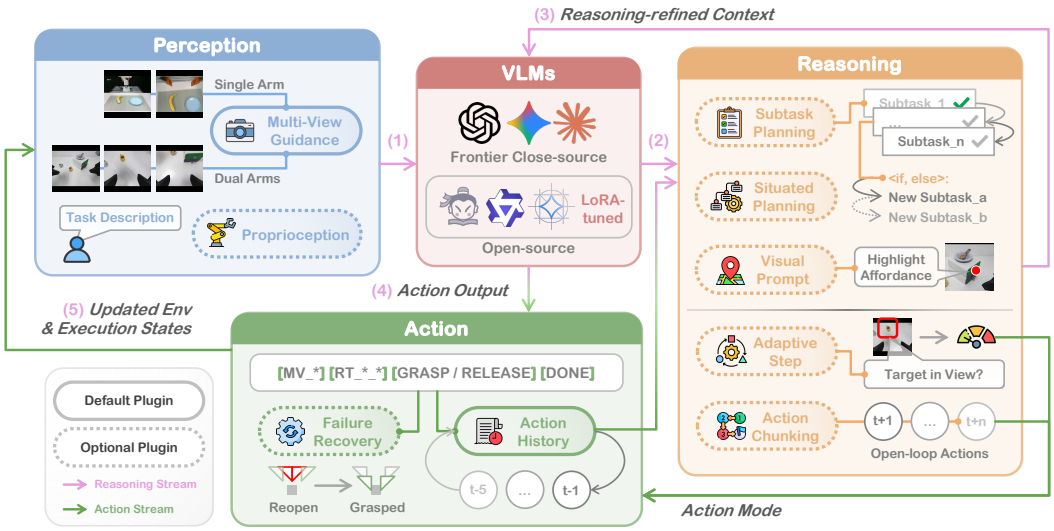


Figure 2: **The Show-Harness architecture.** A modular perceive–reason–act loop connects foundation VLMs to robot control through a shared semantic action interface.

robot interaction, Show-Harness organizes its plugins into three configurable stages: *Perception*, *Reasoning*, and *Action*. Tab. 1 provides an overview of the function of each plugin, while detailed designs are given in the Appendix. Given a language instruction ℓ, at step t the system captures an observation $o_t = (\mathcal{I}_t, p_t)$, where $\mathcal{I}_t$ denotes the multi-view visual observation and p_t the robot's proprioceptive state. The system first processes the current observation and a compact interaction history h_t through a configurable set of reasoning plugins $\mathcal{P}$, producing a reasoning-refined context

$$c_t = \Phi_{\mathcal{P}}(\ell,\, o_t,\, h_t)\,. \tag{1}$$

The central VLM then selects a semantic action conditioned on the refined context c_t,

$$a_t = \pi(c_t) \in \mathcal{A}, \tag{2}$$

where $\mathcal{A}$ is a compact set of fine-grained, actionable units that defines the semantic interface between the VLM and the robot. Each unit specifies a directly interpretable end-effector movement or gripper intent without exposing embodiment-specific control variables to the model (Sec. 3.2). The resulting semantic decision a_t is then passed to an embodiment-specific and model-agnostic interpreter, which deterministically grounds it into executable robot control,

$$u_t = g_E(a_t;\, s_t), \tag{3}$$

where E denotes the robot embodiment and s_t is the interpreter's internal setpoint state. Executing u_t updates the robot and environment, producing new observations and execution states that are fed back into the next interaction step.

3.2 Physically Grounded Semantic Action Interface

Semantic action space. The shared action space $\mathcal{A}$ in Eq. 2 consists of a compact set of semantic action units. At each step, the system determines a reference view from the current observations to define end-effector movement directions. Relative to this view, `MV_FWD`/`MV_BACK`, `MV_LEFT`/`MV_RIGHT`, and `MV_UP`/`MV_DOWN` move the end effector one step along the corresponding direction. For tasks that require end-effector reorientation, `RT_YAW_CW`/`RT_YAW_CCW`, `RT_PITCH_CW`/`RT_PITCH_CCW` and `RT_ROLL_CW`/`RT_ROLL_CCW` incrementally rotate the end effector clockwise or counterclockwise about the corresponding axis. `GRASP` and `RELEASE` close and open the gripper, while `DONE` indicates task completion. The vocabulary is designed around 3 properties. (i) *Incremental*: each unit induces a small, localized physical change, keeping the VLM situated in the control loop through observable action effects and enabling precise behavior through successive corrections. We empirically find that VLMs can flexibly switch between step granularities without fine-tuning (Sec. 4.3). (ii) *Interpretable and embodiment-agnostic*: actions

Table 1: Harness plugins organized along the perceive–reason–act loop.

Plugin	Function
Perception	
Multi-View Guidance	Guides the VLM on the roles and appropriate use of different camera views
Proprioception	Translates robot state, contact, and gripper status into textual feedback
Reasoning	
Subtask Planning	Maintains an ordered subtask plan with visually checkable completion
Situated Planning	Defers uncertain decisions and selectively replans as execution unfolds
Action Chunking	Adaptively chunks actions to balance precision and efficiency
Adaptive Step	Adaptively adjusts step size to balance precision and efficiency
Visual Prompt	Highlights and verifies task-relevant visual targets
Action	
Action History	Summarizes recent actions and discourages oscillatory behavior
Failure Recovery	Detects execution failures and triggers corrective recovery

are represented as semantic symbols rather than numeric targets, while embodiment-specific low-level control is delegated to the downstream interpreter, keeping the model-facing space compact, interpretable, and reusable across robots. (iii) *Visually grounded*: movement directions are defined relative to observable views, allowing spatial reasoning to map onto actions.

Embodiment grounding. Each semantic decision $a_t \in \mathcal{A}$ made by the VLM is then passed to an embodiment-specific interpreter g_E, which deterministically grounds the unit a_t into robot control. For an action unit, the interpreter updates the 6-DoF Cartesian pose setpoint $s_t = (\mathbf{x}_t, Q_t)$ as

$$s_{t+1} = \Pi_E(\mathbf{x}_t + \sigma_t R_E d_a,\ \exp(\theta_t [R_E r_a]_\times)\, Q_t)\,, \tag{4}$$

where $\mathbf{x}_t \in \mathbb{R}^3$ and $Q_t \in \mathrm{SO}(3)$ denote position and orientation. d_a and r_a encode translation and rotation, respectively: $d_a = 0$ for rotation, while $r_a = 0$ for translation and otherwise $r_a \in \{\pm\mathbf{e}_x, \pm\mathbf{e}_y, \pm\mathbf{e}_z\}$. Here $[\cdot]_\times$ is the skew-symmetric matrix operator. The calibrated increments σ_t and θ_t set the translation and rotation magnitudes, while R_E maps the semantic directions and axes into the motion frame of embodiment E. The projection Π_E enforces embodiment-specific workspace and per-step translation or rotation limits. Gripper units bypass the pose update and map directly to open or close commands. Different embodiments realize the same semantic units through different low-level controllers. For example, Franka tracks Cartesian setpoints with impedance control, AgileX uses inverse kinematics and streamed joint targets, and the simulator executes corresponding operational-space commands. By isolating embodiment-specific control inside g_E, the semantic model interface remains unchanged across robots. Adapting to a new embodiment therefore requires only a new interpreter. Safety bounds, such as workspace and table-height limits, can be flexibly configured in the interpreter, and actions that violate them are blocked before execution to ensure safe physical interaction.

3.3 Two Modes on One Interface

Frontier VLM as zero-shot agents (ZS mode). A frontier VLM can directly control robots through the harness without fine-tuning. This mode directly unlocks frontier-model capabilities for physical control, providing a scalable path that inherits advances in increasingly capable foundation models.

Fine-tuning small VLMs (FT mode). The same interface supports lightweight adaptation of a small open-source VLM to predict semantic action units directly. Given demonstrations $\mathcal{D}$ collected in the shared action space, the policy minimizes the token-level cross-entropy of the target unit,

$$\mathcal{L}(\theta) = -\sum_{(\ell, \mathcal{I}, h, a) \in \mathcal{D}} \log \pi_\theta(a \mid \Phi_{\mathcal{P}_{\min}}(\ell, \mathcal{I}, h))\,, \tag{5}$$

where $\mathcal{P}_{\min}$ intentionally retains only a minimal decision context to facilitate controlled comparison and analysis, consisting of the task instruction ℓ, the multi-view images $\mathcal{I}$, and a short action history h, without proprioceptive state. Crucially, semantic actions are predicted through the VLM's native vocabulary, without dedicated action heads or special tokens, enabling lightweight low-rank adaptation from limited demonstrations and stronger generalization than representative VLA baselines (as shown in Sec. 4), while the step size stays with the interpreter and can change at test time.

3.4 Extension to GUI-based Manipulation Interface

Since the semantic action space is interpretable and directly operable by both humans and VLMs, it unifies VLM-based robot control and GUI-style human control under a shared action interface. Building on this property, we develop GUMI, a GUI-based manipulation interface that allows humans and agents to operate robots and collect demonstrations through simple keyboard inputs mapped to the same semantic action units. Unlike conventional teleoperation pipelines that rely on specialized hardware (Zhao et al., 2023; Wu et al., 2023; Chi et al., 2024) or simulation-specific controls (Zhu et al., 2020; Mandlekar et al., 2021; Liu et al., 2023b), GUMI records demonstrations directly in the shared semantic action space, so that the same demonstrations can be reused across embodiments when paired with corresponding low-level interpreters, while the same workflow applies in both simulation and the real world without specialized teleoperation hardware. Its lightweight, digitally accessible design also supports remote data collection without requiring physical colocation with the robot. More details can be found in the appendix.

4 Experiments

4.1 Experimental Setup

Hardware. We conduct experiments on two robot platforms: a Franka Research 3 with exocentric and wrist views, and a bimanual AgileX rig with one shared egocentric and two wrist views. Franka is used by default unless otherwise specified. Local models run on a single RTX 5090 GPU, while frontier models are accessed via APIs. Further details are provided in the appendix.

Tasks and metrics. We construct ten real-robot manipulation tasks by pairing five objects with two target receptacles (a plate and a bowl). Each task requires locating, approaching, grasping, transporting, and placing appropriately. Objects span diverse physical properties: rigid geometry (block), irregular shape (banana), rolling dynamics (tennis ball), deformability (teddy bear), and precision manipulation (chess piece). For fine-tuned open-sourced VLMs, only the block, banana, and tennis ball appear in the demonstrations. The teddy bear and chess piece are held out for OOD evaluation. Beyond these tasks, Secs. 4.3 and A.2 introduce targeted scenarios to probe broader capabilities and design choices. We report success rate and average steps per episode. Unless noted otherwise, we run 10 trials per task with randomized object placements. Episodes are capped at 50 steps, with timeouts counted as failures.

VLM agents and harness. (1) **Frontier VLM as zero-shot agents (ZS mode).** We use Gemini-3.1 Pro (Google DeepMind, 2026) by default, and study alternative frontier models and reasoning effort levels in our ablations. The Adaptive Step plugin uses two translation step sizes: a 2 cm fine step when the target is visible in the wrist view and a 4 cm coarse step otherwise. This simple configuration suffices for basic pick-and-place tasks, while the VLM can flexibly adapt to finer step granularities without fine-tuning (Sec. 4.3). Action history retains the five most recent actions. All plugins in Sec. 3.1 are enabled by default, except Situated Planning and Visual Prompt, which are activated only when needed. (2) **Fine-tuning small VLMs (FT mode).** We use Qwen3.5-2B (Qwen Team, 2026) by default and study model scaling in our ablations. We apply LoRA (Hu et al., 2021) to all language-model linear layers while freezing the vision encoder and multimodal projector, updating only about 3% of the parameters.

Baselines. We compare against three representative families of robot-control systems: (1) **VLA** baselines directly map observations and instructions to low-level robot actions, including $\pi_{0.5}$ (Black et al., 2025) and GR00T (Bjorck et al., 2025). For a controlled comparison, both are fine-tuned on continuous end-effector trajectories converted from the same demonstrations used to train our VLM policy. (2) **VLA-centric agents** retain a VLA as the main physical executor while adding an outer agentic layer for planning, monitoring, or intervention, represented by Harness VLA (Zhang et al., 2026c) (H-VLA in tables) and Goal-VLA (Chen et al., 2025a) (G-VLA in tables). (3) **Code-as-policy agents** translate high-level reasoning into executable programs or API calls, represented by CaP-X (Fu et al., 2026) and RATS (Zhang et al., 2026a).

