# ROBOICL: EMBODIED IN-CONTEXT LEARNING WITH GPT-6 ASTRA

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

In-context learning enables LLMs to adapt to new tasks from demonstrations without updating model parameters. We study whether the same principle can support robot control, where demonstrations must connect language and vision to continuous actions and subsequent decisions must account for execution outcomes. GPT-6 Astra provides a general-purpose multimodal starting point, but its zero-shot performance varies substantially across manipulation tasks. We introduce RoboICL, a framework for closed-loop control that combines a static prefix of compressed expert demonstrations with a bounded dynamic context of the robot's own interactions. Both contexts use the same observation–action call–execution receipt–next observation format, allowing the model to distinguish proposed actions from those actually executed. Bounded memory retains temporally distributed interactions and recent feedback while marking omitted intervals. At each decision round, GPT-6 Astra predicts Cartesian end-effector action blocks from the current observation and retained context, without task-specific model updates or an auxiliary learned action policy. In RoboDojo simulations, adding demonstrations is associated with gains on the evaluated shot-study tasks. Across three single-arm real-robot tasks, mean normalized progress rises from 14.45 at zero shot to 78.89 with three demonstrations. A separate exploratory study coupling JEV with RoboICL substantially improves inference speed on RoboDojo while preserving control capability. These results show that demonstrations and execution-grounded interaction history can serve as an effective inference-time interface for direct robot control.

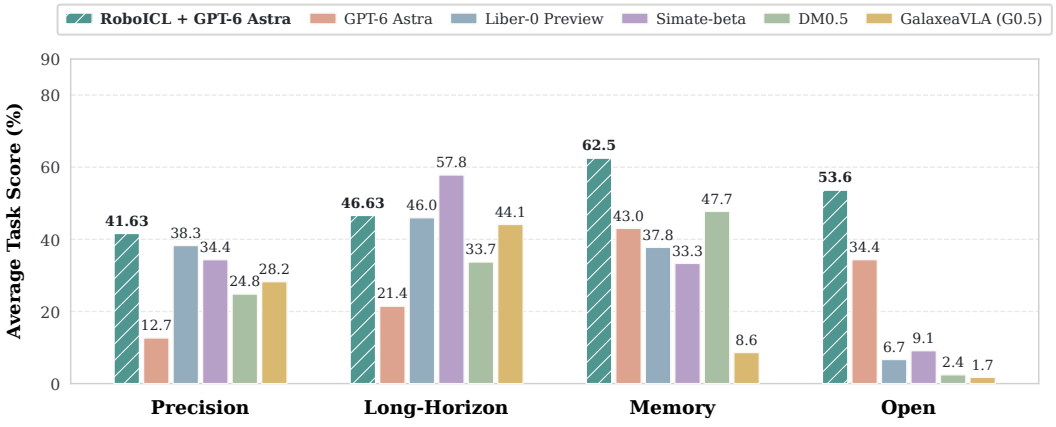


Figure 1: **RoboDojo category-level comparison.** Baseline scores are reproduced from the official RoboDojo (Zhang et al., 2026) leaderboard [1]. For category Open, which lacks released training demonstrations, RoboICL uses evolving online context only (zero-shot); other categories use full context, adding 1-shot demonstration [2]. In this preliminary cross-protocol comparison, RoboICL substantially improves on the official zero-shot GPT-6 Astra implementation, leads all other baselines on Precision, Memory and Open, and approaches second place on Long-Horizon.



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

# 1 INTRODUCTION

Large language models demonstrate strong in-context learning (ICL) on new tasks: given a few textual demonstrations, a fixed model can infer and follow the demonstrated task pattern without parameter updates (Brown et al., 2020). Extending this ability to embodied robot control is challenging because each decision must integrate language, images, proprioception, and continuous actions, while every command changes the scene, contact state, and robot configuration. Unlike language modeling, embodied control still lacks a broadly applicable foundation model that can be deployed across tasks without robotics-specific adaptation; many vision-language-action and world-action-model systems instead obtain their capabilities through substantial robot-specific training or post-training (Fu et al., 2024; Vosylius & Johns, 2025; Sridhar et al., 2025; Zhou et al., 2026; Generalist Team, 2026). Recent results from GPT-6 Astra suggest emerging zero-shot capabilities in 3D visual reasoning and direct robot control (Zhang et al., 2026). Nevertheless, its strongly task-dependent behavior on RoboDojo illustrates the remaining gap between general multimodal competence and embodied control (Chen et al., 2026b; Zhang et al., 2026). Reliable manipulation requires the model not only to understand a task, but also to infer action conventions, coordinate arm motion and gripper control, track task progress, and adapt when realized execution differs from the plan.

These limitations motivate transferring ICL from textual examples to closed-loop robot interaction. Existing context-conditioned robot-control approaches commonly realize deployment-time adaptation through demonstrations that convey task identity or desired behavior, including demonstration-conditioned imitation, policies trained to use context, and frozen language or vision-language models prompted with task examples (Duan et al., 2017; Fu et al., 2024; Vosylius & Johns, 2025; Sridhar et al., 2025; Di Palo & Johns, 2024; Yin et al., 2025; Cheng et al., 2026). Concurrent VLM-based robot-control systems further broaden the design space of context, action interfaces, and feedback (Chen et al., 2026c). Despite this progress, a demonstration-centered formulation captures only one source of embodied context. Physical execution continuously creates another: the model observes the consequences of its own actions, including progress, contact errors, controller interruptions, and states outside the demonstrations. Embodied ICL must therefore explain both how expert examples guide control and how experience emerging within the current episode conditions subsequent decisions.

To incorporate both sources of embodied experience, we introduce *RoboICL*, a framework for closed-loop robot control that combines an evolving *dynamic online context* with an optional *static prefix context* of compressed expert demonstrations. The two contexts play different temporal roles but follow the same interaction grammar: observation $\rightarrow$ `Act` call $\rightarrow$ execution feedback $\rightarrow$ next observation. The dynamic online context forms the zero-shot foundation by allowing each new decision to condition on a bounded record of earlier interactions from the same episode. In the few-shot setting, the static prefix context augments this evolving record with expert trajectories that provide task structure, action semantics, and procedural guidance. Few-shot control consequently extends interaction-conditioned adaptation rather than replacing it.

RoboICL uses a unified interface to present static and online interactions in the same format and to record the outcome of each predicted action block. At each decision round, a fixed multimodal model receives the task instruction, the optional static prefix context, the bounded dynamic online context, and the current visual and proprioceptive state before directly predicting a $15 \times 14$ matrix of bimanual Cartesian actions. Once the execution system processes this proposal, the interface records the interaction as an observation, an `Act` call, an execution receipt, and the resulting observation. The receipt distinguishes the proposed action block from the portion that the controller actually executes, preventing an interrupted or rejected suffix from entering context as an accomplished transition. Because an unbounded interaction history would make the request grow with episode length, the online memory combines temporally distributed anchors with the latest completed interaction and marks omitted intervals with explicit gaps. In the few-shot setting, compressed expert trajectories use the same transition format and remain fixed at the beginning of the request. Deterministic code validates and executes admissible commands without supplying learned action proposals, so deployment-time adaptation arises from the two contexts rather than an auxiliary learned robot policy. Before execution, static prefix context provides expert task knowledge; during execution,

[1] https://robodojo-benchmark.com/report/gpt-6-astra-eval

[2] The evaluation comprises 50 official layouts per task for Open and 20 for the other three categories.

dynamic online context supports progress tracking, error correction, and strategy revision from observed action outcomes. RoboICL therefore extends fixed demonstration prompting by conditioning each decision on prior demonstrations and an evolving interaction history.

We evaluate RoboICL in three dimensions: benchmark performance, physical-world applicability, and inference efficiency. On ten simulated RoboDojo tasks, one-shot RoboICL reaches a macro-average progress score of 62.00 versus 38.50 for the externally reported direct GPT-6 Astra configuration. The protocols differ, so this comparison provides context rather than a matched causal estimate. On a physical robot, expert demonstrations raise mean progress across three tasks from 14.45 at zero shot to 78.89 at three shots, including complex long-horizon manipulation. On simulated RoboDojo tasks, coupling RoboICL with JEV substantially improves inference speed while preserving control capability. These studies assess effectiveness, deployability, and efficiency.

