



Auriga Technical Report: A Task-Agnostic Agent Harness for Zero-Shot Embodied Manipulation

StellarEdge AI Technical Team

Date Nov. 1, 2026

 [Project Page](#) |  [GitHub](#)

Abstract

Vision-language models can interpret instructions and plan manipulation tasks, yet struggle when acting directly as robot policies. On RoboDojo, a large *planning–execution gap* separates high-level task decisions from physical execution: removing arm execution raises GPT-6 Astra’s success rate from 22.5% under the reference control interface to 76.2%. We present **Auriga**, a task-agnostic agent harness that closes 68.3% of this gap in zero-shot manipulation without target-task trials or resets. In Auriga, the model makes all task decisions over an explicit task belief and acts through either direct tool calls or synthesized multi-step Python programs, while the harness supplies calibrated 3D measurements, finger-referenced dual-arm control, and a single admission gateway to the robot. Before motion, the model inspects a visual trajectory preview while simulation is paused; during execution, the harness checks declared stage tolerances and grasp bounds, stopping the robot on divergence and returning the measured discrepancy for in-place recovery. Developed across 13 cross-task capability iterations and frozen for evaluation, Auriga ranks first on the RoboDojo leaderboard with a 57.7% success rate (64.0 Score), more than doubling the reference harness on the same model, and remains first under scene randomization while learned policies drop by up to 33.9%. On RoboCasa365, the same frozen harness reaches a 73.2% success rate, outperforming VLA-assisted agents by up to 23.1% on composite tasks.



1 Introduction

“Plans are worthless, but planning is everything.” — Dwight D. Eisenhower

In physical manipulation, a failed attempt changes the scene, often irreversibly. Unlike simulated settings where an agent can call `env.reset` repeatedly on a target task to tune coordinate offsets, control scripts, or failure notes before evaluation [11, 20, 31, 49], a deployed robot must complete an unseen task within a single episode. Tuning procedures across repeated trials on a fixed scene also fits to that scene’s layout, and performance drops when object positions are randomized or the task changes. Generalization in manipulation therefore separates two problems: across tasks during development, using failures to improve the shared perception, control, and verification tools rather than task-specific scripts; and during execution on an unseen



Figure 1 Auriga completes manipulation tasks zero-shot within a single episode. A general-purpose model plans and acts through a shared harness, with no target-task exploration, demonstrations, or resets. The model localizes targets by multi-view triangulation, complemented by depth and point-cloud estimates (1), and reviews candidate motions before they are executed (2). The harness verifies each execution and stops when an action fails (3); the model then revises its plan and recovers from the current state (4–5) until the task succeeds (6).

task, completing the task in one episode without prior trials, demonstrations, or resets (Sec. 2.1).

On a 210-episode diagnostic across all 42 tasks of the RoboDojo benchmark [4], the main bottleneck is physical execution rather than task reasoning (Sec. 2.2 and Tab. 2). When GPT-6 Astra is given a privileged action that moves objects along chosen waypoints without the robot arm, it succeeds in **76.2%** of episodes; when it drives the arm through the benchmark’s reference end-effector interface (RoboProbe [60]), it succeeds in **22.5%**. The model usually identifies the right object and destination, but the arm misses a grasp by a few millimeters, loses the object in flight, or hits a container rim during transport. Common control interfaces do not catch these errors in time. Step-wise and chunked action interfaces [36, 60, 62] issue motions from the latest frames without showing the full route in advance, so collisions appear only after the arm has moved. Program generation [9, 21] hands execution to a script that cannot see the scene, where normal program exit does not mean the intended contact succeeded.

We present **Auriga**, a task-agnostic agent harness that narrows this planning–execution gap within a single zero-shot episode (Figs. 1 and 2). In Auriga, the model makes all task decisions, while the harness measures the scene, checks and executes motions, and reports what happened:

- 1. Task-agnostic harness and explicit task belief (Sec. 3).** Auriga contains no task-specific code, prompts, or heuristics, and exposes no privileged simulator state. One persistent model session plans the task, chooses actions, and recovers from errors. As an episode grows long, the model records its current plan, open questions, and task premises (pending, supported, contradicted, or unknown) in a concise *task belief* B_t , while full interaction logs stay in an off-context *evidence record* E_t retrieved on demand.
- 2. Preview and verification through a single admission gateway (Sec. 4).** Every physical motion,

whether called directly or from a model-written program, passes through the admission gateway $\mathcal{A}$ in a *preview–execute–verify* loop. While simulation is paused, $\mathcal{A}$ plans a multi-stage dual-arm sequence into a trajectory preview showing the path, step cost, and clearance warnings. During execution, the previewed trajectory runs while $\mathcal{A}$ checks model-set arrival tolerances $\tau_k = (\delta_p, \delta_\theta)$, gripper openings, and observation checkpoints. If a stage misses its tolerance, $\mathcal{A}$ stops the robot and returns a receipt r_t with the completed stages, measured deviations, unexecuted stages, and fresh camera views so the model can recover from the current state (Fig. 2).

3. **3D geometric measurements and finger-referenced actions (Sec. 5).** On the input side, camera calibration, VGGT-Omega [46] depth, and SAM 3.1 [2] segmentation provide 3D object estimates, ray–plane intersection with height sensitivity, and two-view triangulation, together with in-hand slip measurements and warnings when a motion re-enters a past failure region. On the output side, the model sets goals at the finger contact region rather than the wrist, chooses tolerances per stage, and coordinates both arms on a shared timeline.

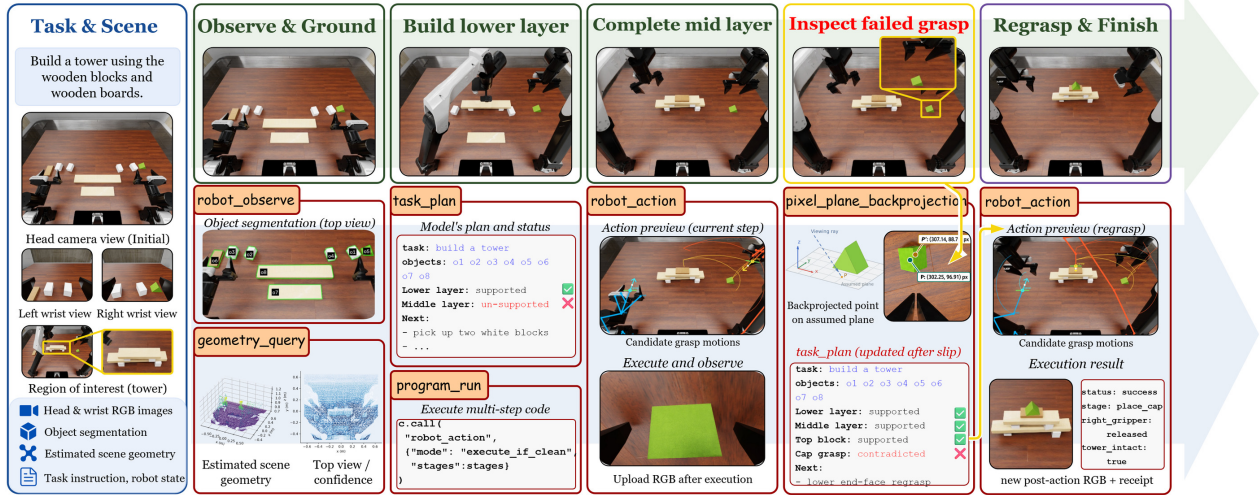


Figure 2 Planning, execution, and recovery within a single tower-building episode. From left to right, the model grounds the task in visual and geometric observations, constructs a plan, and builds the tower through direct tool calls and generated programs. Action previews and execution feedback support a *preview–execute–verify* loop. After a failed grasp, the model inspects the scene, updates its plan, and regrasps the object to complete the tower. The harness provides measurements and execution checks; the model selects actions and recovery strategies. The scene is never reset.

Auriga separates cross-task tool development from single-episode execution (Sec. 2.3). Over 13 development iterations (Fig. 3), manipulation failures were used to identify missing capabilities in 3D localization, contact checking, and state tracking, raising zero-shot performance across all 42 RoboDojo tasks from 36.2% SR (45.5 Score) at iteration # 4 to 57.7% SR (64.0 Score) in the final harness. During evaluation, the harness is frozen and runs under the single-episode zero-shot protocol (Secs. 2.1, 6 and 6.1), with no target-task exploration, demonstrations, or resets:

- **RoboDojo [4].** Across all 42 tasks and five dimensions (6,300 episodes over three seeds), Auriga with GPT-6 Astra reaches an average **Score of 64.0** and **SR of 57.7%**, ranking **first overall on the leaderboard** across learned policies and agent methods, more than doubling the GPT-6 Astra reference harness (**29.0 Score, 22.5% SR**), and leading all 12 columns of the agent block (Tab. 4). On the 210-episode diagnostic set, Auriga closes **68.3%** of the gap between reference arm control and arm-free object motion (Tab. 2).
- **Scene randomization.** Across the 12 Generalization tasks (Fig. 9 and Tab. 17), learned policies drop by up to 33.9% in SR on randomized layouts, whereas **Auriga ranks first on both standard scenes**

(52.3% SR) and random scenes (45.6% SR) and achieves the highest generalization score ($G = 46.3$).

- **RoboCasa365 [32]**. On the 50 Target50 mobile manipulation tasks in unseen kitchens (250 episodes), Auriga reaches **73.2% Overall SR (92.2% Atomic-Seen, 67.5% Composite-Seen, 57.5% Composite-Unseen)** without a learned policy or cross-episode memory, outperforming VLA-assisted agents RPent [61] (59.2%) and PyRUA-Lean [43] (56.4%) by **19.4%** and **23.1%** on composite tasks (Tab. 5).

2 Problem Setting and Positioning

We study zero-shot manipulation, in which a general-purpose model uses a shared robot harness to construct and execute a task procedure from the task specification and interaction feedback. We first define the single-episode zero-shot evaluation protocol (Sec. 2.1), then quantify the planning–execution gap that motivates our harness (Sec. 2.2), and finally position cross-task harness evolution relative to within-task trial search (Sec. 2.3).

2.1 Zero-Shot Single-Episode Protocol

Task and interaction. The unit of evaluation is a *case*, defined by a task instruction and a specified scene variant, random seed, and layout. Each case starts from an initial scene and is evaluated under the benchmark’s execution budget and success criteria. The agent receives the task specification and accesses observations and robot actions through the shared harness. It uses the available observations and interaction history to choose subsequent tool calls or programs, then receives feedback from their execution. This interaction continues until success, termination, or budget exhaustion. The benchmark’s evaluator determines task success from the resulting execution.

Zero-shot protocol. We define zero-shot evaluation through three conditions:

1. **No target-task exploration or development before evaluation.** Evaluation starts with a pretrained model, a fixed shared harness, and task-independent operating guidance. No preparatory exploration or trials are conducted on the target task, including its other layouts, to develop or select task-specific prompts, programs, skills, or policies. Model parameters remain fixed throughout evaluation.
2. **No demonstrations at evaluation time.** The agent receives the task specification and permitted observations, without additional solution examples or expert demonstrations. For memory tasks, the benchmark-defined event sequence forms part of the case’s observation input.
3. **One execution per case.** Each case is initialized and executed once, without a reset or rerun. Its outcome is recorded whether the episode succeeds or fails.

Within-episode adaptation. An episode comprises successive model decisions, observations, and physical actions. During this interaction, the agent may inspect the scene, write and revise programs, preview motions, revisit its history, and recover from a failed grasp or placement. Recovery continues from the physical state left by earlier actions under the remaining execution budget, rather than resetting the scene to start over. Plans, generated code, and interaction memory evolve within the current episode and are not transferred to subsequent evaluation cases. Tab. 1 compares six aspects of target-task preparation and information available to the test agent across representative manipulation settings (scope and per-setting qualifications are detailed in App. A).

Table 1 Target-task preparation and evaluation inputs across the specified settings.

Method / setting	Before evaluation		Available to the test agent			
	Task exploration	Cross-trial refinement	Task code / skills	Trial-derived memory	Demo-derived artifacts	Demo references
GPT-6 Astra (zero-shot) [60]	X	X	X	X	X	X
GPT-6 Astra (image-demo ICL) [60]	X	X	X	X	X	✓
GPT-Policy (video ICL) [5]	X	X	X	X	X	✓
GPT-GEAR (full-context ICL) [64]	X	X	X	X	✓	✓
SenseNova (RoboDojo ICL) [42]	X	X	✓	X	✓	✓
PyRUA-Lean (RoboDojo) [43]	–	–	X	X	X	X
SEAR (hardware AaP) [49]	✓	✓	✓	✓	X	X
HexaShell (RoboDojo) [11]	✓	✓	✓	✓	X	X
EmbodiedEvo (frozen harness) [20]	✓	✓	✓	✓	X	X
WorldScape R ² SI (release) [29]	✓	✓	✓	X	X	X
Auriga (ours)	X	X	X	X	X	X

Notes. ✓: included; X: absent from the specified setting; –: not reported. Scope and qualifications are given in App. A.

2.2 The Planning–Execution Gap

Deciding without the arm. To separate what a vision-language model decides from how the robot arm executes it, we run the same harness with one additional privileged action, which moves an object along waypoints given by the model without using the arm. The model still chooses the object and every target pose from its own perception. The object moves under simulated physics, stops at a collision, and settles under gravity at the end, after which the action reports where the object came to rest. Success under this action therefore estimates what the model’s own decisions achieve when execution by the arm is taken out of the loop. We compare it with two executed systems on the same 210 episodes, five for each of the 42 RoboDojo tasks (Tab. 2). The first is RoboProbe [60], the reference harness released with the evaluation of GPT-6 Astra on RoboDojo [4], and the second is Auriga. Averaged over the five dimensions, the success rate is 76.2% with the privileged action, 22.5% for RoboProbe, and 59.2% for Auriga. The model therefore decides considerably better than it executes, and Auriga recovers 68.3% of this planning–execution gap in SR (70.6% in Score).

Table 2 Planning–execution gap on 210 episodes, five for each of the 42 tasks: SR (%) and Score (0–100) per dimension and their average over the five dimensions, as in Tab. 4. With direct object motion, the model moves objects through a privileged action instead of the arm. All methods use GPT-6 Astra, RoboProbe at medium and the others at high reasoning effort. [§]Memory over 29 episodes, since one episode has no counterpart in the reference results.

Method	Average		Generalization		Precision		Long-Horizon		Memory		Open	
	SR	Score	SR	Score	SR	Score	SR	Score	SR	Score	SR	Score
Direct object motion (privileged)	76.2	79.6	81.7	83.3	65.0	72.0	70.0	75.4	96.7	96.7	67.5	70.6
RoboProbe [60]	22.5	28.5	31.7	34.1	2.5	8.5	7.5	22.1	40.8 [§]	45.8 [§]	30.0	32.1
Auriga	59.2	64.6	43.3	46.2	50.0	55.6	45.0	59.5	90.0	90.5	67.5	71.2

Why common paradigms leave the gap open. Two common ways of placing a general model on a robot address execution only indirectly. In *step-wise action output*, as in the reference harness and in VLA policies [36, 60, 62], the model emits one motion step or chunk per call from the latest frames. It never sees the route of a motion before committing to it, and a failed motion is noticed only when the model reads the next frame, by which time the arm has already moved. In *code as policy* [9, 21], the model writes a program that commands the control interface directly. Execution judgment then moves into a program that cannot

see the scene. A failure is detected only if the program was written to check for it, and a program may exit normally even after the motions it requested have failed. Auriga keeps the judgment with the model, which describes a complete action sequence in each call rather than a single step. Before execution, the model can see the whole trajectory, its cost, and its projection on the camera image (Sec. 4.1), and the harness checks the measured outcome after every movement (Sec. 4.2).

2.3 General-Purpose Harness Evolution

Improvement in embodied agents can target either a procedure for a particular task or the shared capabilities used across many tasks [8, 63].

Within-task trial search versus cross-task tool evolution. Recent work on recursive self-improvement (RSI) groups autonomous improvement into six levels from L0 to L5 [8]:

- **L0–L2 (within-task search and refinement):** Under a single-task objective, the system repeats trials to adjust a trajectory, script, or task note, so the resulting gains stay mainly on the practiced task.
- **L3–L4 (cross-task experience and tool evolution):** The system turns recurring failures across different tasks into reusable tools, interfaces, and verification steps that apply to unseen tasks.
- **L5 (meta-improvement):** The system revises its own perception, control, and verification mechanisms so that later tasks are solved more reliably.

Target-task refinement methods such as HexaShell, EmbodiedEvo, and SEAR [11, 20, 49] work in the L0–L2 regime: they reset the target scene across trials to tune coordinate offsets, scripts, or failure notes for that task, which improves performance on the practiced layout but can overfit when layouts change. Auriga instead focuses on L3–L5 evolution during development. Failures such as a jammed pin insertion or a slipped cloth grasp are used not to add a coordinate offset for that scene, but to diagnose missing capabilities in 3D localization, contact checking, or in-hand state tracking and to upgrade the shared tools accordingly.

Role of the harness. An agent harness connects the model to the robot: it translates the model’s intent into motion and stops the robot when contact or slippage occurs, without making task decisions on the model’s behalf. Existing harnesses often either run generated commands open-loop without checking the physical outcome, or add task-specific rules and thresholds in the backend to pass particular tasks. Because manipulation involves both tight rigid clearances and deformable objects, an offset or grasp rule tuned for one task often interferes with another. Auriga keeps the harness task-agnostic through three rules: (1) *no task-specific logic or privileged state*, so the same code and prompts run across all tasks and randomized layouts; (2) *general measurement and control tools*, including calibrated multi-view triangulation, paused-world trajectory preview, stage tolerances, and in-hand slip checks; and (3) *in-place recovery*, stopping when a motion leaves its tolerance and returning the measured deviation so the model can continue from the current state.

Two complementary timescales and capability trajectory. System development and within-episode adaptation operate at complementary timescales. Across 13 development iterations (#0 to #12, Fig. 3), recurring execution bottlenecks across diverse tasks drove task-agnostic upgrades to the shared harness, lifting the RoboDojo Score from 21.5 (4.0% SR on the 50-case diagnostic set) at the initial base-model loop (#0) to 45.5 (36.2% SR across all 42 tasks) at #4 and 64.0 (57.7% SR across 6,300 episodes) at #12, with consistent gains across all five capability dimensions (+16.8 Score in Generalization, +5.6 in Precision, +29.5 in Long-Horizon, +30.8 in Memory, and +9.8 in Open; inset of Fig. 3). These 13 milestones organize into the three architectural pillars detailed in the next three sections:

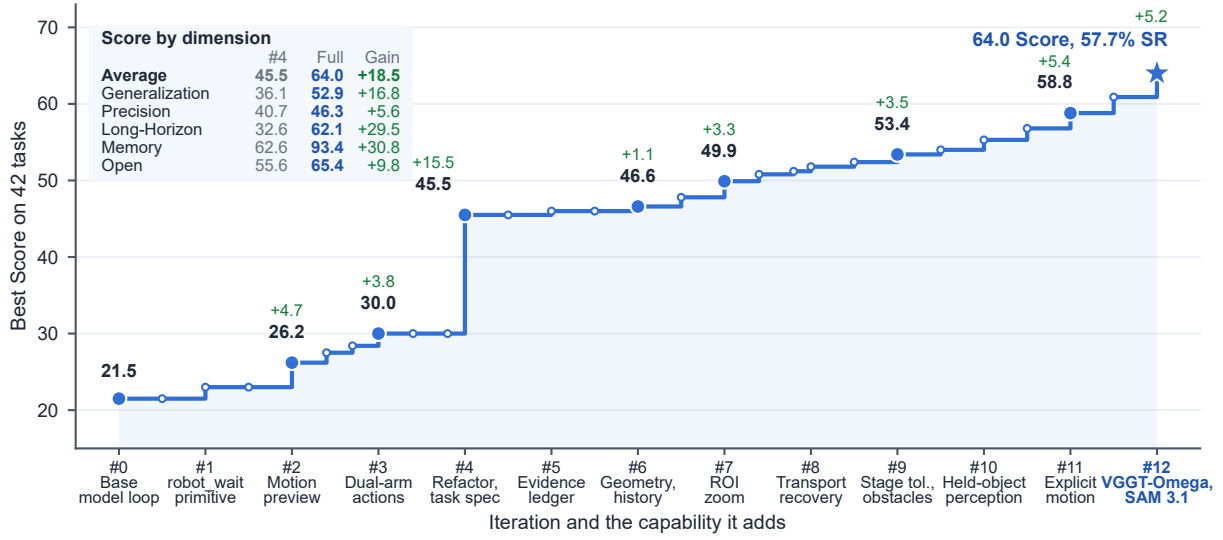


Figure 3 General-purpose harness evolution of Auriga across 13 capability iterations (#0 to #12). The horizontal axis lists the task-agnostic tools and harness mechanisms introduced at each iteration, and the vertical axis tracks the resulting performance (overall Score on RoboDojo). Larger nodes, labeled with their Score and gain, mark major capability milestones: motion preview (#2), unified dual-arm actions (#3), harness refactoring with verbatim task specifications (#4), calibrated 3D geometry and history retrieval (#6), ROI zoom (#7), stage tolerances and obstacle checks (#9), and explicit motion control (#11), culminating in the frozen Auriga harness with VGGT-Omega and SAM 3.1 perception (#12, star marker: 64.0 Score, 57.7% SR across 6,300 episodes). Top-left inset: 42-task Score by capability dimension from iteration #4 to the full frozen harness.

1. **Structured task belief and evidence records (Sec. 3):** refactoring the harness around verbatim task specifications (#4, 45.5 Score), maintaining an explicit evidence ledger (#5), and adding structured history retrieval (#6, 46.6 Score);
2. **Admission gateway with preview and stage verification (Sec. 4):** previewing candidate trajectories while simulation is paused (#2, 26.2 Score), supporting in-place transport recovery (#8), and enforcing stage arrival tolerances and obstacle clearance checks (#9, 53.4 Score);
3. **Calibrated 3D perception and expressive action interfaces (Sec. 5):** unifying dual-arm timelines (#3, 30.0 Score), adding multi-view 3D triangulation (#6), ROI zoom (#7, 49.9 Score), held-object perception (#10), explicit motion control (#11, 58.8 Score), and VGGT-Omega / SAM 3.1 geometric perception (#12, 64.0 Score).

During evaluation, the harness version (#12) is completely frozen across all cases. Within each evaluated episode, the model uses these shared tools to inspect the scene, diagnose discrepancies, and revise its plan or generated programs from execution feedback, continuing from the physical state left by previous actions under the remaining budget without an environment reset or cross-episode memory transfer.

3 System Overview

Large vision-language models can reason about task structure, but reliable manipulation also requires metric scene information, contact-sensitive execution, and feedback after each action. Auriga addresses this gap with a simple closed-loop workflow: the model measures and previews while the simulation is paused, submits a physical action through a single admission gateway, and then uses the returned receipt and observation to decide what to do next (Fig. 4). The model retains authority over planning and recovery; the harness supplies

measurements, enforces the execution boundary, and records what occurred. This section first formalizes the decision process (Sec. 3.1), then describes the three system layers (Sec. 3.2), the explicit task belief maintained by the model (Sec. 3.3), and the three feedback loops that connect planning to execution (Sec. 3.4). Sec. 4 details execution preview and verification through the admission gateway, and Sec. 5 specifies the perception and action interfaces.

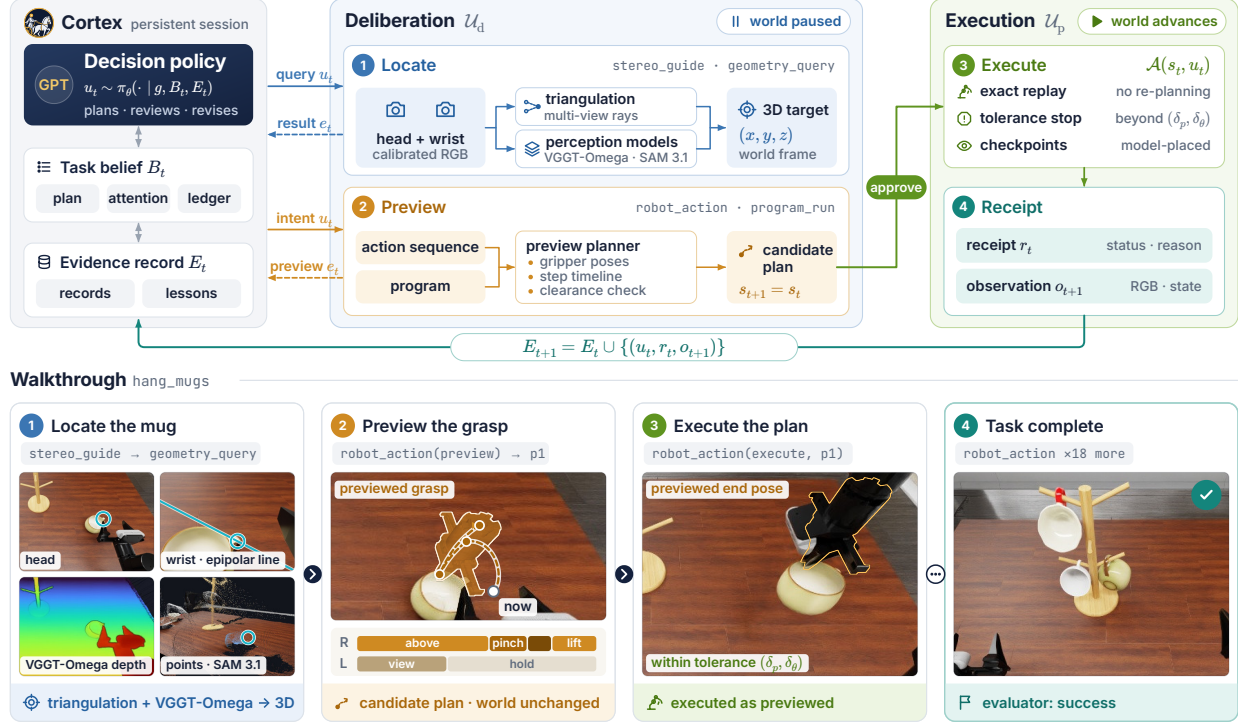


Figure 4 Overview of Auriga. *Top*: The model maintains a task belief and an evidence record, uses observations to localize targets, and previews actions while the world is paused. Every physical motion passes through the admission gateway $\mathcal{A}$, which returns a receipt and a new observation. *Bottom*: Closed-loop execution on hang_mugs: (1) localization, (2) grasp preview, (3) execution, and (4) placement on the rack.