Demonstration data collection. For open-source VLM training and trainable VLA baselines, we collect demonstrations through GUMI using human keyboard control and frontier-VLM rollouts via the browser interface. In total, we collect 164 real-robot episodes (7.8K decision steps) under a shared 2 cm translation: 101 episodes on the 7-DoF Franka (5.0K steps), 63 episodes on the single-arm AgileX (2.8K steps). Our fine-tuned policy and compared VLA baselines jointly train on demonstrations from both embodiments. For sim-to-real experiments, all compared models are

Table 2: Performance across three levels of generalization on real robots. ZS and FT denote Show-Harness in zero-shot mode (Gemini-3.1 Pro w/ medium effort) and fine-tuned mode (Qwen3.5-2B).

Setting	VLA		VLA-centric agent		Code-as-policy agent		**Show-Harness**	
	$\pi_{0.5}$	GR00T	H-VLA	G-VLA	CaP-X	RATS	ZS	FT
Cross-Task *10 object–receptacle tasks; †: unseen in fine-tuning; 10 trials / task*								
Block → Plate	6 / 10	5 / 10	7 / 10	2 / 10	5 / 10	6 / 10	10 / 10	10 / 10
Banana → Plate	7 / 10	7 / 10	8 / 10	3 / 10	8 / 10	8 / 10	10 / 10	10 / 10
Tennis → Plate	3 / 10	3 / 10	3 / 10	1 / 10	4 / 10	6 / 10	8 / 10	8 / 10
Teddy† → Plate	5 / 10	4 / 10	5 / 10	1 / 10	4 / 10	6 / 10	10 / 10	8 / 10
Chess† → Plate	1 / 10	0 / 10	3 / 10	0 / 10	2 / 10	4 / 10	10 / 10	9 / 10
Block → Bowl	5 / 10	4 / 10	6 / 10	2 / 10	5 / 10	5 / 10	10 / 10	10 / 10
Banana → Bowl	6 / 10	6 / 10	7 / 10	2 / 10	7 / 10	8 / 10	6 / 10	7 / 10
Tennis → Bowl	2 / 10	2 / 10	3 / 10	1 / 10	3 / 10	5 / 10	7 / 10	8 / 10
Teddy† → Bowl	3 / 10	4 / 10	5 / 10	1 / 10	4 / 10	5 / 10	8 / 10	7 / 10
Chess† → Bowl	1 / 10	0 / 10	3 / 10	0 / 10	2 / 10	4 / 10	10 / 10	9 / 10
Average (%)	39.0	35.0	50.0	13.0	44.0	57.0	89.0	86.0
Cross-Environment *2 tasks (Block or Banana → Plate); 10 trials / task*								
Background	11 / 20	10 / 20	15 / 20	4 / 20	12 / 20	14 / 20	20 / 20	18 / 20
Lighting	9 / 20	9 / 20	14 / 20	5 / 20	11 / 20	13 / 20	20 / 20	19 / 20
Viewpoint	10 / 20	7 / 20	10 / 20	2 / 20	10 / 20	12 / 20	20 / 20	19 / 20
Distractors	10 / 20	8 / 20	12 / 20	1 / 20	9 / 20	13 / 20	20 / 20	19 / 20
Sim-to-real	0 / 20	0 / 20	–	–	–	–	–	13 / 20
Average (%)	40.0	34.0	63.8	15.0	52.5	65.0	100.0	88.0
Cross-Embodiment *5 Plate tasks; 10 trials / task on each arm*								
Franka (7-DoF)	22 / 50	19 / 50	26 / 50	7 / 50	23 / 50	30 / 50	48 / 50	45 / 50
AgileX (6-DoF)	19 / 50	17 / 50	23 / 50	4 / 50	20 / 50	22 / 50	45 / 50	42 / 50
Average (%)	41.0	36.0	49.0	11.0	43.0	52.0	93.0	87.0

only trained on 230 simulated episodes (13.5K steps) collected from ManiSkill (Tao et al., 2024) and RoboLab (Yang et al., 2026b). More details are in the Appendix A.5.

4.2 Main Results: Generalization across Task, Environment, and Embodiments

We evaluate three levels of generalization as summarized in Tab. 2. Cross-task covers all ten object–receptacle tasks. Cross-environment evaluates background, lighting, viewpoint, distractor, and sim-to-real shifts not seen during training or the standard test setting. For sim-to-real, trainable methods use only simulated demonstrations collected through the same GUMI interface and are evaluated on the real Franka. Cross-embodiment evaluates transfer between the Franka and AgileX arms: the zero-shot agent switches embodiments solely through the embodiment-specific interpreter, while the fine-tuned policy is co-trained on demonstrations from both embodiments and evaluated directly on each arm. Tab. 2 shows that both ZS (zero-shot frontier models) and FT (fine-tuned small open-source models) consistently outperform representative baselines across task, environment, and embodiment shifts. The advantage persists on held-out object combinations and extends to sim-to-real transfer, where FT succeeds using only simulated demonstrations while trainable VLA baselines fail. Cross-embodiment results further show that the same semantic interface transfers effectively between Franka and AgileX. Together, these results suggest that Show-Harness provides a scalable interface spanning frontier zero-shot agents and small fine-tuned models.

4.3 Capability Analysis: Physical and Semantic Adaptability

Sec. 4.2 establishes broad generalization across task, environment, and embodiment shifts. We next probe a harder question: **how well can Show-Harness enable foundation models to adapt to new physical and semantic demands?** Through targeted scenarios, we study two complementary forms of adaptability: *physical adaptability* (✋), which accommodates changes in motion precision, composition, workspace, and embodiment without policy retraining, and *semantic adaptability* (🧠), which handles tasks requiring reasoning or in-context learning from novel demonstrations (Fig. 3).

Fine-grained control. We evaluate block stacking and peg insertion, which require finer motion than standard pick-and-place. We simply reduce the interpreter step size from 2 cm to 1 cm, without



Figure 3: Capability analysis of Show-Harness across physical (✋) and semantic (🧠) adaptability.

changing the VLM–action interface or retraining either model. This improves ZS from 60% to 80% and FT from 40% to 65%. In contrast, $\pi_{0.5}$ achieves only 15% with the same demonstrations as FT, reaching 60% only after additional fine-grained training. This highlights the benefit of separating semantic decisions from metric execution: Show-Harness-equipped VLM agents adapt to new precision requirements through the interpreter alone, without task-specific relearning.

Action composition. On the five Plate tasks, composing two orthogonal translation units into one diagonal displacement substantially reduces execution steps without a notable drop in success. This shows that new action compositions can be introduced through the interpreter without retraining.

Rotation extrapolation. We test compositional extrapolation on a carrot-grasping task with the carrot oriented at 0°, 45°, or an unseen 90° relative to the gripper, while each rotation unit changes orientation by 15°. ZS remains robust across all angles, while FT reaches 70% at the unseen 90° orientation despite training only on 0° and 45° demonstrations; $\pi_{0.5}$ reaches 20% (Fig. 3) showing that incremental rotation enables larger unseen orientation changes through repeated unit composition, whereas continuous action regression remains more tied to the demonstrated angle range.

Workspace shift. On the five Plate tasks, we evaluate nested workspace regions following Chen et al. (2026b), from the core 25% (S@1) to 90% near the boundary (S@3). Show-Harness degrades only mildly as the workspace expands, whereas $\pi_{0.5}$ drops sharply. This suggests that visually grounded semantic decisions make Show-Harness-equipped VLM agents less sensitive to workspace shifts, while $\pi_{0.5}$ remains more tied to the training distribution.

Multi-arm coordination. We evaluate two 20-trial tasks on an AgileX pair: *tidy the table*, where each arm clears nearby objects, and *pass the banana*, requiring handover before placement. We compare two independent single-arm agents with a joint policy that predicts both actions at each step. Joint decision making substantially improves success on both tasks while eliminating collisions, showing that Show-Harness enables explicit inter-arm coordination through joint action prediction.

Reasoning-intensive tasks. We evaluate two tasks requiring high-level reasoning before manipulation: locating a block hidden under one of three cups and arranging scattered letters into "SHOW". ZS with Situated Planning achieves 85%, while FT and $\pi_{0.5}$ alone reach only 10% and 0%. With

the same Gemini-generated subtask instructions, FT rises to 70%, whereas $\pi_{0.5}$ remains at 5%. This suggests that Show-Harness better preserves the instruction-following flexibility of VLMs by grounding novel instructions through a reusable semantic action space.

In-context learning from video. We ask the policy to put away three objects in the order shown by a human or robot demonstration. Without a demonstration, ZS receives only the instruction *tidy up* and succeeds in just 20%, since the required order is unspecified. With a video demonstration, ZS follows the demonstrated order and succeeds in 20 / 20 trials from either source, while FT reaches 90% when conditioned on the task outline extracted by the same planner. This shows that Show-Harness enables strong visual in-context learning for robot control.

4.4 Ablation on Action Interfaces

We compare our semantic action interface against four representative alternatives: `L0` emits a free Cartesian displacement in centimeters, the quantity continuous VLAs regress or tokenize (Brohan et al., 2023; Kim et al., 2024; Black et al., 2025). `L1` emits an absolute end-effector pose in the robot frame, as in methods that name a target configuration and leave a solver to realize it (Huang et al., 2023b; 2024). `L2` emits an image point with a height, realized by depth back-projection and inverse kinematics, as in pointing-based agents (Yuan et al., 2024). `L3` emits semantic action units, grounded by the action interpreter (ours). `L4` invokes scripted high-level skills, following the code-as-policies paradigm (Fu et al., 2026; Liang et al., 2023). The library includes `move_above(target)`, `pick(object)`, `place(target)`, and gripper open / close skills. An extended variant additionally includes `insert(object, target)`, tailored for peg insertion. We evaluate three task groups: **Coarse** (Block / Teddy → Plate), **Fine** (three-block stacking and tight peg insertion), and **Reasoning** (hidden-block search and arranging letters into "SHOW"). Each task is evaluated over 20 trials. For ours (`L3`), step sizes are set by the Adaptive Step plugin (Tab. 1): 4 / 2 cm on Coarse and Reasoning, 1 cm on stacking, and 4 / 1 / 0.25 cm on peg-tight.

Table 3: Action-interface ablation with Gemini-3.1 Pro (GPT-6 Astra).

Action interface	Coarse	Fine	Reasoning	Overall
`L0` Numeric displacement	77.5 (82.5)	35.0 (45.0)	57.5 (65.0)	56.7 (64.2)
`L1` Absolute pose	70.0 (75.0)	30.0 (40.0)	50.0 (57.5)	50.0 (57.5)
`L2` Image point	85.0 (90.0)	50.0 (55.0)	47.5 (52.5)	60.8 (65.8)
`L3` Semantic units (ours)	**97.5** (92.5)	**80.0** (**85.0**)	**85.0** (**85.0**)	**87.5** (**87.5**)
`L4` High-level skills	92.5 (**95.0**)	42.5 (47.5)	77.5 (80.0)	70.8 (74.2)
`L4` High-level skills (extended)	92.5 (**95.0**)	62.5 (70.0)	77.5 (80.0)	77.5 (81.7)

Compared with low-level interfaces (L0–L2), our semantic units keep the VLM's decisions in a space natively interpretable to VLMs, yielding substantially stronger performance. At the same time, control remains local and fine-grained, so the agent can make local corrections from visual feedback rather than committing to an entire high-level skill. The action interpreter further decouples motion magnitude from semantic prediction, allowing the same unit to be realized at different step sizes as needed, without fine-tuning. By contrast, skill-level interfaces often require specialized high-precision skills (e.g., `insert` for peg insertion). More ablations are in the appendix.

5 Conclusion and Limitations

In this work, we introduced **Show-Harness**, an *Embodied Harness* that unlocks embodied capability from foundation VLMs through a compact semantic action interface. Show-Harness exposes an action space that is semantic enough for VLMs to naturally reason over yet fine-grained enough for direct physical control, keeping the VLM engaged in stepwise physical decisions within a closed perception–reasoning–action–feedback loop. Extensive experiments demonstrate strong generalization and adaptability of Show-Harness-enabled VLM agents, suggesting that the right interface can unlock substantial embodied capability already present in foundation VLMs, with minimal embodiment-specific adaptation.

Despite its effectiveness, Show-Harness is currently evaluated primarily on single- and dual-arm manipulation with parallel-jaw grippers. Extending the framework to more complex embodiments (e.g., humanoids or dexterous hands) is a valuable direction. Future work may also enrich the perception side with additional embodied modalities, such as tactile and force feedback, to support more contact-rich and fine-grained physical interaction.

AI USE STATEMENT

AI-assisted tools were used solely for language polishing to improve the clarity and readability of the manuscript. All scientific content was reviewed and verified by the authors, who take full responsibility for the final manuscript.