Our contributions are threefold. First, we formulate RoboICL by combining dynamic online and static prefix contexts for fixed-parameter robot control. Second, we introduce a unified action–execution–observation format and deterministic bounded memory that preserve action outcomes and mark omitted trajectory spans. Third, we evaluate GPT-6 Astra with RoboICL in simulation, on a physical robot, and with JEV-based inference.

## 2 Related work

**Robot learning from context.** In-context and demonstration-conditioned robot policies adapt at deployment from sensorimotor examples. ICRT, Instant Policy, RICL, Zero-WAM, and GEN-1.5 obtain this capability through robot-specific training or large-scale physical-interaction pretraining (Fu et al., 2024; Vosylius & Johns, 2025; Sridhar et al., 2025; Zhou et al., 2026; Generalist Team, 2026). RoboICL instead elicits action blocks from a frozen general-purpose VLM without training a robot-specific policy or action head. Its expert demonstrations provide static prefix context, while execution during the current episode produces dynamic online context.

**Frozen models and interaction feedback.** Frozen language and vision-language models control robots through skills, programs, spatial plans, or numerical actions (Ahn et al., 2022; Liang et al., 2022; Huang et al., 2023; Di Palo & Johns, 2024; Yin et al., 2025). Recent systems extend these interfaces with demonstrations and closed-loop feedback (Chen et al., 2026c; Su et al., 2026; Cheng et al., 2026), while ReAct, Reflexion, Zeva, and Reflective VLA illustrate the broader role of feedback and memory in sequential decision making (Yao et al., 2023; Shinn et al., 2023; Chen et al., 2026a; Lian et al., 2026). RoboICL differs in its combination of short Cartesian-increment sequences, explicit execution receipts, and bounded retention of temporally distributed interactions. This design makes proposed actions, realized execution, and omitted trajectory intervals explicit within a common context interface. Appendix A provides an extended discussion of these connections and action-interface differences.

## 3 RoboICL

### 3.1 Framework overview

RoboICL integrates a dynamic online context with an optional static prefix context within a single context-conditioned control process. Let $t$ index model decision rounds, $g$ denote the task instruction, and $(o_t, z_t)$ denote the visual observation and robot proprioceptive state available at the beginning of round $t$. The dynamic online context $\mathcal{M}_t$ contains selected interactions accumulated before round $t$, whereas the static prefix context $\mathcal{D}_K = \{D^{(k)}\}_{k=1}^{K}$ contains $K$ expert demonstrations and remains unchanged throughout the episode. The zero-shot setting uses $K = 0$ and builds context from online interaction, while the few-shot setting uses $K > 0$ and places the expert demonstrations before the same evolving online context. A fixed-parameter multimodal model $\pi_\theta$, with parameters $\theta$, receives the serialized request $Q_t$ and predicts an action block $A_t$:

$$Q_t = \mathrm{Serialize}(g, \mathcal{D}_K, \mathcal{M}_t, o_t, z_t), \qquad A_t \sim \pi_\theta(\cdot \mid Q_t), \quad A_t \in \mathbb{R}^{H\times 14}. \tag{1}$$

Because $\theta$ remains fixed throughout deployment, all task adaptation arises from the static prefix context and dynamic online context rather than from gradient updates.

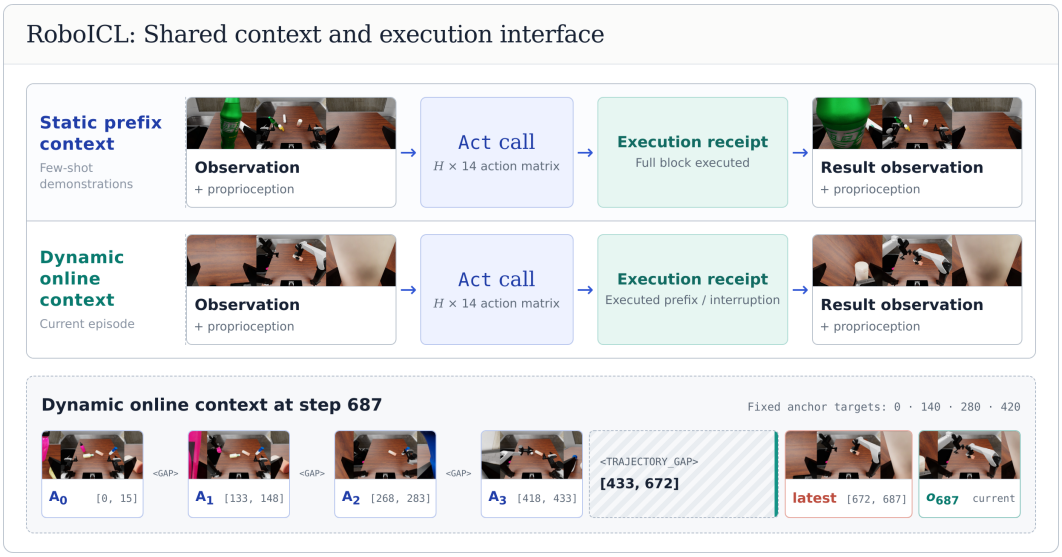


Figure 2: **RoboICL: Shared context and execution interface.** Static prefix context contains few-shot demonstrations, while dynamic online context records the model-generated interactions within the current episode. Both represent each interaction through an observation and proprioceptive state, an `Act` call containing an $H \times 14$ bimanual action block, an execution receipt, and the resulting observation, where $H$ is the action horizon. Online receipts report the executed prefix and any interruption. Images concatenate left-wrist, head, and right-wrist camera views. The lower panel illustrates a *Put bottles in a bin* request at step 687: four fixed anchors and the latest completed interaction preserve selected history, followed by the current observation. Explicit gap markers distinguish omitted intervals from consecutive interactions.

The matrix $A_t$ contains $H$ controller targets, where the horizon is fixed within each task evaluation. Its $i$-th row, for $i \in \{1, \ldots, H\}$, specifies a bimanual command:

$$a_{t,i} = [\Delta p^L_{t,i}, \Delta r^L_{t,i}, u^L_{t,i}, \Delta p^R_{t,i}, \Delta r^R_{t,i}, u^R_{t,i}], \quad \Delta p, \Delta r \in \mathbb{R}^3, \quad u \in [0, 1]. \tag{2}$$

Here, superscripts $L$ and $R$ identify the left and right arms, $\Delta p$ and $\Delta r$ denote world-frame translation and rotation-vector increments, and $u$ denotes an absolute normalized gripper target. At each decision round, the model submits one `Act` call containing $A_t$ and a concise execution note, after which deterministic code validates the command shape and bounds, solves inverse kinematics, and executes the admissible prefix without supplying any learned action proposal.

### 3.2 Context and Execution Interface

To make expert demonstrations and current-episode experience jointly available without conflating the two sources, each request places the task instruction and optional static prefix context first, followed by the dynamic online context and current robot state. Within both contexts, an interaction indexed by $j$ uses the common record

$$c_j = \big((o_j, z_j), A_j, r_j, (o_{j+1}, z_{j+1})\big), \tag{3}$$

where $(o_j, z_j)$ and $(o_{j+1}, z_{j+1})$ are the states before and after the interaction, $A_j$ is the proposed action block, and $r_j$ is its execution receipt. The receipt is a structured controller response recording the executed prefix, any discarded suffix, and any interruption. It therefore preserves the causal link between a proposal and its realized consequence when execution stops early. In static prefix context, the record describes an expert transition; in dynamic online context, it describes the model's transition in the current episode.

Every observation concatenates synchronized left-wrist, head, and right-wrist RGB views into a $1920 \times 480$ triptych. The accompanying proprioceptive state contains joint states, gripper command states, and end-effector poses, and the request provides head-camera calibration. To make adaptation

depend on observable interaction feedback, the request excludes rewards, task scores, success labels, simulator object poses, evaluator code, and layout metadata. Camera geometry, action bounds, inverse kinematics, and deterministic validation remain outside the learned policy.