3.1 Manipulation as Closed-Loop Decision Making

For each benchmark episode, the model must construct a procedure from the task description and the evidence gathered during that episode; no task-specific solution is prepared in advance. Consequently, each decision uses the information available so far, and each outcome must be returned in a form that can inform the next decision.

Formulation. A case specifies a task description g and an initial world state s_0 (Sec. 2.1). The world state s_t is hidden from the model, which instead receives observations o_t comprising calibrated multi-view RGB images and the measured robot state. At decision step t , a pretrained model with fixed parameters θ selects a tool call

$$u_t \sim \pi_\theta(\cdot | g, B_t, E_t), \quad (1)$$

where B_t is its *task belief* (Sec. 3.3) and E_t is the episode’s *evidence record*, the interactions accumulated so far. The available calls have two roles. *Deliberative* calls $u_t \in \mathcal{U}_d$ inspect the scene, measure quantities, compute, retrieve records, revise the belief, or preview a motion. They return a result e_t while leaving the world unchanged:

$$s_{t+1} = s_t, \quad E_{t+1} = E_t \cup \{(u_t, e_t)\}. \quad (2)$$

For example, before hanging a mug, the model can localize the mug and rack, inspect the handle orientation, and preview a grasp without advancing simulated time. *Physical* calls $u_t \in \mathcal{U}_p$ request robot motion or the passage of simulated time. They reach the world only through the admission gateway, written as the operator $\mathcal{A}$, which executes an admissible request and returns a receipt r_t and a fresh observation:

$$(s_{t+1}, r_t, o_{t+1}) = \mathcal{A}(s_t, u_t), \quad E_{t+1} = E_t \cup \{(u_t, r_t, o_{t+1})\}. \quad (3)$$

If $\mathcal{A}$ rejects a request, the world remains unchanged and the receipt reports the reason. The model parameters θ , tool set, and task-independent operating guidance remain fixed throughout an episode; adaptation comes from the evolving B_t and E_t . Because the simulation is paused during deliberation, reasoning and deliberative calls consume no simulated time.

Quantitative error feedback. A physical request specifies both an intended result and the required precision. Each stage k has a target $\hat{x}_k$ and a model-chosen tolerance τ_k (Sec. 5.2). During execution, $\mathcal{A}$ measures the reached state x_k and halts when the deviation $\epsilon_k = x_k - \hat{x}_k$ leaves τ_k . The resulting receipt reports the completed stages, measured deviations, active tolerance, and unexecuted stages (Sec. 4.2). Thus, if a grasp approach in `hang_mugs` stops short or the gripper opening differs from the expected value, the model receives the measured discrepancy rather than a generic failure flag. It can then decide whether to inspect the scene, revise the approach, or execute the remaining stages; $\mathcal{A}$ does not choose the recovery.

Information boundary. The harness exposes only the published task text, calibrated RGB images, the robot’s measured state, and the public robot model and camera calibration. From these inputs, it may compute learned depth, per-object 3D estimates, and point clouds, reporting each as an estimate with its reliability (Sec. 5.1). The model and its tools cannot access simulator depth, hidden object poses, contact state, or evaluator internals. The benchmark evaluator, rather than the model’s own assessment, determines task success.

3.2 Separating Decision, Capability, and Admission

Auriga separates task reasoning from measurement and execution so that a general-purpose model can build a procedure while every physical action remains reviewable.

Division of responsibility. The harness is organized into three layers (Fig. 4): a model session that makes every task decision, a capability layer of general tools, and an admission gateway that is the only path to the robot. Each layer has a distinct responsibility:

- *Cortex* is one persistent model session for the entire episode. It receives task-independent operating guidance, the official task text, and the robot’s capabilities and coordinate conventions. It chooses task decompositions, measurements, action targets, and recovery steps without human intervention or a task-specific prompt, program, or skill.
- The *capability layer* provides general tools to measure, act, compute, and recall (Tab. 3). For `hang_mugs`, these tools can estimate mug and rack locations, compare views, preview a grasp, or retrieve an earlier receipt; they do not encode a solution for hanging mugs.
- The *admission gateway* implements $\mathcal{A}$. It is the only path that moves the robot. It receives a reviewed physical request, determines whether the request is admissible, monitors execution, and records the outcome.

One-step dataflow. At each decision step, the model first gathers the evidence needed for the next choice through the capability layer (Sec. 5.1). It then submits either a complete multi-stage action sequence or a program that requests one (Sec. 5.2). Both routes converge on $\mathcal{A}$, so a program-generated motion is previewed and reviewed in the same way as a direct request (Sec. 4.1). After a physical call, the harness stores the reviewed request, receipt, observations before and after execution, and measured displacement in

Table 3 Capabilities available to the model. Only physical calls advance the world, and they do so only through the admission gateway $\mathcal{A}$.

Capability	Role	Representative tools	World	Details
Measure	Return images, robot state, object estimates, and geometric or state quantities with their reliability	robot_observe, geometry_query, stereo_guide	–	Sec. 5.1
Act	Preview and execute multi-stage action sequences for one or both arms, or let simulated time pass	robot_action, robot_wait	$\mathcal{A}$	Secs. 4.1, 4.2 and 5.2
Compute	Run programs for loops, search, and numerical work; their physical requests are reviewed by the model	program_run	$\mathcal{A}$	Sec. 4.1
Recall	Revise the task belief, retrieve earlier interactions, and record or query operation lessons	task_plan, history_*, operation_memory	–	Secs. 3.3 and 5.1

one interaction record ([Sec. 5.1](#)). These records make completed motion, normal program termination, and benchmark success separately observable instead of conflating them.

Transparent execution. The model sets task decomposition, targets, and the need for additional evidence; the harness exposes the thresholds that admit, pause, or stop an action and reports their effects. Every physical request, whether direct or program-generated, follows the same admission path. The harness neither substitutes a task-specific rule nor silently changes a parameter supplied by the model. The records constitute E_t , remain outside the model context, and can be retrieved on demand ([Sec. 5.1](#)). The model may attach short operation lessons to them, but it rechecks each lesson against the current scene before using it.

3.3 Deliberation over an Explicit Task Belief

Long-horizon manipulation accumulates observations, receipts, and intermediate decisions. Without a compact representation of the current task understanding, a model can lose track of an earlier objective or confuse an issued command with a verified outcome. Auriga therefore maintains a revisable task belief B_t alongside the detailed evidence record E_t .

Structure. The belief has three concise components. The *plan* records the model’s current understanding of the goal, its intended approach, and the visible evidence that would indicate completion. The *attention* list records the uncertainties that matter now, the evidence that could resolve each one, and the part of the procedure that depends on it. The *ledger* records a small number of task-critical premises. Each premise includes an expected event, a status (*pending*, *supported*, *contradicted*, or *unknown*), the permitted evidence for that status, and the next observation or action that could change it (the labeled premises in [Fig. 1](#)). The harness admits no physical action until the model has submitted a belief.

For example, in `hang_mugs`, the plan can state “grasp the mug by its handle, lift it, and place it on the rack hook.” An attention item can require a view that resolves the handle orientation before the approach. A ledger premise can state that the mug is retained by the gripper: it is initially *pending*, and becomes *supported* only after permitted evidence, such as the expected gripper opening together with a confirming lift observation, is available.

Belief versus evidence. B_t is a compact, revisable working state rather than an action script or a source of evidence. Concrete poses, arm timing, and gripper commands remain part of individual actions, so the model can change them as new measurements arrive. Likewise, a belief entry must be checked against fresh

observations before it supports a decision: a closed gripper alone does not verify that a mug is retained. When the conversation is compacted to fit the context window, the harness reinserts the latest B_t as the model’s provisional judgment, while the details of earlier interactions remain retrievable from E_t .

3.4 Closing the Loop at Three Time Scales

Errors can arise within a movement, between successive actions, or only after a task-level premise has been tested. Auriga addresses these cases with three nested loops that have distinct responsibilities: the admission gateway stops unsafe or off-target execution, while the model diagnoses the result and chooses the recovery. Fig. 5 shows how the three loops interleave within one episode.

Stage loop. Within a single action, $\mathcal{A}$ contains errors: applying the stage check of Sec. 3.1 during execution, it halts at the first stage whose deviation ϵ_k leaves τ_k , and also pauses at a failed grasp check or at a checkpoint placed by the model (Sec. 4.2). For a `hang_mugs` grasp, it can stop the sequence immediately when the measured gripper opening indicates that the mug handle was not secured. The stage loop keeps an error from propagating to later stages; it does not repair the motion.

Action loop. Between actions, the model diagnoses outcomes: after each physical call, it compares the receipt and fresh observation with its intended result and forms a cause hypothesis before acting again. It may continue the remaining previewed stages, replan from the stopped stage without repeating completed stages, change the contact, route, target, arm, or tolerance, or collect an observation that distinguishes competing explanations. For example, after an unsuccessful mug grasp, the model can inspect whether the handle orientation, approach route, or target localization caused the discrepancy before choosing a new approach. Whenever possible, normal task progress tests the hypothesis; a separate probe is used only when its result would alter the next decision.

Belief loop. Across the episode, the model revises its premises: action outcomes update the ledger and, when necessary, the plan in B_t . If repeated evidence contradicts a premise, the model changes its task-level explanation and selects a different mechanism, such as another contact, route, or target, rather than simply restating the same hypothesis. In `hang_mugs`, evidence that a top-down approach repeatedly fails can update the belief so that the model seeks a handle-oriented grasp or a different approach route. This loop keeps the model’s working understanding aligned with the evidence accumulated during the episode. Sec. 4 next describes the admission path in which the stage loop runs.

4 Admission Gateway: Preview and Verification

To close the planning–execution gap quantified in Sec. 2.2 and Tab. 2, Auriga binds physical execution to a trajectory preview that the model can inspect before committing (Sec. 4.1) and verifies the measured state as the trajectory runs (Sec. 4.2). Auriga follows a single rule, enforced by the admission operator $\mathcal{A}$: the motion that reaches the robot is exactly the motion compiled into its preview, and whenever execution departs from it beyond the declared tolerance, the harness stops and reports the measured state rather than repairing it on the model’s behalf.

4.1 Preview Before Acting

Preview gives the model a first guarantee, namely that every motion, whether requested by the model or by a program it has written, is compiled into a preview before it runs, and that nothing else is executed. The model reviews this preview by default and can skip the review to save a call. Below, we describe how the model sees a motion and how execution is held to it.

Seeing the whole motion. Rather than issuing one movement per call, the model describes a complete action

Input: task description g , initial observation o_0 , task-independent operating guidance, fixed model π_θ .
Initialize: $E_0 \leftarrow \{o_0\}$; the model observes, measures, or previews and submits a belief B_0 before any physical call.

repeat *belief loop*

1. *Decide:* the model selects $u_t \sim \pi_\theta(\cdot \mid g, B_t, E_t)$ (equation (1)).
2. *Deliberate:* if $u_t \in \mathcal{U}_d$, return an observation, measurement, computation, retrieved record, or preview e_t ; the world is unchanged (equation (2)).
3. *Act:* if $u_t \in \mathcal{U}_p$, issued directly or by a program, apply $\mathcal{A}$ (equation (3)). *stage loop*
 For each stage k : execute, measure x_k , and halt if ϵ_k leaves τ_k , if the grasp check fails, or at a model checkpoint. Return r_t and o_{t+1} , and append the record to E_{t+1} .
4. *Assess:* the model compares r_t and o_{t+1} with its intent and states a cause hypothesis for any deviation. *action loop*
 It continues, replans from the stopped stage, changes the contact, route, target, arm, or tolerance, or gathers a distinguishing observation.
5. *Revise:* the model updates the status of affected premises and, if its understanding has changed, the plan in B_{t+1} .

until the evaluator reports termination or the execution budget is exhausted.

Figure 5 The three nested loops of Auriga within one episode. The stage loop runs inside $\mathcal{A}$ and stops execution when a declared condition is violated; the model closes the action and belief loops. Deliberative calls extend the evidence record without advancing the world.

sequence for one arm or for both arms at once, such as approaching an object, closing the gripper, lifting, transporting, and releasing it. The harness plans the entire sequence into a robot trajectory before anything moves and returns a preview of what would happen. Any adjustment the harness makes to how a movement is realized happens at this stage, so the preview already shows the motion that will run. The preview draws the planned gripper paths (Fig. 6) and arm configurations on the camera image and reports how long each part of the sequence would take on each arm. It also estimates the clearance between the robot, the objects it holds, and the obstacles the model has declared, such as whether the two arms come close to each other. Such estimates are given as advice, and the decision whether to proceed stays with the model. If any part of the sequence cannot be planned, nothing is executed, and the preview states which part failed and why.

Committing to it. Once the model accepts a preview, the harness replays the previewed trajectory without replanning it. Because the simulation is paused while the model inspects the preview, the robot is still in the state from which the preview was planned, and looking ahead costs no execution time. The robot therefore performs the previewed motion, so every later report refers to a motion the model can inspect. In free space the robot follows the previewed trajectory closely, which means that a deviation during execution almost always indicates contact, the case that Sec. 4.2 is built around.

4.2 Verify While Acting

Preview fixes what will be executed, but not how the world will respond. Contact, slip, and objects displaced by earlier actions can still make the outcome differ from the plan. Verification gives the second guarantee, namely that every divergence of the robot from its previewed motion beyond the tolerance is caught at the end of the movement in which it occurs and handed back to the model together with the facts needed to act on it.

Catching divergence. After each movement, the harness compares where the gripper actually is with where the preview placed it. The tolerance τ_k of a movement (Sec. 3.1) consists of a position part δ_p and an orientation part δ_θ . The model can set both for each movement, by default $\delta_p=10$ mm and $\delta_\theta=0.15$ rad. If the

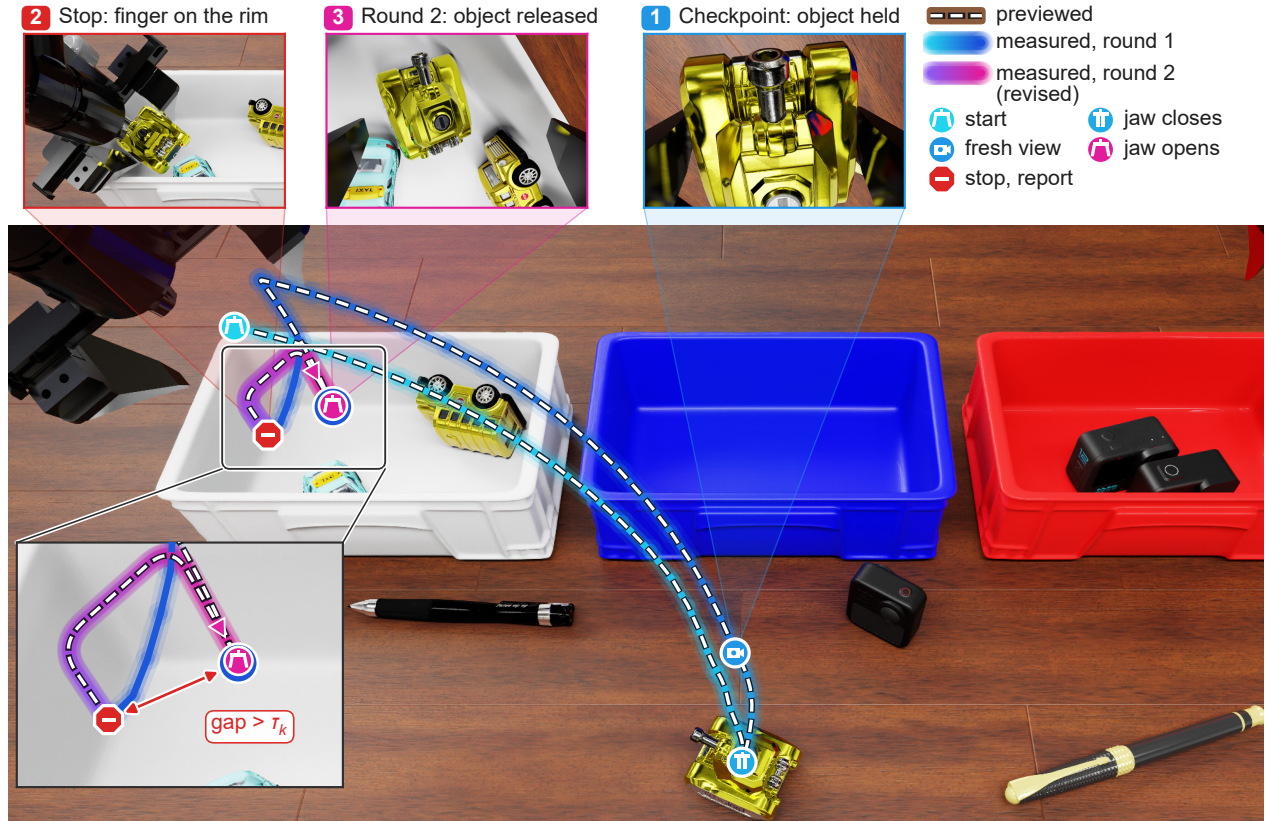


Figure 6 Preview and verification in two rounds, drawn on the head-camera image. In the first round, the previewed gripper path moves to the object, closes on it, and lifts it to a checkpoint placed by the model, where the robot pauses and the model receives a fresh wrist view (inset 1). The measured path follows the preview until a finger lands on the basket rim while the object is lowered. Once the gap exceeds the tolerance, the harness stops the arm and reports with a fresh view (inset 2), so the gripper does not open. In the second round, the model previews a revised motion from the stop that lifts clear of the rim and lowers into the basket, where the gripper opens (inset 3). The rim contact and the second round are illustrative.

difference exceeds the tolerance set for that movement, the robot stops and holds its pose instead of continuing with the rest of the sequence. The receipt r_t that follows states which movement failed, by how much it missed, and which parts did not run. It also adds a fresh camera view from the stopping point and images of the measured and the previewed route. Movements that already completed are kept rather than undone. In Fig. 6, a finger lands on a basket rim while the object is lowered, and the measured path leaves the previewed one beyond its tolerance. The harness stops the arm and reports only this miss, and the model identifies the rim as the cause from the fresh view. It then previews a revised motion that clears the rim, which completes the placement in a second round. This check targets the main source of divergence, contact with an object or fixture, which the planner does not model, so the harness detects its consequence instead of predicting it.

Looking where the model chooses. Some divergences leave the robot on its planned path and show only in the scene, such as an object that slips or shifts during a grasp. The model can therefore insert checkpoints at moments it considers critical, either unconditionally or only when the measured gripper opening after a grasp falls outside the range it expects, for example when the fingers close on nothing. At a checkpoint the robot stops and the model receives a fresh view, after which it either continues the remaining sequence as previewed or replaces it. In Fig. 6, the model places a checkpoint after the lift, and the sequence continues as previewed once the fresh wrist view shows the object still held. Checkpoints are placed by the model rather than at fixed points, since pausing after every gripper action by default fragmented long tasks. Acting on what a stop or

a checkpoint reveals, however, often requires facts that the model cannot read reliably from RGB, such as whether an object is held and where it sits relative to the gripper, which the following section provides.

5 Capability Layer: Perception and Action Interfaces

Faithful execution helps only if the motion the model chooses is the right one, which depends on what the model can perceive and on how it can express what it wants. A general-purpose vision-language model reads a scene and plans a task well, but it is weak at the quantitative side of manipulation. This section describes how the harness compensates on both sides, by turning physical state into facts the model can read (Sec. 5.1) and by letting the model state its intent in terms of the task (Sec. 5.2), as summarized in Fig. 7. Two principles guide both sides. The harness *informs but does not decide*, reporting measurements and estimates together with how reliable they are and leaving the choice of what to do with them to the model. It also *attaches facts at the point of decision*, so that information that should change the next command arrives with the preview or the report instead of waiting to be requested.

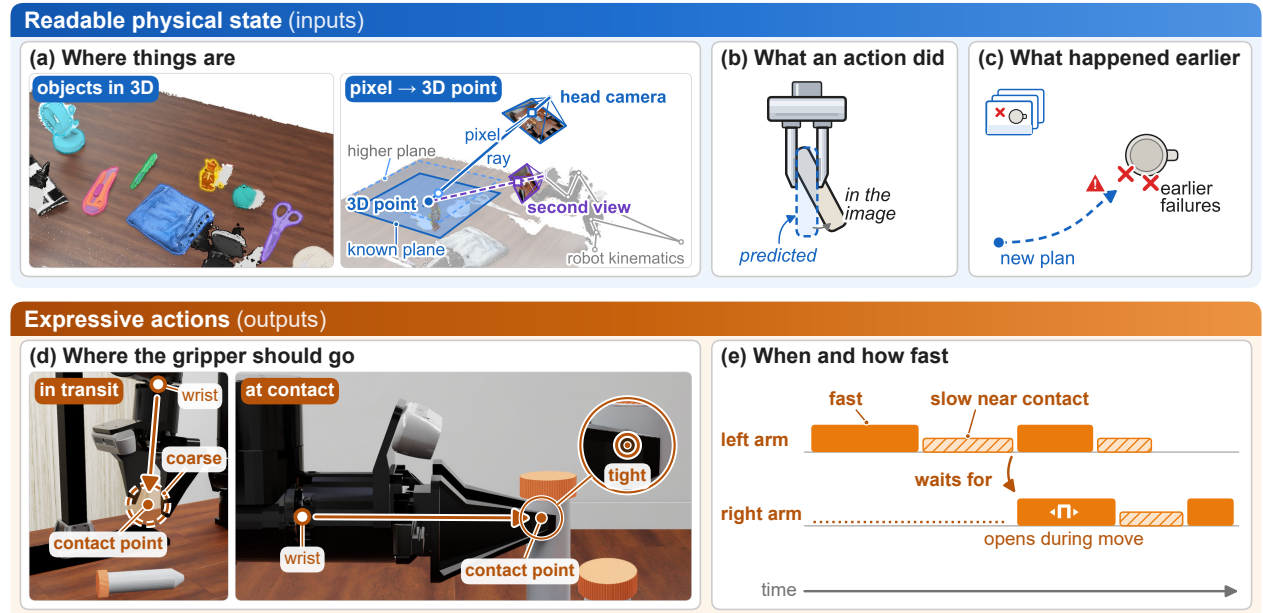


Figure 7 Perception and action interfaces in Auriga. (a) Objects separated and placed in 3D from the head-camera image, and the position of a selected point from its pixel ray and a plane of known height or a second view. (b) The object estimates keep a held object at its pose in the hand at the grasp, and a measurement against this reference shows that it has turned in the hand. (c) A new plan that returns to where earlier motions failed carries a warning, and earlier frames remain retrievable. (d) A goal refers to the contact point on the fingers rather than the wrist, with a coarse tolerance in transit and a tight one at contact. (e) Both arms share one timeline, with declared waits, a speed per movement that slows near contact, and a gripper that opens during a movement.

5.1 Readable Physical State

On the input side, every quantity the model needs for its next decision reaches it as a measured, computed, or estimated fact together with how reliable it is, rather than as something the model has to infer from pixels. We describe how the model locates things, learns what an action did, and recalls what happened earlier.

3D localization. The model understands the layout of a scene well from RGB, but an image does not show metric 3D positions directly, so any estimate made from appearance alone remains uncertain. A deployed robot, however, comes with physical quantities that are known exactly, namely the intrinsic and extrinsic

calibration of every camera, the height of the work surface, and the kinematic model of the robot itself. With the calibration, every pixel (u, v) of a camera with intrinsics K , orientation R , and center $\mathbf{c}$ lies on a known ray with direction $\mathbf{d} = R K^{-1}(u, v, 1)^\top$, so locating it in 3D reduces to finding its distance along this ray. The harness provides two tools that supply this distance at different levels of precision (Fig. 7a). The first estimates every visible object in 3D. VGGT-Omega [46] predicts a depth for every pixel of the head-camera image, which places the pixel on its ray, and the depths are scaled so that the table lies at its known height. SAM 3.1 [2] separates the objects in these points, and each object is reported with its 3D position, orientation, and size and with how reliable the estimate is, in every full observation and every action report. Where such an estimate is not precise enough, as for placing the fingers on a narrow edge, the second tool computes the position of a point that the model selects in an image. The model fixes its distance either with a plane $\mathbf{n}^\top \mathbf{p} = h$ of known height,

$$\mathbf{p} = \mathbf{c} + \frac{h - \mathbf{n}^\top \mathbf{c}}{\mathbf{n}^\top \mathbf{d}} \mathbf{d}, \quad \frac{\partial \mathbf{p}}{\partial h} = \frac{\mathbf{d}}{\mathbf{n}^\top \mathbf{d}}, \quad (4)$$

or by marking the same point in a second view, such as the wrist camera, from which the harness triangulates it. The two tools meet in the plane height. For a point on the table, the height is known exactly. For a point on an object, the model either takes the height from the learned depth of that object, in which case the harness reports how far the point would move if this height were off, or marks the point in a second view, which needs no height. Any 3D point, including the fingers placed by the kinematic model, can be projected back into the image to check it.

Action outcomes and in-hand state. After an action, the model needs its physical outcome. For most of the scene, the next image shows it, and each report also lists the objects whose estimated position changed during the action. A grasp is harder, since the fingers often hide the object, so the image may not show the state of the object and how it interacts with the gripper. The harness therefore complements the image with measurements of both. For the gripper, every report includes its measured opening. For the object, the model can measure on request where a visible part of it, such as its lower end, lies relative to the gripper, the axis of the object in the 3D points around the fingers, and how far a part marked in a wrist image before an action has moved after it. The object estimates also keep the pose that the object had in the hand at the grasp, as a reference for these measurements. Together, these tell the model whether the object is held, how it sits in the hand, and whether it has turned or slipped since the grasp.

