ARE FRONTIER LLMS READY FOR OUT-OF-THE-BOX ROBOT MANIPULATION?

Anonymous authors
Paper under double-blind review

ABSTRACT

Embodied AI systems are often organized into System 1 and System 2. System 1 is typically a pretrained policy that generates actions at high frequency, whereas System 2 is often instantiated as a vision-enabled language model for high-level planning. We ask whether a large language model (LLM) can act as the policy for robot manipulation without task-specific finetuning. We call this setting LLM as policy. We evaluate three LLMs on all 42 RoboDojo tasks and compare their scores with 40 public policies. Astra and GPT-5.5 use the official 50-episode-per-task protocol; DeepSeek-Flash uses 10 episodes per task. GPT-6 Astra achieves 22.48% average success rate and 28.97 Score over 2,100 trials, ranking above every public entry. Yet GPT-5.5 and DeepSeek-Flash reach only 0.88% and 1.92% average success rate with the same post-processing. We find that Astra exhibits a sharply polarized capability profile. It generalizes well to tasks that require semantic understanding but not high-precision control. In contrast, it performs poorly on tasks that require precision, dynamic control, or complex bimanual coordination. In-context experiments show no aggregate benefit from one-shot demonstrations, while selected interaction traces show within-episode corrections under perturbations. Overall, the evaluated LLMs vary substantially in manipulation performance. Astra stands out and provides initial evidence for the potential of a general-purpose manipulation model, although reliable precision and dynamic control remain limitations in the evaluated setting.

1 INTRODUCTION

Robot manipulation systems are commonly organized as a hierarchy between fast execution and slow deliberation (Kahneman, 2011). System 1 is typically a pretrained policy that maps observations and instructions to high-frequency actions (Chi et al., 2023; Zhao et al., 2023). Vision Language Action (VLA) models (Brohan et al., 2023; Kim et al., 2025; Black et al., 2024) and World Action Models (WAMs) (Ye et al., 2026; Kim et al., 2026) instantiate this role. System 2 is typically a vision-enabled language model that interprets the scene, selects subgoals, or writes programs (Driess et al., 2023; Liang et al., 2023). Practical systems combine the two: the language model decides what should happen, and a pretrained policy or scripted skill executes the decision. SayCan, Inner Monologue, Code as Policies, and VoxPoser follow this hierarchical pattern (Ahn et al., 2023; Huang et al., 2023b; Liang et al., 2023; Huang et al., 2023a).

This division has been challenged repeatedly. An early example used GPT-4 (OpenAI, 2023) to generate dense end-effector trajectories without motion primitives or robot fine-tuning, but it relied on separate detection and segmentation models and a 30-task evaluation (Kwon et al., 2024). More recent reports suggest that direct control is improving. Anthropic found that frontier models made increasing subgoal progress on LIBERO-40, although full-task success remained between 0 and 5.5% (Liu et al., 2023; Berman et al., 2026). Reports released with GPT-6 Astra (OpenAI, 2026) showed strong results on selected real-robot tasks, but used small custom task suites and different interfaces (Robocurve, 2026; Jia et al., 2026). These studies establish feasibility and motivate a sharper question, but they do not place frontier LLMs against a complete public policy leaderboard under one benchmark protocol.

We ask whether a frontier large language model (LLM) can itself serve as the policy for manipulation. We call this setting *LLM as policy*: every executable motion target is selected by the model, with only simple non-learned post-processing before execution. Real-robot tests alone cannot support the