### 3.3 Dynamic and static context construction

After decision round $t$, execution receipt $r_t$ and the next observation $(o_{t+1}, z_{t+1})$ complete interaction $c_t$. The dynamic online context incorporates this evidence through

$$\mathcal{M}_{t+1} = \mathrm{Update}(\mathcal{M}_t, c_t), \tag{4}$$

where the deterministic update operator applies a bounded retention policy. RoboICL allocates $B$ online slots: the first $B-1$ retain fixed temporal anchors with targets $\tau_i = \lfloor iT/B \rfloor$ for $i = 0, \ldots, B-2$, and the final slot retains the latest interaction. This combination preserves long-range progress and immediate feedback without allowing the request to grow with episode length. Explicit `TRAJECTORY_GAP` records mark omitted intervals so that temporally separated observations do not appear adjacent.

The few-shot setting places the static prefix context $\mathcal{D}_K$ before this evolving online history. Each of its $K$ expert demonstrations remains fixed throughout the episode and uses the same interaction record in Equation 3. To fit long trajectories within the multimodal context budget, RoboICL selects $J$ non-overlapping action blocks per demonstration, retaining the start observation, expert actions, execution receipt, and result observation for each block. Gap markers identify omitted trajectory segments. The two contexts therefore share an action-conditioned representation while serving different roles: expert blocks provide stable procedural guidance, and online blocks expose rollout-specific progress and errors.

Before deduplication, the static prefix contributes at most $2JK$ endpoint triptychs and the online memory contributes at most $2B$, for a combined visual bound of $2JK + 2B$. The current three-shot construction uses $K = 3$, $J = 5$, and $B = 5$, yielding at most 40 triptychs. Appendix C provides the anchor-assignment rules, request example, and implementation details. Section 4.1 evaluates how performance changes when the context budget is allocated between these two sources.

## 4 Experiments

### 4.1 Experimental Setup

We examine how a finite multimodal context budget is divided between static expert demonstrations and dynamic online history. With $K$ demonstrations, $J$ retained blocks per demonstration, and $B$ online blocks, the total interaction budget is $C = KJ + B$. The one-shot allocation study fixes $C = 24$ and varies $(J, B) \in \{(8, 16), (12, 12), (16, 8), (22, 2)\}$. The zero-shot panel separately varies $B \in \{5, 15, 25\}$ and therefore measures online-memory scaling rather than a matched fixed-budget comparison.

Figure 3 shows that the preferred allocation is task dependent. In particular, *Build Tower* and *Insert Tubes* perform worse at $(J, B) = (22, 2)$ than at $(12, 12)$, indicating that denser expert coverage does not compensate for removing most online history on these tasks. Conversely, no single allocation dominates every task. These results support treating static procedural guidance and rollout-specific feedback as complementary resources rather than reducing context quality to demonstration count alone. The five-layout means include historical or provisional points and do not isolate individual context components, so Appendix D reports the complete protocol, provenance caveats, and planned matched controls.

### 4.2 Evaluation setting and benchmark results

We evaluate on simulated bimanual manipulation tasks from RoboDojo (Chen et al., 2026b). Its native evaluator returns a progress score and a separate complete-task success indicator. We report progress on a 0–100 scale, so a score of 82 does not imply an 82% success rate. The ten-task study uses five configurations per task at `eval_seed=0`; the 1-shot condition uses one task-specific expert trajectory when available.

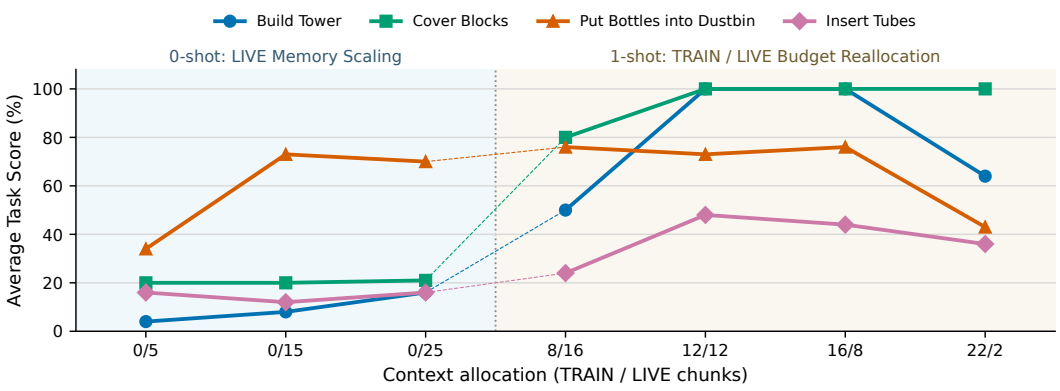


Figure 3: **Static–dynamic context allocation.** Mean RoboDojo progress score multiplied by 100 over five seed-0 layouts per task. The left region varies zero-shot online memory; the right region reallocates a 24-block one-shot budget between demonstration blocks ($J$) and online blocks ($B$). Lines connect settings within each task and do not denote interpolation. These development runs include historical configurations with differing horizons or task-documentation settings and should be read as descriptive evidence rather than a matched component ablation.

Table 1: **Ten-task simulation comparison.** Native RoboDojo progress score multiplied by 100 (higher is better); Overall is the unweighted mean of the ten task scores. RoboICL uses five seed-0 layouts per task. External VLA/WAM, hybrid, and direct-GPT-6 Astra columns reproduce published results (Zhang et al., 2026; Su et al., 2026). One-shot RoboICL obtains 62.00, compared with 62.60 for GPT-6 Astra $+\pi_{0.5}$. Blue and orange identify zero- and one-shot RoboICL.

| | VLA / WAM | | | Hybrid | 0-shot | | 1-shot |
|---|---|---|---|---|---|---|---|
| **Task** | Galaxea G0.5 | OpenWAM $\alpha$ | $\pi_{0.5}$ | GPT-6 Astra $+\pi_{0.5}$ | GPT-6 Astra Direct | GPT-6 Astra **RoboICL** | GPT-6 Astra **RoboICL** |
| Organize the table | 46.33 | 62.50 | 23.33 | 60.00 | 30.00 | 45.00 | 50.00 |
| Classify by language | 1.07 | 1.33 | 0.60 | 38.00 | 60.00 | 44.00 | 44.00 |
| Imitate sorting sequence | 1.67 | 2.90 | 1.60 | 53.00 | 0.00 | 90.00 | 90.00 |
| Arrange largest number | 4.11 | 4.36 | 2.29 | 50.00 | 57.00 | 71.00 | 72.00 |
| Pack objects into a box | 17.12 | 20.83 | 18.36 | 50.00 | 50.00 | 16.00 | 36.00 |
| Classify objects | 10.33 | 5.53 | 24.67 | 71.00 | 100.00 | 100.00 | 71.00 |
| Build Tower | 82.93 | 52.53 | 37.73 | 64.00 | 12.00 | 16.00 | 100.00 |
| Make a Kong in Mahjong | 90.00 | 32.00 | 26.67 | 40.00 | 0.00 | 0.00 | 0.00 |
| Fold clothes | 32.75 | 51.31 | 29.12 | 100.00 | 40.00 | 4.00 | 84.00 |
| Put bottles in a bin | 96.30 | 94.03 | 79.93 | 100.00 | 36.00 | 70.00 | 73.00 |
| **Overall** | **38.26** | 32.73 | 24.43 | **62.60** | 38.50 | 45.60 | **62.00** |

Table 1 includes all ten tasks, including low-scoring cases. The 1-shot RoboICL macro-average is 62.00, compared with 38.50 for the externally reported Direct configuration. Task-level values are higher on six tasks, lower on three, and equal on one. Because external columns follow their original protocols and the RoboICL entries combine selected historical batches, these differences provide broad context rather than a matched causal comparison.

Table 2 reports all eight RoboDojo Precision tasks. RoboICL uses one expert demonstration, 12 retained demonstration blocks, 12 online-memory blocks, GPT-6 Astra with `xhigh` reasoning, public task descriptions. We first average layouts within each task and then weight the eight tasks equally.