Episode history and warnings. In long episodes the context is compacted and early images are dropped, while recovery often depends on what earlier attempts actually achieved. The model can therefore retrieve from the evidence record E_t (Sec. 3.1) any earlier camera frame, including a magnified region of it, a list of past actions with the displacement that each one measured, and the planned and measured trajectories of any action. Full history is retrieved on demand rather than pushed into every turn, since forced reviews of history produced idle calls. The exception is history that bears directly on the next command. A report of a stopped or failed action carries a short summary of recent outcomes, and a preview that returns to a place where a motion has failed carries a warning, so that the model does not unknowingly repeat a failed attempt.

5.2 Expressive Actions

On the output side, the model states what it wants in terms of the task, such as where the fingers should touch and how the two arms should cooperate, and the harness turns these statements into motion. We describe how the model specifies the goal of each movement and how it sets the timing and speed of a sequence.

Finger-referenced goals. A robot interface typically takes the pose of the wrist as its goal, whereas what matters for a task is where the fingers touch the object and how precisely they get there. In Auriga, each movement states its goal in these terms. The model gives where the gripper should end up and how it should be turned in a frame fixed to the workspace, together with the part of the fingers that should make contact,

such as the fingertip or the pad center, so that the goal refers to the point that touches the object rather than to the wrist. It also states how precisely the gripper must arrive, so that a tight insertion and a coarse transit are checked differently.

Shared timeline and speed. Goals alone do not fix how a sequence unfolds in time. When both arms take part, or when a movement passes from free space to contact, the order and the speed of its parts matter as much as its goals. In Auriga, the stages of both arms belong to the same sequence and advance on a shared timeline. Stages of the two arms run at the same time unless the model declares that one must wait for another. The model also sets the speed of each movement, typically slower near contact, and can let the gripper open or close during a movement, for example opening it on the way to an object. Speed and gripper timing are fixed in the preview like the path itself, so they are part of the reviewed motion.

6 Experiments

We evaluate Auriga under the zero-shot protocol of [Sec. 2.1](#) on two simulation benchmarks: dual-arm tabletop manipulation on **RoboDojo** [4] ([Sec. 6.1.1](#)) and kitchen mobile manipulation on **RoboCasa365** [32] ([Sec. 6.1.2](#)). After describing the experimental setup ([Sec. 6.1](#)), we report main results on RoboDojo ([Sec. 6.2](#)) and RoboCasa365 ([Sec. 6.3](#)), mechanism ablations ([Sec. 6.4](#)), execution efficiency and tool usage ([Sec. 6.5](#)), and qualitative behavior and failure modes ([Sec. 6.6](#)).

6.1 Experimental Setup

6.1.1 RoboDojo

RoboDojo [4] simulates a dual-arm ARX X5 robot with one head camera and two wrist cameras in Isaac Sim.

Capability dimensions. RoboDojo groups its 42 tasks into five capability dimensions: Generalization (12 tasks, each with standard and random scenes), Precision (8), Long-Horizon (8), Memory (6), and Open (8); [Tab. 13](#) in [App. A](#) lists what each dimension covers and its tasks. As on the official leaderboard, results are first averaged across the tasks of each dimension and then across the five dimensions, with equal weight at both levels. For Generalization, the value of each task and of the dimension is the equal-weight mean of its standard-scene average and its random-scene average.

Evaluation set. Following [Sec. 2.1](#), a case specifies a task, a scene variant (standard or random), an evaluation seed, and a layout, and an episode is one execution of a case by the agent. The evaluation set follows the protocol of the official leaderboard, with three evaluation seeds and, in each seed, 50 cases for each of the 42 tasks, that is, 2,100 cases per seed and 6,300 in all. In each seed, every Generalization task contributes 25 cases with standard scenes and 25 with random scenes, and every other task contributes 50 with standard scenes. Each case counts with its first official score, including scores of zero, and any later episode of the same case is ignored. When an infrastructure fault ends an episode before the evaluator has scored it, the case is re-run under the same configuration.

Metrics. *SR* is the share of cases that the benchmark evaluator judges fully successful, over a fixed denominator: a case without an official score stays in the denominator and does not count as a success. *Score* is the official graded process score of the evaluator, from 0 to 100, averaged over the cases with an official score. Both metrics are first computed for each task in each seed, and a task’s value is the mean of its three per-seed values; the dimension values and the five-dimension average then follow from the task values as in [Sec. 6.1.1](#). *Steps* are the native steps that the simulator executes in an episode, also reported as a percentage of the official native-step limit of the task, which ranges from 200 to 1,900 steps; a single action request may use at most 250 steps. *Calls* are the model calls (the model responses in an episode) and the direct tool calls issued by the model; tool calls issued by model-written programs are counted separately. *Cost* comprises the input tokens

of the model session (total, share served from the prompt cache in %, and uncached input tokens), the output tokens, and the wall-clock time of the episode; Steps, Calls, and Cost are reported as means per episode.

Agent configuration. The RoboDojo results (Secs. 6.2 and 6.5) use Auriga as described in Secs. 3 to 5, with the model `gpt-6-astra` at reasoning effort high. An episode may use at most 170 model responses and the official native-step limit of its task, and the simulation is paused while the model deliberates (Sec. 3.1). The model observes RGB images from the three cameras, proprioception, and the public camera calibration, together with the object estimates that the harness computes from the head-camera image (Sec. 5.1); simulator depth and hidden object state are withheld. Its task text is the native instruction of the task, the official task page with its description and scoring rules, and the task-independent operating guidance of the harness (Secs. 3.1 and 3.2). The ablations in Sec. 6.4 remove or restrict individual mechanisms of this configuration while keeping the rest fixed.

6.1.2 RoboCasa365

RoboCasa365 [32] evaluates household mobile manipulation with a single Franka Panda arm mounted on an Omron mobile base across kitchen environments, spanning 365 tasks (65 atomic and 300 composite).

Tasks and task groups. Evaluation focuses on the benchmark’s 50 Target50 tasks, organized into three groups: 18 *Atomic-Seen* tasks involving single manipulation skills (e.g., opening a drawer or operating a microwave); 16 *Composite-Seen* tasks that chain two to fifteen subtasks whose templates appear in the benchmark’s pretraining demonstrations; and 16 *Composite-Unseen* tasks of comparable horizon whose templates are absent from the pretraining demonstrations.

Evaluation protocol and metrics. Whereas the standard policy leaderboard [40] evaluates demonstration-trained policies in the 2,500 pretraining kitchens, agent evaluations are conducted in ten unseen target kitchens with novel layouts, fixtures, and objects. We follow the protocol of the published agent baselines [43, 61]: every task is evaluated at seeds 1–5, giving 250 episodes (90 Atomic-Seen, 80 Composite-Seen, and 80 Composite-Unseen). We report the success rate (SR, %) of each group, of all 160 composite episodes, and *Overall* as the mean over the 50 tasks, which equals the SR over all 250 episodes. The benchmark environment provides camera images, robot proprioception, world-coordinate maps derived from rendered depth, the source code of the task’s success check, and live values of the predicates tested by that check.

Baselines. Both published agent baselines plan with an LLM agent and hand contact-rich phases, such as grasping an object or turning a knob, to the RLDX-1 vision-language-action (VLA) policy [17]. RPent [61] also retrieves task memory, built by exploring each task on seed 0 before evaluation. PyRUA-Lean [43] uses no cross-episode memory. We also list RLDX-1 on its own and, as in the comparison of RPent, three learned policies evaluated in the pretraining kitchens.

Agent configuration. Auriga runs `gpt-6-astra` at high reasoning effort with at most 100 model requests per episode. Unlike both agent baselines, it uses no VLA policy and no cross-episode memory. Alongside the benchmark-provided observations and task predicates, Auriga segments camera images via SAM 3.1 [2] from model-chosen text prompts and projects each mask onto the world coordinate map, without reading ground-truth object poses. All physical motions are executed through multi-stage action programs under the preview-and-verification gateway of Sec. 4, coordinating stages for the arm, gripper, torso, and mobile base.

6.2 Main Results on RoboDojo

Tab. 4 places Auriga on the official leaderboard [41], reporting Score and SR on average and in each dimension. The table has two blocks: the eight leading entries that use a learned policy, either a world action model (WAM) or a vision-language-action (VLA) policy, alone or with an agent; and four agent methods, which act through tools or code, together with Auriga. Averaged across the five dimensions, Auriga reaches a Score of

64.00 and an SR of 57.74%. Across both blocks of Tab. 4, Auriga achieves the highest overall averages, ahead of GO-3 Nano Preview, which combines an agent with a world action model, at 61.90 and 56.42%. Within the agent block, Auriga leads all twelve columns, including both averages (ahead of PyRUA-Lean at 55.26 and 50.35%). According to their reports [11, 20, 43, 60], PyRUA-Lean, HexaShell-Preview, EmbodiedEvo, and RoboProbe all use `gpt-6-astra`, the same base model as Auriga (marked [†] in Tab. 4), with reasoning effort high for PyRUA-Lean, xhigh for EmbodiedEvo, and medium for RoboProbe; Auriga uses high. Against RoboProbe [60], the reference harness released with the evaluation of GPT-6 Astra on RoboDojo [4], Auriga raises the average Score from 28.97 to 64.00 and the average SR from 22.48% to 57.74% on the same model (Tab. 4).

Table 4 Results on the RoboDojo simulation benchmark, with external entries from the official leaderboard [41] (Sec. 6.1.1), reporting Score (0–100) and SR (%) in two blocks ordered by ascending average Score: the eight leading entries that use a world action model (WAM) or a vision-language-action (VLA) policy, alone or with an agent, and four agent methods on the leaderboard (Agent, acting through tools or code) together with Auriga. In each column, the highest value is in bold and the second highest is underlined. [†]Uses `gpt-6-astra`.

Method	Type	Average		Generalization		Precision		Long-Horizon		Memory		Open	
		Score	SR	Score	SR	Score	SR	Score	SR	Score	SR	Score	SR
<i>Learned policies: WAM or VLA, alone or with an agent</i>													
Rex-M1 Preview	VLA	33.79	27.44	34.05	26.05	32.82	23.83	46.89	33.00	48.53	48.22	6.67	6.08
Simate-beta	VLA	33.95	27.96	35.09	27.95	34.35	26.92	57.84	43.42	33.33	33.00	9.12	8.50
Awomo-0.5	WAM	35.34	29.64	30.31	23.84	39.65	33.50	40.36	26.75	42.27	41.22	24.09	22.92
PhysicalRSI	Agent + VLA	36.27	31.38	21.30	15.62	38.05	32.50	46.37	37.33	46.74	46.56	28.88	24.92
VPP2	WAM	39.26	32.26	36.38	29.50	43.03	35.17	56.28	37.00	56.01	55.56	4.61	4.08
WALL-OSS-1.0-lite	VLA	42.59	34.97	46.66	37.94	38.16	29.08	49.94	32.25	57.42	56.00	20.76	19.58
WorldScape-R ² SI	Agent + WAM	46.58	39.35	42.95	35.83	23.51	15.50	51.48	34.25	65.78	63.67	49.16	47.50
GO-3 Nano Preview	Agent + WAM	<u>61.90</u>	<u>56.42</u>	36.45	33.66	78.65	76.00	68.21	53.50	64.50	63.67	<u>61.71</u>	55.25
<i>Agent methods, acting through tools or code</i>													
RoboProbe [†]	Agent	28.97	22.48	33.36	30.50	12.65	4.00	21.45	8.25	43.04	38.67	34.36	31.00
EmbodiedEvo [†]	Agent	45.08	38.95	35.22	30.83	25.15	16.25	33.83	21.75	79.40	76.67	51.81	49.25
HexaShell-Preview [†]	Agent	53.18	46.85	39.22	34.50	41.25	32.00	42.91	31.50	<u>84.53</u>	<u>81.00</u>	57.97	55.25
PyRUA-Lean [†]	Agent	55.26	50.35	<u>48.27</u>	<u>43.17</u>	44.77	38.25	54.30	43.75	68.40	68.33	60.56	<u>58.25</u>
Auriga [†]	Agent	64.00	57.74	52.90	48.89	<u>46.30</u>	<u>38.50</u>	<u>62.10</u>	<u>45.24</u>	93.40	92.99	65.40	63.08

By dimension. Across both blocks of Tab. 4, Auriga has the highest Score and SR in Generalization, Memory, and Open, ahead of the next entry by 5.72, 11.99, and 4.83 percentage points in SR. Precision and Long-Horizon remain the most challenging dimensions for Auriga, at 38.50% and 45.24% SR, against 76.00% and 53.50% for GO-3 Nano Preview, which leads both, though Auriga ranks second overall in both Score and SR across Precision (46.30 and 38.50%) and Long-Horizon (62.10 and 45.24%).

Profile against other methods. Fig. 8 compares the five-dimension profile of Auriga with seven other methods of Tab. 4: the four highest-scoring entries of the WAM / VLA block (GO-3 Nano Preview, WorldScape-R²SI, WALL-OSS-1.0-lite, and VPP2) and the three highest-scoring other entries of the agent block (PyRUA-Lean, HexaShell-Preview, and EmbodiedEvo); Fig. 20 in App. D plots the same values as bars, together with RoboProbe. GO-3 Nano Preview has the opposite profile to Auriga: its Score exceeds that of Auriga by 32.35 in Precision and 6.11 in Long-Horizon, while falling below Auriga in the other three dimensions, most in Memory (64.50 vs. 93.40) and Generalization (36.45 vs. 52.90). Against PyRUA-Lean, Auriga achieves a higher Score in all five dimensions, with gains of 1.53 in Precision, 4.63 in Generalization, 4.84 in Open, 7.80 in Long-Horizon, and 25.00 in Memory.

Per task. Tab. 18 and Fig. 21 in App. E compare Auriga task by task with PyRUA-Lean, HexaShell-Preview, EmbodiedEvo, and RoboProbe. Against each of the four, Auriga is higher in Score on more of the 42 tasks

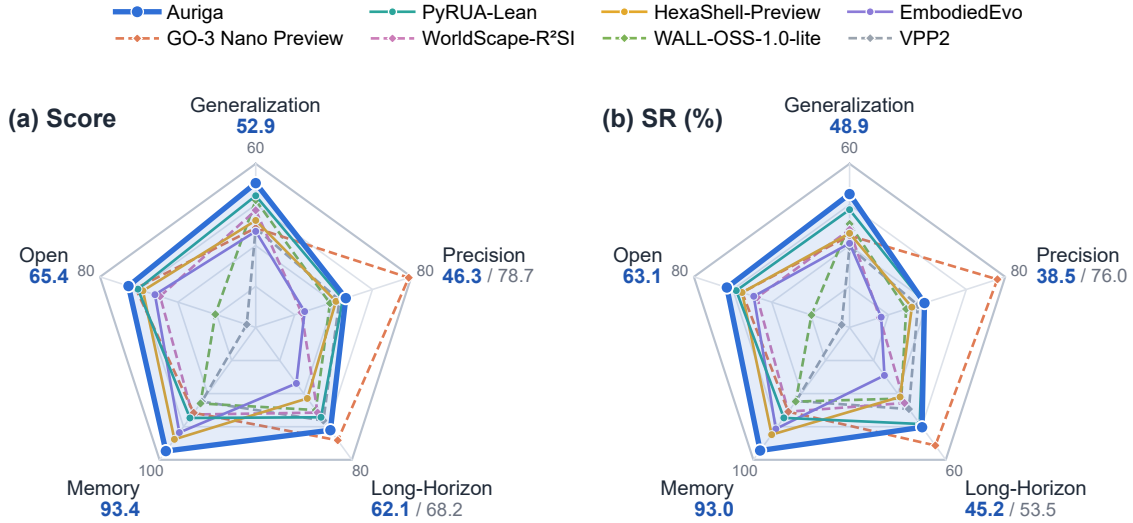


Figure 8 Five-dimension profile of Auriga and seven other methods of Tab. 4, in Score (a) and SR in % (b). Each axis runs from zero at the center to the value at its outer vertex, with four evenly spaced rings. Below each dimension name is the value of Auriga, followed after a slash by the highest value of the other methods where it is higher. Auriga is the thick blue line with the shaded area; the other agent methods are drawn with solid lines and the WAM / VLA methods with dashed lines.

than it is lower, from 24 vs. 17 tasks against PyRUA-Lean to 41 vs. 1 against RoboProbe. In SR, it is likewise higher on more tasks against all four methods: PyRUA-Lean (22 vs. 16), HexaShell-Preview (28 vs. 9), EmbodiedEvo (32 vs. 6), and RoboProbe (38 vs. 1).

Generalization under randomization. Fig. 9 compares the entries of Tab. 4 on the 12 Generalization tasks across the standard and random scenes, reporting the SR change $\Delta\text{SR} = \text{SR}_{\text{rand}} - \text{SR}_{\text{std}}$ in panel (a) and the composite generalization metric $G = \frac{1}{2}(S_{\text{std}} + S_{\text{rand}}) - |S_{\text{rand}} - S_{\text{std}}|$ in panel (b), which rewards a high mean Score and penalizes the gap between the two scene types (Tab. 17 in App. D lists the SR and Score on both scene types). Auriga achieves the highest G in Fig. 9 at 46.27, ahead of PyRUA-Lean at 43.87 and WorldScape-R²SI at 35.61. Across both scene types, Auriga achieves the highest SR and Score of all 13 entries: 52.28% SR and 56.22 Score on the standard scenes, and 45.61% SR and 49.58 Score on the random scenes, with a change of -6.7 percentage points in SR (-13% relative) and -6.6 points in Score (Tab. 17). By contrast, the learned WAM and VLA entries suffer steep drops under scene randomization, losing up to 33.9 percentage points in SR (31.5 points in Score) for VPP2 and 18.7 percentage points (-43% relative) for GO-3 Nano Preview (Fig. 9 and Tab. 17).

6.3 Main Results on RoboCasa365

RoboCasa365 shifts the evaluation to a mobile manipulator operating doors, drawers, knobs, and appliances across unseen kitchens over chains of up to fifteen subtasks (Sec. 6.1.2). Tab. 5 and Fig. 10 compare Auriga with the published agent baselines and with RLDX-1 on the 250 Target50 episodes (Sec. 6.1.2). All three agent methods use gpt-6-astra at high reasoning effort, and the baseline results are those reported by RPent [61] and PyRUA-Lean [43]. As in the comparison of RPent, Tab. 5 also lists WorldDreamer, $\pi_{0.5}$, and π_0 , which are evaluated in the pretraining kitchens; for these policies, Composite All is the mean of the two composite groups. These policies, together with the leaderboard entry Xiaomi-Robotics-1, are also listed in Tab. 15 in App. C.

Comparison with agent baselines. Auriga succeeds in 73.2% of the 250 episodes, against 59.2% for RPent

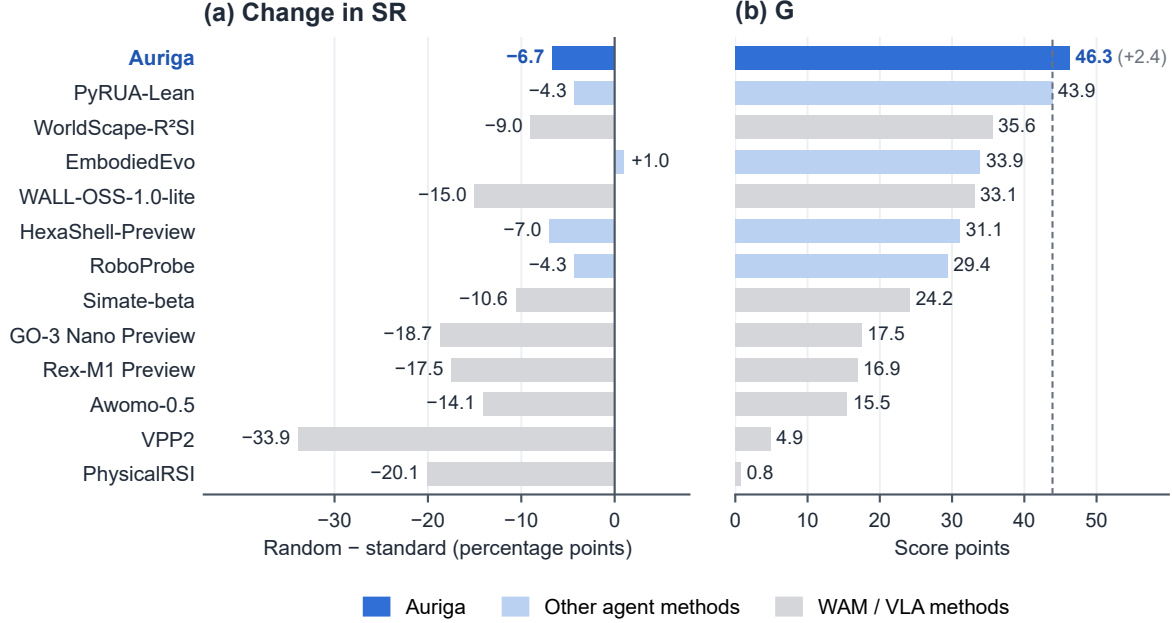


Figure 9 Generalization of the entries of Tab. 4 on the 12 Generalization tasks under scene randomization, ordered by G . (a) Change in SR (percentage points, $\Delta\text{SR} = \text{SR}_{\text{rand}} - \text{SR}_{\text{std}}$) from standard to random scenes. (b) $G = \frac{1}{2}(S_{\text{std}} + S_{\text{rand}}) - |S_{\text{rand}} - S_{\text{std}}|$, the mean of the standard- and random-scene Scores minus their absolute difference.

Table 5 Success rates (%) on the 50 Target50 tasks of RoboCasa365 [32], at seeds 1–5 per task (250 episodes), ordered by ascending Overall SR within each evaluation setting. Overall is the mean over the 50 tasks, and Composite All covers the 160 composite episodes. Agent + VLA methods hand contact-rich phases to the RLDX-1 policy. WorldDreamer, $\pi_{0.5}$, and π_0 are evaluated in the pretraining kitchens, all other methods in the target kitchens. –: not reported. Bold: highest; underlined: second highest.

Method	Type	Overall	Atomic-Seen	Composite		
				Seen	Unseen	All
<i>Pretraining kitchens: learned WAM or VLA policies</i>						
π_0	VLA	14.8	34.6	6.1	1.1	3.6
$\pi_{0.5}$	VLA	16.9	39.6	7.1	1.2	4.2
WorldDreamer	WAM	35.3	66.3	26.7	9.0	17.9
<i>Target kitchens: VLA policy and agent methods</i>						
RLDX-1	VLA	30.0	60.0	21.3	5.0	13.1
PyRUA-Lean	Agent + VLA	56.4	86.7	–	–	39.4
RPent	Agent + VLA	<u>59.2</u>	<u>87.8</u>	<u>43.8</u>	<u>42.5</u>	<u>43.1</u>
Auriga	Agent	73.2	92.2	67.5	57.5	62.5

and 56.4% for PyRUA-Lean with the same model and reasoning effort, and 30.0% for RLDX-1 alone. It has the highest success rate in every group of Tab. 5. The margin is largest on the composite tasks, which chain several subtasks: Auriga completes 62.5% of these episodes, 19.4 percentage points more than RPent and 23.1 more than PyRUA-Lean. Both baselines rely on the RLDX-1 policy for contact-rich phases, and RPent also on task memory gathered before evaluation; Auriga uses neither.

Where Auriga gains and where it fails. The composite gap reflects how contact-rich phases are executed. The baselines hand them to RLDX-1, which on its own completes 21.3% of the Composite-Seen and 5.0% of the Composite-Unseen tasks, while Auriga builds every phase, including grasps, placements, and fixture operations, from action programs whose stages are checked as they execute (Sec. 4). Of the 67 failed episodes

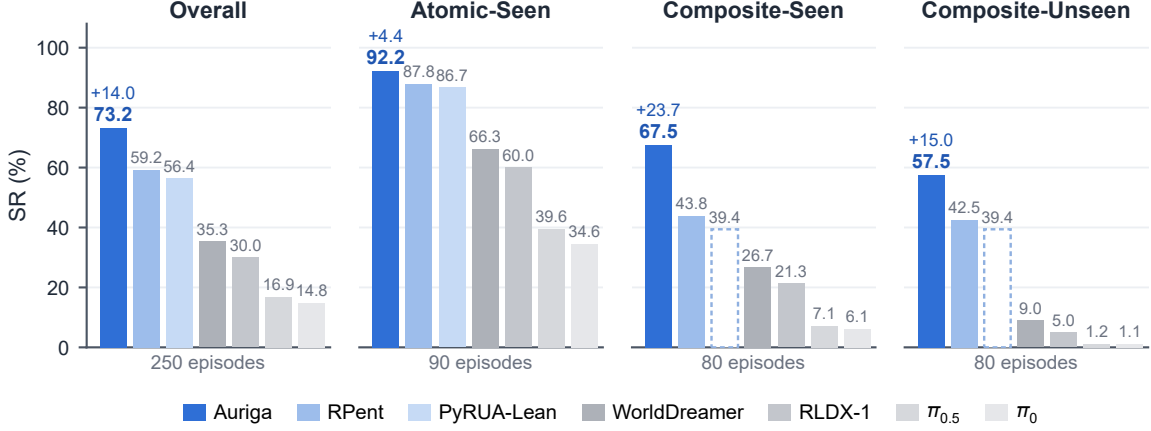


Figure 10 Success rates (%) of the methods of Tab. 5 on RoboCasa365 Target50, in descending order of Overall SR. Grey bars are learned policies, and light blue bars are agent methods that hand contact-rich phases to the RLDX-1 policy. PyRUA-Lean reports composite tasks only as a whole, so its two composite bars are outlined and both show that rate. Above the bar of Auriga is its difference from the highest other method of the group.

of Auriga, 23 ended when the 100 model requests were used and 44 after the model stopped acting for several turns (Fig. 19 in App. C). Four composite tasks fail in all five episodes, PackIdenticalLunches and StoreLeftoversInBowl among the seen and MakeIceLemonade and SeparateFreezerRack among the unseen, while every Atomic-Seen task succeeds in at least three of its five episodes.