REPRODUCIBILITY STATEMENT

The paper provides the details needed to reproduce our results: the robot platforms, task set, and evaluation protocol in Sec. 4.1, the fine-tuning hyperparameters in Appendix A.3, and the complete prompt for each model role in Appendix A.7. The demonstration corpus, the embodiment interpreters, and the evaluation code will be released.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Michael Ahn, Debidatta Dwibedi, Chelsea Finn, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Karol Hausman, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, et al. Autort: Embodied foundation models for large scale orchestration of robotic agents. *arXiv preprint arXiv:2401.12963*, 2024.

Anthropic. Claude Opus 5 System Card. System Card, July 2026. URL https://www.anthropic.com/claude-opus-5-system-card.

Abrar Anwar, John Welsh, Joydeep Biswas, Soha Pouya, and Yan Chang. Remembr: Building and reasoning over long-horizon spatio-temporal memory for robot navigation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2838–2845. IEEE, 2025.

Erik Bauer, Elvis Nava, and Robert K Katzschmann. Latent action diffusion for cross-embodiment manipulation. *arXiv preprint arXiv:2506.14608*, 2025.

Shmuel Berman, Michael Ilie, Jia Deng, and Daniel Freeman. Claude plays robotics. *Anthropic Research*, 2026. https://www.anthropic.com/research/claude-plays-robotics.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y Galliker, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *9th Annual Conference on Robot Learning*, 2025.

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

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

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.

Haonan Chen, Jingxiang Guo, Bangjun Wang, Tianrui Zhang, Xuchuan Huang, Boren Zheng, Yiwen Hou, Chenrui Tie, Jiajun Deng, and Lin Shao. Goal-vla: Image-generative vlms as object-centric world models empowering zero-shot robot manipulation. *arXiv preprint arXiv:2506.23919*, 2025a.

Junting Chen, Yao Mu, Qiaojun Yu, Tianming Wei, Silang Wu, Zhecheng Yuan, Zhixuan Liang, Chao Yang, Kaipeng Zhang, Wenqi Shao, et al. Roboscript: Code generation for free-form manipulation tasks across real and simulation. *arXiv preprint arXiv:2402.14623*, 2024a.

Siyi Chen, Hugo Hadfield, Alex Zook, Mikaela Angelina Uy, Chan Hee Song, Erwin Coumans, Xuning Yang, Faisal Ladhak, Qing Qu, Stan Birchfield, et al. Volo: A physical orchestrator for open-vocabulary long-horizon manipulation. *arXiv preprint arXiv:2606.07723*, 2026a.

Yanzhe Chen, Kevin Yuchen Ma, Qi Lv, Yiqi Lin, Zechen Bai, Chen Gao, and Mike Zheng Shou. Escaping the diversity trap in robotic manipulation via anchor-centric adaptation. *arXiv preprint arXiv:2605.07381*, 2026b.

Yi Chen, Yuying Ge, Weiliang Tang, Yizhuo Li, Yixiao Ge, Mingyu Ding, Ying Shan, and Xihui Liu. Moto: Latent motion token as the bridging language for learning robot manipulation from videos. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19752–19763. IEEE, 2025b.

Yongchao Chen, Jacob Arkin, Charles Dawson, Yang Zhang, Nicholas Roy, and Chuchu Fan. Autotamp: Autoregressive task and motion planning with llms as translators and checkers. In *2024 IEEE International conference on robotics and automation (ICRA)*, pp. 6695–6702. IEEE, 2024b.

Yongchao Chen, Yilun Hao, Yang Zhang, and Chuchu Fan. Code-as-symbolic-planner: Foundation model-based robot planning via symbolic code generation. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 19248–19254. IEEE, 2025c.

Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Robotics: Science and Systems*, 2024.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

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

Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation. *arXiv preprint arXiv:2410.00371*, 2024a.

Jiafei Duan, Wentao Yuan, Wilbert Pumacay, Yi Ru Wang, Kiana Ehsani, Dieter Fox, and Ranjay Krishna. Manipulate-anything: Automating real-world robots using vision-language models. *arXiv preprint arXiv:2406.18915*, 2024b.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

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

Liane Galanti, Dhruv Shah, and Tri Dao. Addressing the orchestration gap in generalist robots via physical agency. *arXiv preprint arXiv:2607.21725*, 2026.

Google DeepMind. Gemini 3.1 pro. Model Card, February 2026. URL https://deepmind.google/models/model-cards/gemini-3-1-pro/.

Ankit Goyal, Hugo Hadfield, Xuning Yang, Valts Blukis, and Fabio Ramos. Vla-0: Building state-of-the-art vlas with zero modification. *arXiv preprint arXiv:2510.13054*, 2025.

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. *arXiv preprint arXiv:2106.09685*, 2021.

Jiaheng Hu, Mohit Shridhar, Caden Lu, Dhruv Shah, Hao-Tien Lewis Chiang, Jie Tan, and Annie Xie. What matters in orchestrating robot policies: A systematic study of hierarchical vla agents. *arXiv preprint arXiv:2606.10267*, 2026a.

Yutong Hu, Jan-Nico Zaech, Nikolay Nikolov, Yuanqi Yao, Sombit Dey, Giuliano Albanese, Renaud Detry, Luc Van Gool, and Danda Paudel. Ar-vla: True autoregressive action expert for vision-language-action models. *arXiv preprint arXiv:2603.10126*, 2026b.

Siyuan Huang, Zhengkai Jiang, Hao Dong, Yu Qiao, Peng Gao, and Hongsheng Li. Instruct2act: Mapping multi-modality instructions to robotic actions with large language model. *arXiv preprint arXiv:2305.11176*, 2023a.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

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

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024.

Yuheng Ji, Huajie Tan, Jiayu Shi, Xiaoshuai Hao, Yuan Zhang, Hengyuan Zhang, Pengwei Wang, Mengdi Zhao, Yao Mu, Pengju An, et al. Robobrain: A unified brain model for robotic manipulation from abstract to concrete. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1724–1734. IEEE, 2025.

Tengyue Jiang, Chunpu Xu, Jiayue Kang, and Yao Mu. Sa-vla: State-aware tokenizer for improving vision-language-action models' performance. *arXiv preprint arXiv:2606.30113*, 2026.

Miracle Kang, Lights Shi, Lucy Liang, Roy Gan, Dongxiu Liu, Pushi Zhang, Sylas Chen, Shawn Qin, Yinan Zheng, Jinliang Zheng, et al. X-tokenizer: A multimodal action tokenizer for vision-language-action pretraining. *arXiv preprint arXiv:2606.14752*, 2026.

Elis Karcini, Faisal Mehrban, Quang Nguyen, Mac Schwager, Arash Ajoudani, Cesar Cadena, Jan Peters, Marco Hutter, and Haitham Bou-Ammar. Robots need more than vla and world models. *arXiv preprint arXiv:2606.06556*, 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.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *European Conference on Computer Vision*, pp. 104–120, 2020.

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pp. 80–93. PMLR, 2023.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li E Li, Ruohan Zhang, et al. Embodied agent interface: Benchmarking llms for embodied decision making. *Advances in Neural Information Processing Systems*, 37:100428–100534, 2024.

Ruiying Li, Yunlang Zhou, YuYao Zhu, Kylin Chen, Jingyuan Wang, Sukai Wang, Kongtao Hu, Minhui Yu, Bowen Jiang, Zhan Su, et al. Roboclaw: An agentic framework for scalable long-horizon robotic tasks. *arXiv preprint arXiv:2603.11558*, 2026a.

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. *arXiv preprint arXiv:2503.00200*, 2025.

Siyao Li, Jiawei Gu, Shuai Liu, Kairui Hu, Zekun Li, Linjie Li, Chengcheng Tang, Po-Chen Wu, Ivan Shugurov, Lingni Ma, et al. Humanclaw: Can vision-language models act through a body? *arXiv preprint arXiv:2607.27180*, 2026b.

Ziang Li, Dongzhou Cheng, Yibin Wang, Shiyue Wang, Xiaoyang Xu, Lingxuan Weng, Juan Wang, and Jiaqi Wang. Light-wam: Efficient world action models with state-fusion action decoding. *arXiv preprint arXiv:2606.08242*, 2026c.

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.

Oscar Lima, Marc Vinci, Martin Günther, Marian Renz, Alexander Sung, Sebastian Stock, Johannes Brust, Lennart Niecksch, Zongyao Yi, Felix Igelbrink, et al. Agentic ai for robot control: Flexible but still fragile. In *Proceedings of the AAAI Symposium Series*, volume 8, pp. 465–473, 2026.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for gui visual agent. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19498–19508. IEEE, 2025.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023a.

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

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.

Peiqi Liu, Yaswanth Orru, Jay Vakil, Chris Paxton, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Ok-robot: What really matters in integrating open-knowledge models for robotics. *arXiv preprint arXiv:2401.12202*, 2024.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 29982–30009, 2025.

Zeyi Liu, Arpit Bahety, and Shuran Song. Reflect: Summarizing robot experiences for failure explanation and correction. In *Conference on Robot Learning*, 2023c.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning*, 2021.

Juncheng Mu, Sizhe Yang, Hojin Bae, Feiyu Jia, Qingwei Ben, Boyi Li, Huazhe Xu, and Jiangmiao Pang. One-policy-fits-all: Geometry-aware action latents for cross-embodiment manipulation. *arXiv preprint arXiv:2603.14522*, 2026.

Mingyu Ouyang, Siyuan Hu, Kevin Qinghong Lin, Hwee Tou Ng, and Mike Zheng Shou. Gameworld: Towards standardized and verifiable evaluation of multimodal game agents. *arXiv preprint arXiv:2604.07429*, 2026.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Bikang Pan, Fan Liu, Haotao Lu, Jingya Wang, and Ye Shi. Selfwam: A self-grounded unified world action model for fast robot control. *arXiv preprint arXiv:2608.00725*, 2026.

Jiaqi Peng, Xiqian Yu, Delin Feng, Yuqiang Yang, Wenzhe Cai, Jing Xiong, Ganlin Yang, Jinliang Zheng, Jiafei Cao, Xueyuan Wei, et al. Cortex: A bidirectionally aligned embodied agent framework for long-horizon manipulation. *arXiv preprint arXiv:2607.05377*, 2026.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

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

Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning. *arXiv preprint arXiv:2307.06135*, 2023.

Vitor Gaboardi dos Santos, Ibrahim Khadraoui, Ibrahim Farhat, Hamza Yous, Samy Teffahi, and Hakim Hacid. Alrm: Agentic llm for robotic manipulation. *arXiv preprint arXiv:2601.19510*, 2026.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9338–9346, 2019.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pp. 785–799. PMLR, 2023.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

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. *arXiv preprint arXiv:2209.11302*, 2022.

StarVLA Community. StarVLA: A lego-like codebase for vision-language-action model developing. *arXiv preprint arXiv:2604.05014*, 2026.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, Yuan Gao, Xuanlin Li, Tongzhao Mu, Nan Xiao, Arnav Gurha, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. ManiSkill3: GPU parallelized robotics simulation and rendering for generalizable embodied AI. *arXiv preprint arXiv:2410.00425*, 2024.

Gemini Robotics Team, Abbas Abdolmaleki, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Ashwin Balakrishna, Nathan Batchelor, Alex Bewley, Jeff Bingham, et al. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Sai H Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. Chatgpt for robotics: Design principles and model abilities. *Ieee Access*, 12:55682–55696, 2024.

Zihao Wang, Xujing Li, Yining Ye, Junjie Fang, Haoming Wang, Longxiang Liu, Shihao Liang, Junting Lu, Zhiyong Wu, Jiazhan Feng, et al. Game-tars: Pretrained foundation models for scalable generalist multimodal game agents. *arXiv preprint arXiv:2510.23691*, 2025.

Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. GELLO: A general, low-cost, and intuitive teleoperation framework for robot manipulators. *arXiv preprint arXiv:2309.13037*, 2023.

Fan Yang, Yuting Su, Xiaobo Wang, Yuncheng You, Fugui Fan, Yuting Wu, Minghui Wu, Chenxu Zhao, JiaHong Ning, and Peiguang Jing. Lila-wam: Lightweight latent reasoning world-action model for robotic manipulation. *arXiv preprint arXiv:2608.03701*, 2026a.

Jenai Xuning Yang, Rishit Dagli, Alex Zook, Hugo Hadfield, Ankit Goyal, Stan Birchfield, Fabio Ramos, and Jonathan Tremblay. Robolab: A high-fidelity simulation benchmark for analysis of task generalist policies. *arXiv preprint arXiv:2604.09860*, 2026b.

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

Zhejian Yang, Yongchao Chen, Xueyang Zhou, Jiangyue Yan, Dingjie Song, Yinuo Liu, Yuting Li, Yu Zhang, Pan Zhou, Hechang Chen, et al. Agentic robot: A brain-inspired framework for vision-language-action models in embodied agents. *arXiv preprint arXiv:2505.23450*, 2025b.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. In *International Conference on Learning Representations*, volume 2025, pp. 28213–28239, 2025.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023.