RoboICL attains the highest displayed macro-average, 41.63, which is 3.35 points above Liber-0 Preview. It has the highest displayed task score on *Deposit Coin*, *Insert Key*, and *Play Xylophone*, but scores zero on *Plug in Charger* and *Pour Balls into Vase*. The aggregate result therefore reflects strong gains on several precision tasks rather than uniformly reliable contact-rich control.

Table 2: **RoboDojo Precision results.** Native progress score multiplied by 100 (higher is better). RoboICL uses one demonstration with 12 static and 12 online blocks; Its task means use 20 seed-0 standard layouts; Overall is the unweighted mean of the eight task means. External columns reproduce task means from the public RoboDojo leaderboard. Bold marks the largest displayed value in each row.

| Task | **RoboICL** | GPT-6 Astra | Liber-0 Preview | Simate-beta | DM0.5 | Galaxea G0.5 |
|---|---|---|---|---|---|---|
| Build Tower | 75.00 | 16.40 | 84.33 | **87.40** | 55.20 | 82.93 |
| Insert Tubes | 52.00 | 6.00 | **82.53** | 41.20 | 71.73 | 58.53 |
| Deposit Coin | **92.00** | 16.00 | 18.67 | 16.40 | 13.87 | 10.93 |
| Fasten Screws | 49.00 | 36.40 | **61.80** | 40.93 | 40.20 | 30.00 |
| Insert Key | **15.00** | 14.40 | 0.90 | 10.90 | 4.90 | 14.90 |
| Play Xylophone | **50.00** | 8.00 | 0.00 | 3.33 | 0.67 | 0.00 |
| Plug in Charger | 0.00 | 0.00 | 27.33 | **29.33** | 2.67 | 0.67 |
| Pour Balls into Vase | 0.00 | 4.00 | 30.67 | **45.33** | 9.33 | 28.00 |
| **Overall** | **41.63** | 12.65 | 38.28 | 34.35 | 24.82 | 28.25 |

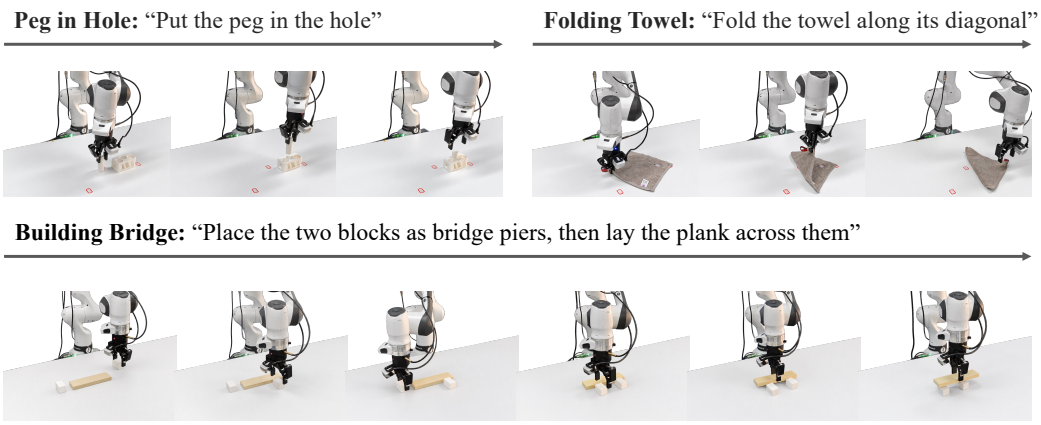


Figure 4: Real-robot execution of three manipulation tasks.

### 4.3 Real-robot experiments

We evaluate whether RoboICL transfers beyond simulation using a Franka Research 3 arm. A human expert collects three demonstrations per task through GELLO, and observations combine a wrist-mounted D405 camera with an external D515 camera. We test `Peg in Hole`, `Folding Towel`, and `Building Bridge`, spanning precision, deformable-object, and long-horizon manipulation. Each zero-, one-, and three-shot condition contains five trials with randomized object positions and orientations. The reported metric assigns credit to predefined task stages and averages normalized progress across trials.

Demonstrations substantially improve physical-robot performance: mean progress rises from 14.45 at zero shot to 63.33 at one shot and 78.89 at three shots (Table 3). Yet the gains do not guarantee completion: three of five three-shot `Peg in Hole` trials stop after pickup, and RGB feedback provides weak cues for small contact corrections. `Building Bridge` remains lowest across settings, reflecting the challenge of representing long procedures with bounded context.

To examine robustness beyond the demonstrated object instance, we further evaluate `Folding Towel` in the three-shot setting with two unseen towels. Performance remains high on Unseen 1, whereas the larger Unseen 2 scores 40, which is 60 points below the seen towel, and only one of its five trials receives full credit (Table 4). Although the high-level goal remains a diagonal fold, the model does not reliably adapt its motion to the change in object scale. This behavior suggests

Table 3: **Real-robot progress by demonstration count.** Mean normalized score ($\times 100$) over five trials per condition.

| Task | 0-shot | 1-shot | 3-shot |
|---|---|---|---|
| Peg in Hole | 10.00 | 60.00 | 70.00 |
| Folding Towel | 30.00 | 90.00 | 100.00 |
| Building Bridge | 3.34 | 40.00 | 66.67 |
| Mean | 14.45 | 63.33 | 78.89 |

Table 4: **Real-robot Folding Towel OOD performance.** Three-shot mean progress ($\times 100$); five trials per condition.

| Test condition | Score |
|---|---|
| Seen 1 | 100.00 |
| Unseen 1 | 90.00 |
| Unseen 2 (larger towel) | 40.00 |
| Mean across unseen conditions | 65.00 |

Table 5: **Jev-gated execution.** Complete-task successes over five layouts and paired wall time for layouts successful in both conditions.

| Task | Pure GPT-6 Astra | GPT-6 Astra+Jev | Paired wall time (s) |
|---|---|---|---|
| Align Blocks | 3/5 | 2/5 | 1128/865 ($n = 1$) |
| General Pickup | 4/5 | 5/5 | 3642/2327 ($n = 4$) |

that the frozen model uses demonstrations to ground broad folding knowledge in the embodiment and action space but may follow the demonstrated motion too closely. Appendix E provides the complete protocol, failure analysis, and OOD examples.

### 4.4 Exploratory Jev-gated execution

We tested Jev, a compact decision model that gates reuse of a GPT-6 Astra-predicted action suffix, in a separate *zero-shot* simulator panel. This is an execution-policy study rather than an in-context demonstration ablation. For *Align Blocks* and *General Pickup*, pure GPT-6 Astra and GPT-6 Astra+Jev each ran once on seed-0 standard layouts 0–4 (five episodes per task and arm). Both used GPT-6 Astra with `xhigh` reasoning, the same task prompt, cameras, action constraints, 20 predicted actions per request, five executed actions per chunk, and 25 live-history anchors. Pure GPT-6 Astra replanned after each chunk. Jev could approve one further five-action chunk from the current GPT-6 Astra plan; it observed proprioception, execution tracking, and the plan note, but no RGB. This Jev configuration was selected after inspecting outcomes on the same layouts, so the comparison is exploratory rather than held-out.

We required Jev to lose no more than one successful layout relative to pure GPT-6 Astra; it met this criterion on both tasks (Table 5). On the four jointly successful *General Pickup* layouts, GPT-6 Astra+Jev's observed wall time was 36.1% lower, despite independent stochastic trajectories. *Align Blocks* had only one jointly successful pair; its 1128-to-865-second difference is not a stable speed estimate. Across all five layouts, GPT-6 Astra requests fell from 181 to 121 for *Align Blocks* and from 116 to 60 for *General Pickup*; provider-reported GPT-6 Astra tokens fell from 9.53 to 5.08 million and from 4.95 to 1.39 million, respectively. Those totals combine successful and failed episodes. On *General Pickup* layout 3, GPT-6 Astra+Jev used 103 executed steps versus pure GPT-6 Astra's 75 despite 13 versus 15 requests; fewer model calls do not imply fewer robot actions. Appendix F reports every native outcome and its runtime measurements. With one run per slot, reused tuning layouts, and external API latency variation, we do not claim a general or causal speedup.