6.4 Ablation Studies

Whereas Fig. 3 (Sec. 2.3) tracked the cumulative gains across development iterations, Tabs. 6 to 10 isolate the contribution of each mechanism of Secs. 3 to 5 on the frozen configuration of Sec. 6.1.1. Each table groups related mechanisms and evaluates its variants on ten tasks (Tab. 14 in App. B): Tabs. 6 to 9 on five standard layouts of each task (50 cases), and Tab. 10 on 30 cases. Each variant removes or restricts the mechanisms marked in its row while keeping the rest of the configuration fixed, and the *Full* row reports the main configuration on the same cases. Alongside the number of successes and the mean Score, each table reports a mechanism-specific diagnostic per episode for the ablated variants (and for all rows in Tab. 10), taken from the execution receipts r_t , the evidence record E_t , or the model calls (Secs. 3.1 and 3.2).

Spatial measurement. Tab. 6 evaluates the mechanisms that provide metric 3D positions (Sec. 5.1): the camera calibration and 3D object estimates, together with `geometry_query` for ray-plane intersection (equation (4)), two-view triangulation, and 3D-to-image projection. In *No geometry tools*, the calibration, the 3D object estimates, and `geometry_query` are all removed, so the model locates targets from the RGB images and the robot state alone (Sec. 5.1). In *No triangulation*, the calibration, the 3D object estimates, two-view triangulation, and 3D-to-image projection are removed while ray-plane intersection (equation (4)) remains available at one plane height per pixel. Compared with *Full* (46/50 successes, mean Score 94.9), *No geometry tools* reaches 30/50 successes and a Score of 67.1, with 65.3% of first grasps held. With ray-plane intersection available, *No triangulation* reaches 40/50 successes and a Score of 84.0, with 85.7% of first grasps held; *Full* adds two-view triangulation, 3D-to-image projection, the calibration, and the 3D object estimates (Sec. 5.1).

Collision handling. Tab. 7 evaluates two mechanisms that keep execution clear of unintended contact (Secs. 4.1 and 4.2). In *No clearance review*, the preview no longer estimates the clearance between the arms, held objects, and declared obstacles before motion (Sec. 4.1). In *No divergence stop*, the stage loop in $\mathcal{A}$

Table 6 Spatial measurement (Sec. 5.1 and equation (4)): successes over the 50 cases, mean Score, and the share of first grasps that hold (%). The highest successes and Score are in bold. ✓: available; ×: removed.

Variant	Plane intersection	Triangulation	Calibration and object estimates	Successes	Score	First grasps held (%)
No geometry tools	×	×	×	30/50	67.1	65.3
No triangulation	✓	×	×	40/50	84.0	85.7
Full	✓	✓	✓	46/50	94.9	–

no longer halts the sequence when a movement exceeds its arrival tolerance $\tau_k = (\delta_p, \delta_\theta)$, so later stages continue from the displaced pose (Secs. 3.4 and 4.2). Compared with *Full* (36/50 successes, Score 79.8), *No clearance review* reaches 31/50 and 69.1, with 1.20 tolerance stops per episode; without the clearance review, the stage check during execution is the remaining guard against contact (Sec. 4.2). Compared with *Full*, *No divergence stop* reaches 34/50 and 75.1, with 0.00 stops and 0.92 out-of-tolerance movements per episode as sequences run on, whereas stage verification in *Full* halts at the first exceeded tolerance and returns a receipt r_t and fresh view (Secs. 3.4 and 4.2).

Table 7 Collision handling (Secs. 4.1 and 4.2): successes over the 50 cases, mean Score, and the stops per episode at a movement that exceeded its arrival tolerance $(\delta_p, \delta_\theta)$. The highest successes and Score are in bold. ✓: available; ×: removed.

Variant	Clearance review	Divergence stop	Successes	Score	Stops per episode
No clearance review	×	✓	31/50	69.1	1.20
No divergence stop	✓	×	34/50	75.1	0.00
Full	✓	✓	36/50	79.8	–

Preview and checkpoints. Tab. 8 evaluates two mechanisms of the admission operator $\mathcal{A}$ (Secs. 3.2 and 4): previewing a trajectory before execution (Sec. 4.1) and pausing only at model-placed checkpoints (Sec. 4.2). In *No preview*, an action request is planned and executed in one call, so the model sees the trajectory only after it has run (Sec. 4.1). In *Pause after every gripper action*, $\mathcal{A}$ inserts a checkpoint after every gripper stage that a later movement depends on, rather than only where the model placed one (Sec. 4.2). Compared with *Full* (42/50 successes, Score 88.5), *No preview* reaches 36/50 and 74.7, with 0.96 checkpoint pauses and 4.31 unreviewed motions per episode, 0.98 of which carry a clearance risk that preview reports before motion (Sec. 4.1). Compared with *Full*, *Pause after every gripper action* reaches 37/50 and 78.7 with 13.82 pauses per episode, whereas model-placed checkpoints in *Full* pause only at stages selected for inspection (Sec. 4.2).

Table 8 Preview and checkpoints in $\mathcal{A}$ (Secs. 3.2, 4.1 and 4.2): successes over the 50 cases, mean Score, and the checkpoint pauses per episode. The highest successes and Score are in bold. ✓: available; ×: removed.

Variant	Preview	Checkpoints	Successes	Score	Pauses per episode
No preview	×	placed by the model	36/50	74.7	0.96
Pause after every gripper action	✓	after every gripper action	37/50	78.7	13.82
Full	✓	placed by the model	42/50	88.5	–

Expressive actions. Tab. 9 evaluates the two output-side mechanisms of Sec. 5.2: finger-referenced movement goals and the shared dual-arm timeline. In *Wrist-flange goals*, each movement specifies the wrist-flange position and quaternion rather than a finger contact point and workspace rotations (Sec. 5.2), and the overlay that projects the finger positions into the image (Sec. 5.1) is removed. In *One arm at a time*, an action request may contain stages for only one arm, so the two arms no longer advance on a shared timeline (Sec. 5.2). Compared with *Full* (39/50 successes, Score 83.9), *Wrist-flange goals* reaches 37/50 and 77.1 with 3.42

unplannable stages per episode, whereas finger-referenced goals in *Full* let the model specify the finger contact region and workspace orientation directly (Sec. 5.2). Compared with *Full*, *One arm at a time* reaches 35/50 and 74.8 with 1.14 unplannable stages per episode, whereas the shared timeline in *Full* coordinates both arms with declared waits in one sequence (Sec. 5.2).

Table 9 Expressive actions (Sec. 5.2): successes over the 50 cases, mean Score, and the planning failures per episode (requested stages that could not be planned). The highest successes and Score are in bold. ✓: available; ×: removed.

Variant	Finger-referenced goals	Shared timeline	Successes	Score	Planning failures per episode
One arm at a time	✓	×	35/50	74.8	1.14
Wrist-flange goals	×	✓	37/50	77.1	3.42
Full	✓	✓	39/50	83.9	–

Programs. Tab. 10 evaluates how model-written programs interact with $\mathcal{A}$ (Secs. 3.2 and 4.1): in *Full*, a program computes and calls deliberative tools ($\mathcal{U}_d$) freely, while any physical request ($\mathcal{U}_p$) enters $\mathcal{A}$, waits for the model to review its preview (Sec. 4.1), and returns a receipt r_t (Secs. 3.1 and 3.2). In *No programs*, the Compute capability (program_run, Tab. 3) is removed, so the model issues every call directly. In *Programs cannot act*, programs still run for computation and deliberative queries, but $\mathcal{A}$ rejects their physical requests, so the model issues every physical action directly (Sec. 3.2). In *Code as policy*, physical requests from a program run without preview review and return no receipts r_t to the program or the model, leaving only the program’s output and exit status, as in the paradigm contrasted with Auriga in Secs. 2.2 and 3.2. Compared with *Full* (30/30 successes, Score 100.0, 46.2 model calls per episode), *No programs* reaches 19/30 and 75.3 with 66.0 calls per episode (one episode exhausting the 170-response limit), and *Programs cannot act* reaches 19/30 and 73.0 with 67.1 calls per episode; in *Full*, programs issue 28.8 tool calls per episode while the model executes 5.9 physical actions directly (Sec. 3.2). Compared with *Full*, *Code as policy* reaches 20/30 and 77.2 with 71.8 calls per episode. Its programs issue 120 physical actions over the 30 episodes, of which 2 halt when a movement exceeds its arrival tolerance and 1 stops at a checkpoint; without receipts r_t , each of these programs still exits with code 0, one of them after issuing 8 further actions (Secs. 3.2 and 4.2).

Table 10 Programs (Secs. 2.2, 3.2 and 4.1): successes over the 30 cases, mean Score, and the model calls per episode. The highest successes and Score are in bold. ✓: available; ×: removed.

Variant	Programs	Physical actions from programs	Review and receipts	Successes	Score	Model calls per episode
Programs cannot act	✓	×	✓	19/30	73.0	67.1
No programs	×	×	✓	19/30	75.3	66.0
Code as policy	✓	✓	×	20/30	77.2	71.8
Full	✓	✓	✓	30/30	100.0	46.2

6.5 Execution Efficiency and Tool Usage

Episode cost across methods and dimensions. Of the other methods in Tab. 4, PyRUA-Lean [43], HexaShell-Preview [11], and EmbodiedEvo [20] report the test-episode cost of their RoboDojo runs. Tab. 11 lists the cost per test episode of these three methods and of Auriga (measured over the 6,299 scored episodes of Tab. 16), together with each method’s overall average Score in Tab. 4. Of the methods that report input tokens, Auriga achieves the highest average Score (64.00) and uses 5.63 million input tokens per episode, of which 85.4% are served from the prompt cache, leaving 0.82 million uncached input tokens per episode (against 3.94 million total and 0.09 million uncached for EmbodiedEvo at 45.08 Score, and 0.75 million total for PyRUA-Lean at 55.26 Score). An Auriga episode averages 47.5 model calls and 18.8 minutes, compared with 25.7 calls and 12.7 minutes for PyRUA-Lean and a median of 12.3 minutes for HexaShell-Preview. Note that cross-trial

evolution methods (Tab. 1) also incur pre-evaluation trial costs that are not included in their per-test-episode numbers, whereas Auriga runs entirely within the evaluated episode.

Fig. 11 compares the five-dimension Score of Auriga (Tab. 4) against the mean input tokens and uncached input tokens per episode across dimensions (Tab. 16 in App. D). Memory, the dimension with the highest Score (93.40), is also the cheapest, with 30.0 model calls, 3.11 million input tokens (0.45 million uncached), and 11.3 minutes per episode, and Open, with the second-highest Score (65.40), is the second cheapest, with 40.7 model calls, 4.67 million input tokens (0.65 million uncached), and 14.9 minutes. Long-Horizon and Generalization episodes are the most expensive, with 54.5 and 54.2 model calls, 6.70 and 6.57 million input tokens, and 23.0 and 21.6 minutes, and across all 6,299 episodes Auriga uses 79.7% of the native-step limit on average (Tab. 16).

	Auriga	PyRUA- Lean	HexaShell- Preview	Embodied- Evo
Reasoning effort	high	high	–	xhigh
Score	64.00	55.26	53.18	45.08
Model calls	47.5	25.7	–	–
Input (M)	5.63	0.75	–	3.94
Cached (%)	85.4	–	–	97.6
Uncached (M)	0.82	–	–	0.09 [‡]
Output (k)	18.3	9.0	–	19 [‡]
Time (min)	18.8	12.7	12.3 [§]	–

Table 11 Cost per RoboDojo episode of Auriga (over the 6,299 episodes in Tab. 16) and of the three methods in Tab. 4 that report it, with each method’s average Score from Tab. 4; the quantities are defined in Sec. 6.1.1. All four use gpt-6-astra; a dash marks a quantity the method does not report. [‡]Derived from reported totals or shares. [§]Median.

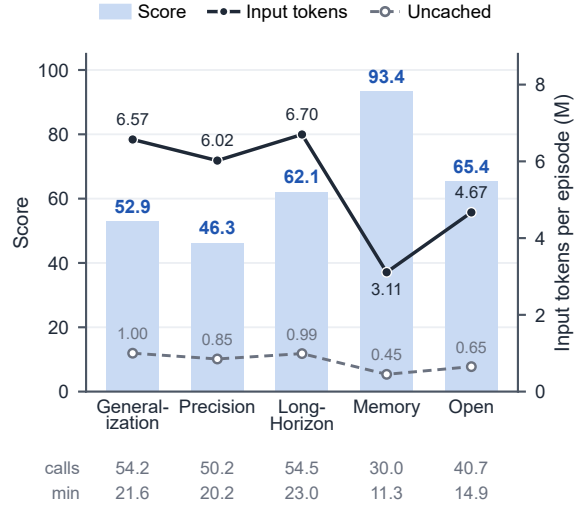


Figure 11 Five-dimension Score (Tab. 4) and per-episode cost of Auriga by dimension (over the 6,299 episodes in Tab. 16): five-dimension Score (bars, left axis), mean input tokens per episode in millions (solid line, right axis), and uncached input tokens in millions (dashed line, right axis). Below the axis: model calls and wall-clock minutes per episode.

Tool usage across capabilities and episode progress. Figs. 12 and 13 group the tool calls of the model by the four capabilities of Tab. 3: Measure, Act, Compute, and Recall. An episode makes 38 direct tool calls on average, from 45 in Long-Horizon and Generalization episodes to 23 in Memory episodes (Fig. 12). Act takes the largest share (45%), followed by Measure (24%), Compute (17%), and Recall (15%). Within each capability, spatial measurement (geometry_query) accounts for 16% of all direct tool calls and scene observation (robot_observe, alongside stereo_guide) for 8% under Measure; under Act, trajectory preview ($\mathcal{U}_d$, equation (2)) accounts for 19% and physical execution ($\mathcal{U}_p$, equation (3)) for 24% in robot_action (with robot_wait at 2%), so 44% of all robot_action calls preview a candidate sequence without advancing the world; under Compute, running programs (program_run) accounts for 17% (with programs issuing 7 additional tool calls per episode on average, rising to 10 in Long-Horizon); and under Recall, updating the task belief (task_plan, Sec. 3.3) accounts for 13% while querying evidence records (history_* and operation_memory, Sec. 3.2) accounts for 2%. Across normalized episode progress $t \in [0, 1]$ (Fig. 13), Measure takes 87% of calls before $t = 0.05$ as the model inspects the scene and localizes targets before acting; from $t = 0.15$ onward, Act (46%–54%) and Compute (17%–22%) account for over 60% of calls while Recall stays between 13% and 17%.

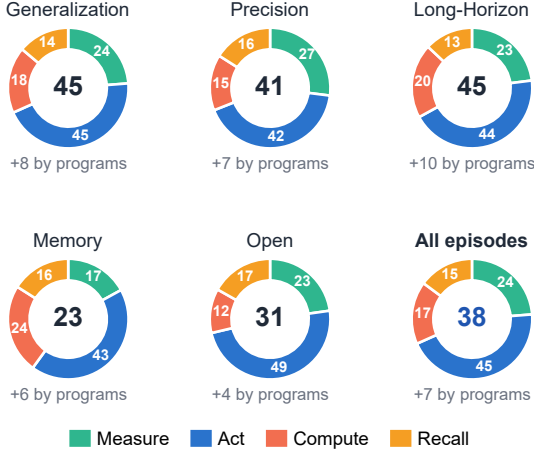


Figure 12 Tool calls per episode by dimension (6,299 episodes): mean number of direct tool calls (center), share of each capability in % (ring segments), and mean number of additional calls made by programs (below each ring).

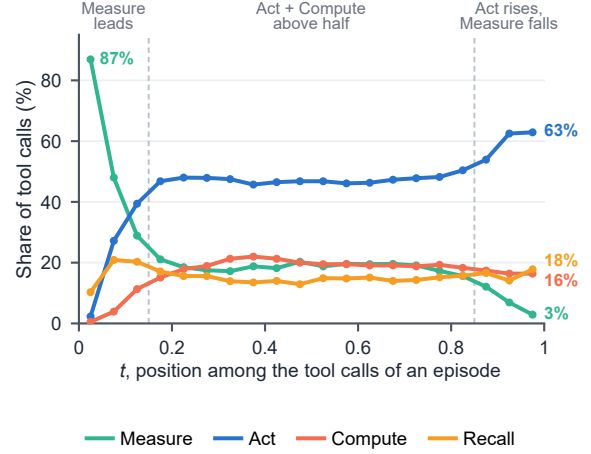


Figure 13 Share of each capability (%) along an episode: the equal-weight mean across episodes (6,298 episodes). The tool calls of each episode are spread in order over $t \in [0, 1]$ and split into 20 equal segments.

Execution granularity against the reference harness. To compare model interaction and physical execution granularity on matched zero-shot scenes, [Tab. 12](#) evaluates Auriga against the published RoboProbe release [60] across the 15 standard-variant scenes of `swap_T` (full step-by-step execution traces and mechanism statistics are detailed in [App. F](#)). Auriga completes 14 of the 15 scenes (93.3% SR), compared with 3 of 15 (20.0% SR) for RoboProbe (+73.3 percentage points), while using fewer model calls per episode (25.4 vs. 36.6 mean, 24 vs. 38 median). Across the 15 episodes, RoboProbe issues 501 physical execution batches (33.4 per episode, 10.9 native steps per batch), whereas Auriga issues 47 batches (3.1 per episode, 107.3 native steps per batch)—a 90.6% reduction in physical execution invocations for a comparable number of native steps (336.2 vs. 365.1), because approach, grasp, transport, and release are previewed and verified as multi-stage operations rather than single end-effector steps.

Table 12 Task outcomes and execution profiles on 15 matched standard-variant T-block exchange (`swap_T`) scenes (detailed trace analysis in [App. F](#)).

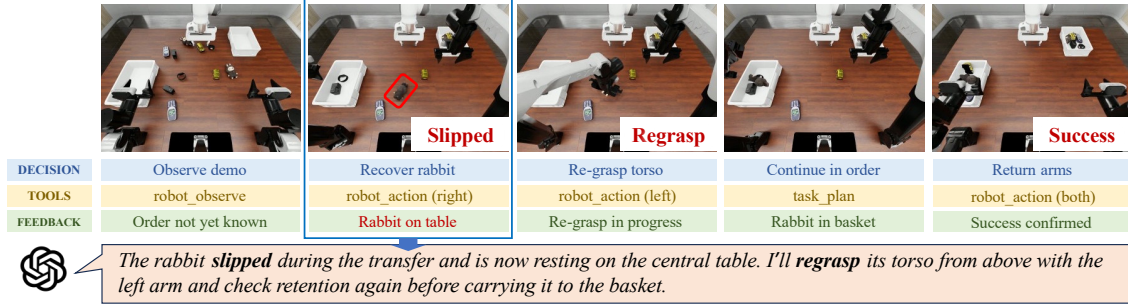
System	Task completion			Model calls / ep.		Physical execution		
	Successes	SR (%)	Exclusive	Mean	Median	Batches / ep.	Steps / ep.	Steps / batch
RoboProbe [60]	3/15	20.0	0	36.6	38	33.4	365.1	10.9
Auriga	14/15	93.3	11	25.4	24	3.1	336.2	107.3

SR: success rate; Exclusive: scenes solved only by that system. Batches count action invocations that advance the environment, including early stops; steps are native actions, and steps/batch is their aggregate ratio.

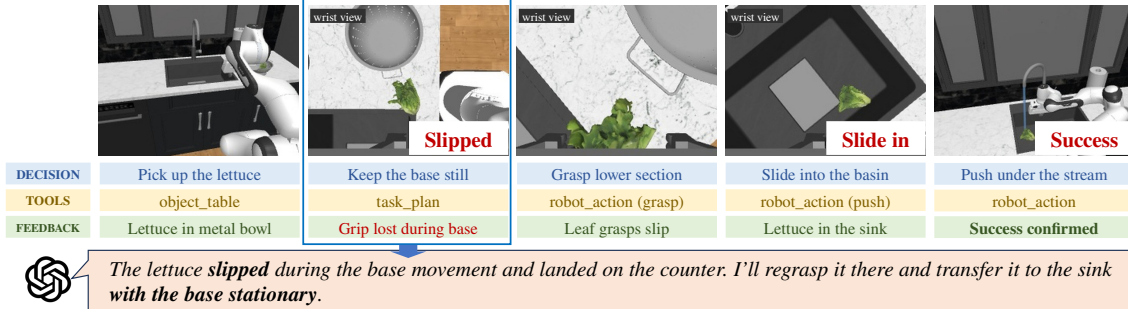
6.6 Qualitative Analysis and Failure Modes

We complement the aggregate metrics and ablations with qualitative episode traces across both RoboDojo and RoboCasa365, organized around four operational challenges: recovery after grasp failure and transport slippage ([Sec. 6.6](#)), scope-matched replanning and budget-aware preview ([Sec. 6.6](#)), 3D geometry-guided manipulation with VGGT-Omega and SAM 3.1 ([Sec. 6.6](#)), and failure modes when repeated corrections under multi-body coupling exhaust the episode budget ([Sec. 6.6](#)). Full step-by-step trace tables on `swap_T` and additional case studies are provided in [Apps. F](#) and [G](#).

Recovery from grasp failure, transport slippage, and arm handover. Grasp acquisition, retention during transport, and stable support after release are tracked as distinct conditions in the task belief B_t , allowing the model to revise how an object is grasped or transported without losing already verified subgoals (Fig. 14).



(a) RoboDojo sequence reproduction (imitate_sorting_sequence): preserving sequence order in B_t and switching from a rim-blocked head grasp to a cross-torso grasp after a handover slip.



(b) RoboCasa365 lettuce washing (WashLettuce): keeping the mobile base still and replacing an aerial lift with a counter-supported slide into the sink after transport slippage.

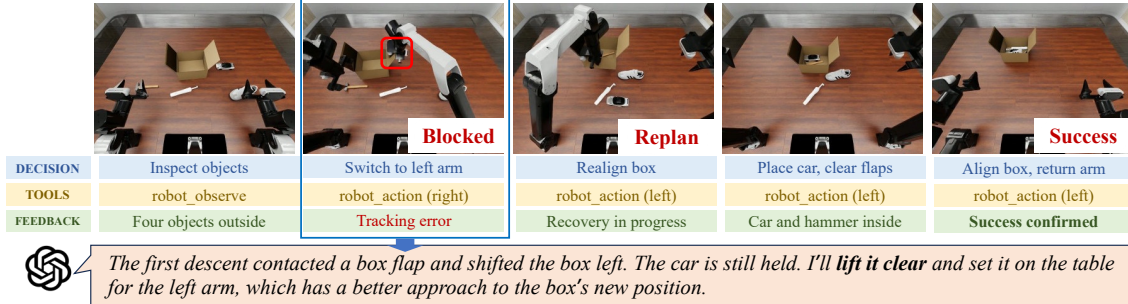
Figure 14 Recovery from grasp failure and transport slippage across RoboDojo and RoboCasa365. Rather than repeating the failed command or discarding prior progress, the model updates the contradicted retention condition in B_t and revises the contact region, active arm, or transport mode.

In RoboDojo `imitate_sorting_sequence` (Fig. 14a), the model must reproduce a five-object placement order (wristwatch, black device, rabbit, phone, gold vehicle). After the first two objects are verified inside the basket, the rabbit lies outside the left arm’s reach, so the right arm transfers it toward the center. When the rabbit slips during transfer and lands outside the basket’s far rim, the task belief keeps the third slot marked `pending` rather than advancing to the phone; after a head grasp catches the rim and triggers a `tracking_error` stop, the model inspects the wrist view, rotates the jaws across the rabbit’s torso in a clearer region, verifies placement inside the basket, and only then proceeds to the remaining two objects. Similarly, in T-block pose exchange (`swap_T`, Tab. 19 in App. F), when the blue block slips onto the table during a right-arm transfer whose robot motion nonetheless reports `completed` with zero final gripper aperture, the model marks blue retention as `contradicted` while preserving the already placed pink block as `supported`, switches recovery to the nearby left arm, and completes the exchange to the saved initial pose.

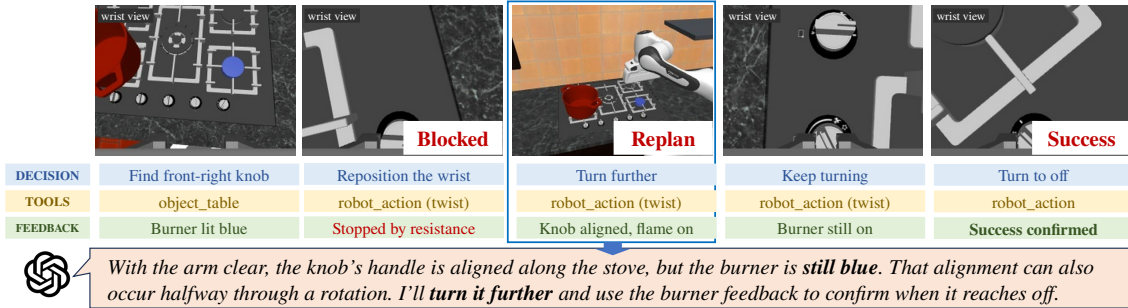
In RoboCasa365 `WashLettuce` (Fig. 14b), transport slippage prompts the model to change the transport mode altogether. After the initial grasp, moving the mobile base causes the head of lettuce to slip onto the counter, as flagged by the execution receipt and wrist-camera object table. When holding the mobile base stationary still leaves three upper-leaf re-grasps blocked or slipping, the model targets a narrower lower section of the lettuce beside the sink and replaces an aerial transfer with a counter-supported slide directly into the basin, before turning on the faucet and pushing the lettuce under the water stream to complete the task (additional slippage recovery cases on `pick_from_conveyor_by_image`, `stack_bowls`, and

PickPlaceDrawerToCounter are analyzed in App. G).

Scope-matched replanning and budget-aware trajectory preview. When an execution receipt and post-action observation invalidate a physical assumption, the model matches the scope of its revision to the contradicted premise (Fig. 15).



(a) RoboDojo object packing (pack_objects_into_box): restoring a box displaced by flap contact to the central workspace and revising descent clearances.



(b) RoboCasa365 burner shutoff (TurnOffStove): re-posturing the wrist when mechanical resistance halts rotation, then continuing past a deceptive halfway handle alignment while the flame remains blue.

Figure 15 Scope-matched replanning after contact and state discrepancies. Each revision addresses the specific assumption contradicted by the execution receipt or visual state—restoring a displaced container, re-posturing the wrist for rotational range, or overriding an ambiguous handle angle.