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

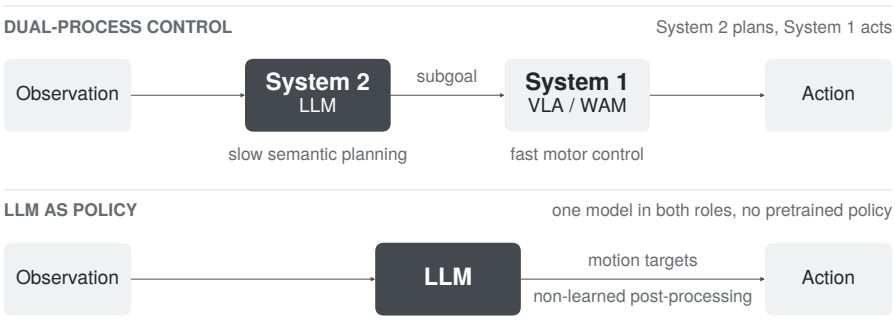


Figure 1: Two ways to turn an observation into an executable action. Top: a dual-process controller separates a language model for semantic planning from a pretrained motor policy for execution. VLA denotes a vision language action model; WAM denotes a world action model. Bottom: in the LLM-as-policy setting, the language model selects motion targets directly. Non-learned post-processing converts these targets into executable commands without a pretrained motor policy.

required comparison. They are slow, hard to scale, and unsafe actions can stop data collection; in our tests, such actions damaged equipment (Section 5). Deploying dozens of pretrained baselines on the same hardware is also impractical. We therefore use RoboDojo simulation for the primary evaluation and hardware for diagnostic evidence. RoboDojo provides 42 tasks, five capability axes, and a fixed 50-episode protocol, yielding 2,100 trials for each fully evaluated model and a public board of 40 pretrained policies (RoboDojo Team, 2026). Figure 1 contrasts the two system designs; Appendix H states the remaining input and serving differences.

The answer is model-specific. GPT-6 Astra reaches 22.48% average success rate (SR) and 28.97 Score over 2,100 trials, ranking above all 40 public policies. This confirms at benchmark scale the manipulation ability suggested by recent Astra reports. Yet GPT-5.5 and DeepSeek-Flash reach only 0.88% and 1.92% average SR with the same action interface and post-processing. This approximately 25-fold spread shows that the shared action conversion alone does not explain the performance difference. The positive results should not be generalized to all frontier LLMs. To our knowledge, this is the first complete public-benchmark evaluation to rank an LLM-as-policy controller against the benchmark's full public policy board.

We therefore analyze Astra as a case study. Its aggregate lead hides a sharply split capability profile: it is strongest on semantic and open-ended tasks, but remains weak on contact-rich, precision-critical, dynamic, and coordinated bimanual tasks (Section 4). In our one-shot experiments, demonstrations reduce aggregate success under the tested protocol. Separately, selected perturbation traces show Astra revising its actions after observing their outcomes (Section 4.5). These observations are consistent with within-episode adaptation, but do not isolate its mechanism or establish an advantage over pretrained policies. Astra thus challenges a strict System 1/System 2 division without eliminating the need for fast and precise pretrained control.

2 Related Work

Large pretrained robot policies. Vision language action (VLA) models map visual observations and language instructions to robot actions (Brohan et al., 2022; Open X-Embodiment Collaboration et al., 2024; Ghosh et al., 2024). RT-2 established web-scale vision–language transfer to control, OpenVLA provided an open 7B model, and π_0 paired a vision–language backbone with a flow-matching action expert (Brohan et al., 2023; Kim et al., 2025; Black et al., 2024). World action models (WAMs) additionally predict future visual states: DreamZero jointly generates video and actions, while Cosmos Policy represents future observations, actions, and values in a shared video-model latent space (Ye et al., 2026; Kim et al., 2026).

Language models as high-level planners. A second line uses a language model to guide a specialized controller. SayCan selects learned skills, Inner Monologue plans with language, Code as Policies writes programs over control APIs, and VoxPoser constructs spatial value maps for a motion planner (Ahn et al., 2023; Huang et al., 2023b; Liang et al., 2023; Huang et al., 2023a). Hi Robot

makes this hierarchy explicit by generating language subgoals for a low-level VLA policy (Shi et al., 2025). Helix instead passes a continuous semantic latent from a slow vision–language model to a 200 Hz visuomotor policy (Figure AI, 2025). In both designs, the slower model guides execution but does not issue the executable targets.

Language models as policies. Direct action generation predates the current generation of frontier models. GPT-4 generated dense end-effector trajectories without motion primitives or robot fine-tuning, but relied on separate detection and segmentation models (Kwon et al., 2024). Recent evaluations show that frontier models can also issue actions directly from robot observations, while reliability remains uneven across tasks (Berman et al., 2026; Ilie et al., 2026; Robocurve, 2026; Jia et al., 2026; Tsui et al., 2026). These studies establish feasibility, not a general replacement for pretrained control.