### 4.5 Qualitative case studies

**Contact-rich manipulation.** In the selected *Fasten Screws* layout-2 pair, zero-shot control receives 0.20, whereas the three-shot rollout receives 1.00. The latter brings the yellow and white nuts to their corresponding screws, uses repeated short clockwise motions with release-and-regrasp resets, and then handles the red assembly with an additional recovery. The zero-shot rollout spends much of its longer execution searching for a stable grasp. This example illustrates how demonstrations can provide a useful manipulation sequence and contact strategy. The task requires tight geometric alignment, but no calibrated millimeter-error measurement is claimed here.

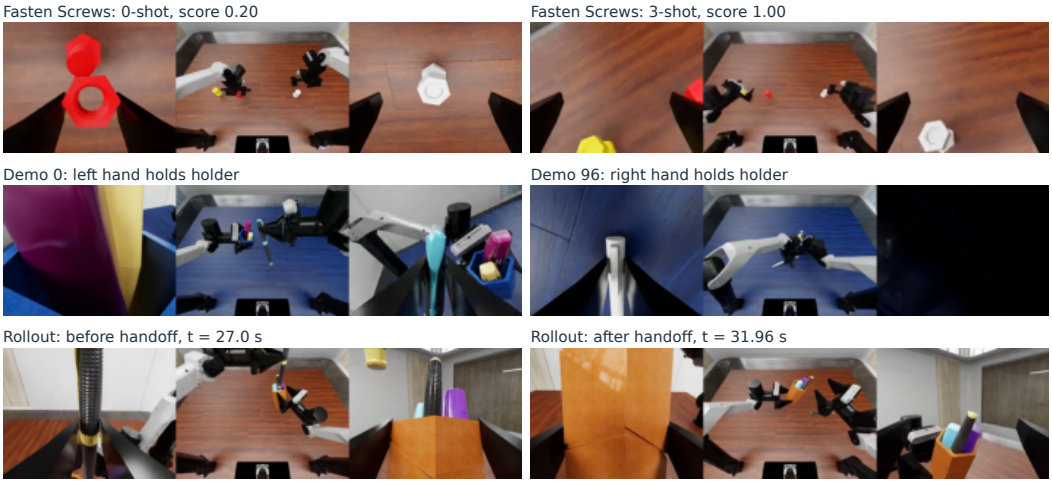


Figure 5: **Selected case-study frames.** Top: terminal views of the zero-shot and three-shot screw-fastening clips. Middle: reference episodes with opposite pen-holder hand assignments. Bottom: two moments around the rollout's right-to-left transfer.

**Within-rollout hand-role adaptation.** For *Fill Pen Holder*, demonstration episode 0 holds the holder with the left hand and inserts pens with the right, while episodes 96 and 99 reverse those roles. In the seed-1/layout-0 rollout, the model initially follows the second assignment, then brings the holder to the left gripper, establishes a new grasp, and releases the right gripper. The selected sequence shows a mid-air transfer while the holder retains its pens. The demonstrations exhibit the two role assignments separately rather than the transition shown in the rollout. This provides qualitative evidence of recombining demonstrated behaviors within an episode.

## 5 Discussion and Limitations

RoboICL makes demonstration and deployment experience comparable at the level of executed interaction. Its single action interface identifies the learned source of each Cartesian proposal, while explicit receipts preserve what the robot actually did. The observed score changes and hand-role transfer motivate studying the content of context—action semantics, feedback, and temporally distributed history—alongside model scale. The present evidence supports this research direction without establishing which component is responsible for the gains.

The main empirical limits are concentrated in evaluation coverage and control: few layouts in the shot study, one broader task sweep, historical batch selection in the ten-task table, and incomplete matched interface and memory ablations. Repeated seeds, a matched zero-shot baseline, equal-budget context controls, and full failure accounting are the next priorities. A continuation signal remains in the current receipt. The Jev panel measures simulator wall time and provider-reported tokens, but its configuration was selected using the same five layouts and does not establish general speedup or monetary cost. Embodiment transfer and reliability beyond the tested simulator and real-robot tasks require further measurement. Demonstrations may also carry dataset-specific conventions that fail under visual or dynamical shift. These limitations motivate controlled follow-up experiments rather than additional unsupported claims in the current draft.

## 6 Conclusion

RoboICL conditions a general-purpose multimodal model on execution-grounded demonstrations and bounded live interaction history to produce direct bimanual controls. The existing results show promising task-level adaptation and concrete examples of contact handling and strategy recombination. The framework makes action provenance, context construction, and execution feedback explicit, providing a basis for controlled studies of embodied in-context learning.

### AI USE STATEMENT

Generative AI tools were used to assist with drafting and editing manuscript text, organizing the presentation and related-work discussion, formatting references, preparing figures, and checking LaTeX and analysis code. These tools also helped articulate the two-context framing and reorganize the presentation of the method and experiments. Generative AI was not used to collect or generate experimental data, execute robot evaluations, or autonomously determine the scientific conclusions. GPT-6 Astra is additionally the policy model evaluated in the experiments, as described in the method. The authors verified AI-assisted text, citations, code, figures, and numerical summaries against the underlying sources and experimental records, and take responsibility for the final content of the paper and its artifacts.

### ETHICS STATEMENT

The experiments comprise simulated RoboDojo tasks and controlled real-robot manipulation, and do not involve a human-subject study or personal data. Model requests contain robot observations, task instructions, proprioceptive state, and interaction records. Released artifacts exclude credentials, personal information, and provider-private reasoning content. The RoboDojo software and simulated assets are used under the applicable MIT License, with the required notices retained in redistributed materials. Direct control with a general-purpose model introduces risks from perception errors, unexpected motions, communication failures, and incomplete recovery. The reported deterministic validation and constrained experimental setting reduce execution risk but do not constitute a safety certification. The real-robot results should therefore not be interpreted as evidence that the system is safe for unsupervised deployment or operation around people.

### REPRODUCIBILITY STATEMENT

Sections 3 and 4 specify the context interface, action representation, memory policy, evaluation metrics, and reported experimental protocols. Appendices B, C, D, E, F, and G provide the interaction contract and memory algorithm, detailed context construction, ablation configurations, the real-robot protocol, complete Jev-panel outcomes, and a sanitized request specimen. The supplementary source package includes the numerical data snapshots and scripts used to reconstruct figures and validate reported aggregates. Protocol mismatches, historical-data limitations, and unavailable measurements are identified explicitly rather than presented as matched or reproducible results.

## REFERENCES

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

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a Visual Language Model for Few-Shot Learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2204.14198.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler,

Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Fu Chen, Xin Ding, Bingjia Huang, Xiangyu Li, Mingju Wang, Jiawei He, Kun Li, Wei Sun, Yunxin Liu, Hao Wu, and Ting Cao. Zeva: In-Context Causal Learning for Generalizable Embodied Manipulation. *arXiv preprint arXiv:2608.30880*, 2026a. URL https://arxiv.org/abs/2608.30880.

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*, 2026b. URL https://arxiv.org/abs/2607.04434.

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

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.

Norman Di Palo and Edward Johns. Keypoint Action Tokens Enable In-Context Imitation Learning in Robotics. In *Proceedings of Robotics: Science and Systems*, 2024. URL https://arxiv.org/abs/2403.19578.

Yan Duan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-Shot Imitation Learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/ba3866600c3540f67c1e9575e213be0a-Abstract.html.

Letian Fu, Huang Huang, Gaurav Datta, Lawrence Yunliang Chen, William Chung-Ho Panitch, Fangchen Liu, Hui Li, and Ken Goldberg. In-Context Imitation Learning via Next-Token Prediction. *arXiv preprint arXiv:2408.15980*, 2024. URL https://arxiv.org/abs/2408.15980.

Generalist Team. GEN-1.5: Embodied Foundation Models are One-Shot Learners. *Generalist AI Blog*, August 2026. URL https://generalistai.com/blog/gen-1.5. Published August 19, 2026. Accessed September 21, 2026.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models. In *Proceedings of the 7th Conference on Robot Learning*, volume 229, pp. 540–562. PMLR, 2023. URL https://proceedings.mlr.press/v229/huang23b.html.