Haoqi Yuan, Yu Bai, Yuhui Fu, Bohan Zhou, Yicheng Feng, Xinrun Xu, Yi Zhan, Börje F Karlsson, and Zongqing Lu. Being-0: A humanoid robotic agent with vision-language models and modular skills. *arXiv preprint arXiv:2503.12533*, 2025.

Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. *arXiv preprint arXiv:2406.10721*, 2024.

Zhihao Zhan, Yuhao Chen, Jiaying Zhou, Qinhan Lyu, Hao Liu, Keze Wang, Liang Lin, and Guangrun Wang. Stable language guidance for vision–language–action models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4137–4159, 2026.

Junyi Zhang, Jiaxin Ge, Hanjun Yoo, Letian Fu, Zihan Yang, Yaowei Liu, Raj Saravanan, Shaofeng Yin, Justin Yu, Dantong Niu, et al. Playful agentic robot learning. *arXiv preprint arXiv:2606.19419*, 2026a.

Kuan Zhang, Dongchen Liu, Qiyue Zhao, Tianyu Xin, Yue Su, Haisheng Wang, Han Yin, Hongbo Ma, Peize Li, Tianjun Gu, et al. Towards generalist game players: An investigation of foundation models in the game multiverse. *arXiv preprint arXiv:2605.09965*, 2026b.

Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, et al. Harness vla: Steering frozen vlas into reliable manipulation primitives via memory-guided agents. *arXiv preprint arXiv:2607.08448*, 2026c.

Rongfeng Zhao, Xuanhao Zhang, Zhaochen Guo, Xiang Shao, Zhongpan Zhu, Bin He, and Jie Chen. Rosclaw: A hierarchical semantic-physical framework for heterogeneous multi-agent collaboration. *arXiv preprint arXiv:2604.04664*, 2026.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Jinliang Zheng, Jianxiong Li, Dongxiu Liu, Yinan Zheng, Zhihao Wang, Zhonghong Ou, Yu Liu, Jingjing Liu, Ya-Qin Zhang, and Xianyuan Zhan. Universal actions for enhanced embodied foundation models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22508–22519. IEEE, 2025.

Yifan Zhong, Fengshuo Bai, Shaofei Cai, Xuchuan Huang, Zhang Chen, Xiaowei Zhang, Yuanfei Wang, Shaoyang Guo, Tianrui Guan, Ka Nam Lui, et al. A survey on vision-language-action models: An action tokenization perspective. *arXiv preprint arXiv:2507.01925*, 2025.

Jian Zhou, Xunyi Zhao, Gengze Zhou, Zerui Li, Sihao Lin, Jiajun Liu, and Qi Wu. Embodied agents take control: Minimal-interface zero-shot agents rival industrial-scale policies in vision-and-language navigation. *arXiv preprint arXiv:2607.26148*, 2026.

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. *arXiv preprint arXiv:2504.02792*, 2025.

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

A Appendix

A.1 Harness Plugin Details

Here we provide more details of the harness plugins listed in Tab. 1 of the main paper.

Perception. These plugins turn raw sensor streams into a context the model can reason over:

(1) Multi-View Guidance tells the VLM how different camera views should be used and prioritized. For example, a global egocentric/exocentric view provides scene-level context, while a wrist-mounted view offers close-up visual evidence for fine-grained alignment and manipulation.

(2) Proprioception translates the robot's internal state into concise textual feedback, including the current gripper height, the displacement induced by one action step, phase-aware execution hints (e.g., descending first when still high), contact state, and gripper state.

Reasoning. Reasoning plugins structure the task into actionable intermediate decisions and refine the current context for fine-grained action-level decisions:

(1) Subtask Planning first invokes the VLM as a planner to decompose the task instruction ℓ into an ordered sequence of subtasks with completion criteria. During execution, subtask transitions belong to the model: at every step it checks the completion criterion against the current images, keeps acting while it is unmet, and advances the plan by declaring the subtask complete.

(2) Situated Planning defers uncertain decisions and selectively triggers replanning when the required information becomes available during execution. Rather than committing to all future branches upfront, the initial plan leaves such decisions unresolved until the relevant evidence becomes observable. The VLM then resolves the branch from the current observation and updates the remaining plan accordingly. This enables conditional tasks to adapt to the evolving execution state without replanning at every interaction step.

(3) Action Chunking adaptively reduces model-query frequency when fine-grained feedback is not yet required (e.g., when the target is still far away), allowing the VLM to emit a short sequence of semantic actions that are executed open-loop before the next model query.

(4) Adaptive Step dynamically adjusts the step size to balance efficiency and precision, using larger steps when the target is distant and smaller steps for close-range alignment.

(5) Visual Prompt uses a dedicated model call to convert an ambiguous verbal target into a visual reference highlighting the relevant affordance, allowing subsequent reasoning to ground on the visual marker. This plugin is enabled when the interaction region is difficult to specify in language.

Action. Building on the physically grounded semantic action interface introduced in Sec. 3.2, the action stage augments execution with lightweight interaction memory and recovery mechanisms that support robust closed-loop control. Two plugins operate at this stage:

(1) Action History carries recent actions into the next interaction step as lightweight context, together with simple usage guidance such as avoiding oscillation between opposite moves. This compact history provides the model with the temporal memory needed across steps.

(2) Failure Recovery automatically detects grasp failures and recovers by resetting the gripper state and rolling back to the relevant grasp subtask.

A.2 Ablation Studies

A.2.1 Frontier VLM Choice and Thinking Effort

We evaluate different frontier VLMs on the five Plate tasks, and further probe the strongest three models on a chess-cannon placement task (20 trials) that separates planning from fine-grained grounding. As shown in Fig. 4 (a), zero-shot performance generally improves with stronger frontier VLMs, broadly tracking their underlying model capability, while increasing thinking effort mainly reduces redundant interaction steps with little gain in success and can incur higher wall-clock cost (e.g., $3.4\times$ for GPT-5.6-sol). We also find that all models follow instructions reliably, with over 98% of responses producing valid action units. Fig. 4 (b) shows that planning is also not the main

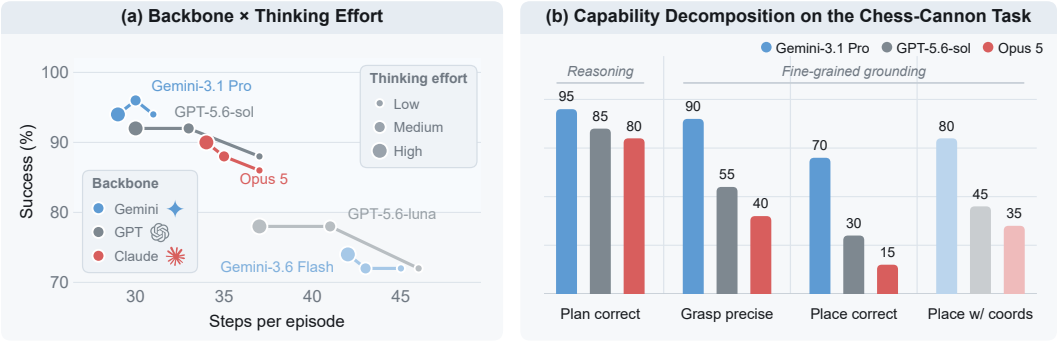


Figure 4: Frontier VLM choice and thinking effort. (a) Success against steps per episode on the five Plate tasks (10 trials each), at three thinking efforts. (b) *Placing the two cannons on their opening squares* (20 trials, three strongest models): plan correctness measures reasoning, grasp and placement measure grounding; pale bars give the target squares as board coordinates.

bottleneck; errors concentrate on fine-grained grasping and placement. Stating the target squares as board coordinates raises success for all three models, showing the value of explicit grounding cues.

A.2.2 Fine-Tuned Backbone Scaling

This study raises the episode cap from 50 to 80 steps, leaving room for the extra corrections of the smallest backbones; timeouts still count as failures. As shown in Fig. 5, performance is already strong at 2B, while larger backbones mainly help fine-grained tasks such as stacking. The 1B-level models instead suffer from excessive local adjustments near the target, leading to longer episodes. Smaller models can nevertheless outperform larger ones on the tennis-ball task, where timely correction of moving objects matters more than fine precision. Overall, the 2B backbone offers a good balance between precision and responsiveness, and we use it by default.



Figure 5: (a) Effect of fine-tuned backbone scaling over the five Plate tasks under an 80-step cap; bubble area is proportional to parameter count. (b) Three Qwen3.5 capacities compared on precision (stacking and peg insertion), responsiveness (tennis ball), and inference latency.

A.2.3 Harness Plugins

Fig. 6 ablates the harness plugins on the real Franka arm, one at a time. Each panel reports success with the plugin switched off and on, in the order discussed below.

Multi-View Guidance. Turning the plugin off removes the wrist view together with its view-role guidance and leaves the global view alone, which substantially degrades performance, particularly for small objects requiring precise alignment. The wrist view gives the most useful cues for fine-grained alignment, and the guidance tells the model which view to rely on at each stage.



Figure 6: Plugin ablations on the real Franka arm, zero-shot with Gemini-3.1 Pro. Default plugins are removed one at a time on the five Plate tasks; Visual Prompt and Situated Planning are instead added to the default and tested on handle-aware grasping and hidden-object search (20 trials each).

Proprioception. Removing proprioceptive state substantially degrades performance, particularly on objects where visual cues alone are ambiguous. Compact state signals such as gripper height and contact provide reliable guidance when appearance or depth makes visual estimation uncertain.

Subtask Planning. Without planning, success drops to 60%, with the model often dragging objects toward the plate without lifting. Subtask decomposition makes the intended manipulation sequence explicit through locally valid stages and visually checkable completion criteria.

Action Chunking. Disabling chunking preserves 96% success but increases model calls, whereas forcing it throughout reduces calls but drops success to 74%. The result favors selective chunking: compress redundant transport steps while preserving fine-grained feedback near interaction.

Adaptive Step. Fine-only control is precise but inefficient and prone to timeouts, while coarse-only control is faster but prone to overshoot. Adaptive switching based on target visibility balances both, achieving 96% success with 30 steps per episode on average.

Visual Prompt. The plugin has little effect on regular tasks, supporting its default-off design, but substantially improves handle-aware grasping (40% to 85%). The gain comes not from the visual marker alone, but from explicitly aligning the language instruction with the marked interaction point.

Situated Planning. The plugin has no effect on regular tasks, but substantially improves hidden-object search (35% to 85%). It defers unresolved decisions until sufficient visual evidence is available, enabling targeted exploration instead of premature planning.

Action History. Removing action history reduces success and increases timeouts, often due to oscillation between opposing actions. Recording recent actions exposes such loops and prevents repeated reversals, providing lightweight memory for more stable closed-loop control.

Failure Recovery. Removing recovery reduces success to 72%, with the largest drops on hard-to-grasp objects. The main failure mode is an undetected empty grasp; empty-grasp detection triggers a retry, preventing the agent from continuing with an empty gripper.

A.2.4 Protocol of the Action-Interface Ablation

All interfaces in Tab. 3 run with the same controller model, observations (two views, proprioception, and five-step history), planner, failure recovery, workspace limits, and sampling settings; behavioral guidance is translated into each action format, and every interface starts from the same 20 layouts per task. Budgets are 60 model calls or 10 minutes for Coarse and Reasoning, and 80 calls or 15 minutes for Fine. Stacking uses 40 mm cubes, and the peg-tight fixture is a 19.0 mm peg in a 22.0 mm bore, 25 mm deep with a 0.5 mm chamfer (1.5 mm radial clearance), under impedance control without force sensing. Peg-tight is a tighter fixture than the peg insertion task of Sec. 4.3 and Fig. 3, so success rates on the two are not comparable. `L4` localizes named objects with an open-vocabulary detector and segmenter on the same two views and executes a fixed approach–descend–close–lift sequence. When `insert` is unavailable in the skill library, `L4` executes peg-tight as `place(hole)`. For `L3`, the 0.25 cm step switches on once the model reports the peg tip near the hole, and disabling this single step lowers peg-tight success across both models from 72.5% to 27.5% (b=20, c=2, p=0.0001).[1] Values in parentheses in Tab. 3 repeat the study with GPT-6 Astra. On Coarse, semantic units and high-level skills stay within five points under either model and swap order between the two; with GPT-6 Astra, semantic units lose two of 40 episodes on Coarse and hold level on Reasoning, while the other interfaces gain slightly. On Fine and Reasoning, semantic units lead under both models, with the largest margins on Fine.