In-context robot learning. Prior work commonly treats a demonstration as the context from which a robot should infer a task. ICRT uses sensorimotor trajectories, RoboPrompt uses textual action examples, and HOST, Skild S1, and GEN-1.5 use one-shot human or sensorimotor demonstrations (Fu et al., 2024; Yin et al., 2024; Chen et al., 2026; Skild AI, 2026; Generalist, 2026). RoboTTT extends this paradigm with fast weights that absorb long demonstration or interaction histories (Jiang et al., 2026). A related line adapts from deployment interaction by inferring latent system configurations, conditioning on histories across trials, or optimizing latent prompts from interaction data (Wang et al., 2026; Liu et al., 2025; Zhang et al., 2026). Our perturbation experiments examine whether GPT-6 Astra revises actions in response to feedback within an episode, without updating its weights or carrying memory across episodes.

3 Study design: evaluating LLM as policy

3.1 What counts as LLM as policy

We define the LLM-as-policy setting using two rules. First, no learned policy appears in the control path between the language model and the robot. Second, the language model's weights remain fixed, with no robot-specific or task-specific fine-tuning performed for this evaluation. The main benchmark runs are zero-shot; Section 4.5 separately studies demonstrations and interaction history. A model fine-tuned to map robot observations to actions would instead fall under the conventional definition of a vision language action model. The model may act through either Cartesian end-effector targets or joint-space targets. For end-effector control, an inverse kinematics solver may convert the target into executable joint commands; this conversion is geometric only and does not involve a learned policy.

3.2 Our implementation

Prompt composition. Figure 2 summarizes our closed-loop implementation. Each episode starts with a fixed system message, Goal message, and tool definitions. The system message specifies the controller role, interaction rules, and embodiment. The Goal message combines the official task instruction with a *task recipe* derived from the RoboDojo wiki. The recipe describes the task and its scoring stages without providing an action sequence. The system message also provides task-independent operating advice, including using an idle wrist camera to inspect the work area.
Observation and context management. At each turn, the model receives the dynamic text history and the current observation. The text history retains all prior observation text, model-written notes, tool calls, and tool results. Each observation contains three RGB views, the remaining step budget, and the robot state. The state consists of 14 grasp-point dimensions and 12 read-only joint angles. RGB images are retained only for the two most recent observation turns.
Tools. The model must call either `move_eef` or `give_up` at every turn. `move_eef` requests a robot motion, while `give_up` ends an episode that the model judges unrecoverable. We do not provide pick, place, or pregrasp primitives because they would solve part of the task outside the model.
Action space. The `move_eef` target may specify any subset of the position, orientation, and gripper dimensions for either arm. Unspecified dimensions retain their observed values. Specifying both arms produces a simultaneous motion. Appendix A summarizes the tools, units, and bounds.
Action conversion and feedback. Non-learned software rejects invalid targets and attempts to plan a joint-space path to each valid target. Accepted paths are resampled for execution at 25 Hz. The