Qing Lian, Kent Yu, and Lei Zhang. Reflective VLA: In-Context Action Consequences Make VLAs Generalize. *arXiv preprint arXiv:2606.25215*, 2026. URL https://arxiv.org/abs/2606.25215.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as Policies: Language Model Programs for Embodied Control. *arXiv preprint arXiv:2209.07753*, 2022. URL https://arxiv.org/abs/2209.07753.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. *arXiv preprint arXiv:2303.11366*, 2023. URL https://arxiv.org/abs/2303.11366.

Kaustubh Sridhar, Souradeep Dutta, Dinesh Jayaraman, and Insup Lee. RICL: Adding In-Context Adaptability to Pre-Trained Vision-Language-Action Models. In *Proceedings of the 9th Conference on Robot Learning*, volume 305, pp. 5022–5038. PMLR, 2025. URL https://proceedings.mlr.press/v305/sridhar25a.html.

Jiayi Su, Yixin Zheng, Mi Yan, Li Yi, Zhizheng Zhang, and He Wang. GPT 6 Astra as an embodied policy. Technical report and code, 2026. URL https://github.com/anonymous-report-421/GPT-as-Policy. Report: https://anonymous-report-421.github.io/public-website/. Accessed September 21, 2026.

Maria Tsimpoukelli, Jacob Menick, Serkan Cabi, S. M. Ali Eslami, Oriol Vinyals, and Felix Hill. Multimodal Few-Shot Learning with Frozen Language Models. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/01b7575c38dac42f3cfb7d500438b875-Abstract.html.

Vitalis Vosylius and Edward Johns. Instant Policy: In-Context Imitation Learning via Graph Diffusion. In *International Conference on Learning Representations*, 2025. URL https://www.robot-learning.uk/instant-policy.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Yida Yin, Zekai Wang, Yuvan Sharma, Dantong Niu, Trevor Darrell, and Roei Herzig. In-Context Learning Enables Robot Action Prediction in LLMs. In *IEEE International Conference on Robotics and Automation*, 2025. URL https://arxiv.org/abs/2410.12782.

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. RoboDojo technical report, September 2026. URL https://robodojo-benchmark.com/report/gpt-6-astra-eval. Published September 16, 2026. Accessed September 21, 2026.

Jiaming Zhou, Qihang Zhang, Gangwei Xu, Cunxin Fan, Yujie Zhao, Ruilin Wang, Yiming Luo, Shuai Yang, Xing Zhu, Yujun Shen, Junwei Liang, and Yinghao Xu. Zero-WAM: In-Context World-Action Modeling from Human Videos for Open-Ended Task Generalization. *arXiv preprint arXiv:2608.26103*, 2026. URL https://arxiv.org/abs/2608.26103.

# A Extended related work

**Demonstration-conditioned robot learning.** Few-shot language modeling establishes examples as an inference-time interface (Brown et al., 2020), subsequently extended to multimodal inputs by Frozen and Flamingo (Tsimpoukelli et al., 2021; Alayrac et al., 2022). In robotics, ICRT and Instant Policy learn context-conditioned control from sensorimotor sequences or demonstration pairs (Fu et al., 2024; Vosylius & Johns, 2025). RICL combines additional training of a pretrained VLA with demonstration retrieval (Sridhar et al., 2025). Zero-WAM learns from paired human videos and robot trajectories with a future-chunk prediction objective (Zhou et al., 2026), while GEN-1.5 reports physical prompting after large-scale pretraining on physical interaction sequences (Generalist Team, 2026). These approaches provide different training routes to deployment-time conditioning. RoboICL studies the complementary setting of eliciting action chunks from a frozen general-purpose VLM through demonstration and interaction context, without training a robot-specific policy or action head.

**Frozen general-purpose models for robot control.** Frozen language and vision-language models control robots through skill selection, program generation, and spatial planning (Ahn et al., 2022; Liang et al., 2022; Huang et al., 2023), or predict numerical actions from structured spatial representations (Di Palo & Johns, 2024; Yin et al., 2025). Recent work further explores semantic action interfaces and direct or hybrid VLM control (Chen et al., 2026c; Su et al., 2026). Concurrent systems GPT-Policy (Cheng et al., 2026) combine frozen VLMs with demonstrations and online feedback. GPT-Policy generates absolute Cartesian targets or waypoint sequences with separate gripper calls. RoboICL instead generates Cartesian increments with gripper commands at each control step. Its common demonstration and online-interaction format records proposed action blocks, executed prefixes, and observations at the actual execution endpoints.

**Language-model control and interaction history.** ReAct and Reflexion use feedback and memory to guide agent decisions (Yao et al., 2023; Shinn et al., 2023). In robotics, Zeva uses interaction memory (Chen et al., 2026a), while Reflective VLA trains policies on observation–action–consequence histories (Lian et al., 2026). RoboICL retains temporally distributed interactions and the latest feedback within a bounded visual context, while explicit gap records prevent omitted intervals from appearing as continuous execution.

# B Interaction contract and memory update

Table 6 records the concrete underlying interface for RoboDojo.

Table 6: Documented observation and action contract.

| Component | Representation |
|---|---|
| RGB | Left wrist, head, right wrist; each $640 \times 480$; concatenated $1920 \times 480$. |
| Robot state | Joint state, commanded gripper state, and end-effector poses. |
| Tool | One `act` call, with an action matrix and an execution note. |
| Action matrix | Eeach row has 7 values per arm: translation delta, rotation-vector delta, and gripper target. |
| Reference point | Cartesian end-effector convention documented by the harness; link-6 convention must be distinguished from a fingertip midpoint. |
| World-frame update | $p' = p + \Delta p$ and $q' = \mathrm{Exp}(\Delta r) \otimes q$, with Exp mapping a rotation vector to a unit quaternion. |
| Control rate | 25 Hz environment actuation; model-call latency is separate. |
| Receipt | Executed prefix length, discarded suffix, interruptions, and episode-continuation information. |

The following pseudocode formalizes the retained-chunk semantics; it is not a replacement for the executable harness or its failure handling.

**Algorithm 1: Bounded anchored interaction memory**
**Input:** episode budget $H$, slot budget $B$, completed chunks $c_j$ with intervals $[s_j, e_j]$.
1. Set $\tau_i = \lfloor iH/B \rfloor$ for $i = 0, \ldots, B-2$; initialize empty anchors.
2. On completion of $c_j$, retain it as the latest chunk.
3. If it is the first completed chunk, store it as anchor zero.
4. For each unfilled later anchor $i$, if $s_j < \tau_i \leq e_j$, store $c_j$ at that anchor.
5. Form the chronological union of filled anchors and the latest chunk, deduplicating repeated chunks.
6. Replace omitted intervals with explicit gap records; preserve their interval boundaries and compact execution metadata.
7. Serialize retained chunks, gap records, and the current observation together with fixed TRAIN context and the task instruction.
**Output:** the next model request and an updated bounded set of full live chunks.

The initial target is special: a positive-length chunk starting at zero does not satisfy $s < 0 \leq e$, so it is retained explicitly. A chunk can coincide with both an anchor and the latest slot; it must not be duplicated. Robodojo use $H = 700$ for the bottle task and $H = 1050$ for tower construction, giving targets $[0, 140, 280, 420]$ and $[0, 210, 420, 630]$, respectively.

## C DETAILED CONTEXT CONSTRUCTION

**Dynamic online context.** In-episode adaptation requires the dynamic online context to expose how earlier decisions change the environment. After decision round $t$, the execution system returns receipt $r_t$ and the next observation $(o_{t+1}, z_{t+1})$, thereby completing interaction $c_t$. The deterministic update operator inserts this interaction and applies the retention policy in Algorithm 1, allowing a later decision to relate the current state to earlier actions, task progress, unsuccessful contacts, and execution interruptions.