In RoboDojo pack_objects_into_box (Fig. 15a), the robot must pack a car, shoe, hammer, and toothbrush into a box with their fronts facing left. During the first car descent, wrist contact with a box flap stops the arm with an 8.2 cm vertical residual (tracking_error) and shifts the box across the table. Rather than repeatedly targeting coordinates measured before contact, the model treats the displaced container as state that must be restored: it sets down the held car, repositions the box to the central workspace, and revises subsequent trajectories to keep the wrists above the flaps with vertical descents and withdrawals.

In RoboCasa365 TurnOffStove (Fig. 15b), two distinct assumptions are contradicted in sequence during knob manipulation. First, after a secure grip turns the front-right knob only a few degrees before the execution receipt reports that rotation was stopped by mechanical resistance, the model keeps the verified knob target and releases only to reposition its wrist posture for a wider rotational range. Second, after the next turn brings the knob handle aligned horizontally along the stove—a pose that visually resembles the off position—the burner flame remains visibly blue in the camera view and the burner-on predicate stays true; recognizing that a symmetric handle alignment can occur halfway through a rotation, the model commands a further turn guided by the burner state until the flame extinguishes. Furthermore, when late corrections face a tight step limit—as in a swap_T episode where a 94° stem misalignment is detected at native step 345 with only 55 steps remaining (Tab. 21 in App. F)—paused-world preview rejects an initial 95-step correction and a 57-step revision *before* moving the robot, enabling the model to compress approach, rotation, release, and

withdrawal into a 51-step coordinated trajectory that succeeds at step 396 (further replanning episodes on `plug_in_charger`, `DeliverStraw`, and `SetUpCuttingStation` are detailed in App. G).

Geometry-guided manipulation with VGGT-Omega and SAM 3.1. Fig. 16 compares recorded episodes with and without 3D geometric perception (Sec. 5.1) on matched scenes in RoboDojo and RoboCasa365.

On a randomized RoboDojo `pack_objects_into_box` scene (Fig. 16a), the VGGT-Omega/SAM 3.1 object table reports an initial 18° box yaw misalignment, prompting the model to grasp the front wall and rotate the box from 109° to 90° before packing; to grasp the hammer, querying pixel (267, 274) via `geometry_query` with a VGGT-derived center-height plane yields $\mathbf{p}_{\text{query}} \simeq (-0.1051, -0.1827, 0.7795)$ m, guiding a finger-referenced grasp at $(-0.105, -0.183, 0.778)$ m to pack all four objects, whereas the run without VGGT/SAM exhausts the step limit.

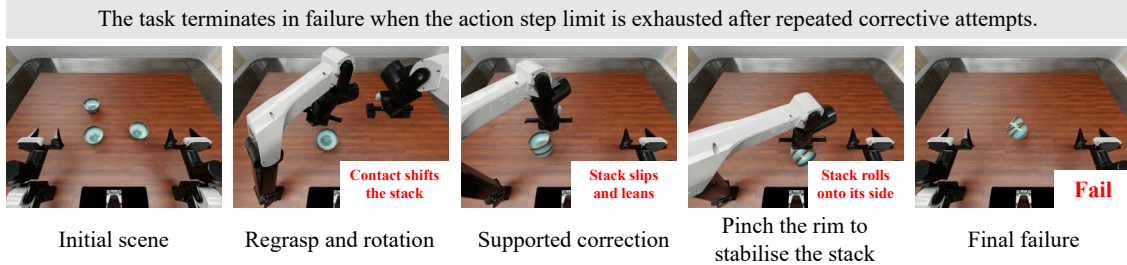
On RoboCasa365 `StackBowlsCabinet` (Fig. 16b), the SAM 3.1 object table locates the open cabinet



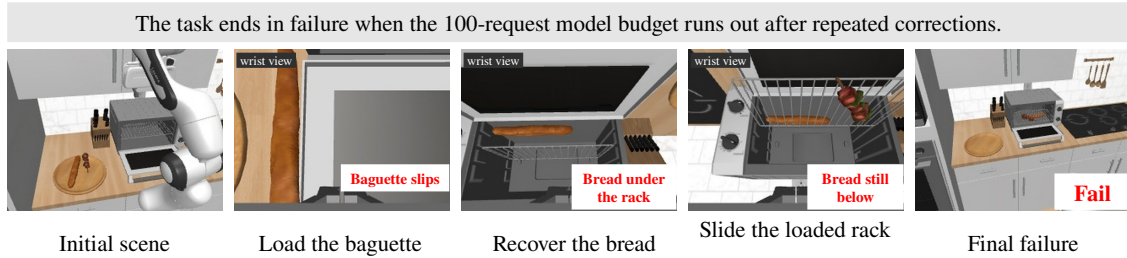
Figure 16 3D geometry-guided manipulation on matched scenes. Estimated 3D bounding boxes, orientations, and surface heights guide container alignment, below-rim grasps, and shelf clearance across both benchmarks.

shelf top at 1.422 m, the large-bowl rim at 0.991 m ($x = 5.431$ m), and the small-bowl rim at 0.967 m ($x = 5.502$ m); the model grasps each bowl 2 cm below its measured rim (0.968 m and 0.948 m) and places them at 1.477 m inside the cabinet to succeed in 29 responses, whereas without the object table the large bowl slips at the cabinet opening, settles inverted on the counter, and eventually falls to the floor.

Failure modes under multi-body coupling and expanding recovery scope.



(a) RoboDojo bowl stacking (`stack_bowls`): leveling a tilted third bowl lifts and rolls the nested stack, exhausting the 800-step native budget.



(b) RoboCasa365 toaster loading (`HeatKebabSandwich`): the baguette lands below the wire rack, and pulling the thin rack rail consumes the 100-request budget.

Figure 17 Failure modes after repeated corrections under multi-body coupling. When a recovery action displaces an already assembled stack or traps an object beneath a fixture rack, local motion completions do not restore cumulative task progress before the budget expires.

Fig. 17 illustrates two representative failure episodes where local corrections perturb coupled structures and exhaust the episode budget (step-by-step budget accounting in Tabs. 22 and 23 in App. G). In RoboDojo `stack_bowls` (Fig. 17a), after the model recovers from an initial slip to form a two-bowl stack (242 steps used, 558 left), placing the third bowl leaves it tilted; attempting to level the top bowl at step 440 lifts the nested three-bowl stack together, which slips and rolls onto its side during rotation, and subsequent supported pushes roll again upon arm withdrawal until the 800-step native budget expires (Score 15/100).

In RoboCasa365 `HeatKebabSandwich` (Fig. 17b), while loading a baguette into a toaster oven, contact with the oven entrance causes the bread to slip and land inside the oven *below* the wire rack at request 41, leaving its rack-contact predicate false. To expose the bread underneath, the model must pull out the wire rack—expanding the problem to grasping a thin front rail that repeatedly slips between requests 63 and 86; although the kebab is placed on the rack at request 50 and the rack finally slides outward at request 86, only 14 requests remain and the 100-request budget expires before the baguette can be lifted onto the rack. These failures highlight the boundary of kinematic trajectory execution: while stage verification and belief updates reliably detect discrepancies, recovery actions that disturb coupled multi-body assemblies or introduce secondary fixture obstacles can consume the remaining budget without achieving stable support.

7 Related Work

7.1 LLMs and VLMs for Embodied Reasoning

Robotic applications of language and vision-language models span task planning, spatial grounding, and direct action generation. Alongside general-purpose models such as GPT-6 [34] and Claude Opus 5.5 [1], embodied foundation models specialize perception and reasoning for physical interaction [6, 12, 19, 45, 58]. Evaluations of GPT-6 Astra [10, 60] investigate direct numerical control and cooperation with learned policies, revealing a gap between semantic task understanding and precise physical execution. Other work adapts robot behavior through context: GPT-Policy [5] preserves task-relevant demonstration transitions, while RoboICL [24] organizes demonstrations and interaction memory around observations, actions, and execution receipts.

7.2 Agentic Harnesses for Robotics

Agentic harnesses organize model interaction with robot tools, programs, and execution feedback. *Trial-and-error interaction* operates within a task: Agent as Policy [16] and CaP-Agent0 [9] within CaP-X allow agents to construct programs and revise execution using observations and tool results, while RoboHarness [38] emphasizes a visual-geometric control interface. The agent may also delegate action generation: PhysicalRSI [31] provides task-aware memory and skills backed by code policies or $\pi_{0.5}$ [36], and FastHarness [22] adapts the frequency of VLM supervision over a frozen vision-language-action (VLA) model.

Cross-trial evolution uses rollout feedback to revise programs, tools, or policies for subsequent executions. HarnessPAI [48] keeps the generated program fixed within a rollout, including its observation-dependent checks, and revises it across rollouts. HexaShell [11] separates an acting agent from an optimizer that repairs and tests executable tools; EmbodiedEvo [20] separates exploration from evaluation, passing distilled code skills and failure lessons to independent test agents. Both freeze the acquired experience for evaluation without updating model weights. Related approaches evolve skills and context code, attribute execution failures, or select informative physical trials [7, 44, 47]; rollout scale and candidate selection also affect harness improvement [51]. WorldScape-R²SI [29] jointly improves reusable skills and a reactive action model, while ENPIRE [53] lets agents revise policy-training algorithms and infrastructure using real-robot evaluations.

Skill libraries complement cross-trial improvement by storing reusable procedures for later retrieval and composition. ASPIRE [26] stores validated repair knowledge as reusable in-context guidance, including applicability conditions, repair strategies, and optional code sketches; RATs [59] stores code skills acquired through self-directed play and retrieves them from a frozen library for downstream tasks. Teach-and-Grow [33] derives closed-loop Skill Blocks from subgoals shared across demonstrations and retains execution conditions and repairs in a separate experience memory. Real2Gym [39] retains task procedures, object-relative motions, and recovery strategies from interaction in simulation environments reconstructed from videos, then reuses them on physical robots. GaP [3] focuses on composition, assembling existing perception, planning, and control modules into computation graphs refined through simulation. Auriga organizes interaction within a persistent model session, where generated programs and direct tool calls share a physical execution interface and return observations and outcomes to guide subsequent decisions.

7.3 Vision-Language-Action and World Action Models

Learned action models map visual and language inputs to robot actions and can serve as policies invoked by an agent. Recent VLA work explores heterogeneous-data co-training and cross-embodiment adaptation [36, 37, 52, 54, 56, 62], experience-driven policy improvement [35], and joint reasoning and action generation [25]. InternVLA-A1.5 [28] further introduces latent foresight supervised by a pretrained video model while retaining semantic training objectives.

World action models (WAMs) incorporate predictive modeling or supervision from future observations into

action learning. Video-based approaches jointly model actions and future observations [18, 50, 55]; VPP2 [14] strengthens manipulation-specific video prediction, and X-WAM [13] adds multi-view RGB-D prediction with asynchronous action decoding. Future supervision can also shape action representations without requiring future-video generation at deployment, as in Being-H0.7, Fast-WAM, and InternW0- Δ [27, 30, 57]. JEPA-WAM [23] couples latent transition prediction to action generation, while Long-WAM [15] studies causal pretraining and longer observation histories. When these policies are used within a harness, the agent coordinates their invocation, evaluates execution outcomes, and revises the task procedure.

8 Conclusion

We have presented **Auriga**, a task-agnostic agent harness for zero-shot manipulation within a single episode, closing 68.3% of the planning–execution gap of vision-language models on RoboDojo (from 22.5% under the reference interface toward the 76.2% arm-free upper bound) without target-task exploration, demonstrations, or environment resets. In Auriga, the model makes all task decisions over an explicit task belief B_t and acts through direct tool calls or synthesized multi-step Python programs, while the harness supplies calibrated 3D perception, finger-referenced dual-arm control, and a single admission gateway $\mathcal{A}$ that previews trajectories while simulation is paused and verifies stage tolerances and grasp bounds during execution for in-place recovery. Developed across 13 cross-task capability iterations and frozen for evaluation, Auriga ranks first overall on the 42-task RoboDojo leaderboard with a 64.0 Score and 57.7% success rate, maintains first place under scene randomization ($G = 46.3$) while learned policies drop by up to 33.9%, and achieves a 73.2% success rate on RoboCasa365 mobile manipulation. Future work will extend this closed-loop execution architecture toward continuous force-sensitive manipulation, lower deliberation latency, and automated cross-task tool synthesis.

Author List

Contributors

Chengcheng Wang, Yuheng Shi, Siyu Xu, Yutong Wang, Jinjing Zhao, Chenghao Xia, Jinqi Liao, Jiankun Peng, Ligong Bi, Yuechuan Ma, Jiaqi Liu, Wenbo Wang, Zijian Wang, Yuemin Wu, Minjing Dong, Xiu Su

Project Leads

Yunke Wang, Chengbin Du, Tao Huang, Jianyuan Guo, Chang Xu

References

- [1] Anthropic. System Card: Claude Opus 5.5. System card, September 2026. URL <https://anthropic.com/claude-opus-5-5-system-card>.
- [2] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. SAM 3: Segment Anything with Concepts. arXiv preprint arXiv:2511.16719, 2026. URL <https://arxiv.org/abs/2511.16719>.
- [3] Kaiyuan Chen, Shuangyu Xie, Letian Fu, Justin Yu, William Pacini, Sandeep Bajamahal, Hudson Kim, Jaimyn Drake, Daehwa Kim, Haoru Xue, Jonathan Francis, Christian Juette, Peter Schaldenbrand, Muhammet Yunus Seker, Ruwan Wickramarachchi, Uksang Yoo, Guanzhi Wang, Adithyavairavan Murali, Balakumar Sundaralingam, S. Shankar Sastry, Yuke Zhu, Linxi "Jim" Fan, Spencer Huang, and Ken Goldberg. GaP: A Graph-as-Policy Multi-Agent Self-Learning Harness For Variational Automation Tasks. arXiv preprint arXiv:2607.05369, 2026. URL <https://arxiv.org/abs/2607.05369>.
- [4] Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, Weijie Wan, Baijun Chen, Songling Liu, Haowen Yan, Honghao Su, Zhiyang Dou, Kaixuan Wang, Dandan Zhang, Yunze Liu, Yan Qin, Qiwei Liang, Qiwei Wu, Zijian Lin, Wenwei Lin, Yuran Wang, Minghua He, Tianshu Wu, Ruihai Wu, Jingquan Zhou, Kai-Chong Lei, Haibao Yu, Yuanfeng Ji, Weiyang Jin, Guanyu Lin, Xiaofan Li, Qi Xiong, Renjing Xu, Zhongyu Li, Wenhao Chai, Enze Xie, Ziwei Wang, Yao Mu, Hao Dong, Wojciech Matusik, Mingyu Ding, Wenbo Ding, Ping Luo, and Masayoshi Tomizuka. RoboDojo: A Unified Sim-and-Real Benchmark for Comprehensive Evaluation of Generalist Robot Manipulation Policies. arXiv preprint arXiv:2607.04434, 2026. URL <https://arxiv.org/abs/2607.04434>.
- [5] Dongzhou Cheng, Taoran Yi, Ye Fang, Xingwu Zhang, Fan Feng, Yixuan Li, Gengxiong Zhuang, Rongze Wang, Shuai Yang, Wei Song, Weizhi Xue, Minyan Wu, Jie Gui, Jiaqi Wang, and Tong Wu. In-Context Robot Learning with VLM Agents. arXiv preprint arXiv:2609.19138, 2026. URL <https://arxiv.org/abs/2609.19138>.
- [6] DeepCybo Team, Yu Bin, Haipeng Cao, Zheng Chang, Kai Chen, Youning Chen, Kailin Deng, Yichao Du, Xiaotong Fu, Haoyang Ge, Yunlong Guo, Chenliu Hao, Jiyan He, Xuguo He, Yakun Hou, Kai Hu, Cong Huang, Tuopusen Huang, Yu Huang, Hong Li, Peize Li, Shijie Lian, Xiaopeng Lin, Yun Lin, Haibao Liu, Haochen Liu, Qiuzhi Liu, Shengcai Liu, Zhiqiang Liu, Tao Luo, Peng Ren, Shuo Ren, Chaoyi Ruan, Zhaolong Shen, Yukun Shi, Qiyuan Su, Yuxuan Tian, Yining Wang, Changti Wu, Hao Wu, Xueyin Xu, Ruoqi Yang, Zhaoyang Yang, Hang Yuan, Zhaoyang Zeng, Hanwen Zhang, Ruimeng Zhang, Yao Zhang, Yibo Zhang, Yuxiang Zhang, Zhirui Zhang, Ziyi Zhang, Zubin Zheng, and Zishen Zhuang. PhysBrain 1.5: From Vision-Language Models to Physical Foundation Models. arXiv preprint arXiv:2609.14973, 2026. URL <https://arxiv.org/abs/2609.14973>.
- [7] Haoyuan Deng, Jiebin Liu, Tengxiao Zhang, Langning Yan, Hongye Cao, and Ziwei Wang. DynaHarness: A Dynamic Physical Harness for Self-Evolving Robot Agents. arXiv preprint arXiv:2609.40306, 2026. URL <https://arxiv.org/abs/2609.40306>.
- [8] Yi Duan, Ying Liu, Zirui Tang, Haodong Chen, Jun Zhou, Yumou Liu, Bangrui Xu, Yukai Wu, Sidi Chen, Yuhan Zhou, Haoyu Wang, Xiaoyou Yu, Shaokun Han, Xuzhou Zhu, Le Zhou, Bolin Lu, Wei Zhou, Jiachen Liu, Nuozhou Fang, Jiaxin Tian, Ruoyu Chen, Yuxuan Li, Kai Zuo, Kaiyan Zhang, Qianyu Yang, Zijie Wang, Jiantao Qiu, Conghui He, Guoliang Li, Bowen Zhou, Zhiyuan Liu, Zhoufutu Wen, Jihua Kang, Xuanhe Zhou, and Fan Wu. The Last AI Built by Humans: Toward Genuine Recursive Self-Improvement. arXiv preprint arXiv:2609.11873, 2026. URL <https://arxiv.org/abs/2609.11873>.
- [9] Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Dantong Niu, Fei-Fei Li, Guanya Shi, Jiajun Wu, Shankar Sastry, Yuke Zhu, Ken Goldberg, and Linxi "Jim" Fan. CaP-X: A Framework for Benchmarking and Improving Coding Agents for Robot Manipulation. arXiv preprint arXiv:2603.22435, 2026. URL <https://arxiv.org/abs/2603.22435>.

- [10] Galbot Team, Xuchuan Chen, Xiaoqian Cheng, Yu Deng, Lihe Ding, Shaocong Dong, Xiangjun Gao, Haozhe Jia, Zekai Li, Zhoujian Li, Yunrui Lian, Sikai Liang, Chenghuai Lin, Dairu Liu, Jiahang Liu, Qingtao Liu, Yuxuan Ma, Zekun Qi, Jiayi Su, He Wang, Ruochen Xu, Tianyu Xu, Xudong Xu, Zhe Xu, Mi Yan, Siming Yan, Li Yi, Ruixi Yu, Jinlu Zhang, Yintianrun Zhang, Zhikai Zhang, Zhizheng Zhang, Yixin Zheng, and Weiyi Zhu. Systematically Exploring the Capabilities of GPT-6 Astra as Embodied Policies. arXiv preprint arXiv:2609.38537, 2026. URL <https://arxiv.org/abs/2609.38537>.
- [11] Hongcheng Gao, Alex Zhou, Jay Zhu, Zelin Zheng, Shijia Ge, Xiping Li, Di Wu, Yazhe Wang, Jianshu Zeng, Xuan Shangguan, Yongli Wang, Yexing Wan, Lingyu He, Zhiqi Jia, Sihang Wu, and Xiao He. HexaShell: A Thin Harness for Exploring the Limits of VLM Agents as Robot Coding Agents. arXiv preprint arXiv:2610.23479, 2026. URL <https://arxiv.org/abs/2610.23479>.
- [12] Gemini Robotics Team, Abbas Abdolmaleki, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Ashwin Balakrishna, Nathan Batchelor, Alex Bewley, Jeff Bingham, Michael Bloesch, Konstantinos Bousmalis, Philemon Brakel, Anthony Brohan, Thomas Buschmann, Arunkumar Byravan, Serkan Cabi, Ken Caluwaerts, Federico Casarini, Christine Chan, Oscar Chang, London Chappellet-Volpini, Jose Enrique Chen, Xi Chen, Hao-Tien Lewis Chiang, Krzysztof Choromanski, Adrian Collister, David B. D’Ambrosio, Sudeep Dasari, Todor Davchev, Meet Kirankumar Dave, Coline Devin, Norman Di Palo, Tianli Ding, Carl Doersch, Adil Dostmohamed, Yilun Du, Debidatta Dwibedi, Sathish Thoppay Egambaram, Michael Elabd, Tom Erez, Xiaolin Fang, Claudio Fantacci, Cody Fong, Erik Frey, Chuyuan Fu, Ruiqi Gao, Marissa Giustina, Keerthana Gopalakrishnan, Laura Graesser, Oliver Groth, Agrim Gupta, Roland Hafner, Steven Hansen, Leonard Hasenclever, Sam Haves, Nicolas Heess, Brandon Hernaez, Alex Hofer, Jasmine Hsu, Lu Huang, Sandy H. Huang, Atil Iscen, Mithun George Jacob, Deepali Jain, Sally Jesmonth, Abhishek Jindal, Ryan Julian, Dmitry Kalashnikov, M. Emre Karagozler, Stefani Karp, Matija Kecman, J. Chase Kew, Donnie Kim, Frank Kim, Junkyung Kim, Thomas Kipf, Sean Kirmani, Ksenia Konyushkova, Li Yang Ku, Yuheng Kuang, Thomas Lampe, Antoine Laurens, Tuan Anh Le, Isabel Leal, Alex X. Lee, Tsang-Wei Edward Lee, Guy Lever, Jacky Liang, Li-Heng Lin, Fangchen Liu, Shangbang Long, Caden Lu, Sharath Maddineni, Anirudha Majumdar, Kevis-Kokitsi Maninis, Andrew Marmon, Sergio Martinez, Assaf Hurwitz Michaely, Niko Milonopoulos, Joss Moore, Robert Moreno, Michael Neunert, Francesco Nori, Joy Ortiz, Kenneth Oslund, Carolina Parada, Emilio Parisotto, Amaris Paryag, Acorn Pooley, Thomas Power, Alessio Quaglino, Haroon Qureshi, Rajkumar Vasudeva Raju, Helen Ran, Dushyant Rao, Kanishka Rao, Isaac Reid, David Rendleman, Krista Reymann, Miguel Rivas, Francesco Romano, Yulia Rubanova, Peter Pastor Sampedro, Pannag R Sanketi, Dhruv Shah, Mohit Sharma, Kathryn Shea, Mohit Shridhar, Charles Shu, Vikas Sindhwani, Sumeet Singh, Radu Soricut, Rachel Sterneck, Ian Storz, Razvan Surdulescu, Jie Tan, Jonathan Tompson, Saran Tunyasuvunakool, Jake Varley, Grace Vesom, Giulia Vezzani, Maria Bauza Villalonga, Oriol Vinyals, René Wagner, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Chengda Wu, Markus Wulfmeier, Fei Xia, Ted Xiao, Annie Xie, Jinyu Xie, Peng Xu, Sichun Xu, Ying Xu, Zhuo Xu, Jimmy Yan, Sherry Yang, Skye Yang, Yuxiang Yang, Hiu Hong Yu, Wenhao Yu, Wentao Yuan, Yuan Yuan, Jingwei Zhang, Tingnan Zhang, Zhiyuan Zhang, Allan Zhou, Guangyao Zhou, and Yuxiang Zhou. Gemini Robotics 1.5: Pushing the Frontier of Generalist Robots with Advanced Embodied Reasoning, Thinking, and Motion Transfer. arXiv preprint arXiv:2510.03342, 2025. URL <https://arxiv.org/abs/2510.03342>.
- [13] Jun Guo, Qiwei Li, Peiyan Li, Zilong Chen, Nan Sun, Yifei Su, Heyun Wang, Yuan Zhang, Xinghang Li, and Huaping Liu. Unified 4D World Action Modeling from Video Priors with Asynchronous Denoising. arXiv preprint arXiv:2604.26694, 2026. URL <https://arxiv.org/abs/2604.26694>.
- [14] Yanjiang Guo, Haodong Yan, Zhide Zhong, Zhongru Zhang, Qingyuan Yang, Qingzhou Lu, Xiaoyu Chen, Yen-Jen Wang, Shuying Deng, Chenghan Yang, Puzhen Yuan, Chenxin Liu, Tun Ban, Xiang Zhu, Yichen Liu, Kun Feng, Haoang Li, and Jianyu Chen. Video Prediction Policy 2: Predict Better, Act Better. arXiv preprint arXiv:2610.10270, 2026. URL <https://arxiv.org/abs/2610.10270>.
- [15] Wei Huang, Bohan Zhang, Chenzhi Liu, Isabella Liu, Shuai Yang, Weian Mao, Luozhou Wang, Yicheng Xiao, Weifeng Lin, Qixin Hu, Bryan Chu, Sifei Liu, Linxi Fan, Xiaojuan Qi, Song Han, and Yukang Chen. Long-WAM: Scaling the Context of World-Action Models. arXiv preprint arXiv:2610.10528, 2026. URL <https://arxiv.org/abs/2610.10528>.
- [16] Mengzhao Jia, Yang Lin, Xixin Zhang, Zhihan Zhang, Xiaobai Liu, and Meng Jiang. Agent as Policy for Robotic Manipulation. arXiv preprint arXiv:2609.12541, 2026. URL <https://arxiv.org/abs/2609.12541>.