A.2.5 Interface versus Harness

The controller prompt (Appendix A.7) carries behavioral guidance, such as moving along the direction of largest deviation and descending once the target is centered. To separate what the interface contributes from what the harness contributes, we cross the action interface with three harness levels. The minimal context (S0) keeps the two views, proprioception, five-step history, action definitions, and interpreter, but drops all guidance and uses a fixed 2 cm step for semantic units. S1 adds the prompt's behavioral guidance, written in each interface's output format, and the full harness (S4) further adds Subtask Planning, Adaptive Step, Action Chunking, and Failure Recovery. Gemini-3.1 Pro runs semantic units (`L3`) and numeric displacement (`L0`, Sec. 4.4) on Block, Tennis, and Chess → Plate, with 20 shared layouts per task and a 50-step budget.

Table 4: Success rates (%) of the two interfaces at three harness levels. Gemini-3.1 Pro, 60 episodes per cell from 20 shared layouts per task. The last column gives the interface gain.

Harness level	`L0` Numeric displacement	`L3` Semantic units	Interface gain
Minimal context (S0)	18.3	43.3	+25.0
+ Behavioral guidance (S1)	28.3	63.3	+35.0
Full harness (S4)	66.7	**95.0**	+28.3

(1) The interface gain holds at every harness level. Semantic units lead numeric displacement by 25.0 points without any guidance (b=18, c=3, p=0.0015), by 35.0 with guidance, and by 28.3 with the full harness, so the advantage does not come from the prompt.

(2) The guidance works through the interface. The same guidance adds 20.0 points to semantic units (b=15, c=3, p=0.0075) but only 10.0 to numeric displacement (b=9, c=3, p=0.15): it speaks in directions, which semantic units execute directly, whereas numeric displacement must still turn each direction into a metric step, which the guidance cannot supply.

(3) The harness makes the agent reliable. Planning, adaptive execution, and failure recovery lift both interfaces by a further 32–38 points without closing the gap between them. On semantic units they mainly end unproductive search: from S0 to S4, timeouts fall from 14 to 1 in 60 episodes and reversals from 3.9 to 0.5 per episode. Both layers are part of Show-Harness: the interface determines what the agent can do, and the harness how reliably it does it.

[1] Here and below, b and c count the shared layouts on which only the first or only the second condition succeeds, and p is from a two-sided exact McNemar test.

A.2.6 Effect of Action-Space Naming and Explicit Rule Guidance

We study the effects of action-space naming and explicit rule guidance. All variants use Gemini-3.1 Pro and differ only in the naming/instruction of the six translational action units. A 2×2 factorial design crosses action naming (semantic, e.g., `MV_LEFT`, vs. arbitrary, e.g., `A`) with instruction mode (explicit rule guidance vs. self-discovery). For self-discovery, with all directional guidance removed, the agent performs 12 pre-task exploratory probes, transferring learned conventions across the session's three tasks. All conditions use 20 identical layouts on Tennis $\rightarrow$ Plate, three-block stacking (1 cm), and peg-tight. Budgets are 60 model calls for Tennis and stacking and 80 for peg-tight, so Tab. 5's default row differs from the Fine group in Tab. 3: with fewer stacking calls it completes 16 rather than 18 of 20 stacks, while peg-tight, with an 80-call budget in both, matches at 14 of 20.

The results in Tab. 5 show that both semantic naming and explicit rule guidance matter. Arbitrary names reduce overall success by 20.0 points, while self-discovered rules reduce it by 23.3 points. When both cues are removed, performance drops by 41.7 points, suggesting that semantic priors and explicit convention grounding provide complementary support for reliable action selection.

Table 5: Effect of action space naming and explicit rule guidance.

Action naming	Rule guidance	Tennis	Stack	Peg-tight	Overall (%)
Semantic	**Stated**	**17 / 20**	**16 / 20**	**14 / 20**	**78.3**
Arbitrary	Stated	13 / 20	12 / 20	10 / 20	58.3 (−20.0)
Semantic	Self-discovered	12 / 20	11 / 20	10 / 20	55.0 (−23.3)
Arbitrary	Self-discovered	9 / 20	8 / 20	5 / 20	36.7 (−41.7)

A.2.7 Effect of Decoupling Step Size from Action Selection

Tab. 6 isolates the step size. Both rows use the same semantic units, the same stated rules, and the same budgets; they differ only in where the magnitude of each move comes from. In the first row (our default), the model selects only the unit, the Adaptive Step plugin (Sec. 3.1) sets the step size, and the action interpreter executes the unit at that size deterministically. In the second row, the model also chooses the step size itself, from the same set of candidate sizes available to the Adaptive Step plugin. Predicting the magnitude lowers overall success from 78.3% to 51.7% (b=18, c=2, p=0.0004), with most of the loss on peg-tight (14 / 20 to 5 / 20). The model's steps stay near 2 cm (median 2.0 cm), and only 7% of its moves near the hole are 0.5 cm or shorter, the range in which Adaptive Step switches to its finest setting. This highlights the benefit of decoupling step size from the model's decisions: the step size is set outside the model (i.e., by the Adaptive Step plugin), and the interpreter applies it deterministically, so the model only decides which way to move and never has to estimate metric magnitudes. Tab. 7 complements this finding in policy learning.

Table 6: Effect of decoupling step-size selection from semantic action selection. Both rows use semantic units with stated rules and the same candidate step sizes, and the interpreter executes the selected size deterministically.

Model outputs	Step size from	Tennis	Stack	Peg-tight	Overall (%)
Action unit	**Adaptive Step plugin**	**17 / 20**	**16 / 20**	**14 / 20**	**78.3**
Action unit + step size	Model	14 / 20	12 / 20	5 / 20	51.7 (−26.7)

A.2.8 Ablation on Action Representations for Policy Learning

In the zero-shot setting of Appendix A.2.6, the agent knows what each action does only from its name and written convention. A fine-tuned policy can instead learn this from demonstrations, so the action representation may matter differently after fine-tuning. We ask three questions: how semantic units compare with other representations of the same end-effector delta, whether the semantic labels still matter once the policy is trained, and whether the baselines are held back by learning from GUMI demonstrations rather than expert teleoperation.

Setup. Every row fine-tunes the same Qwen3.5-2B backbone as in Sec. 4.1 using the training pipeline of (StarVLA Community, 2026); all settings are fixed except the delta representation and

corresponding action decoder for controlled ablation. Ours selects one semantic unit per step: a single-axis delta of fixed size set by the interpreter, or a gripper command, named by its effect, such as `MV_LEFT` or `GRASP`. Two variants change only how this unit is written: arbitrary labels replace every unit name with a fixed letter (`A`, `B`, ...) in both targets and action history and drop the directional descriptions, and fixed-step numbers write the same unit as a signed number (`dx=-2`). A third keeps one decision per step but lets the model set the step: free numeric steps predict the end-effector displacement recorded at each decision (e.g., `dx=-1.8 dy=0.3 dz=0`). The three baselines instead predict the full delta, translation and rotation plus the gripper command, and differ in how its magnitude is encoded. OpenVLA-style (Kim et al., 2024) discretizes each dimension into 256 bins with a dedicated token per bin, VLA-0-style (Goyal et al., 2025) writes each dimension as an integer in plain text, and π-style (Black et al., 2025) generates continuous delta chunks with a flow-matching action expert. Unless noted otherwise, every policy is trained on our mixed GUMI demonstration corpus (Appendix A.5). To examine whether trajectories collected with GUMI affect policy learning compared with conventional teleoperation, we additionally conduct a controlled comparison. For each task, the GUMI and teleoperation data contain the same number of expert episodes with similar target layouts and initial positions. Since trajectory lengths and sampling rates may differ across the two sources, we do not match the number of control steps or collection time. Evaluation uses Tennis $\rightarrow$ Plate, three-block stacking at 1 cm, and peg-tight, with the same 20 held-out layouts per task as in the zero-shot study of Tab. 5.

Table 7: Action representations for policy learning. Every row fine-tunes Qwen3.5-2B in the same pipeline and pools three training seeds (60 trials per task, 180 per row). Brackets are Wilson intervals over the pooled trials, which treat seeds that share layouts as independent.

Action representation	Magnitude decoupling	Output per decision	Success Rate
Trained on our mixed GUMI demonstrations			
Ours: Semantic unit (`MV_LEFT`)	✓	One of 9 units	**66.1** [58.9, 72.6]
Arbitrary label (`A`)	✓	One of 9 labels	63.3 [56.1, 70.0]
Fixed-step number (`dx=-2`)	✓	One of 9 units	61.7 [54.4, 68.5]
Free numeric step (`dx=-1.8`)	×	3-D displacement	38.9 [32.1, 46.2]
OpenVLA-style: Binned tokens	×	7-D delta, 256 bins	8.9 [5.5, 14.0]
VLA-0-style: Integer text	×	7-D delta, 1k integers	12.8 [8.7, 18.4]
π-style: Continuous chunks	×	7-D delta chunk	20.6 [15.3, 27.0]
Trained on episode-matched expert arm-to-arm demonstrations			
OpenVLA-style: Binned tokens	×	7-D delta, 256 bins	8.3 [5.1, 13.3]
VLA-0-style: Integer text	×	7-D delta, 1k integers	11.7 [7.8, 17.2]
π-style: Continuous chunks	×	7-D delta chunk	20.0 [14.8, 26.4]

(1) Predicting the step length costs the most. The three representations that leave the step length to the interpreter stay within 4.4 points of one another (semantic units against fixed-step numbers: b=17, c=9, p=0.17), while predicting the step length drops success to 38.9% (b=56, c=7, p<0.0001), and the baselines, which regress every dimension, reach 8.9–20.6%. The gap opens on the fine tasks: trained on 2 cm demonstrations, the free-step policy keeps taking about 2 cm steps and completes 18 / 60 stacks and 11 / 60 insertions, against 42 / 60 and 28 / 60 for semantic units, which only need a new interpreter setting; halving its outputs by hand helps little (41.1%). Free steps also allow off-axis motion and learn from recorded rather than commanded displacements, so this gap bundles the three factors; the zero-shot diagnostic of Tab. 6, which keeps the same units and adds only a scalar step length, isolates the magnitude and costs 26.7 points. Simulation (13 tasks $\times$ 50 trials $\times$ 3 seeds) repeats the ordering: semantic units lead free numeric steps 71.4 to 52.7, and 63.5 to 24.5 once execution switches to 1 cm steps for tight placement.

(2) Once trained, labels matter much less. Arbitrary labels trail semantic units by only 2.8 points (b=19, c=14, p=0.49), and by one to two of 60 episodes in every seed (38, 37, 39 against 39, 39, 41), against 20.0 points in the zero-shot setting (Tab. 5), since the demonstrations show the policy what each label does. Semantic names still speed up learning, reaching 90% held-out step accuracy at epoch 9 against epoch 17 for arbitrary labels, and they let people and frontier agents collect demonstrations through GUMI in the first place.

(3) GUMI and teleoperation data train the baselines comparably. For every baseline, GUMI data performs comparably to episode-matched expert data (OpenVLA-style 8.9 vs. 8.3, VLA-0-style 12.8

vs. 11.7, π-style 20.6 vs. 20.0), so on these tasks learning from GUMI rather than teleoperation does not explain the baselines' low scores, which Appendix A.2.9 traces to regressing magnitudes.

A.2.9 Baseline Protocol and Sanity Checks

Every learned method sees the same two views and starts from the same layouts. Our policy and the decoder variants of Tab. 7, which share its training pipeline, receive no proprioceptive state, while $\pi_{0.5}$ and GR00T also receive the robot's proprioceptive state. All trainable methods learn from the same corpus, about nine demonstrations per task on average (Tab. 14), the few-demonstration regime that Show-Harness targets. For the continuous baselines, GUMI episodes become trajectories of 7-D end-effector deltas, linearly interpolated from the measured poses at 10 Hz with zero-velocity pauses over 0.2 s removed. $\pi_{0.5}$ (openpi base checkpoint) and GR00T (N1.7 base checkpoint) are fine-tuned with their official recipes for 30K steps at batch size 32, three times the updates of our policy, and actions are rescaled per dimension to $[-1, 1]$ with training-set statistics: the 1st and 99th percentiles for $\pi_{0.5}$, the minimum and maximum for GR00T. Every learned method, ours included, is evaluated at its final checkpoint, so no checkpoint is selected on evaluation layouts. Both baselines predict 8-action chunks and execute 4 before replanning. Tab. 8 reports step-wise action accuracy on training and held-out demonstrations.

Table 8: Step-wise action accuracy (%). A step is jointly correct when the active axis moves the right way within 0.5 cm, the other axes stay within 0.5 cm and 3°, and the gripper command is right; drift is the mean off-axis displacement on held-out demonstrations.