Retaining every completed interaction would make the request grow with episode length. RoboICL instead allocates $B$ slots to the dynamic online context. The first $B-1$ slots provide fixed temporal anchors across the episode, while the final slot preserves the latest completed interaction. With $T$ denoting the maximum number of environment steps and $i \in \{0, \ldots, B-2\}$ indexing a fixed anchor, its target is $\tau_i = \lfloor iT/B \rfloor$. The initial nonempty interaction occupies anchor zero, and every later anchor stores the first completed interaction $c_j$ whose interval satisfies $s_j < \tau_i \leq e_j$. The latest-interaction slot supplies the most recent outcome for local correction and is deduplicated if the same interaction occupies an anchor.

Sparse retention can otherwise make observations separated by many environment steps appear adjacent. Explicit `TRAJECTORY_GAP` records therefore mark intervals between retained interactions, omitting full images and action arrays while preserving interval boundaries and compact execution metadata. For $T = 700$ and $B = 5$, targets are $0, 140, 280, 420$. The released request retains intervals $[0, 15]$, $[133, 148]$, $[268, 283]$, $[418, 433]$, and the latest interval $[672, 687]$. At most $B$ complete online interactions are retained, requiring at most $2B$ endpoint triptychs before deduplication. This bound concerns detailed visual history rather than all serialized tokens.

**Static prefix context.** Expert demonstrations complement self-generated experience by supplying task knowledge before execution begins. The $K$ demonstrations in $\mathcal{D}_K$ remain unchanged throughout the episode while $\mathcal{M}_t$ evolves. Each selected expert interaction uses the same format as an online interaction: an `Act` call specifies the expert action block, and an execution receipt and result observation record its outcome. The shared representation conveys task structure, action semantics, and procedural guidance as action-conditioned transitions.

Serializing complete demonstrations would consume the multimodal context budget before online interaction begins. RoboICL selects $J$ non-overlapping action blocks from each trajectory. Each selected block preserves the source actions and its start and result observations, while explicit gaps identify omitted intervals. Every block contributes at most two endpoint triptychs, giving $2JK$ triptychs for the prefix and a combined upper bound of $2JK + 2B$ with online memory. The current three-shot construction uses $K = 3$, $J = 5$, 15 controller steps per expert block, and $B = 5$, yielding 30 expert triptychs and at most 40 triptychs overall before deduplication.

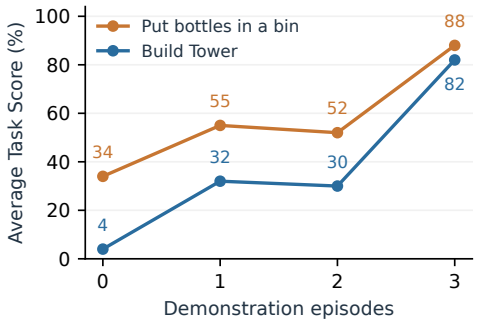


Figure 6: **Demonstration scaling with growing context.** Average Task Score over five layouts at zero to three shots. Each demonstration adds five expert blocks while online memory remains at five blocks. Reused batches preclude a matched causal interpretation.

## D Detailed ablation protocols and additional studies

**Allocation protocol.** A retained interaction block contains its start observation, action block, execution receipt, and result observation. With $K$ demonstrations, $J$ blocks per demonstration, and $B$ online blocks, the total budget is $C = KJ + B$. The one-shot configurations in Figure 3 use $C = 24$, corresponding to at most 48 endpoint triptychs before deduplication. The allocation grid uses $(J, B) \in \{(8, 16), (12, 12), (16, 8), (22, 2)\}$. The zero-shot reference $(0, 25)$ has a different total budget. We stop at $B = 2$ because one slot stores the initial interaction and one rolling slot stores the latest interaction. The study reports Average Task Score on *Build Tower*, *Cover Blocks*, *Put Bottles into Dustbin*, and *Insert Tubes* over five layouts per task.

Increasing $J$ provides denser coverage of the demonstrated procedure but removes earlier online actions, interruptions, failed grasps, and recovery attempts. A trend favoring larger $J$ would indicate that expert procedural coverage dominates, whereas a trend favoring larger $B$ would favor online state tracking and correction. An interior optimum would support complementarity. The curves are task dependent: *Build Tower* and *Insert Tubes* score lower at $(22, 2)$ than at $(12, 12)$. The $0/5$ tower and bottle values come from an earlier preview with different horizon and task-documentation settings, and the *Cover Blocks* $0/15$ display value still requires reconciliation with its raw-run mean. These development means therefore do not establish a universal allocation or isolate a single transition's effect.

**Within-episode context scaling.** The available zero-shot sweep uses $B \in \{5, 15, 25\}$. A planned matched control uses $B \in \{2, 4, 8, 12, 16, 24\}$, retaining the initial and latest interaction at every point while additional slots introduce uniformly distributed anchors. Earlier configurations retained five online blocks and frequently omitted intermediate failures and corrections. Increasing memory to 25 blocks coincides with stronger RoboDojo Open results but also with a shorter task-adaptive action horizon, so the historical comparison does not establish a causal memory effect. The matched control holds the action horizon fixed.

**Fixed-budget shot scaling.** A planned control separates demonstration diversity from total capacity using $1 \times 12$ expert blocks plus 12 online blocks for one shot, $2 \times 6 + 12$ for two shots, and $3 \times 4 + 12$ for three shots. Demonstration sets remain nested and ordered, so shot count changes without increasing the number of expert blocks or endpoint images.

**Demonstration scaling with growing context.** Figure 6 reports *Build Tower* and *Put Bottles in a Bin*. Each expert demonstration contributes five uniformly sampled blocks, while online memory retains five blocks. Scores at zero, one, two, and three shots are $4/32/30/82$ for tower and $34/55/52/88$ for bottles. Both tasks improve substantially from zero to three shots, although the decrease from one to two shots shows that performance need not scale monotonically. This study varies total context and therefore does not isolate shot count under a fixed image budget.

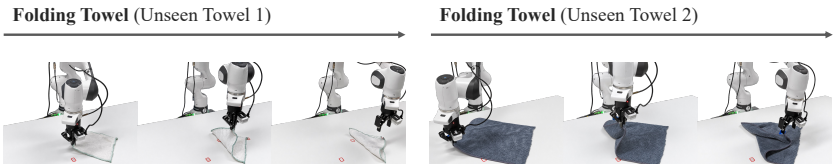


Figure 7: Out-of-distribution evaluation of `Folding Towel` using two unseen towels.

Taken together, the studies distinguish adaptation from expert interactions across episodes and adaptation from action-conditioned feedback within the current episode. The working hypothesis is that performance depends on coverage of informative transitions rather than shot count alone. Expert demonstrations expose task phases and contact strategies, while online memory supports state estimation, verification, and correction of execution-specific errors.

## E Real-robot protocol and additional analysis

**Platform and demonstrations.** Experiments use a Franka Research 3 arm. A human expert collects three demonstrations per task using GELLO as the teleoperation interface. Visual observations comprise RGB images from a wrist-mounted D405 camera and an external third-person D515 camera. Zero-shot evaluation provides no demonstration, one-shot evaluation uses the same selected demonstration in every trial, and three-shot evaluation uses all three demonstrations.

**Tasks and evaluation protocol.** The three tasks span high-precision, deformable-object, and long-horizon manipulation. `Peg in Hole` requires insertion with a nominal peg–hole tolerance of $1\,\mathrm{mm}$. `Folding Towel` requires folding a towel along its diagonal. `Building Bridge` requires positioning two bridge piers before placing the deck. Each task and shot setting contains five trials, with object positions randomized within a $10\,\mathrm{cm} \times 10\,\mathrm{cm}$ region and orientations randomized about the gravity-aligned axis.

Each trial receives a progress score in $[0, 1]$. For `Peg in Hole`, picking up and inserting the peg each contribute 0.5. For `Folding Towel`, picking up the towel and completing the fold each contribute 0.5. For `Building Bridge`, each of the six pickup or placement stages contributes $1/6$. Table 3 reports mean progress multiplied by 100 rather than complete-task success.