- [17] Dongyoung Kim, Huiwon Jang, Myungkyu Koo, Suhyeok Jang, Taeyoung Kim, Beomjun Kim, Byungjun Yoon, Changsung Jang, Daewon Choi, Dongsu Han, Donguk Lee, Heeseung Kwon, Hojin Jeon, Jaehyun Kang, Jaekyoung Bae, Jihyuk Lee, Jimin Lee, John Won, Joonwoo Ahn, Junhyeong Park, Junyoung Sung, Kyungmin Lee, Minseong Han, Minsung Yoon, SeJune Joo, Seonil Son, Seungcheol Park, Seunggeun Cho, Seungjun Moon, Seungku Kim, Yonghoon Dong, Yongjin Cho, Youngchan Kim, Chang Hwan Kim, Dohyeon Kim, Heecheol Kim, Heewon Lee, Hensen Ahn, Hyungkyu Ryu, Hyunsoo Choi, Hyunsoo Shin, Jaheon Jung, Jaewoo Kim, Jinwook Kim, Joochul Chang, Joonsoo Kim, Junghun Park, Jungwoo Park, Junho Cho, Junhyeok Park, Junwon Lee, Kangwook Lee, Kwanghoon Kim, Kyoungwhan Choe, Manoj Bhadu, Nayoung Oh, Sangjun Kim, Sangwoo Kim, Seunghoon Shim, Seunghyun Kim, Seungjun Lee, Seungyup Ka, Sungryol Yang, Wook Jung, Yashu Shukla, Yeonjae Lee, Yeonwoo Bae, and Jinwoo Shin. RLDX-1 Technical Report. arXiv preprint arXiv:2605.03269, 2026. URL <https://arxiv.org/abs/2605.03269>.
- [18] Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos Policy: Fine-Tuning Video Models for Visuomotor Control and Planning. arXiv preprint arXiv:2601.16163, 2026. URL <https://arxiv.org/abs/2601.16163>.
- [19] Kehan Li, Bohan Hou, Minghao Zhu, Tianyi Zhang, Zesen Cheng, Zhikai Wang, Sicong Leng, Xin Li, Xiao Lin, Biying Yao, Minghua Zeng, Jiangpin Liu, Ronghao Dang, Jiayan Guo, Siteng Huang, Haoyu Zhao, Heng Ping, Yaxi Zhao, Tong Zhao, Kexiang Wang, Tong Lu, Shengke Xue, Jiahao Tang, Yulei Wang, Zejing Wang, Jianwei Gao, Shijian Lu, Chengju Liu, Jianfei Yang, Mingxiu Chen, and Deli Zhao. RynnBrain 1.1: Towards More Capable and Generalizable Embodied Foundation Model. arXiv preprint arXiv:2607.17977, 2026. URL <https://arxiv.org/abs/2607.17977>.
- [20] Wenyi Li, Yizhe Chi, Youjie Zheng, Chunyu Liu, Xiaoqi Wang, Mingju Gao, Zheng Jiang, Deyao Hong, Calvin Xiao, Kaisen Yang, and Qinhuai Na. EmbodiedEvo: Self-Evolving Coding Agent For Physical Intelligence. arXiv preprint arXiv:2610.18867, 2026. URL <https://arxiv.org/abs/2610.18867>.
- [21] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as Policies: Language Model Programs for Embodied Control. arXiv preprint arXiv:2209.07753, 2023. URL <https://arxiv.org/abs/2209.07753>.
- [22] Tao Lin, Yuxin Du, Haoxiao Wang, Zile Zhong, and Bo Zhao. FastHarness: Experience-Driven Review Scheduling for Vision-Language-Action Policies. arXiv preprint arXiv:2610.31427, 2026. URL <https://arxiv.org/abs/2610.31427>.
- [23] Yihan Lin, Jiawei He, Shifeng Bao, Chen Zhao, Yang Li, Xiaobo Wang, Yan Wang, Cheng Chi, and Jing Zhang. JEPA-WAM: Learning Vision-Language-Action Policies with Joint-Embedding World Modeling. arXiv preprint arXiv:2608.09381, 2026. URL <https://arxiv.org/abs/2608.09381>.
- [24] Fangcheng Liu, Yeqing Shen, Anda Cheng, Weishi Mi, Chao Tang, Chenyuan Liu, Yushun Xiang, Tingguang Li, Yong-Lu Li, and Yehui Tang. RoboICL: Embodied In-Context Learning with GPT-6 Astra. arXiv preprint arXiv:2609.34261, 2026. URL <https://arxiv.org/abs/2609.34261>.
- [25] Yicheng Liu, Zibin Dong, Baijun Ye, Tianyuan Yuan, Tao Jiang, Anqi Yang, Shicheng Cao, Haonan Liu, Yue Sun, Zihan Guo, Xiao Liu, Dong Ke, Changxun Pan, Chenru Wu, Tailai Cheng, Xiaoshu Ren, Xinlei Zhang, Jianning Cui, Zijie Zhao, Haoyu Zhang, Kaiming Xu, Haodong Yang, Bowen Zhang, Jiahui Niu, Shaoting Zhu, Shiduo Zhang, and Hang Zhao. G0.5: One Autoregressive Stream for Robot Reasoning and Action. arXiv preprint arXiv:2608.11739, 2026. URL <https://arxiv.org/abs/2608.11739>.
- [26] Runyu Lu, Yubo Wu, Ethan Kou, Haotian Lin, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, Mosharaf Chowdhury, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Guanzhi Wang. ASPIRE: Agentic /Skills Discovery for Robotics. arXiv preprint arXiv:2607.00272, 2026. URL <https://arxiv.org/abs/2607.00272>.
- [27] Hao Luo, Wanpeng Zhang, Yicheng Feng, Sipeng Zheng, Haiweng Xu, Chaoyi Xu, Ziheng Xi, Yuhui Fu, and Zongqing Lu. Being-H0.7: A Latent World-Action Model from Egocentric Videos. arXiv preprint arXiv:2605.00078, 2026. URL <https://arxiv.org/abs/2605.00078>.
- [28] Haoxiang Ma, Junhao Cai, Xiaoxu Xu, Hao Li, Yuyin Yang, Yang Tian, Jiafei Cao, Hongrui Zhu, Zherui

- Qiu, Zhaxizhuoma, Yuqiang Yang, Jiaqi Peng, Xueyuan Wei, Yangkun Zhu, Jiahao Jiang, Xing Gao, Hanqing Wang, Feng Yuan, Kailin Li, Xueyue Zhu, Tai Wang, Yan Ding, Jiangmiao Pang, Jia Zeng, Jingjing Zhang, Bowen Zhou, Yao Mu, Chunhua Shen, and Weinan Zhang. InternVLA-A1.5: Unifying Understanding, Latent Foresight, and Action for Compositional Generalization. arXiv preprint arXiv:2607.04988, 2026. URL <https://arxiv.org/abs/2607.04988>.
- [29] Manifold AI Team. R^2 SI: Recursive Robot Self-Improvement via a Bi-Level Agentic System with Agent-Model Co-Evolution. Technical report, Manifold AI, October 2026. URL <https://r2si.worldscape.io/report.pdf>.
- [30] Xingyu Miao, Zizun Li, Baole Fang, Kaiwen Song, Tenghui Wang, Hanxue Zhang, Yating Wang, Xudong Li, Yuping He, Xueyuan Wei, Chao Gao, Xijie Yang, Yingxiang Xu, Kerui Ren, Wenqi Guo, Jianjun Zhou, Xinzhe Wang, Weiguang Zhao, Ni Yang, Zetao Cai, Yufei Xue, Hengjie Li, Zeyu He, Yuanzhen Zhou, Rong Fu, Jianyang Zhang, Siwei Cui, Fuxian Huang, Yunsong Zhou, Xing Gao, Yifei Yao, Qiaojun Yu, Kailin Li, Ming Zhou, Mu Huang, Xinyue Li, Wenze Cui, Bingqi Jiang, Xueyue Zhu, Junting Dong, Haoyu Guo, Tao Lu, Mulin Yu, Bowen Zhou, Bin Zhao, Tianfan Xue, Weinan Zhang, and Chunhua Shen. InternW0- Δ : A World Action Model Bridging Predictive Dynamics and Actions with 20K+ Hours of Open Data. arXiv preprint arXiv:2609.31394, 2026. URL <https://arxiv.org/abs/2609.31394>.
- [31] MMLab & PhysicalRSI contributors. Physical RSI 1.0: Recursive Self-Harness for Scaling Embodied Skills. Project website and open-source software, 2026. URL <https://mmlab.hk/research/PhysicalRSI>. Preview release.
- [32] Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. RoboCasa365: A Large-Scale Simulation Framework for Training and Benchmarking Generalist Robots. arXiv preprint arXiv:2603.04356, 2026. URL <https://arxiv.org/abs/2603.04356>.
- [33] Chang Nie, Zhe Liu, and Hesheng Wang. Teach and Grow: An Agent-Centered Architecture for General Robot Learning. arXiv preprint arXiv:2608.17209, 2026. URL <https://arxiv.org/abs/2608.17209>.
- [34] OpenAI. GPT-6 Astra System Card. System card, September 2026. URL <https://deploymentsafety.openai.com/gpt-6-astra>.
- [35] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi_{0.6}^*$: a VLA That Learns From Experience. arXiv preprint arXiv:2511.14759, 2025. URL <https://arxiv.org/abs/2511.14759>.
- [36] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}^*$: a Vision-Language-Action Model with Open-World Generalization. arXiv preprint arXiv:2504.16054, 2025. URL <https://arxiv.org/abs/2504.16054>.
- [37] Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, Vedant Choudhary, Foster Collins, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Maitrayee Dhaka, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachlan Groom, Haroun Habeeb, Hunter Hancock, Karol Hausman, Gashon Hussein, Victor Hwang, Brian Ichter, Connor Jacobsen, Szymon Jakubczak, Rowan Jen, Tim Jones, Gregg Kammerer, Ben Katz, Liyiming Ke, Mairbek Khadikov, Chandra

- Kuchi, Marinda Lamb, Devin LeBlanc, Brendon LeCount, Sergey Levine, Xinyu Li, Adrian Li-Bell, Vladislav Lialin, Zhonglin Liang, Wallace Lim, Yao Lu, Enyu Luo, Vishnu Mano, Nandan Marwaha, Aikys Mongush, Liam Murphy, Suraj Nair, Tyler Patterson, Karl Pertsch, Allen Z. Ren, Gavin Schelske, Charvi Sharma, Baifeng Shi, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Jiaming Tang, Jimmy Tanner, Shalom Tekeste, Marcel Torne, Kyle Vedder, Quan Vuong, Anna Walling, Haohuan Wang, Jason Wang, XuDong Wang, Chris Whalen, Samuel Whitmore, Blake Williams, Charles Xu, Sukwon Yoo, Lili Yu, Wuming Zhang, Zhuoyang Zhang, and Ury Zhilinsky. $\pi_{0.7}$: a Steerable Generalist Robotic Foundation Model with Emergent Capabilities. arXiv preprint arXiv:2604.15483, 2026. URL <https://arxiv.org/abs/2604.15483>.
- [38] Bince Qu, Wei Chen, and Bo Zhang. RoboHarness: A Simple Harness Outperforms VLA and World Action Models. arXiv preprint arXiv:2610.14018, 2026. URL <https://arxiv.org/abs/2610.14018>.
- [39] Kerui Ren, Yingxiang Xu, Kaiwen Song, Linning Xu, Bo Dai, Mulin Yu, and Tao Lu. Real2Gym: Building Gyms from Videos, Bringing Skills to Robots. arXiv preprint arXiv:2609.37089, 2026. URL <https://arxiv.org/abs/2609.37089>.
- [40] RoboCasa Team. RoboCasa365 leaderboard. <https://robocasa.ai/leaderboard.html>, 2026. Accessed 2026-10-30.
- [41] RoboDojo. RoboDojo Leaderboard, 2026. URL <https://robodojo-benchmark.com/leaderboard>. Accessed 2026-10-31.
- [42] SenseNova, Joint Research Team, Zhongang Cai, Ziang Cao, Tailai Chen, Han Chong Chua, Linjun Dai, Hanming Deng, Hokin Deng, Haiwen Diao, Chenjing Ding, Weichen Fan, Xiao Fu, Zerui Gong, Chenyang Gu, Zexin He, Zhi Hou, Bo Li, Qinyue Li, Quanzhou Li, Xiaowan Li, Zhichao Liao, Dahua Lin, Zilong Lin, Bo Liu, Chunxiao Liu, Pengfei Liu, Tianqi Liu, Ziwei Liu, Chen Change Loy, Lewei Lu, Jiapeng Luo, Yuexiao Ma, Liang Pan, Xingang Pan, Oscar Qian, Ziyi Shang, Yuyao Shao, Dawei Su, Dacheng Tao, Qingyi Tao, Quan Wang, Ruisi Wang, Xiaogang Wang, Ziyi Wang, Beichen Wen, Donghao Wu, Jiaxu Wu, Silei Wu, Xiaoming Wu, Zhonghua Wu, Hongcan Xiao, Weiye Xu, Atsushi Yamashita, Lei Yang, Zhe Yang, Baiqiao Yin, Wanqi Yin, Hanfei Zhao, Tianbo Zhong, Long Zhuo, and Jing Zuo. Has GPT-6-Astra Achieved Embodied Intelligence? arXiv preprint arXiv:2610.22471, 2026. URL <https://arxiv.org/abs/2610.22471>.
- [43] Ruiyang Si, Jianxin Bi, Shunyu Yang, Rui Ni, Wenbo Huang, Qiang Wang, Shulong Jiang, Yintianrun Zhang, Feixiang Ruan, Kewei Zhang, Duomin Wang, Xiuyu Li, Haiwen Feng, Zhen Dong, and Daquan Zhou. Fewer Tokens, Better Action: GPT-6 Astra Robot Agents with 14% Higher Success Rate but 65% Fewer Tokens. arXiv preprint arXiv:2610.01939, 2026. URL <https://arxiv.org/abs/2610.01939>.
- [44] Python Song, Zhixuan Liang, Kelsey Fu, Mengdi Wang, Junfeng Yang, and Shilong Liu. EmbodiedRSI: Active Continual Robot Learning Through Hypothesis-Guided Co-Evolution. arXiv preprint arXiv:2610.10498, 2026. URL <https://arxiv.org/abs/2610.10498>.
- [45] Tencent Robotics X, HY Vision Team, Xumin Yu, Zuyan Liu, Ziyi Wang, He Zhang, Yongming Rao, Fangfu Liu, Yani Zhang, Ruowen Zhao, Oran Wang, Yves Liang, Haitao Lin, Minghui Wang, Yubo Dong, Kevin Cheng, Bolin Ni, Rui Huang, Han Hu, Zhengyou Zhang, Linus, and Shunyu Yao. HY-Embodied-0.5: Embodied Foundation Models for Real-World Agents. arXiv preprint arXiv:2604.07430, 2026. URL <https://arxiv.org/abs/2604.07430>.
- [46] Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schöninger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. VGGT- Ω . arXiv preprint arXiv:2605.15195, 2026. URL <https://arxiv.org/abs/2605.15195>.
- [47] Peidong Wang, Zhiming Ma, Ying Chang, Xufang Luo, Yiqun Zhang, Zihan Wang, Xiaocui Yang, Shi Feng, Yuqing Yang, and Dongsheng Li. Self-Evolving Embodied Agents via Skill-Harness Evolution. arXiv preprint arXiv:2608.11350, 2026. URL <https://arxiv.org/abs/2608.11350>.
- [48] Xin Wang, Wenhao Wu, Menghao Zhang, Zhi Wang, Kun Shao, Jian Luan, Yang Li, Qing Li, Shangding Gu, Huichi Zhou, Shuqing Shi, Fei Ni, Shuo Lu, Weicheng Meng, Kang Li, Jin Wu, Kang Zhao, Shangmin Guo, Gen Li, Yongqiang Tang, Zhizhong Zhang, Yuan Xie, and Heng Qu. HarnessPAI: An Evolving Harness for Physical AI. arXiv preprint arXiv:2609.29166, 2026. URL <https://arxiv.org/abs/2609.29166>.

- [49] Xinyi Wang, Bangzheng Li, Hanyu Wang, Lin Zhang, Andrea Yaoyun Cui, Zhenggang Tang, Shyamal Buch, and Dejie Xu. SEAR: Self-Evolving Agentic Robotics. arXiv preprint arXiv:2610.18792, 2026. URL <https://arxiv.org/abs/2610.18792>.
- [50] Yuran Wang, Siqiao Huang, Mingleyang Li, Chenhao Zhang, Jiaqi Liang, Weiyang Jin, Yue Chen, Xuemin Chi, Donghao Zhou, Qize Yu, Yu-Kai Wang, Yuhan Rui, Shenzhe Yao, Zhen Yuan, Zhenhao Shen, Kefei Zhu, Zijie Zhu, Ning Gao, Xiaowei Chi, Guanqi He, Shanghang Zhang, Hao Dong, Lin Shao, and Hang Zhao. OpenWAM: An Open, Modular Exploration Towards Systematic World-Action Model Pretraining. arXiv preprint arXiv:2609.07398, 2026. URL <https://arxiv.org/abs/2609.07398>.
- [51] Zhijie Wei, Ferris Tan, Jinghui Wang, and Hao Zhou. Scale and Selection: What Makes Automatic Harness Evolution Work for Visual-Interface Robot Agents. arXiv preprint arXiv:2609.39304, 2026. URL <https://arxiv.org/abs/2609.39304>.
- [52] Wei Wu, Fangjing Wang, Fan Lu, He Sun, Shi Liu, Yunnan Wang, Yibin Yan, Yong Wang, Shuailei Ma, Xinyang Wang, Yibin Liu, Shuai Yang, Tianxiang Zhou, Kejia Zhang, Lei Zhou, Cheng Su, Nan Xue, Bin Tan, Han Zhang, Youchao Zhang, Fei Liao, Xing Zhu, Yujun Shen, and Kecheng Zheng. From Foundation to Application: Improving VLA Models in Practice. arXiv preprint arXiv:2607.06403, 2026. URL <https://arxiv.org/abs/2607.06403>.
- [53] Wenli Xiao, Jia Xie, Tonghe Zhang, Haotian Lin, Letian "Max" Fu, Haoru Xue, Jalen Lu, Yi Yang, Cunxi Dai, Zi Wang, Jimmy Wu, Guanzhi Wang, S. Shankar Sastry, Ken Goldberg, Linxi "Jim" Fan, Yuke Zhu, and Guanya Shi. ENPIRE: Agentic Robot Policy Self-Improvement in the Real World. arXiv preprint arXiv:2606.19980, 2026. URL <https://arxiv.org/abs/2606.19980>.
- [54] Xiaomi Robotics Team, Jun Guo, Piaopiao Jin, Jason Li, Peiyan Li, Yingyan Li, Futeng Liu, Wanli Peng, Optimus Qin, Yifei Su, Nan Sun, Qiao Sun, Runze Suo, Heyun Wang, Yunhong Wang, Rujie Wu, Caoyu Xia, Lina Zhang, Jack Zhao, Guoliang Chen, Wenlong Chen, Xinze He, Bin Li, Qing Li, Zhuorong Li, Heng Qu, Wenxuan Song, Diyun Xiang, Yifan Xie, Peiran Xu, Hangjun Ye, Wen Ye, Han Zhao, and Quanyun Zhou. Xiaomi-Robotics-1: Scaling Vision-Language-Action Models with over 100K Hours of Real-World Trajectories. arXiv preprint arXiv:2607.15330, 2026. URL <https://arxiv.org/abs/2607.15330>.
- [55] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World Action Models are Zero-shot Policies. arXiv preprint arXiv:2602.15922, 2026. URL <https://arxiv.org/abs/2602.15922>.
- [56] Haoqi Yuan, Zhixuan Liang, Anzhe Chen, Ye Wang, Haoyang Li, Pei Lin, Yiyang Huang, Zixing Lei, Tong Zhang, Jiazhao Zhang, Jie Zhang, Jingyang Fan, Gengze Zhou, Qihang Peng, Chenxu Lv, Xiaoyue Chen, An Yang, Fei Huang, Junyang Lin, Dayiheng Liu, Jingren Zhou, Chenfei Wu, and Xiong-Hui Chen. Qwen-RobotManip Technical Report: Alignment Unlocks Scale for Robotic Manipulation Foundation Models. arXiv preprint arXiv:2606.17846, 2026. URL <https://arxiv.org/abs/2606.17846>.
- [57] Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do World Action Models Need Test-time Future Imagination? arXiv preprint arXiv:2603.16666, 2026. URL <https://arxiv.org/abs/2603.16666>.
- [58] Yifu Yuan, Yaoting Huang, Xianze Yao, Yutong Li, Shuoheng Zhang, Linqi Han, Pengyi Li, Jiangeng Sun, Wenting Jia, Zhao Zhang, Yuhao Liu, Ruihao Liao, Yucheng Hu, Qiyu Wu, Yuxiao Li, Zibin Dong, Fei Ni, Yan Zheng, Shuyang Gu, Yi Ma, Hongyao Tang, Han Hu, and Jianye Hao. Embodied-R1.5: Evolving Physical Intelligence via Embodied Foundation Models. arXiv preprint arXiv:2606.11324, 2026. URL <https://arxiv.org/abs/2606.11324>.
- [59] Junyi Zhang, Jiaxin Ge, Hanjun Yoo, Letian Fu, Zihan Yang, Yaowei Liu, Raj Saravanan, Shaofeng Yin, Justin Yu, Dantong Niu, Zirui Wang, Roei Herzig, Ken Goldberg, Yutong Bai, David M. Chan, Ion Stoica, Angjoo Kanazawa, Jiahui Lei, Haiwen Feng, and Trevor Darrell. Playful Agentic Robot Learning. arXiv preprint arXiv:2606.19419, 2026. URL <https://arxiv.org/abs/2606.19419>.

- [60] Wenbo Zhang, Kaixuan Wang, Yutao Ouyang, Xiaoyu Huang, Liyang Li, Kailun Su, Weiyang Jin, Wenhao Chai, Haotian Liang, Zhiyang Dou, Yue Chen, and Tianxing Chen. An Unexpected Robot Policy: Early Evaluations of GPT-6 Astra on RoboDojo and Beyond. arXiv preprint arXiv:2609.24170, 2026. URL <https://arxiv.org/abs/2609.24170>.
- [61] Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Yi Nie, Chunyang Zhu, Jiaying Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, Jiayi Rao, Zhengru Fang, Changxu Wei, Yu Wang, Wenbo Ding, and Chao Yu. Harness VLA: Steering Frozen VLAs into Reliable Manipulation Primitives via Memory-Guided Agents. arXiv preprint arXiv:2607.08448, 2026. URL <https://arxiv.org/abs/2607.08448>.
- [62] Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Ya-Qin Zhang, Jiangmiao Pang, Jingjing Liu, Tai Wang, and Xianyu Zhan. X-VLA: Soft-Prompted Transformer as Scalable Cross-Embodiment Vision-Language-Action Model. arXiv preprint arXiv:2510.10274, 2025. URL <https://arxiv.org/abs/2510.10274>.
- [63] Hai Zhou, Fan Yang, Zeyi Chen, Lejun Zhang, Chenyang Fu, Jianping Li, Jiayi Zhu, Xianyang Liu, Yingxuan Yang, Lili Meng, Lu Wang, Xinyang Han, Jiankai Sun, Weirui Ye, Guanchu Wang, Wanhua Li, Ming Zhou, Boyi Li, Ming Jin, Kun Shao, Muhao Chen, Guang Chen, Lei Ma, Joey Tianyi Zhou, Costas Spanos, Xia Hu, Mac Schwager, Weinan Zhang, Qian Feng, and Shangding Gu. Who Improves the Robot? Toward Physical Recursive Self-Improvement. arXiv preprint arXiv:2610.18842, 2026. URL <https://arxiv.org/abs/2610.18842>.
- [64] Youwei Zhou, Bohan Hou, Yankai Fu, Yaoxu Lyu, Yifan Ye, Shilin Shan, Chen Guo, Jiaming Liu, Ran He, Caifeng Shan, Shanghang Zhang, Jianfei Yang, and Zhihe Lu. Large Multimodal Models for Direct Robot Control: GPT-6 as a Beginning. arXiv preprint arXiv:2610.28259, 2026. URL <https://arxiv.org/abs/2610.28259>.

A Benchmark Tasks and Evaluation Scope

Tab. 13 lists the 42 tasks of RoboDojo by capability dimension (Sec. 6.1.1).

Evaluation protocol comparison scope (Tab. 1). Tab. 1 in Sec. 2.1 distinguishes target-task environment trials from offline demonstration analysis. Prepared code/skills include procedures later generalized across tasks; trial-derived memory denotes separate notes inherited from earlier trials. Demo-derived artifacts are distilled text or code, whereas demo references are recorded images or motions, including those retrieved from frozen packages. Task-independent tools, public task specifications, current-episode history, benchmark-required memory events, and training data encoded in model weights are outside these categories. A fixed evaluation agent does not establish the absence of earlier task-driven tool development: PyRUA-Lean uses shared primitives without cross-episode memory, but does not report whether target-task trials informed their development, so its two preparation entries are marked as not reported [43]; HexaShell is shown for non-Open tasks, as its Open setting omits task-specific development and notes [11]; and the R²SI row describes the inspected public release, where outer-loop trials exclude Open tasks and the snapshot supplies no separate memory files or demo context, although the framework supports memory updates and its WAMs use trained weights [29].

Table 13 The five capability dimensions of RoboDojo and their tasks.

Dimension	Tasks	What it covers	Task identifiers
Generalization	12	Manipulation across varied objects and layouts, including randomized scene variants	stack_bowls, push_T, pack_objects_into_box, fold_clothes, hang_mugs, sweep_blocks, pour_liquid_into_cup, make_toast, arrange_largest_number, sort_nesting_dolls_by_size, store_laptop_and_headphones, stack_blocks
Precision	8	Fine-grained manipulation under tight spatial constraints	fasten_screws, plug_in_charger, insert_tubes, pour_balls_into_vase, play_Xylophone, deposit_coin, insert_key, build_tower
Long-Horizon	8	Multi-step tasks that require completing several subgoals	put_bottles_into_dustbin, fill_pen_holder, classify_objects, play_tic_tac_toe, fill_egg_holder, organize_table, make_kong, play_stacking_toy
Memory	6	Tracking state, recalling a sequence, or matching after a delay	cover_blocks, match_and_pick_from_conveyor, swap_blocks, swap_T, press_by_number, imitate_sorting_sequence
Open	8	Open-ended tasks and manipulation conditioned on language or images	align_blocks, general_pickup, stack_blocks_by_language, solve_equation, classify_objects_by_language, pick_from_conveyor_by_image, store_tools_in_toolbox, pour_by_language

B Ablation Details

Tab. 14 lists the ten tasks of each ablation table in Sec. 6.4: five standard layouts per task (50 cases) in Tabs. 6 to 9, and 30 cases in Tab. 10.