Method	Training	Held-out demonstrations		
	Joint	Direction	Joint	Drift (cm)
Ours: Semantic unit	**99.1**	**94.1**	**92.8**	**0.00**
Action decoders in the same pipeline (Tab. 7)				
OpenVLA-style	96.4	86.9	56.1	0.71
VLA-0-style	95.1	87.6	59.6	0.62
π-style	94.6	89.4	61.9	0.48
VLA baselines (Tab. 2)				
$\pi_{0.5}$	93.8	88.8	60.4	0.51
GR00T	92.7	87.1	57.2	0.58

(1) The baselines fit their data and come close to published LIBERO results. Every method fits its training data, with joint accuracy of at least 92.7%; binned tokens reach 96.4%, so single-axis 2 cm moves are easy to discretize. Trained on standard LIBERO in the same pipeline, the three decoders reach 92.5%, 93.0%, and 94.2%, within 1.5–2.9 points of published results with 3–4B backbones (StarVLA Community, 2026; Goyal et al., 2025). Replayed open-loop on the Franka, 23 of 25 converted trajectories reproduce the original outcome; both misses are tennis balls rolling differently after release, not failures of the converted motion.

(2) The gap is in magnitude, not direction. On held-out demonstrations the baselines still choose the right direction 87–89% of the time, but joint accuracy falls to 56–62% because every step regresses all seven dimensions and drifts 0.5–0.7 cm along axes that should stay still. Semantic units keep 92.8% with no drift, since the interpreter fixes the length and moves one axis at a time.

(3) $\pi_{0.5}$ scores comparably on GUMI and teleoperation data. Trained with its official recipe on the three tasks of Tab. 7, $\pi_{0.5}$ completes 15 / 60 episodes with GUMI data and 14 / 60 with episode-matched teleoperation (b=4, c=3, p=1.0). Among the agent baselines, G-VLA's 87 cross-task failures in Tab. 2 have a clear structure: 29 from a generated goal image that does not match the scene, 34 from the executor missing the goal, 17 from failed grasps, and 7 from timeouts.

A.2.10 Cross-Embodiment Transfer without Target Data

The fine-tuned policy in Tab. 2 is co-trained on both arms. Here it is trained on one arm only, with every frame of the other arm excluded, and tested on the arm it has never seen. Directions are defined in the camera image rather than the robot frame (Appendix A.7), and each arm's interpreter maps a unit to its own motion, so the same decision can carry over. The two rigs also differ in their global camera, exocentric on the Franka and egocentric on the AgileX (Fig. 8), so the test changes

the view from which directions are judged as well as the arm. $\pi_{0.5}$ follows the same protocol: it acts in end-effector space, so each arm's inverse kinematics absorbs the difference between seven and six joints, and its deltas are expressed in the target arm's base frame through calibrated extrinsics.

Table 9: Cross-embodiment transfer on the five Plate tasks, 10 trials per task. Single-arm training excludes every frame of the other arm, so the test arm is never seen in training.

Training data	Test arm	FT (Qwen3.5-2B)	$\pi_{0.5}$
Co-trained on both arms (Tab. 2)			
Franka + AgileX	Franka	45 / 50	22 / 50
Franka + AgileX	AgileX	42 / 50	19 / 50
Trained on one arm			
Franka only	Franka	44 / 50	21 / 50
Franka only	**AgileX**	**32 / 50**	4 / 50
AgileX only	**Franka**	**27 / 50**	3 / 50

(1) The policy transfers to an unseen arm. Trained on the Franka alone, the policy completes 32 / 50 on the AgileX (76% of co-training), while $\pi_{0.5}$ completes 4 / 50 (b=29, c=1, p<0.0001). In reverse, the 63 AgileX demonstrations alone give 27 / 50 on the Franka (60% of co-training).

(2) Directions carry over; grasp timing needs a few examples. On AgileX frames, the Franka-only policy loses only 3.7 points of step accuracy on the six direction units (93.0 to 89.3%) but 12.3 points on `GRASP` and `RELEASE`, where the other gripper looks different; $\pi_{0.5}$'s direction-sign accuracy instead falls from 88.8 to 61.2%, since it maps pixels to one robot's frame. Adding 5, 10, and 20 AgileX demonstrations lifts success to 72%, 78%, and 82%, 98% of the co-trained result.

A.2.11 Benchmarking Results on LIBERO-Plus

To further benchmark our fine-tuned policy against published results, we evaluate it on LIBERO-Plus (Fei et al., 2025), which perturbs LIBERO scenes along seven dimensions: camera viewpoint, robot initial pose, language paraphrasing, lighting, background texture, sensor noise, and object layout. Our training set for this benchmark is derived from the LIBERO demonstrations (Liu et al., 2023b): we infer semantic units from the recorded trajectories and reconstruct them from object geometry, which yields a dataset with only 15 discrete actions: the six single-axis translations `MV_FWD`/`MV_BACK`, `MV_LEFT`/`MV_RIGHT` and `MV_UP`/`MV_DOWN`, the six single-axis rotations `RT_YAW_CW`/`RT_YAW_CCW`, `RT_PITCH_CW`/`RT_PITCH_CCW` and `RT_ROLL_CW`/`RT_ROLL_CCW`, and `GRASP`, `RELEASE` and `DONE`. We fine-tune Qwen3.5-2B on it for 4 epochs without proprioceptive input and evaluate it under the LIBERO-Plus official protocol.

Table 10: Success rates (%) on LIBERO-Plus. All baseline rows are the all-suite averages reported by Fei et al. (2025), and $\pi_{0.5}$ from Zhan et al. (2026). Avg. is the overall success rate across the seven perturbation dimensions; ours follows the LIBERO-Plus official evaluation protocol.

Method	Camera	Robot	Language	Light	Background	Noise	Layout	Avg.
OpenVLA	0.8	3.5	23.0	8.1	34.8	15.2	28.5	15.6
π_0	13.8	6.0	58.8	85.0	81.4	79.0	68.8	53.6
OpenVLA-OFT	56.4	31.9	79.5	88.7	**93.3**	75.8	74.2	69.6
$\pi_{0.5}$	64.8	71.8	83.0	**93.5**	92.2	78.8	**85.5**	81.4
Ours	**72.9**	**82.0**	**90.5**	87.3	83.7	**82.1**	80.5	**82.6**

Over all perturbed episodes, Ours reaches 82.6%, on par with $\pi_{0.5}$ (81.4%) and above every other published entry, from OpenVLA at 15.6% to OpenVLA-OFT at 69.6%. It scores higher on Camera, Robot, Language, and Noise, with the largest margins on Robot (+10.2) and Camera (+8.1), the dimensions that move the arm or the viewpoint rather than the appearance of the scene, in line with the viewpoint row of Tab. 2; the margin on Language (+7.5) reflects the instruction following that the VLM backbone retains when it acts through language. The remaining dimensions go to $\pi_{0.5}$ (Light and Layout) and to OpenVLA-OFT (Background), the appearance-level shifts that large-scale robot pretraining covers well.

A.2.12 Open-Weight VLMs as Zero-Shot Agents

The zero-shot agents in the main paper use closed-source frontier models; here, we test whether the same interface also enables 30B-scale open-weight VLMs to act, and examine the gap to API-based frontier models. Specifically, we run Qwen3.5-27B (Qwen Team, 2026) and Gemma-4-31B-it (Team et al., 2026) in the same harness as the API agents. Both models run with thinking enabled in vLLM on a single H200 GPU and are evaluated with absolute end-effector poses (`L1`) and our semantic units (`L3`) defined in Sec. 4.4 on the five Plate tasks, with 10 trials per task from the same layouts. We further conduct an offline probe on logged frames, asking each model to map the movement direction from adjacent observations and measuring classification accuracy. We also record the fraction of decisions that violate explicit action rules specified in the prompt. Tab. 11 shows that neither model completes any episode with absolute poses, while semantic units enable Qwen3.5-27B and Gemma-4-31B-it to complete 16 / 50 and 19 / 50 episodes, respectively. Our interface thus lowers the capability that robot control requires: with absolute poses neither open model completes an episode, while with semantic units each completes about a third.

Table 11: Performance of open-weight VLMs as zero-shot agents on the five Plate tasks. Direction-mapping accuracy comes from the offline probe on logged frames; rule violations are the share of decisions that break an explicit rule stated in the prompt.

Model	Success rate		Direction mapping accuracy (%)	Rule violations (%)
	Absolute pose	Show-Harness		
Open-weight, zero-shot				
Qwen3.5-27B	0 / 50	16 / 50	61.3	12.7
Gemma-4-31B-it	0 / 50	19 / 50	68.8	8.4
API reference				
Gemini-3.1 Pro	22 / 50	48 / 50	91.3	1.9

A.2.13 Runtime Cost and Human Operation

We report execution time and cost on the five Plate tasks, together with human operation through the same semantic interface, as shown in Tab. 12.

Accounting. API cost covers controller and planner calls at each provider's published per-token list price at the time of the runs (e.g., \$2 / \$12 per million input / output tokens for Gemini-3.1 Pro and \$5 / \$25 for Claude Opus 5), and cost per success divides total expenditure, including failed episodes, by the number of successes. A *pass* is one complete model evaluation and latency is its wall-clock duration, defined identically in every row; for $\pi_{0.5}$ a pass is one action-chunk prediction, not the 10 Hz rate at which the chunk is then streamed to the controller. Duration averages all episodes, including failures. Local inference is priced on the same basis as the API, as the rented GPU-hour a deployment would pay (\$1.00 per GPU-hour, held for the whole episode because the accelerator cannot be timeshared mid-episode), and operator time at \$18/h over about three minutes per episode, including scene reset. Three operators, including one without robotics experience and none involved in interface design, issue the same units by keyboard through the same interpreter, each completing ten scenarios per task under the agent's step sizes and budgets after a 15-minute briefing and five practice episodes, which are excluded from the results.

(1) Stepwise control takes as long per episode as continuous control. On the same tasks and arm, our fine-tuned policy and $\pi_{0.5}$ take the same time per episode to within a percent, and ours succeeds on 45 / 50 against 22 / 50. What costs time is calling a frontier API: controller inference takes 77.4% of the episode, and running the same units locally is 4.2× faster and 20.4× cheaper per success than calling the API.

(2) More thinking buys fewer calls, not success. Raising the thinking effort cuts calls on both Gemini-3.1 Pro and GPT-5.6-sol, while success stays flat and time and money climb: GPT-5.6-sol takes 3.36× longer at high effort than at low. We therefore run Gemini-3.1 Pro at medium effort by default; the strongest model buys one extra success at four times the price.

(3) Humans do well through the same interface. Operators complete 148 / 150 Plate trials and 78 / 90 fine-task trials (stacking, chess-cannon placement, and peg-tight) using the same semantic

Table 12: Runtime and cost on five Plate tasks. The pass column counts model calls for API rows, local forward passes, and keyboard action steps for operators; cost per success includes failed trials.

Model / effort	Success	Forward passes	Latency / pass (s)	Duration (min)	Cost / episode	Cost / success
API charges						
Gemini-3.1 Pro / low	47 / 50	27	6.8	4.30	$0.26	$0.28
Gemini-3.1 Pro / medium	**48 / 50**	26	10.0	5.41	$0.47	$0.49
Gemini-3.1 Pro / high	47 / 50	25	13.0	6.60	$0.90	$0.96
GPT-5.6-sol / low	44 / 50	32	3.9	3.60	$0.59	$0.67
GPT-5.6-sol / medium	46 / 50	28	11.6	6.50	$0.90	$0.98
GPT-5.6-sol / high	46 / 50	26	25.1	12.10	$1.62	$1.76
GPT-6 Astra / medium	49 / 50	24	11.5	5.20	$1.94	$1.98
CaP-X	23 / 50	9.4	7.1	1.92	$0.26	$0.57
Local inference, priced as rented GPU-hours						
FT Qwen3.5-2B (RTX 5090)	45 / 50	31.8	0.039	1.28	$0.021	$0.024
$\pi_{0.5}$	22 / 50	168	0.06	1.29	$0.021	$0.049
Human operation through the same interface						
Operators via GUMI	148 / 150	21.4	0.86	1.08	$0.90	$0.91

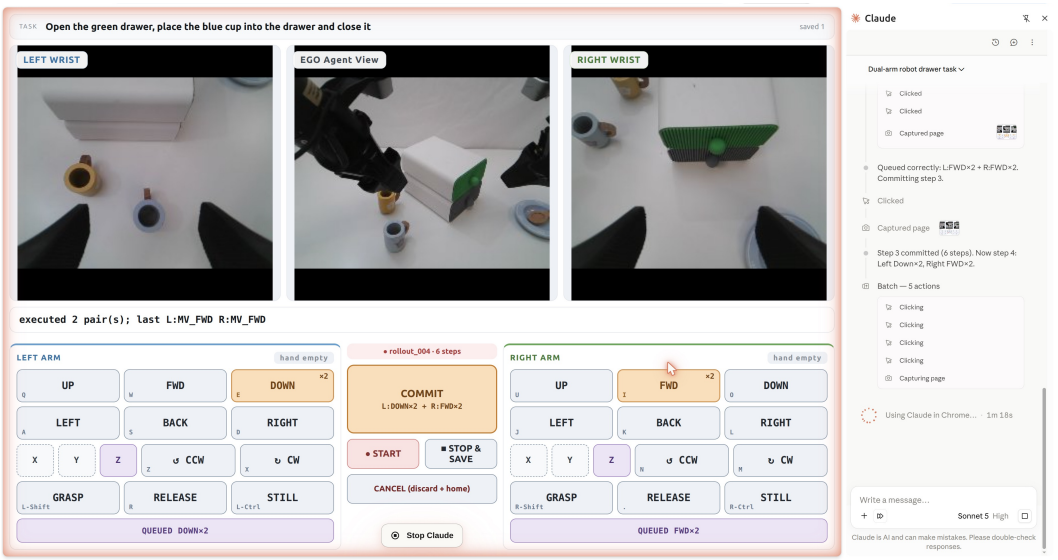


Figure 7: **The GUMI interface.** GUMI is a GUI-based manipulation interface that enables humans and frontier agents to autonomously collect demonstrations through the same semantic controls.

units, and each of the three scores between 91.2% and 97.5% across all eight tasks. They and the agent both do worst on peg-tight and best on stacking, so tight contact is hard for people as well, and force or compliance units are the natural next addition.