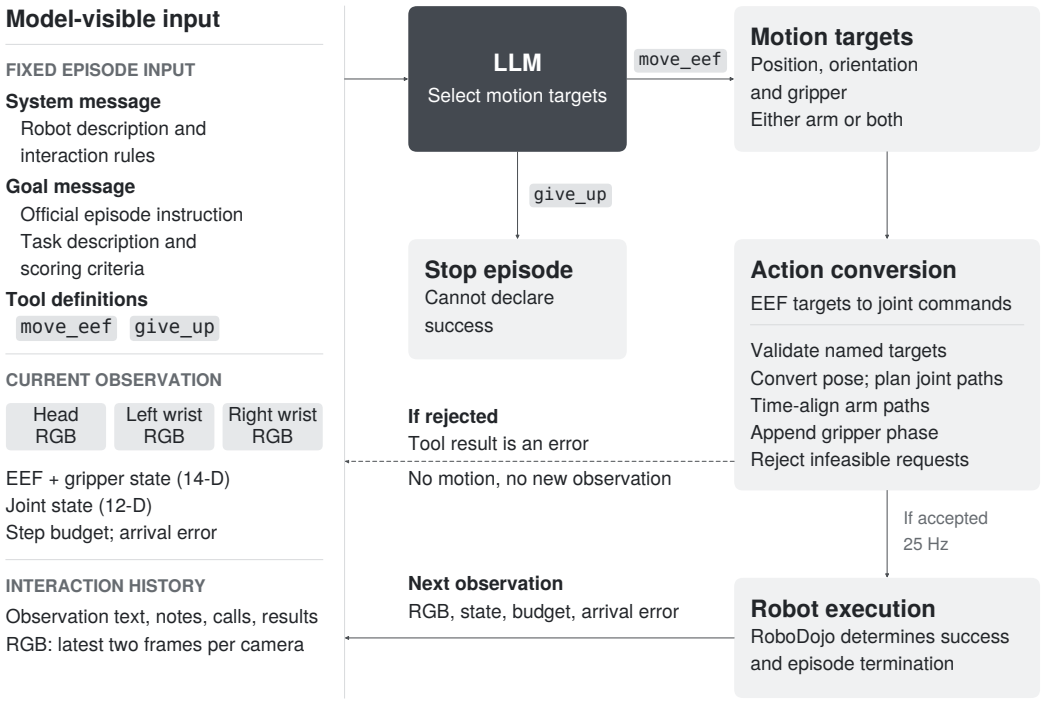


Figure 2: The model-facing context and execution loop. A frozen language model chooses world-frame end-effector (EEF) targets at the grasp point through `move_eef`. The action conversion module is non-learned. It validates the request, converts grasp-point targets to flange poses, plans joint paths, and constructs executable joint commands. Requested gripper changes follow the arm-motion phase. The two return routes are the two branches of a request. A rejected target returns an error as the tool result and causes no motion, so the model retries against the same observation; this is the dashed route. An accepted target returns its planned duration, executes, and only then produces the next observation with the achieved state and arrival error; this is the solid route. `give_up` requests termination, but only the environment determines success. The context retains the latest two RGB frames from each camera.

interface reports whether the target was accepted and the planned execution duration. RoboDojo executes the path and returns the next observation, including the discrepancy between the requested target and the measured end-effector state. This conversion is not a guarantee of collision-free or safe execution. Appendix A describes the model-facing feedback.

Information boundary and termination. The model receives no metric depth, privileged object pose, reward state, layout specification, or layout-specific solution. `give_up` does not declare success or change the reward. Success is determined only by the RoboDojo environment.

3.3 Evaluation protocol

The benchmark LLM runs use the same action space, post-processing, observation format, and context policy, with one evaluation seed per model. We evaluate GPT-6 Astra and GPT-5.5 for 50 episodes on each of the 42 tasks, following the official protocol (RoboDojo Team, 2026). This gives 2,100 trials per model. Because of evaluation cost, we evaluate DeepSeek-Flash for 10 episodes per task, giving 420 trials in total. For each Generalization task, it runs five standard and five randomized episodes. We mark all of its results with †. All benchmark LLM runs use medium reasoning effort and a default budget of 100 model calls per episode. We compare against the public large pretrained robot policies as recorded on the leaderboard on September 10, 2026 (Community et al., 2026).

Score is mean process reward multiplied by 100; SR is the percentage of episodes that satisfy the environment's full-success criterion. The leaderboard Average is the unweighted mean of the five axis-level values. It differs from pooling all episodes because the axes contain different numbers of tasks. Generalization cells average the standard and randomized conditions. The public policies were

not rerun with the wiki-derived text supplied to the LLMs, so the comparison shares outcome metrics but not identical input information.

4 How far can LLM as policy go?

4.1 GPT-6 Astra Leads Overall, While Other LLMs Lag Behind

Table 1 ranks the three LLM controllers alongside the 40 public policies in RoboDojo's official Score/SR% cell format. GPT-6 Astra takes rank 1 at 28.97 Score / 22.48% Average SR (micro $472/2100 = 22.48\%$ SR, 28.72 Score), ahead of all 40 public policy entries in this comparison. GPT-5.5 reaches 1.13/0.88% and DeepSeek-Flash 2.99†/1.92%†. The highest and lowest Average SR among the three LLMs differ by approximately 25-fold.