Table 14 The ten tasks of each ablation table by dimension: five standard layouts per task (50 cases) in [Tabs. 6 to 9](#), and 30 cases in [Tab. 10](#).

Table	Tasks
Tab. 6	Generalization: stack_blocks, stack_bowls, push_T; Long-Horizon: play_tic_tac_toe; Memory: cover_blocks, press_by_number, imitate_sorting_sequence; Open: align_blocks, stack_blocks_by_language, classify_objects_by_language
Tab. 7	Generalization: stack_bowls, sort_nesting_dolls_by_size; Precision: fasten_screws, insert_tubes; Long-Horizon: classify_objects, play_stacking_toy; Memory: cover_blocks, swap_T; Open: classify_objects_by_language, align_blocks
Tab. 8	Generalization: fold_clothes, arrange_largest_number, sort_nesting_dolls_by_size; Precision: fasten_screws; Long-Horizon: classify_objects; Memory: swap_T, cover_blocks, imitate_sorting_sequence; Open: pick_from_conveyor_by_image, classify_objects_by_language
Tab. 9	Generalization: fold_clothes, arrange_largest_number; Precision: fasten_screws, play_Xylophone; Long-Horizon: classify_objects, play_tic_tac_toe; Memory: press_by_number, cover_blocks; Open: classify_objects_by_language, align_blocks
Tab. 10	Generalization: sort_nesting_dolls_by_size, pack_objects_into_box; Precision: fasten_screws; Long-Horizon: play_stacking_toy, play_tic_tac_toe, classify_objects; Memory: press_by_number, imitate_sorting_sequence; Open: pick_from_conveyor_by_image, classify_objects_by_language

C RoboCasa365 Results

[Tab. 15](#) lists four entries of the official RoboCasa365 leaderboard [\[40\]](#), which are evaluated in the pretraining kitchens rather than in the target kitchens of [Tab. 5](#); Xiaomi-Robotics-1 leads all four columns among these entries, with 57.4% Overall. [Fig. 18](#) plots these four entries alongside Auriga, RPent, and RLDX-1 from [Tab. 5](#). [Fig. 19](#) gives the outcome of every task and episode of Auriga.

Table 15 Success rates (%) of four entries of the official RoboCasa365 leaderboard [\[40\]](#) on the 50 Target50 tasks, evaluated in the pretraining kitchens and ordered by ascending Overall SR. Overall is the mean over the 50 tasks.

Method	Type	Overall	Atomic-Seen	Composite-Seen	Composite-Unseen
π_0	VLA	14.8	34.6	6.1	1.1
$\pi_{0.5}$	VLA	16.9	39.6	7.1	1.2
WorldDreamer	WAM	35.3	66.3	26.7	9.0
Xiaomi-Robotics-1	VLA	57.4	80.2	57.1	32.1

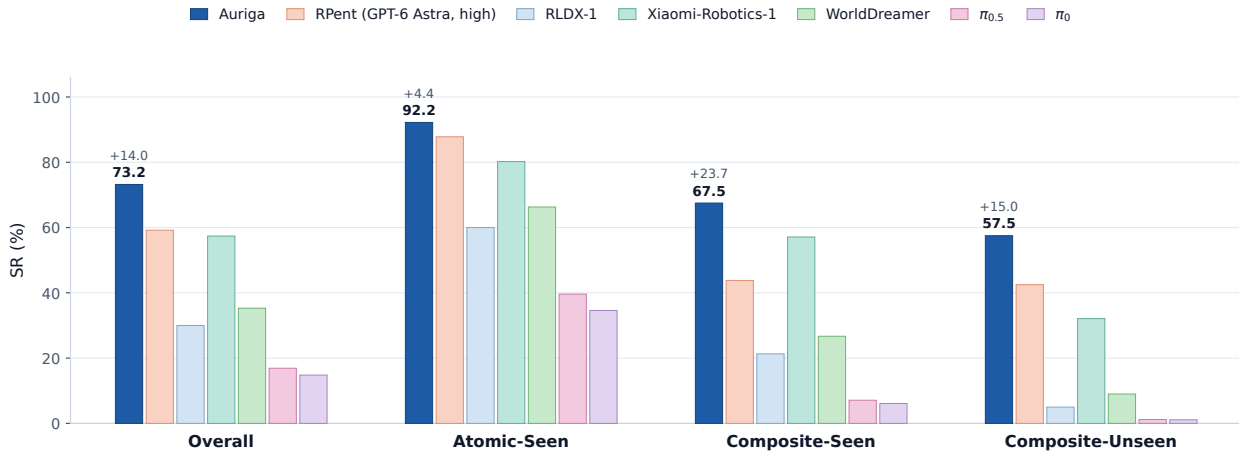


Figure 18 Success rates (%) on RoboCasa365 by group: Auriga, RPent, and RLDX-1 from [Tab. 5](#) (target kitchens, 250 episodes) and the leaderboard entries of [Tab. 15](#) (pretraining kitchens), with Auriga first. In each group, the bar of Auriga shows its rate and, above it, its difference from the highest other method of [Tab. 5](#); another method’s bar is labeled with its rate only where it reaches that of Auriga ([Tabs. 5 and 15](#) list all values).

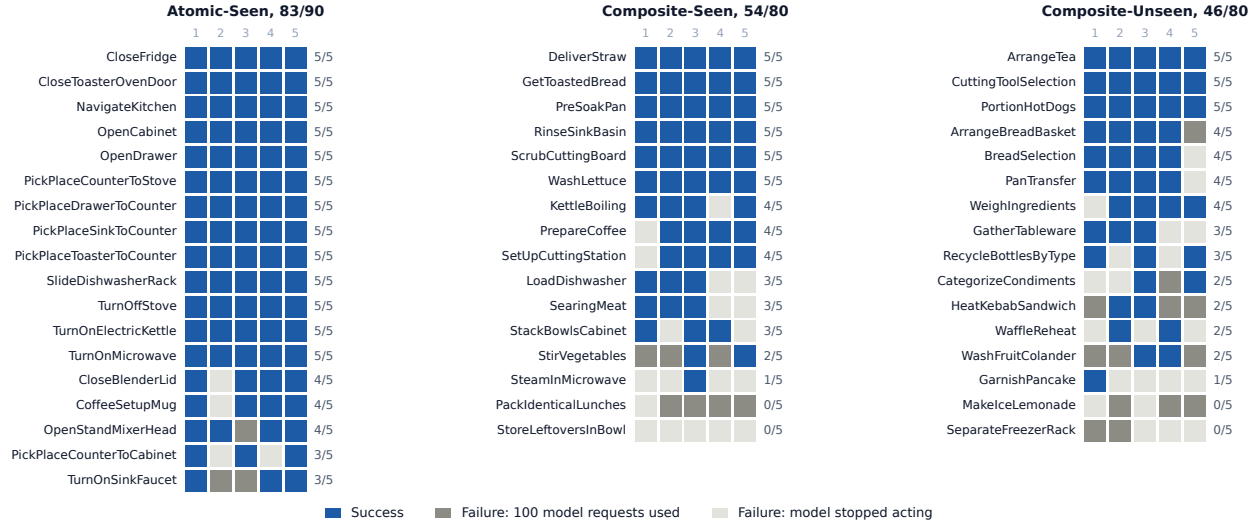


Figure 19 Outcome of every episode of Auriga on the 250 Target50 episodes, by task and seed: success, failure after the 100 model requests of the episode were used, and failure after the model stopped acting for several turns. Within each group, tasks are ordered by their number of successes.

D Additional Results

Tab. 16 lists the episode cost of Auriga by dimension corresponding to Fig. 11. Across the 6,299 episodes in Tab. 16, successful episodes (3,405 episodes) complete efficiently once the task subgoals are verified, averaging 30.3 model calls, 3.10 million input tokens, 11.5 minutes, and 65.6% of the native-step limit without ever exhausting the step budget. The remaining 2,894 episodes, which continue online recovery attempts until the budget expires, average 67.6 model calls, 8.61 million input tokens, 27.5 minutes, and 96.4% of the native-step limit, accounting for 65.5% of all model calls and 70.2% of all input tokens. Tab. 17 lists the SR and Score on the standard and random scenes of the 12 Generalization tasks together with the change between the two scene types (with the SR change and G plotted in Fig. 9), and Fig. 20 plots the Score and SR of the methods in Fig. 8 and of RoboProbe as bars.

Table 16 Cost of an Auriga episode by dimension across the 6,299 evaluated episodes: episodes, mean model calls, input tokens and their cached share, output tokens, the share of the native-step limit used, and wall-clock time.

Dimension	Episodes	Model calls	Input (M)	Cached (%)	Output (k)	Steps used (%)	Time (min)
Generalization	1,800	54.2	6.57	84.8	21.9	79.6	21.6
Precision	1,200	50.2	6.02	85.9	19.5	86.9	20.2
Long-Horizon	1,199	54.5	6.70	85.2	21.8	88.8	23.0
Memory	900	30.0	3.11	85.5	9.0	68.9	11.3
Open	1,200	40.7	4.67	86.1	15.4	71.8	14.9
All	6,299	47.5	5.63	85.4	18.3	79.7	18.8

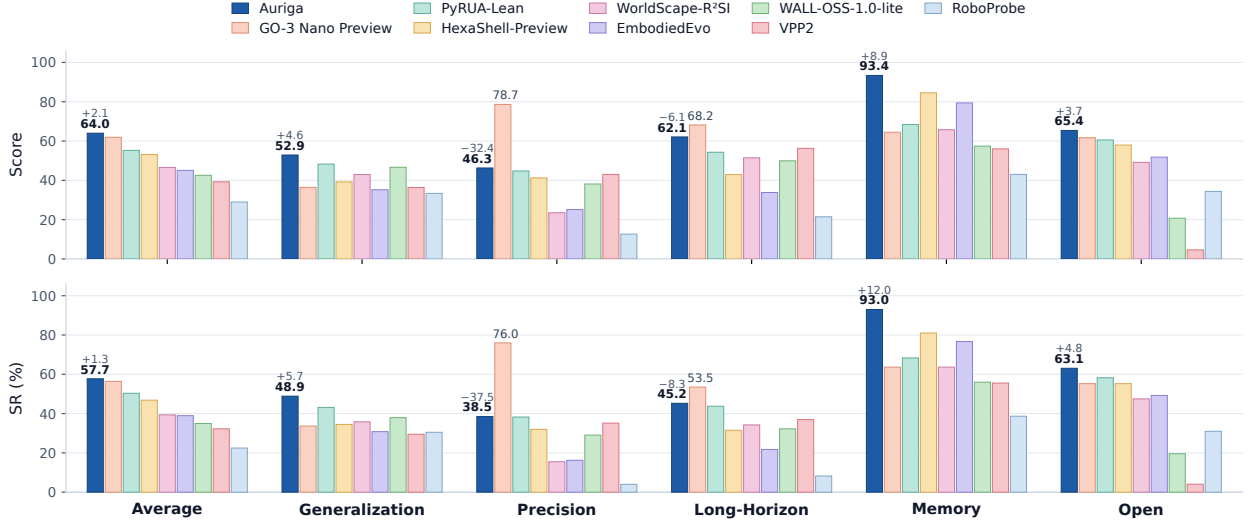


Figure 20 Score (top) and SR in % (bottom) of Auriga, the seven other methods of Fig. 8, and RoboProbe, on average and by dimension. Auriga is the first bar of each group, followed by the other methods in order of average Score. Above the bar of Auriga are its value and, higher up in grey, its difference from the highest of the other eight methods, whose value is given above its own bar where it is higher than Auriga.

Table 17 Generalization under scene randomization on the 12 Generalization tasks for the entries of Tab. 4: SR (%) and Score on the standard and random scenes, with the change (random – standard) and relative SR change. Above Auriga, entries are ordered by ascending composite generalization score G (Fig. 9). Bold: best entry per column.

Method	SR (%)				Score		
	Standard	Random	Change	Rel. change	Standard	Random	Change
PhysicalRSI	25.67	5.56	−20.1	−78%	31.57	11.04	−20.5
VPP2	46.44	12.56	−33.9	−73%	52.14	20.63	−31.5
Awomo-0.5	30.89	16.78	−14.1	−46%	37.70	22.92	−14.8
Rex-M1 Preview	34.78	17.33	−17.5	−50%	42.64	25.46	−17.2
GO-3 Nano Preview	43.00	24.33	−18.7	−43%	45.93	26.97	−19.0
Simate-beta	33.22	22.67	−10.6	−32%	40.54	29.63	−10.9
RoboProbe	32.67	28.33	−4.3	−13%	35.32	31.40	−3.9
HexaShell-Preview	38.00	31.00	−7.0	−18%	43.28	35.15	−8.1
WALL-OSS-1.0-lite	45.44	30.44	−15.0	−33%	53.44	39.88	−13.6
EmbodiedEvo	30.33	31.33	+1.0	+3%	34.55	35.88	+1.3
WorldScape-R ² SI	40.33	31.33	−9.0	−22%	46.62	39.28	−7.3
PyRUA-Lean	45.33	41.00	−4.3	−10%	50.47	46.07	−4.4
Auriga	52.28	45.61	−6.7	−13%	56.22	49.58	−6.6

E Per-Task Results

Fig. 21 compares Auriga with the four other agent methods of Tab. 4 (PyRUA-Lean, HexaShell-Preview, EmbodiedEvo, and RoboProbe) task by task in Score and in SR, and Tab. 18 lists the corresponding per-task Score and SR values.

		Score					SR (%)				
		Auriga	PyRUA-Lean	HexaShell-Preview	EmbodiedEvo	RoboProbe	Auriga	PyRUA-Lean	HexaShell-Preview	EmbodiedEvo	RoboProbe
Tasks: Auriga higher / tied / lower		24 / 1 / 17	31 / 1 / 10	34 / 1 / 7	41 / 0 / 1		22 / 4 / 16	28 / 5 / 9	32 / 4 / 6	38 / 3 / 1	
Generalization											
	stack_bowls	92.5	+18.6	+5.6	+32.6	+26.0	91.3	+21.3	+5.3	+35.3	+29.3
	push_T	86.0	0	-10.1	0	+26.0	86.0	0	-10.0	0	+26.0
	pack_objects_into_box	27.3	-18.7	-5.8	+13.2	+15.7	14.7	-15.3	-1.3	+14.7	+14.7
	fold_clothes	74.5	+15.3	+26.9	-10.7	-1.9	69.3	+17.3	+29.3	-12.7	-2.7
	hang_mugs	19.5	-8.2	+0.5	+12.4	+18.6	4.0	-8.0	0	+4.0	+4.0
	sweep_blocks	40.7	+34.7	+32.7	+38.7	+36.7	40.7	+34.7	+32.7	+38.7	+36.7
	pour_liquid_into_cup	7.3	+3.3	+5.3	-18.7	+3.3	7.3	+3.3	+5.3	-18.7	+3.3
	make_toast	28.7	+7.2	+27.2	+23.7	+28.2	26.0	+12.0	+26.0	+22.0	+26.0
	arrange_largest_number	90.9	+10.8	+60.7	+42.6	+22.2	88.0	+14.0	+66.0	+56.0	+28.0
	sort_nesting_dolls_by_size	61.3	-8.7	+9.3	+47.3	+43.3	61.3	-8.7	+9.3	+47.3	+43.3
	store_laptop_and_headphones	7.1	+2.3	+2.3	+4.3	+5.5	0.0	0	0	0	0
	stack_blocks	98.3	-1.7	+8.8	+26.1	+10.2	98.0	-2.0	+10.0	+30.0	+12.0
Precision											
	fasten_screws	80.7	+0.1	+50.9	+25.1	+44.3	69.3	+1.3	+51.3	+31.3	+53.3
	plug_in_charger	20.7	+4.7	-11.3	+14.7	+20.7	20.7	+4.7	-11.3	+14.7	+20.7
	insert_tubes	61.3	-5.1	+20.5	+15.3	+55.3	46.0	-4.0	+24.0	+14.0	+46.0
	pour_balls_into_vase	38.0	-28.0	-42.0	+16.0	+34.0	38.0	-28.0	-42.0	+16.0	+34.0
	play_Xylophone	77.3	+15.3	+19.3	+67.3	+69.3	77.3	+15.3	+19.3	+67.3	+69.3
	deposit_coin	42.5	-1.5	+16.1	+4.9	+26.5	32.0	-8.0	+16.0	+10.0	+30.0
	insert_key	14.8	+14.8	-0.2	-0.2	+0.4	0.0	0	0	0	0
	build_tower	35.2	+12.0	-12.8	+26.2	+18.8	24.7	+20.7	-5.3	+24.7	+22.7
Long-Horizon											
	put_bottles_into_dustbin	89.3	-2.9	+18.8	+18.1	+54.0	82.7	-5.3	+24.7	+26.7	+72.7
	fill_pen_holder	43.0	+19.5	+36.9	+42.1	+38.0	15.3	+7.3	+15.3	+15.3	+15.3
	classify_objects	97.7	+0.6	+12.4	+62.2	+28.7	96.7	+0.7	+18.7	+80.7	+40.7
	play_tic_tac_toe	95.0	-2.2	-4.0	-5.0	+84.3	83.9	-10.1	-12.1	-16.1	+83.9
	fill_egg_holder	11.1	-10.1	+2.3	-4.7	+5.6	0.7	-5.3	+0.7	+0.7	+0.7
	organize_table	69.0	+44.0	+55.0	+34.0	+29.5	19.3	+15.3	+19.3	+17.3	+19.3
	make_kong	27.3	+27.3	+25.3	+27.3	+27.3	27.3	+27.3	+25.3	+27.3	+27.3
	play_stacking_toy	64.7	-13.5	+7.1	+52.5	+58.1	36.0	-18.0	+18.0	+36.0	+36.0
Memory											
	cover_blocks	84.6	-15.4	-15.4	+0.6	+35.3	83.3	-16.7	-16.7	-0.7	+37.3
	match_and_pick_from_conveyor	96.7	+10.7	+22.7	+36.7	+34.7	96.7	+10.7	+22.7	+36.7	+34.7
	swap_blocks	96.7	+56.7	+18.7	+16.7	+96.7	96.7	+56.7	+18.7	+16.7	+96.7
	swap_T	91.3	-0.7	-4.7	+3.3	+73.3	91.3	-0.7	-4.7	+3.3	+73.3
	press_by_number	96.0	+96.0	0	+8.0	+26.0	95.9	+95.9	-0.1	+7.9	+25.9
	imitate_sorting_sequence	95.3	+2.9	+32.1	+19.5	+36.4	94.0	+2.0	+52.0	+34.0	+58.0
Open											
	align_blocks	90.7	+14.7	+12.7	-7.3	+40.7	90.7	+14.7	+12.7	-7.3	+40.7
	general_pickup	93.3	+7.3	+11.3	+9.3	+9.3	93.3	+7.3	+11.3	+9.3	+9.3
	stack_blocks_by_language	96.8	-3.2	+0.4	+45.2	+48.8	96.0	-4.0	0	+46.0	+56.0
	solve_equation	87.3	+15.3	+1.3	+17.3	+47.3	87.3	+15.3	+1.3	+17.3	+47.3
	classify_objects_by_language	92.1	-2.5	+12.7	+34.1	+46.1	88.7	-3.3	+16.7	+48.7	+58.7
	pick_from_conveyor_by_image	44.7	+22.7	+16.7	-7.3	+40.7	44.7	+22.7	+16.7	-7.3	+40.7
	store_tools_in_toolbox	13.3	-2.2	-0.7	+12.8	+10.8	0.0	0	0	0	0
	pour_by_language	4.9	-13.5	+4.9	+4.5	+4.5	4.0	-14.0	+4.0	+4.0	+4.0

Auriga lower by >30 10-30 0-10 tied 0-10 10-30 >30 Auriga higher by

Figure 21 Auriga against PyRUA-Lean, HexaShell-Preview, EmbodiedEvo, and RoboProbe on each task, in Score (left) and SR in % (right). In each panel, the first column gives the value of Auriga and each remaining column gives the difference (Auriga minus that method): blue where Auriga is higher, red where it is lower, and grey where the two are tied, darker for larger differences. The top row counts the tasks on which Auriga is higher, tied, and lower, with a bar of their shares.

Table 18 Per-task results of Auriga (SR in % and Score) and per-task SR (%) of the four other agent methods in Tab. 4 (Sec. 6.1.1). Bold: highest SR of a task (none where all five are at 0.0).

Task	Auriga		SR (%) of the four other agent methods			
	SR	Score	PyRUA-Lean	HexaShell-Preview	EmbodiedEvo	RoboProbe
<i>Generalization</i>						
stack_bowls	91.3	92.5	70.0	86.0	56.0	62.0
push_T	86.0	86.0	86.0	96.0	86.0	60.0
pack_objects_into_box	14.7	27.3	30.0	16.0	0.0	0.0
fold_clothes	69.3	74.5	52.0	40.0	82.0	72.0
hang_mugs	4.0	19.5	12.0	4.0	0.0	0.0
sweep_blocks	40.7	40.7	6.0	8.0	2.0	4.0

Continued on the next page

Table 18 Per-task results (continued).

Task	Auriga		SR (%) of the four other agent methods			
	SR	Score	PyRUA- Lean	HexaShell- Preview	EmbodiedEvo	RoboProbe
<code>pour_liquid_into_cup</code>	7.3	7.3	4.0	2.0	26.0	4.0
<code>make_toast</code>	26.0	28.7	14.0	0.0	4.0	0.0
<code>arrange_largest_number</code>	88.0	90.9	74.0	22.0	32.0	60.0
<code>sort_nesting_dolls_by_size</code>	61.3	61.3	70.0	52.0	14.0	18.0
<code>store_laptop_and_headphones</code>	0.0	7.1	0.0	0.0	0.0	0.0
<code>stack_blocks</code>	98.0	98.3	100.0	88.0	68.0	86.0
<i>Precision</i>						
<code>fasten_screws</code>	69.3	80.7	68.0	18.0	38.0	16.0
<code>plug_in_charger</code>	20.7	20.7	16.0	32.0	6.0	0.0
<code>insert_tubes</code>	46.0	61.3	50.0	22.0	32.0	0.0
<code>pour_balls_into_vase</code>	38.0	38.0	66.0	80.0	22.0	4.0
<code>play_Xylophone</code>	77.3	77.3	62.0	58.0	10.0	8.0
<code>deposit_coin</code>	32.0	42.5	40.0	16.0	22.0	2.0
<code>insert_key</code>	0.0	14.8	0.0	0.0	0.0	0.0
<code>build_tower</code>	24.7	35.2	4.0	30.0	0.0	2.0
<i>Long-Horizon</i>						
<code>put_bottles_into_dustbin</code>	82.7	89.3	88.0	58.0	56.0	10.0
<code>fill_pen_holder</code>	15.3	43.0	8.0	0.0	0.0	0.0
<code>classify_objects</code>	96.7	97.7	96.0	78.0	16.0	56.0
<code>play_tic_tac_toe</code>	83.9	95.0	94.0	96.0	100.0	0.0
<code>fill_egg_holder</code>	0.7	11.1	6.0	0.0	0.0	0.0
<code>organize_table</code>	19.3	69.0	4.0	0.0	2.0	0.0
<code>make_kong</code>	27.3	27.3	0.0	2.0	0.0	0.0
<code>play_stacking_toy</code>	36.0	64.7	54.0	18.0	0.0	0.0
<i>Memory</i>						
<code>cover_blocks</code>	83.3	84.6	100.0	100.0	84.0	46.0
<code>match_and_pick_from_conveyor</code>	96.7	96.7	86.0	74.0	60.0	62.0
<code>swap_blocks</code>	96.7	96.7	40.0	78.0	80.0	0.0
<code>swap_T</code>	91.3	91.3	92.0	96.0	88.0	18.0
<code>press_by_number</code>	95.9	96.0	0.0	96.0	88.0	70.0
<code>imitate_sorting_sequence</code>	94.0	95.3	92.0	42.0	60.0	36.0
<i>Open</i>						
<code>align_blocks</code>	90.7	90.7	76.0	78.0	98.0	50.0
<code>general_pickup</code>	93.3	93.3	86.0	82.0	84.0	84.0
<code>stack_blocks_by_language</code>	96.0	96.8	100.0	96.0	50.0	40.0
<code>solve_equation</code>	87.3	87.3	72.0	86.0	70.0	40.0
<code>classify_objects_by_language</code>	88.7	92.1	92.0	72.0	40.0	30.0
<code>pick_from_conveyor_by_image</code>	44.7	44.7	22.0	28.0	52.0	4.0
<code>store_tools_in_toolbox</code>	0.0	13.3	0.0	0.0	0.0	0.0
<code>pour_by_language</code>	4.0	4.9	18.0	0.0	0.0	0.0

F Extended Execution Trace Analysis

We examine `swap_T`, a pose-exchange task in the Memory dimension, to study how a remembered objective guides manipulation after the scene changes (complementing the matched-scene baseline comparison in Sec. 6.5 and Tab. 12). The robot must exchange two T-shaped blocks so that each occupies the other’s initial position and orientation. This task combines three demands in a compact setting: preserving a spatial reference after its source object moves, coordinating access to a shared workspace, and correcting the object pose when grasping or release changes it. The asymmetric T shape makes orientation part of the goal, so exchanging positions alone is insufficient. We analyze the 15-scene standard-variant diagnostic set for this task, using their task-belief records, action specifications, execution receipts, and evaluator outcomes.

Preserving object goals across changes in execution. For analysis, let $\hat{T}_{A,0}$ and $\hat{T}_{B,0}$ denote the two initial planar poses comprising planar position and yaw estimated from the calibrated images. The exchange

assigns the object goals

$$G_A = \hat{T}_{B,0}, \quad G_B = \hat{T}_{A,0}. \quad (5)$$

These goals remain fixed when the model changes a grasp or assigns the remaining transfer to another arm. The current object location and its relation to the gripper, however, must be updated from subsequent observations. This distinction matters after slippage: re-localizing the fallen block supplies a new pickup pose, while the saved initial pose still supplies its destination. The recorded `task_plan` separates these roles through the plan, attention, and ledger described in [Sec. 3.3](#). Its updates retain the original placement requirement while revising which physical conditions are supported and which remain unresolved.