A.2.14 GUMI AS AN AGENT INTERFACE

GUMI was built so that people can drive the robot from a web page (Sec. 3.4, Fig. 7); here a computer-use agent drives it the same way. Claude Opus 5 (Anthropic, 2026) either emits semantic units directly or clicks the matching GUMI buttons, and Gemini-3.1 Pro (Google DeepMind, 2026) emitting units directly serves as the default reference.

Setup. All three conditions run the five Plate tasks, 10 trials each from the same layouts, with the same arm, interpreter, step sizes, and 50-step cap. The GUMI page shows the same two camera views, proprioception readout, subtask plan, and five-step history as the direct agent receives, and

the prompt carries the same guidance with button names in place of tokens. Clicks queue in GUMI and execute back to back, so one model turn can issue several units.

Table 13: Driving the robot through GUMI on the five Plate tasks, 10 trials per task from the same layouts. Tokens compare only the Claude rows; parentheses give the cost with prompt caching.

Agent / channel	Success	Steps	Calls	Duration (min)	Input tokens	Output tokens	Cost / success
Gemini-3.1 Pro / direct	48 / 50	30.2	26.0	5.41	87.4K	24.4K	$0.49
Claude Opus 5 / direct	44 / 50	35.0	30.0	6.85	67.2K	29.7K	$1.23
Claude Opus 5 / GUMI	43 / 50	38.6	**10.4**	**3.94**	179.7K	13.1K	$1.43 ($0.69)

(1) Clicking works as well as emitting. Through GUMI, Claude Opus 5 succeeds on 43 / 50 episodes against 44 / 50 when it emits the same units directly (b=3, c=2, p=1.0), so the semantic units serve the model equally well as structured output or as buttons on a page.

(2) Batching makes the GUI route faster. Because clicks queue, the agent issues several units per turn and needs 2.9× fewer model calls, finishing 1.74× faster than the same model emitting directly and 1.37× faster than the default Gemini agent. It sizes each batch itself, about seven units per turn in free space and two near placement, the same pattern as the Action Chunking plugin (Fig. 6). We therefore attribute the speed-up to batching rather than to the GUI channel itself.

(3) The GUI route trades input for calls. Full-page screenshots and a history resent every turn double the tokens per episode: input grows 2.7× while output falls to less than half. Most of that input repeats from turn to turn and can be cached, whereas output tokens, which caching does not discount, make up 69% of the direct route's cost; with prompt caching the GUI route costs $0.69 per success, against $1.23 for direct output and at least $0.88 even if every direct input token were a cache hit.

A.3 VLM Fine-Tuning Details

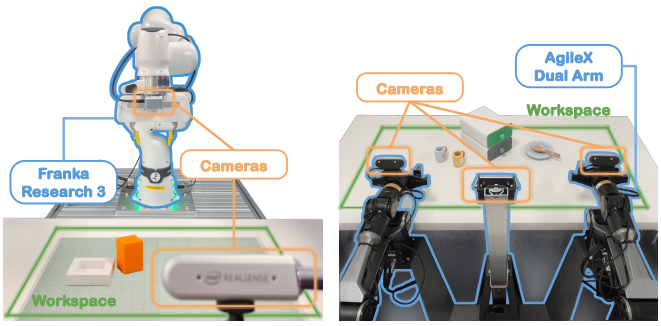
For lightweight VLM fine-tuning, we train for 40 epochs on the 7.9K single-arm samples at a learning rate of 1×10^{-4} under a cosine schedule with a warmup ratio of 0.1, using bf16, 256×256 views, and an effective batch size of 32. The fine-tuning is lightweight and can be performed on 24 GB-class GPUs. Fine-tuning Qwen3.5-2B takes under two hours on a single H200.

Table 14: Demonstration corpus collected through GUMI.

	Platform	Tasks	Episodes	Ep./Task	Steps	Steps/Ep.
Real Robots	Franka (7-DoF)	9	101	11.2	4969	49.2
	AgileX (6-DoF)	10	63	6.3	2805	44.5
	total	19	164	8.6	7774	47.4
Simulation	ManiSkill	1	100	100.0	5840	58.4
	RoboLab	12	130	10.8	7683	59.1
	total	13	230	17.7	13523	58.8

A.4 Hardware

Our experiments use two robot platforms, each equipped with parallel-jaw grippers (as shown in Fig. 8). The first is a 7-DoF Franka Research 3 arm, observed by an exocentric Intel RealSense D435 facing the workspace and a wrist-mounted Intel RealSense D405, providing two views. The second is a bimanual AgileX rig with two 6-DoF arms, observed by three Orbbec Dabai DC1 cameras: an egocentric view shared by both arms and one wrist view per arm. Local policies are served on a single RTX 5090 GPU and the open-weight zero-shot agents (Appendix A.2.12) on a single H200, while frontier models are accessed through their public APIs, with all calls made on or before September 25, 2026. For the viewpoint shift in Tab. 2, we move the exocentric camera and keep the interpreter's R_E as calibrated for the default view, so the system receives no new extrinsics. The



(a) Franka (b) AgileX (dual-arm)

Figure 8: Real-robot rigs, with overlays marking the cameras and the reachable workspace. The 7-DoF Franka Research 3 uses an exocentric RealSense D435 and a wrist-mounted D405; the bimanual AgileX pairs two 6-DoF arms with one shared egocentric view and one wrist view per arm.

wrist camera, which guides the final alignment, moves with the gripper and is unaffected, and since every step is observed before the next decision, a unit whose motion looks rotated in the new view is corrected on the following step.

A.5 Demonstration Data

Tab. 14 breaks the training corpus down per platform. The real-robot demonstrations cover 19 tasks, nine distinct object–target combinations on the Franka and ten on the AgileX, concentrated on pick-and-place with a few stacking and shelf-rearrangement episodes. All real-robot episodes are recorded through GUMI with the same 2 cm translation step size, so a semantic unit denotes the same displacement on the Franka and both AgileX configurations. Each episode stores the third-person and wrist frames for every step together with the emitted unit, the measured end-effector pose, and the gripper width; a training sample pairs the two views at one step with the unit taken at that step, plus one terminal sample per episode that emits `DONE`. Beyond clean executions, 19 Franka episodes (594 steps) are recorded specifically as grasp recovery: the gripper starts short of, past, or to either side of the intended grasp point, and a close that comes up empty is followed by lifting the gripper and re-approaching rather than transporting an empty hand. These episodes are trimmed so that only the corrective segment is kept, which is why they run about 31 steps against 53 for the freely collected ones. The 50-step cap applies only to evaluation, where episodes take about 30 steps (Tab. 12); demonstrations are recorded without it.

For simulation data, we collect demonstrations from ManiSkill (Tao et al., 2024) and RoboLab (Yang et al., 2026b). Across both simulators, we keep the model-facing control and observation conventions consistent with the real Franka setup: each semantic translational action unit corresponds to a 2 cm single-axis Cartesian translation, and camera views are converted to the same input format used at deployment. At the same time, we retain substantial variation in rendering, scene composition, and embodiment appearance to preserve a meaningful sim-to-real gap. ManiSkill provides 100 tabletop pick-and-place episodes involving an orange block and a coaster, while RoboLab provides 130 episodes over twelve tasks, including coloured blocks into bowls, left-versus-right bowl selection, canned food and yogurt into containers, a Rubik's cube, and throwing away an apple.

A.6 Qualitative Analysis

We further provide qualitative results on a diverse set of real-world manipulation scenarios. As shown in Fig. 9, Show-Harness successfully handles a broad range of visual and physical variations, including novel objects, background changes, lighting changes, cluttered scenes, spatial-reasoning tasks, and bimanual control. Beyond standard pick-and-place, the same interface supports more challenging behaviors such as rearranging letter blocks to satisfy a semantic goal, and coordinating two arms to open a drawer and place an object inside. These examples qualitatively demonstrate that the proposed semantic action interface can be reused across substantially different task structures and manipulation requirements without redesigning the model-facing action space.

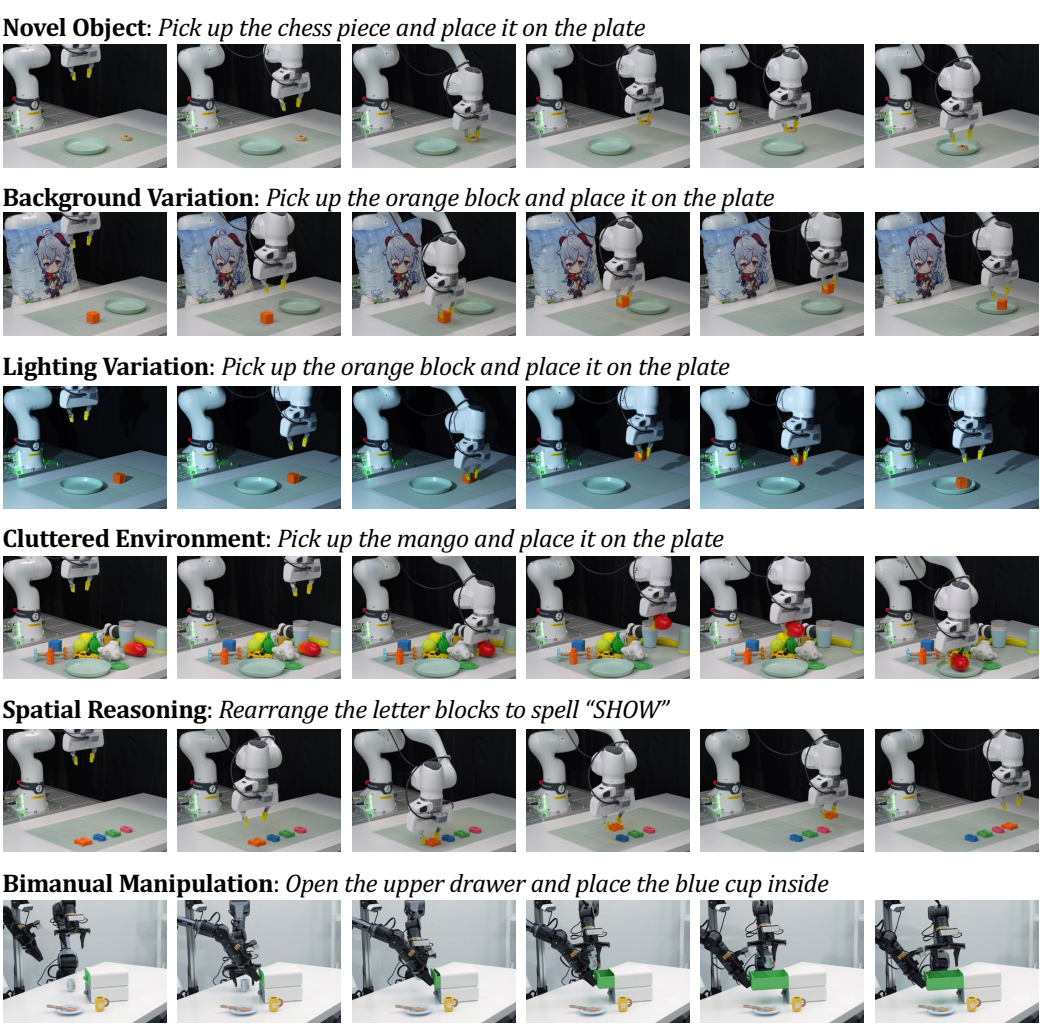


Figure 9: Real-world demonstrations of Show-Harness agents across diverse tasks and conditions.