Table 1: RoboDojo-Sim board, Score/SR% per cell: the top ten of 43 ranked entries plus the two remaining LLM controllers, which rank 28 and 33. Ranks are over all 43; the public submission contains only the 40 policy rows, where the same order gives DM0.5 rank 1. † DeepSeek-Flash is 1 seed × 10 episodes per task, not 50. Policy rows are a 2026-09-10 leaderboard snapshot. The full 43-row board is Table 5.

Rank	Model	Average	Gen.	Prec.	Long-Hor.	Memory	Open
1	**GPT-6 Astra**	28.97/22.48	33.36/30.50	12.65/4.00	21.45/8.25	43.04/38.67	34.36/31.00
2	DM0.5	24.90/19.34	15.77/10.95	24.82/16.75	33.70/19.50	47.74/47.44	2.43/2.08
3	GalaxeaVLA (G0.5)	20.23/14.88	18.46/12.83	28.25/20.42	44.12/32.25	8.61/7.33	1.73/1.58
4	Xiaomi-Robotics-1	20.07/13.93	23.54/17.00	26.69/18.83	38.39/23.67	7.81/6.56	3.94/3.58
5	OpenWAM-α	17.18/11.92	20.71/14.83	18.45/9.25	34.93/25.33	10.41/9.11	1.41/1.08
6	Meituan-Robotics-0	14.95/9.53	13.75/8.17	16.77/7.75	29.61/18.58	10.06/8.89	4.54/4.25
7	Hy-Embodied-0.5-VLA	13.07/8.80	11.78/8.39	13.81/8.00	25.74/14.92	13.37/12.11	0.65/0.58
8	Spatial Forcing	12.38/8.04	14.12/9.34	17.32/10.58	23.26/14.58	5.43/4.11	1.78/1.58
9	Pi-05	11.41/6.91	13.38/8.17	12.40/5.50	23.54/14.67	5.78/4.56	1.98/1.67
10	InternVLA-A1.5	11.15/7.14	10.35/6.83	15.23/10.17	23.80/13.75	4.93/3.56	1.43/1.42
28	**DeepSeek-Flash†**	2.99/1.92	2.42/1.67	1.75/0.00	2.12/0.00	2.17/1.67	6.50/6.25
33	**GPT-5.5**	1.13/0.88	0.19/0.00	0.42/0.00	0.51/0.00	1.67/1.67	2.87/2.75

The remaining analyses focus on Astra as a case study rather than on LLMs in general.

4.2 Zero-shot Evaluation

4.2.1 A Heavily Imbalanced Policy

Astra achieves an Average SR of 22.48%, exceeding DM0.5, the highest-ranked public policy, by 3.14 percentage points. This lead is driven by Open and Generalization. In comparison, it performs worse than DM0.5 on Memory, Precision, and Long-Horizon tasks. Table 2 shows complementary task-level outcomes, including tasks on which Astra records no success despite high public-policy success rates. Its overall lead therefore reflects strengths in particular tasks alongside substantial gaps in others, rather than consistently reliable performance.

Table 2: Complementary task-level outcomes. Left: Astra reaches at least 20% SR while every public policy remains below 5%. Right: the strongest public policy reaches at least 20% while Astra remains below 5%. Xiaomi denotes Xiaomi-Robotics-1; G0.5, GalaxeaVLA (G0.5).

Astra ≥ 20%, policy < 5%			*Policy ≥ 20%, Astra < 5%*		
Task	Astra	Best policy	Task	Astra	Best policy
`push_T`	60.0	0.7 Xiaomi	`make_kong`	0.0	90.0 G0.5
`arrange_largest_number`	60.0	4.7 Xiaomi	`build_tower`	2.0	78.7 G0.5
`align_blocks`	50.0	0.0 DM0.5	`insert_tubes`	0.0	59.3 DM0.5
`solve_equation`	40.0	0.0 DM0.5	`play_tic_tac_toe`	0.0	58.7 OpenWAM
`stack_blocks_by_language`	40.0	2.7 DM0.5	`pour_balls_into_vase`	4.0	46.0 Xiaomi
`imitate_sorting_sequence`	36.0	0.7 Xiaomi	`pour_liquid_into_cup`	4.0	42.0 Xiaomi
`classify_objects_by_language`	30.0	0.7 Xiaomi	`store_laptop_and_headphones`	0.0	22.7 Xiaomi
			`fill_pen_holder`	0.0	22.0 Xiaomi