**Failure analysis.** Eleven of 15 zero-shot trials receive no progress credit, although some rollouts still exhibit basic manipulation. For example, the robot can pick up a towel corner without completing the diagonal fold. Demonstrations increase mean progress to 63.33 with one shot and 78.89 with three shots, but three of five three-shot `Peg in Hole` trials stop after pickup. Inspected rollouts bring the peg near the hole and then stall during insertion. Successive RGB observations change little after contact, providing weak cues for the required correction. Force, torque, or tactile feedback may help, but these modalities are not tested.

`Building Bridge` remains the lowest-scoring task at every shot count. It gains 26.67 points between one and three shots, compared with 10 points on each other task, yet only one of five three-shot trials receives full credit. Its longer trajectory is represented by a bounded set of keyframes and action blocks. Under a fixed budget, adding demonstrations also reduces temporal density within each demonstration, making the allocation between diversity and trajectory coverage especially consequential for long-horizon control.

**Towel out-of-distribution examples.** Figure 7 visualizes the seen and unseen towel conditions evaluated quantitatively in Table 4.

# F EXPLORATORY JEV-GATED EXECUTION: COMPLETE FIVE-LAYOUT RECORD

This panel is separate from the demonstration-count experiments. Each condition uses zero demonstrations, seed 0, standard layouts 0–4, GPT-6 Astra with `xhigh` reasoning, H20 prediction/K5 execution, and 25 live anchors. The two arms share the same GPT-6 Astra request configuration. Jev may approve at most one additional K5 chunk from an H20 plan. It sees proprioception, execution tracking, and the GPT-6 Astra execution note but not the live RGB image. The Jev configuration was selected after examining outcomes on these same five layouts, with no held-out tuning split. All 20 listed runs returned native scores; pure GPT-6 Astra's layout-0 episodes were retained from earlier scored runs, while Jev reran layout 0.

Table 7: **Every Jev-panel layout.** Each cell is native score $\times 100$ / executed robot steps / wall-clock seconds (rounded). Scores in this panel are 0 or 100; a 100 denotes native complete-task success. Failed 200-step episodes are shown but excluded from successful-pair completion-time comparisons.

| Task | Layout | Pure GPT-6 Astra | GPT-6 Astra+Jev |
|---|---|---|---|
| Align Blocks | 0 | 0/200/4489 | 0/200/1135 |
| | 1 | 100/151/1128 | 100/148/865 |
| | 2 | 100/167/1490 | 0/200/1120 |
| | 3 | 100/138/1054 | 0/200/1199 |
| | 4 | 0/200/2358 | 100/141/827 |
| General Pickup | 0 | 100/64/1335 | 100/70/440 |
| | 1 | 100/99/857 | 100/73/557 |
| | 2 | 100/83/792 | 100/84/683 |
| | 3 | 100/75/658 | 100/103/648 |
| | 4 | 0/200/1436 | 100/89/576 |

Table 8: **Total measured model use across each five-layout condition.** GPT-6 Astra request counts and API seconds use completed transport events; tokens are provider-reported response usage. Jev reuse counts only steps with actual execution receipts. These resource totals include successful and failed episodes and are not completion-time estimates.

| Task | Arm | Success | GPT-6 Astra calls | GPT-6 Astra API s | GPT-6 Astra tokens | Jev votes / reused steps |
|---|---|---|---|---|---|---|
| Align Blocks | Pure GPT-6 Astra | 3/5 | 181 | 9407 | 9,534,190 | 0 / 0 |
| | GPT-6 Astra+Jev | 2/5 | 121 | 4101 | 5,083,650 | 113 / 315 |
| General Pickup | Pure GPT-6 Astra | 4/5 | 116 | 4192 | 4,952,887 | 0 / 0 |
| | GPT-6 Astra+Jev | 5/5 | 60 | 2095 | 1,391,331 | 53 / 140 |

**Success-conditioned latency and failure accounting.** Jev and pure GPT-6 Astra both succeed on *Align Blocks* layout 1 (864.91 versus 1127.98 seconds) and *General Pickup* layouts 0–3 (2327.17 versus 3641.54 seconds). Jev also succeeds on *Align Blocks* layout 4, where pure GPT-6 Astra fails, so that episode has no paired completion-time baseline. On *General Pickup* layout 3, Jev takes 103 robot steps versus 75 for pure GPT-6 Astra while using 13 versus 15 GPT-6 Astra requests. Fewer model calls can therefore coexist with more physical actions. Failed runs remain in the success denominator but are excluded from successful-pair time sums. These single episodes do not support uncertainty intervals or a causal component attribution.

**Provenance.** The machine-readable artifact `data/jev-five-layout-2026-09-25.json` contains every reported run ID, native score, wall time, executed steps, request and token totals, Jev votes and receipts, and SHA-256 hashes for the profiles, layouts, and runtime manifest. The pure GPT-6 Astra implementation is recorded at commit `3520194`; the selected Jev implementation at `c66514d`. Full-resolution video, transport logs, and action receipts remain in the

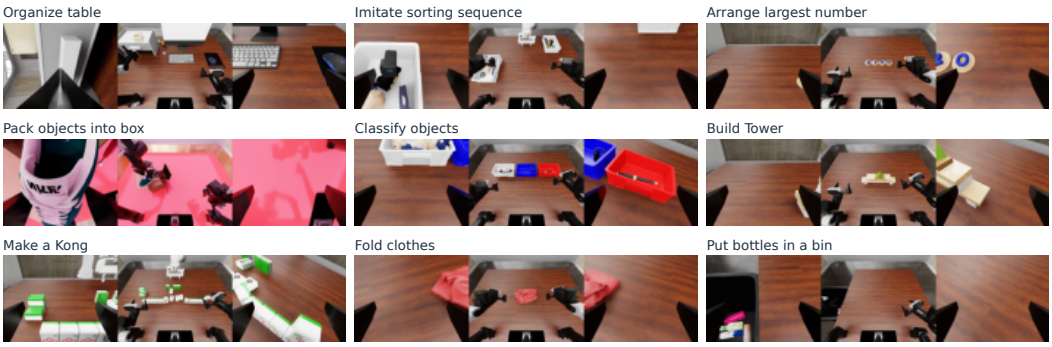


Figure 8: Terminal frames from selected three-shot website clips. Each cell preserves three synchronized views. The set includes failures and partial outcomes; it is a qualitative selection rather than a random sample. Videos and action/receipt traces should accompany the final artifact.

corresponding simulator run directories. API service load and independent model/simulator trajectories are uncontrolled; comparison uses observed wall-clock time, not simulated time or verified monetary cost.

## G Request specimen, runtime accounting, and qualitative artifacts

The released structural specimen is the final two-shot bottle-task request for seed 0, standard layout 3. At model call 49, the current observation is step 687. The request contains 81 input items, 20 reference triptychs, and 10 live triptychs. Its serialized size is approximately 7.6 MB. Recorded API accounting gives 72,522 input tokens, of which 66,889 are cached (92.23%); the episode-level reported cache fraction is 89.15%. These measurements describe one request/episode and must not be presented as average three-shot efficiency.

Its fixed live intervals are $[0, 15]$, $[133, 148]$, $[268, 283]$, and $[418, 433]$; the latest interval is $[672, 687]$. The final action block proposes 15 steps, of which 13 run before the step-700 boundary. The episode receives score 0.4 and is not a success. This failed example is useful for documenting the input contract without selecting a successful trajectory solely for presentation. The sanitized specimen excludes credentials, request identifiers, provider-private reasoning, encrypted content, image payload bytes, and local absolute paths.

**Case-study sampling.** Figure 5 samples the screw clips near their ends (75.8 s and 53.6 s), the two pen-holder demonstrations at 25.0 s and 20.0 s, and the transfer rollout at 27.0 s and 31.96 s. These are video timestamps, not measured model latency. The rollout file's historical name contains "11", but its verified setting is seed 1/layout 0. The asset-generation script records exact paths and sampling times in the data snapshot. No synthetic robot frame is used.




Figure 9: **Detail of the pen-holder transfer.** Head-view crops from the synchronized three-view rollout show the right hand supporting the holder, the new left-hand grasp, and the right gripper's release. The main figure preserves all three views; this enlargement makes the contact sequence easier to inspect. Timestamps identify selected frames, not the exact event onset times.