We further provide a qualitative comparison with $\pi_{0.5}$ in Fig. 10. For a fair comparison, Show-Harness uses a Qwen3.5-2B VLM fine-tuned on our collected demonstrations, while $\pi_{0.5}$ is fine-tuned on the corresponding low-level action trajectories converted from the same demonstration set. Under this matched-data setting, Show-Harness consistently completes the manipulation sequence across the chess piece, tennis ball, and teddy bear examples, whereas $\pi_{0.5}$ exhibits grasping or object-interaction failures, as highlighted in red. These examples qualitatively illustrate the benefit of learning over the shared semantic action interface rather than directly fitting embodiment-specific low-level actions. More showcases can be found in the Anonymous Repository.

A.7 Runtime Prompts

We report one canonical prompt for each distinct model role rather than enumerating every embodiment- or backend-specific variant. Braced fields such as {`task`} are populated with the current task, plan state, measurements, or action history immediately before the model call; images are supplied separately through the model API. By default, rotation-related guidance and actions are omitted from the prompt and enabled only for tasks that require end-effector reorientation.

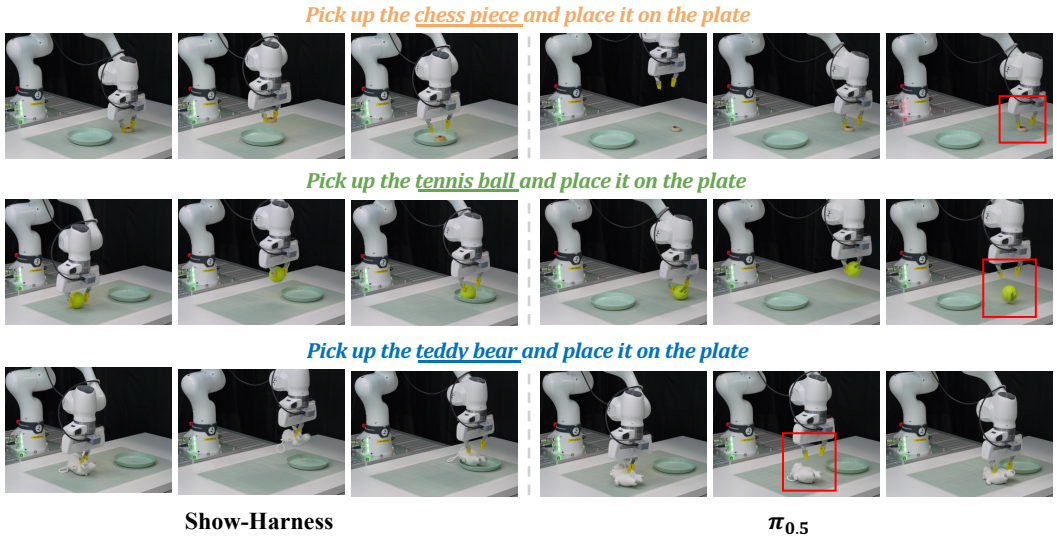


Figure 10: Qualitative comparison with $\pi_{0.5}$. Show-Harness completes all three examples, while $\pi_{0.5}$ fails through gripper–object collisions or unstable grasps (red boxes).

A.7.1 Frontier VLM Agent

Show-Harness wraps the same VLM into two roles within the agentic loop: (1) a planner that decomposes the task into visually verifiable stages, and (2) a controller that selects the next semantic action from the current context to control the robot. At runtime, the controller template is augmented with the active plugin context and an output contract.

Controller prompt ($\mathcal{P}_{\text{ctrl}}$)

```
TASK: {task}
STAGE: {stage}
TARGET: {target}
AFFORD: {affordance}
Stage goal: {description}
DONE WHEN: {completion}
Gripper now: {gripper_state}
{mem_text}
{recovery}
{proprio}

DIRECTION:
Is STAGE for grasping AND TARGET inside the wrist view?
A) YES -> wrist is the primary guide. Judge AFFORD's position vs the {gripper_color}
     gripper and take the direction of LARGEST deviation:
- AFFORD to the gripper's left  -> MV_LEFT
- AFFORD to the gripper's right -> MV_RIGHT
- AFFORD near image bottom and far from the grippers -> MV_FWD
- AFFORD between the image top and the grippers -> MV_BACK
- AFFORD roughly centered between the grippers -> MV_DOWN
{rotation}
B) NO -> AgentView is the primary guide. Judge TARGET's position vs the end effector
     (before GRASP) or holding object (after GRASP), and take the direction of LARGEST
     deviation:
- TARGET near the image left  -> MV_LEFT
- TARGET near the image right -> MV_RIGHT
- TARGET near the image bottom -> MV_FWD
- TARGET near the image top -> MV_BACK
C) MV_UP when:
- Need to lift the object
- too low to reach TARGET
- retreating after a RELEASE

ATTENTION:
{mem_text_rules}
- DONE only when "DONE WHEN" is already visible in the images
```

```
GRIPPER:
- GRASP when BOTH AgentView and Wrist view confirm the {affordance} is clearly between the
    center of two grippers
- RELEASE only when the held object is above its destination and lowered onto it
{variable_step}
{action_chunk}
Think one visual sentence, then commit.
{output_contract}
```

The default structured-decoding route fills {output_contract} as follows; reasoning-oriented backbones use the same action alphabet and recover the final unit from their response.

Atomic action output contract

```
Choose exactly one action:
MV_FWD, MV_BACK, MV_LEFT, MV_RIGHT, MV_UP, MV_DOWN, GRASP, RELEASE, DONE
Return JSON only: {"decision":"ONE_ACTION","reasoning":"one visual sentence"}
```

The default harness adds only short, state-dependent fragments to this controller. The listing below consolidates the fragments that realize Proprioception, Multi-View Guidance, Action Chunking, Adaptive Step, Action History, and Failure Recovery. Bracketed labels identify the owning plugin and are not sent to the model.

Default plugin injections ($\Phi_{\mathcal{P}}$)

```
[proprioception/block]
The gripper is {gap_cm} cm above the table; {step_sizes}. {hint}

[proprioception/step_sizes_fine]
each step moves ~{fine_cm} cm

[proprioception/step_sizes_coarse]
each step moves ~{fine_cm} cm ({coarse_cm} cm)

[proprioception/hint_descend]
If height > {high_cm} cm, MV_DOWN first.

[proprioception/hint_holding]
Holding an object: lift until clear of the table; descend only to place.

[proprioception/descend_stall]
Last MV_DOWN lowered {moved_cm} of {commanded_cm} cm -> already in contact, do NOT MV_DOWN
    again

[wrist_marker/marker]
WRIST CHECK: begin your reasoning with `WRIST: YES` if the TARGET is visible in the wrist
    view, else `WRIST: NO`.

[action_chunk/plan]
ACTION PLAN -- only when WRIST: NO (TARGET far): plan your next {step_num} moves as `PLAN:
    M1, M2, ...` (MV_ tokens only) and set decision to M1. Choose each move from the
    height and step size so the plan does not overshoot (e.g. never plan more MV_DOWN
    than the height above the table allows). When WRIST: YES, decide a single move.

[mem_text/recent]
Recent moves, newest first: {moves}

[mem_text/rules]
- If recent moves show GRASP(empty), do not GRASP in place again; prioritize MV_UP,
    MV_BACK, MV_DOWN, or MV_FWD
- NEVER OSCILLATE: Do NOT choose the opposite of the newest recent move (Pairs:
    MV_LEFT/MV_RIGHT, MV_FWD/MV_BACK)
- When opposite directions appear in recent moves, prioritize MV_DOWN or MV_UP

[recovery/context]
Recovery: {note}

[recovery/note_empty_grasp]
Empty close; do not retry on an edge/corner. Recenter body and confirm depth.

[recovery/note_unverified_grasp]
Grasp not verified; continue GRASP until width and images show a real hold.

[recovery/note_lost_grasp]
```

```
Grasp lost; return to GRASP, recenter the object body, then confirm depth.

[recovery/note_unsettled]
Closed width is unsettled; wait before moving.
```

Subtask Planning prompt ($\mathcal{P}_{\text{plan}}$)

```
ROLE: SubgoalPlanner
TASK: {task}

{video_ref}

Return an ordered JSON plan:
{{
  "subgoals": [
    {{
      "id": "short_snake_case_id",
      "target": "object or destination",
      "affordance": "visible part or placement region",
      "motion": "semantic stage label",
      "description": "visual strategy for this stage",
      "completion": "visible condition that means this stage is complete"
    }}
  ]
}}

### STRICT RULES

1. Stage Segmentation

-Break the task down into meaningful visual milestones (e.g., GRASP, LIFT, MOVE, PLACE,
     RELEASE, RETREAT)
-MERGE: Do NOT split immediate pre-grasp steps. Combine approach, align, lower, and close
     into a single `GRASP` stage
-SEPARATE: Keep lift/clearance after a successful grasp as a separate `LIFT` stage
-RETREAT: After every `RELEASE`, add a `RETREAT` stage that lifts the gripper up

2. Affordance Selection

-ONE specific part -- never alternatives like "left end or middle"
-Must be visible in AgentView & bracketable by open fingers
-Containers/Hollow objects (cups, bowls): Target left/right rim or edge.
-Simple Solid objects (blocks): main body

3. Completion Criteria

-ALL completion conditions MUST be strictly judgeable from raw 2D images
-Movement stages: End with a stable visual spatial relation, NOT a gripper event
-Set-aside placements: If moving an object to another location, ensure it is placed away
     from the origin point on the table
-MUST distinguish among similar objects

Return JSON ONLY.
```

The dual-arm deployment instantiates the same structure once per arm and predicts the two actions jointly, adding `STILL` for an idle arm; we therefore do not repeat the near-isomorphic controller and planner templates. Likewise, the specialized Situated Planning and Visual Prompt roles are used only in their targeted experiments, with full implementation details provided in our code repository.

Video-conditioned in-context planning. Video conditioning introduces a distinct input role for learning task procedures from demonstrations. An analyst converts temporally ordered demonstration frames into a textual task outline inserted at {`video_ref`} in the planner prompt, allowing the agent to infer the demonstrated procedure in context and reuse it during execution. Here we show the single-arm version, while the dual-arm variant differs only by attaching an arm identity to each operation.

Reference-video analyst prompt

```
ROLE: DemoVideoAnalyst
You see {num_frames} frames sampled IN ORDER from ONE reference video demonstrating a
     table-top manipulation performed by ONE arm (a robot gripper, or a human hand
     standing in for it).
```

```
A frame may be a composite of labeled camera panels (front view + wrist view); overlay
     text is auxiliary -- trust what the imagery shows.

Report exactly what is demonstrated, operation by operation, in the order performed.
     Capture the details that make replication faithful:
- object: name it so it cannot be confused with similar ones (color/size/position)
- grasp: the exact part grasped (stem, rim, edge, handle, top face, ...)
- destination: where it ends up, with the placement nuance (which side/half, on/into,
     orientation)
Report only what the frames show; do not invent steps between or beyond them.

Return JSON only:
{{"task":"one line: what the demo achieves","operations":[{{"action":"short verb
     phrase","object":"...","grasp":"...","destination":"..."}}]}}
```

A.7.2 FINE-TUNED LIGHTWEIGHT VLM AGENT

The fine-tuned lightweight VLM directly serves as the controller, without a separate planner by default. It takes only the task instruction, current multi-view observation, recent action history, and shared action vocabulary as input.

Post-trained VLM prompt ($\mathcal{P}_{\text{min}}$)

```
You are controlling a robot arm with two cameras:
- Agentview: overhead view of the robot and workspace
- Wristview: close-up view from the gripper

Task: {task}
Recent moves, newest first: {recent_moves}

Output exactly one action unit:
MV_FWD, MV_BACK, MV_LEFT, MV_RIGHT, MV_UP, MV_DOWN, GRASP, RELEASE, DONE

Check both camera views and choose the next action:
- Use AgentView to locate the target when it is not in wrist view
- Use wrist view to fine-align when target is visible up close
- GRASP when gripper fingers are aligned around the object
- RELEASE when object is above the destination
- DONE when the task is complete: the object is at its destination and the gripper is clear
- Avoid repeating a direction that conflicts with the most recent move

Return the single unit only, no punctuation, no explanation:
```