A completed motion can still require a task-level correction. [Tab. 19](#) follows a successful exchange in which the pink block has already reached the blue block’s initial pose. During the remaining right-arm transfer, the blue block slips onto the table. The crossing stage reports `completed`, but its grip feedback records a zero final aperture, and the returned image places the block to the left of its target. The next belief revision marks blue retention as `contradicted` while retaining the pink placement as `supported`. The model then selects the nearby left arm, re-localizes the blue stem, and specifies a short, slow transfer to the original target. Thus the correction changes the arm, pickup coordinates, and transport motion together, while preserving both the completed placement and the outstanding object goal.

Table 19 Task state, execution feedback, and correction in a successful T-block exchange. Entries condense the recorded task belief and actions; steps are cumulative native actions from receipt budget counters. The destination remains the opposite block’s initial pose throughout.

Step	Recorded state and feedback	Decision or verified outcome
72	Both blocks are lifted; the belief records retention as <code>supported</code> and the exchange as <code>pending</code> .	Preserve both initial outlines; place pink first and inspect its release.
169	Pink has been released at blue’s initial pose; blue remains to be transferred.	Keep the completed placement and move blue with the right arm.
278	The transfer motion completes, but the final aperture is zero and the image shows blue on the table.	Mark blue retention as <code>contradicted</code> ; preserve pink as <code>supported</code> ; switch recovery to the left arm.
332	The left-arm re-grasp and slow lift complete; the right arm returns.	Use a short, slow placement at the saved target, followed by release and withdrawal.
380	The evaluator reports task success within the 400-step budget.	Both object goals are satisfied after the local recovery.

Recovery preserves completed work. The two ledger entries distinguish an unresolved blue transfer from an already supported pink placement. Consequently, the model does not repeat the entire exchange or derive a new destination from the displaced scene. It repairs the remaining relation between one object and its saved goal. This is the operational value of maintaining task state: evidence can invalidate a local assumption without erasing the broader objective or the progress already achieved. It also explains why counting only `tracking_error` receipts would miss an important class of recovery. Here, the discrepancy lies between successful robot motion and unsuccessful object transport, and is exposed by combining the receipt with the resulting observation.

Using feedback to revise feasible operations. [Tab. 20](#) summarizes mechanism use across the 15 episodes. Geometric queries provide coordinates for pickup and placement; belief updates retain the goal and revise progress; previews evaluate candidate operations before execution. All episodes use each of these capabilities. Their roles become concrete in the preceding trace: the saved target survives the transfer failure, a fresh localization supplies the new grasp, and the revised action specification changes which arm performs the remaining work. The table counts usage at the interface where it occurs, including calls made through model-written programs.

Table 20 Mechanism use in the 15 T-block exchange episodes. Belief updates exclude the initial submission. Preview evaluations include unsuccessful candidates; generated plans are the candidates returned with a plan identifier. Plan and guard counts refer to specifications, including revisions. Observation pauses count actual receipt-reported stops.

Mechanism	Interface	Episodes	Count and unit
Task-belief updates	task_plan	15/15	25 updates
Geometric queries	geometry_query	15/15	17 calls
Preview evaluations	robot_action	15/15	106 evaluations
Generated action plans	robot_action	15/15	74 plans
Multi-stage plans	move, gripper, observe	15/15	73 plans
Plans involving both arms	arm	15/15	54 plans
Grasp-guard specifications	grip_guard_preset [†]	15/15	70 stages
Observation pauses	observation_checkpoint [‡]	1/15	1 pause

[†]Includes explicit aperture bounds as well as guard presets. [‡]The receipt reason identifying an observation pause; a pause returns control for inspection and does not itself imply an error.

Coordination is expressed at the scale of an operation. Multi-stage plans account for 73 of the 74 generated plans. The sequence can include pickup, lift, transfer, release, and withdrawal, allowing the model to consider the entire operation before committing to it. The 54 plans involving both arms include staggered motion and one-arm holding while the other places an object. This is well suited to pose exchange: both destinations are initially occupied, and the arms need access to the same central region. The recorded plans repeatedly resolve this dependency by holding one block clear, delivering the other, and then completing the exchange. The relevant coordination is the ordering of access and the preservation of held objects, which can be expressed within one plan even when the arms move at different times.

Preview makes correction cost available before motion. A second successful exchange shows how the model uses the remaining budget when a late observation contradicts placement. After 345 native steps, the blue block is correctly placed, but the red block’s stem points in the wrong direction. The next belief revision preserves the required destination and calls for a right-arm re-grasp and approximately 94-degree clockwise correction. [Tab. 21](#) records the subsequent candidate costs. With 55 steps remaining, the model shortens the route and coordinates gripper motion with the approach and return. The executed specification also contains an explicit release stage. Task success is reached at step 396. The useful change is therefore more specific than issuing another grasp: the correction is reorganized to include the outstanding orientation change, release, and return within the available budget.

Table 21 Recorded preview costs during the late orientation correction. The world remains at native step 345 during these previews, with 55 steps available. The first two candidates are rejected because their planned lengths exceed the remaining 55-step native budget. Candidate rows give planned lengths; the last row gives steps actually used before the evaluator signals success.

Candidate	Native steps	Status
Initial correction	95	Rejected: insufficient budget
Revised route	57	Rejected: insufficient budget
Coordinated approach and return	54	Planned
Final plan with explicit release	55	Planned
Executed correction to task success	51	Success at step 396

The 106 preview evaluations and 74 generated plans in [Tab. 20](#) reflect this distinction between evaluating a candidate and obtaining a plan for execution. The remaining 32 evaluations report unsuccessful candidates without generating such a plan. Likewise, the observation pause provides an opportunity to inspect intermediate results; it is counted separately from grasp-guard specifications and execution errors. The two traces show

how deliberation and physical execution interact: one revises the physical route after object loss, while the other revises the cost and timing of a correction before moving. Together with the matched-scene comparison against RoboProbe in Tab. 12 (Sec. 6.5), these records connect task-level objectives to concrete physical operations.

G Extended Case Studies

This appendix provides the extended RoboDojo and RoboCasa365 case analyses accompanying Sec. 6.6: additional recovery episodes after slippage (Figs. 22 and 23), replanning episodes after contact and execution disruptions (Figs. 24 and 25), and detailed step-by-step budget accounting for the failure episodes (Tabs. 22 and 23).

Additional slippage recovery episodes. Alongside sequence reproduction and lettuce washing (Fig. 14 in Sec. 6.6), Figs. 22 and 23 show how the model separates acquisition, retention during transport, and post-release support on conveyor picking, bowl stacking, and drawer-to-counter transfer.

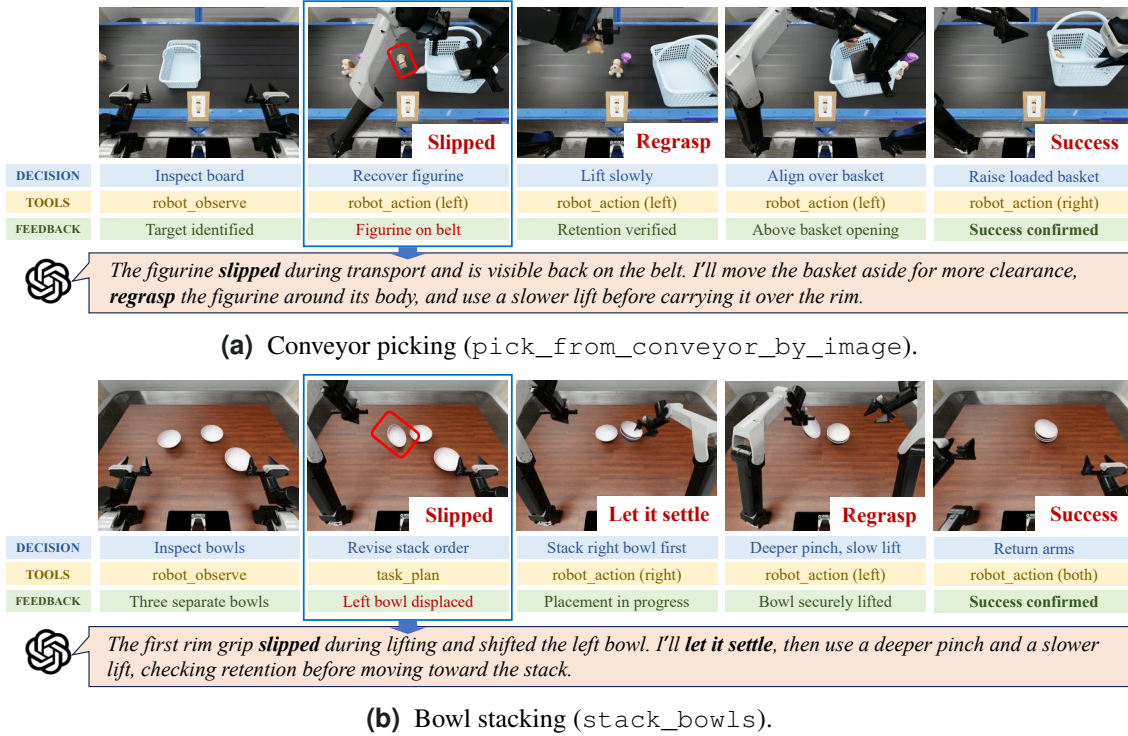


Figure 22 Additional RoboDojo episodes recovering from grasp failure and slippage. The model changes the grasped region for the figurine and uses a deeper pinch and slower lift for the bowl.

Conveyor picking: separating acquisition, retention, and destination access. The model must match a figurine to the board image, place it inside the basket, and lift the basket by more than 8 cm. The moving conveyor first invalidates a basket pickup location, prompting a switch to right-arm interception. For the figurine, an approach contacts the hat before closure; a later empty closure leaves the body exposed. A torso grasp then acquires the object, but the figurine slips during transport (Fig. 22a). The model consequently changes both the grasp and the transfer conditions: it moves the basket aside, re-grasps across the torso, and verifies retention during a slower lift.

Once the figurine is retained, the remaining obstacle is the destination. Moving the basket underneath it causes contact with the image board and tilts the opening. The model keeps the figurine held, moves the

basket upward and away from the board, and then completes the release. A later check against the initial image revises the height judgment because the earlier estimate followed a rim on a tilted basket. The loaded basket is raised farther before the evaluator signals success. These successive revisions address distinct predicates (target acquired, target retained, opening accessible, and required basket elevation), so progress on one predicate can be preserved while another is corrected.

Bowl stacking: changing the assembly order after a failed lift. The initial plan uses the rear middle bowl as the base, adds the left bowl, and then adds the right bowl. The first rim grip slips and displaces the left bowl. The revised plan lets that bowl settle while the robot approaches the right bowl, using a deeper pinch and slower lift (Fig. 22b). After the right bowl is nested into the base and remains supported after release, the model re-localizes the displaced left bowl and adds it last. Thus recovery changes the assembly order as well as the grasp.

This reordering is possible because the task constrains the final stack rather than the order in which bowls enter it. It also gives the displaced bowl time to settle before its next localization. The plan checks co-motion after lifting and upright support after release and withdrawal; the final return supplies another observation of whether the stack persists without gripper support. The evaluator confirms successful stacking. The distinction from sequence reproduction is consequential: the bowl order can change to simplify execution, whereas the observed object order must remain fixed.

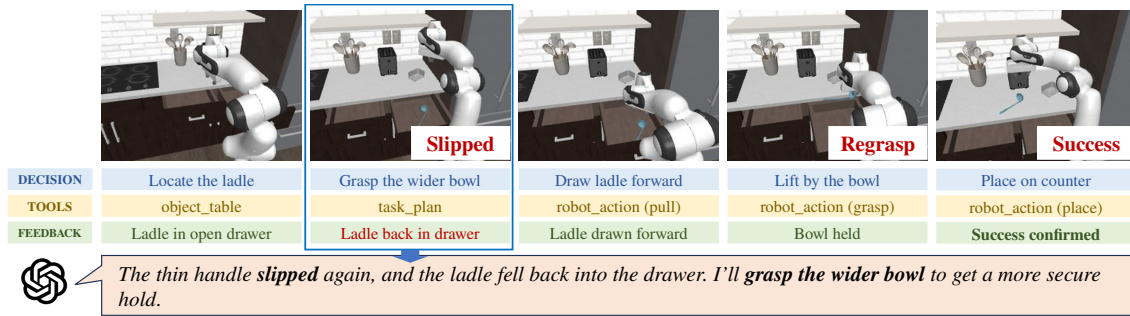


Figure 23 Recovery from grasp failure and slippage in RoboCasa365 drawer-to-counter transfer (PickPlaceDrawerToCounter). After grasps on the thin ladle handle slip during lifting and base motion, the model pulls the ladle forward and switches contact to the wider bowl.

Drawer-to-counter transfer: shifting contact from a thin handle to the wider bowl. The robot must take a ladle from an open drawer and place it on the counter, where the single task predicate tests only whether the ladle rests on the counter and remains false until final placement. An initial grasp on the thin handle slips during the lift, and after a lower re-grasp on a thicker section of the handle, the ladle slips again during mobile-base motion and falls back into the drawer (Fig. 23). Both losses are reported by the execution receipts rather than the task predicate, and after the second slip the object table no longer finds the ladle in the wrist-camera view.

The model revises its task plan to grasp the ladle's wider bowl instead, though roughly ten attempts are blocked by the counter overhang or miss inside the drawer while it pulls the drawer out farther. It then uses the handle to draw the ladle forward within the drawer and grasps the bowl once clear of the overhang. Transferring the ladle by its bowl and releasing it on the counter turns the placement predicate true, and the evaluator confirms task success. This revision corresponds to the acquisition condition at the start of Sec. 6.6: when a thin handle fails to sustain lift and base transport, recovery changes the contact geometry to a wider region of the object.

Additional contact and replanning episodes. Alongside object packing and burner shutoff (Fig. 15 in Sec. 6.6), Figs. 24 and 25 follow the model from a measured discrepancy to a procedure revision across conveyor picking, charger insertion, straw delivery, and cutting-station setup.

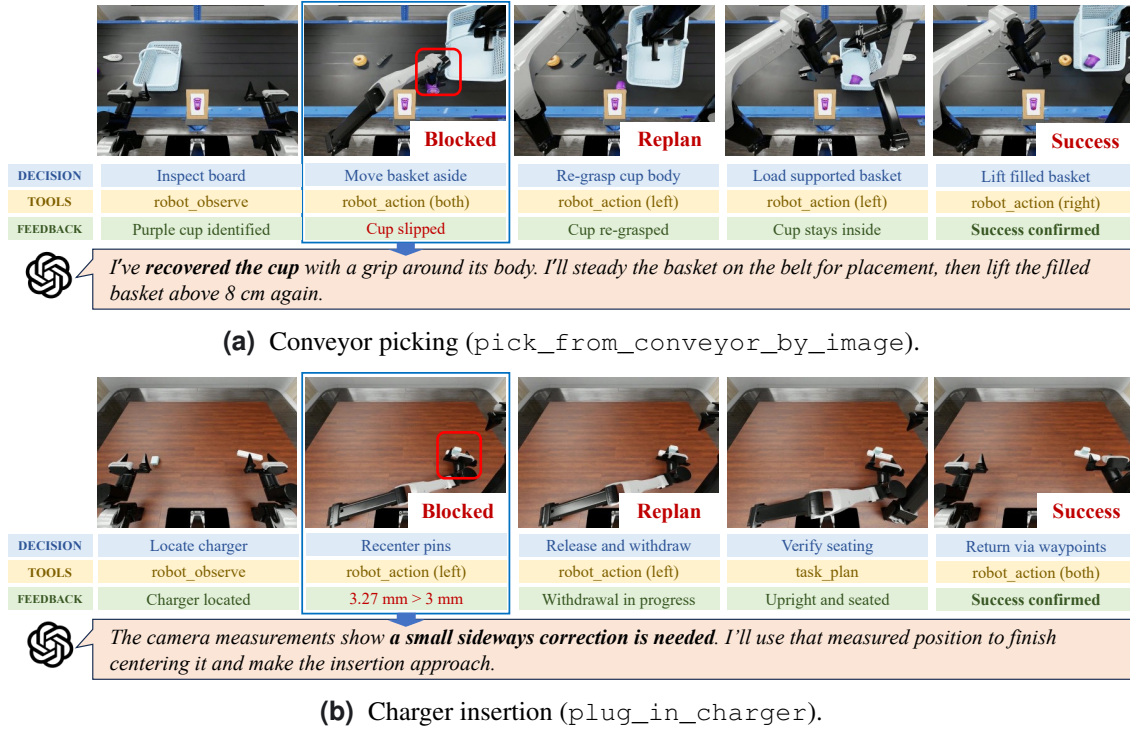


Figure 24 Additional RoboDojo episodes replanning after contact and execution disruptions. The model supports the basket during cup placement and corrects the charger’s alignment before insertion.

Conveyor picking: using support to decouple placement from basket motion. The cup initially remains held while the other arm raises the basket. During approach, the basket swings and the cup shifts in the left grip; a wrist-orientation change then loses the cup. After recovering it with a body grasp, the model preserves that wrist orientation and changes the basket procedure (Fig. 24a). The revised plan lowers the basket onto the belt, loads the cup, and then lifts the filled basket above the required height.

Supporting the basket changes the manipulation problem. Insertion can now be organized around a supported opening, while lifting becomes a separate operation on the loaded container. When a side adjustment loses the handle grip, the plan retains the cup grasp, clears the right arm for placement, and postpones handle acquisition until the cup has been released. The model then verifies that the cup stays inside after left-arm withdrawal, re-grasps the handle from above, and lifts the basket to complete the task. This sequence preserves the final object-in-basket and elevation requirements while removing the need to maintain a suspended basket throughout insertion.

Charger insertion: revising only the part of the procedure contradicted by evidence. The initial plan allows a table-supported handover if the left arm cannot reach the socket. A preview establishes a direct route, so the model keeps the charger in the left hand and moves the right arm outward to clear that route. Near the socket, the `clear_pins` clearance lift returns `tracking_error`: the position error is 3.27 mm, predominantly vertical, exceeding the stage’s tight 3 mm tolerance. The charger remains held, and camera measurements identify a small lateral alignment correction (Fig. 24b). The next action uses the returned position to center the pins and complete the insertion, followed by release and a check that the charger stays seated.

The later return-path failure calls for a different revision. Because seating remains visually supported, the model leaves the inserted charger undisturbed and splits the right-arm return into shorter movements through clear intermediate poses. The evaluator then reports success. This separates a local alignment correction from a route-planning problem: one changes the target relative to the socket, while the other changes how the arm

reaches its home pose. Neither requires repeating the already verified insertion.

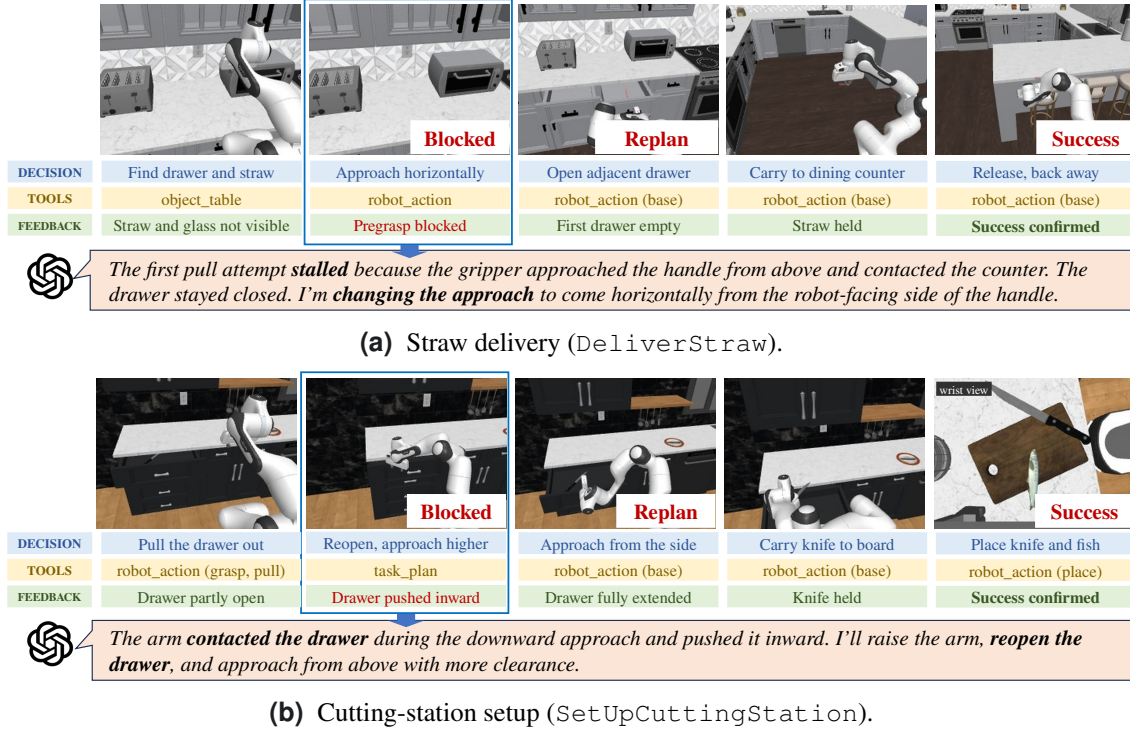


Figure 25 Additional RoboCasa365 episodes replanning after contact and execution disruptions. The model changes the approach to the drawer handle and then switches to the adjacent drawer when the first is empty (DeliverStraw), and reopens a drawer pushed inward by arm contact before approaching from the side (SetUpCuttingStation).

Straw delivery: matching the scope of revision to the invalidated assumption. The task requires taking a straw from a drawer to a glass cup on the dining counter; initially, the object table locates the drawer handles while neither the straw nor the cup is visible. Repeated attempts to open the first drawer return execution receipts for a blocked or stalled approach (Fig. 25a), as approaching the handle from above brings the gripper against the counter. The model switches to a horizontal approach from the robot-facing side of the handle, backs the mobile base away for clearance, and writes that direction into the grasp stage before pulling the drawer open with the base.

Inspecting the opened drawer reveals a different discrepancy when the object table finds no straw inside. The task plan records that this drawer is empty while the adjacent front drawer remains unchecked, and the robot opens that second drawer using the same horizontal handle approach and base pull. After grasping the straw and driving around the counter end, it rights the glass after a missed first insertion, places the straw inside to satisfy the straw-in-cup predicate, and backs away as the evaluator confirms success. As in charger insertion, the two disruptions call for different scopes of revision: a blocked grasp changes the approach route, whereas an empty drawer changes the target container without altering the delivery objective.

Cutting-station setup: restoring a displaced container before revising the approach. The task requires taking a knife from a drawer and a fish from a plate and placing both onto a cutting board. When the arm reaches downward for the knife, the execution receipt reports a blocked approach, and the camera view shows that arm contact has pushed the partly open drawer inward (Fig. 25b). The model initially plans to raise the arm, reopen the drawer, and approach from above with more clearance.

Across roughly ten subsequent attempts, however, passing the arm over the drawer front repeatedly pushes the drawer inward and leaves grasps blocked or empty. The model then pulls the drawer fully open, withdraws

the gripper clear of the handle before turning the wrist to approach from the side, and lowers its grasp slightly to place the finger pads around the knife handle. Delivering the knife satisfies the knife-on-board predicate, and after the fish is re-grasped following a slip during base motion, the evaluator signals task success during the final placement onto the board. As in object packing, the central revision treats the displaced drawer as state that must be restored and paired with a non-colliding approach route, rather than targeting coordinates measured before contact.

Detailed budget accounting for failure episodes. Tabs. 22 and 23 give the step-by-step native motion and model-request budget accounting for the failed `stack_bowls` and `HeatKebabSandwich` episodes shown in Fig. 17 (Sec. 6.6).

Table 22 Progress and remaining motion budget in the failed RoboDojo stacking episode (`stack_bowls`, Fig. 17a). Step counts come from execution accounting; state descriptions condense the observation-linked model records.

Steps used	Steps left	State guiding the next decision
242	558	Two bowls remain nested after withdrawal; add the third.
440	360	The third bowl remains tilted after seating attempts; re-grasp and level it.
564	236	The nested stack was lifted and slipped; restore it using table support.
691	109	The stack has rolled onto its side again during the return; attempt a compact supported correction.
756	44	The rim pinch has not retained the stack; change contact to the raised base.
800	0	Native step limit reached; task failure, Score 15/100.

Table 23 Progress and remaining model budget in the failed RoboCasa365 episode (`HeatKebabSandwich`, Fig. 17b). Request counts are model responses out of the 100 allowed per episode; state descriptions condense the recorded model messages.

Requests used	Requests left	State guiding the next decision
14	86	The baguette has slipped onto the counter beside the open door; re-grasp it with an explicit approach direction.
41	59	The baguette is inside the oven but below the wire rack, so its rack-contact check is false; move the rack to expose the bread.
51	49	The kebab rests on the rack and its check passes; slide the loaded rack inward and recover the baguette from below.
63	37	The rack has not moved because the grip on its thin front rail was not secure; grasp the rail from a raised angle.
86	14	The rack has slid outward with the kebab, leaving the baguette behind it; lift the bread onto the rack.
97	3	The model reports the task unfinished: kebab on the rack, baguette below it, door open; the harness asks it to continue.
100	0	Model-request limit reached; task failure.

Coupled structures and local completion versus cumulative progress. In `stack_bowls` (Tab. 22), the plan at step 440 intends to lift and level the tilted top bowl, but the grasp lifts the entire nested stack, changing the recovery target from seating one bowl to restoring upright support for all three bowls. Although subsequent tipping maneuvers, supported pushes, and contact changes receive `completed` motion receipts, the structure rolls again upon arm withdrawal as the remaining budget drops from 360 steps to 44. Similarly, in `HeatKebabSandwich` (Tab. 23), after the baguette lands below the toaster wire rack at request 41, exposing the bread requires pulling the thin front rail of the rack, which repeatedly slips between requests 63 and 86; although placing the kebab at request 50 and sliding the rack outward at request 86 succeed locally, only 14 requests remain and the 100-request budget expires before the baguette is lifted onto the rack.