4.3 QUALITATIVE ZERO-SHOT CAPABILITIES

Astra exhibits several behaviours without a pretrained motor policy or scripted manipulation skill. Figure 3(a–c) illustrates these through three qualitative episodes:

Semantic understanding. In `make_kong`, Astra selects the tile from its own collection that matches the opponent's play.
Active perception. In `insert_tubes`, Astra uses active perception (Bajcsy, 1988) by repositioning the opposite wrist camera to view the rack when the held tube occludes the other camera. This behaviour follows the general active-view advice in the system prompt; it is not evidence of discovering that strategy without guidance.
Flexible use of both arms. In `classify_objects_by_language`, Astra holds a different object in each hand, demonstrating concurrent use of both arms (Zhao et al., 2023).

These examples establish that the behaviors occur, but do not quantify their frequency or contribution to task success.

4.4 PHYSICAL BOUNDARY.

Observed failures include errors in lift height, approach angle, or correction timing, even in episodes where the model appears to interpret the task correctly. Following Zeng & the Generalist Team (2026), we characterize these difficulties as limitations in *physical commonsense* (Battaglia et al., 2013): an implicit grasp of contact dynamics, force sensitivity, and collision geometry. These weaknesses qualify the interpretation of its aggregate rank, which alone does not establish that LLM-based policies can replace pretrained motor policies. The observations do not separate missing physical knowledge from limitations of perception or the action interface (Appendix H). Figure 3(d–f) illustrates three such limitations through qualitative failure episodes:

Dynamic control. In `pick_from_conveyor_by_image`, Astra misses a moving target during grasping, illustrating the challenge of coordinating motion with a changing scene.
Bimanual coordination. In `sweep_blocks`, a failed two-hand transfer shows that engaging both arms does not ensure coordinated manipulation.
Awareness of surrounding objects. In `build_tower`, Astra knocks over the existing structure while reaching for the next block, illustrating a failure to account for surrounding objects along its motion path.

These episodes identify failure modes that aggregate success rates alone do not distinguish.

4.5 IN-CONTEXT LEARNING FROM DEMONSTRATIONS AND INTERACTION

We examine in-context learning (ICL) (Brown et al., 2020; Min et al., 2022) through demonstrations and interaction histories, two forms of context studied in robot learning (Fu et al., 2024; Yin et al., 2024; Chen et al., 2026; Skild AI, 2026; Generalist, 2026; Jiang et al., 2026; Wang et al., 2026; Liu et al., 2025; Zhang et al., 2026). One-shot demonstrations do not improve Astra's aggregate performance in our probe. Selected interaction traces instead show corrections within an episode. These experiments remain a case study of Astra, not a claim about LLMs in general.

One-shot demonstrations do not improve aggregate performance here. We first test demonstration-conditioned ICL in a one-shot setting. The model receives one example of the desired behaviour. We evaluate 340 matched task–layout pairs across 34 tasks. The example comes from a different layout of the same task. One condition provides images and corresponding end-effector poses. The other describes the same trajectory in text. The zero-shot baseline uses the same 340 pairs from the official campaign. Appendix C gives protocol details and task-level evidence.

Neither demonstration condition improves success over the matched 22.9% zero-shot baseline (Table 3). The task-level split shows that demonstrations rarely produce successes on tasks with no success in the matched zero-shot runs: the image condition succeeds in 2 of these 150 episodes, and the text condition in none. The aggregate decline instead comes from losing episodes on tasks with zero-shot successes (Appendix C.3).

One possible explanation is that a demonstration adds little when execution precision, rather than task interpretation, limits performance. Another is that the model transfers layout-specific geometry from the example to the current scene. These are hypotheses: the experiment does not isolate either cause from effects of demonstration format, context length, or stochastic variation.

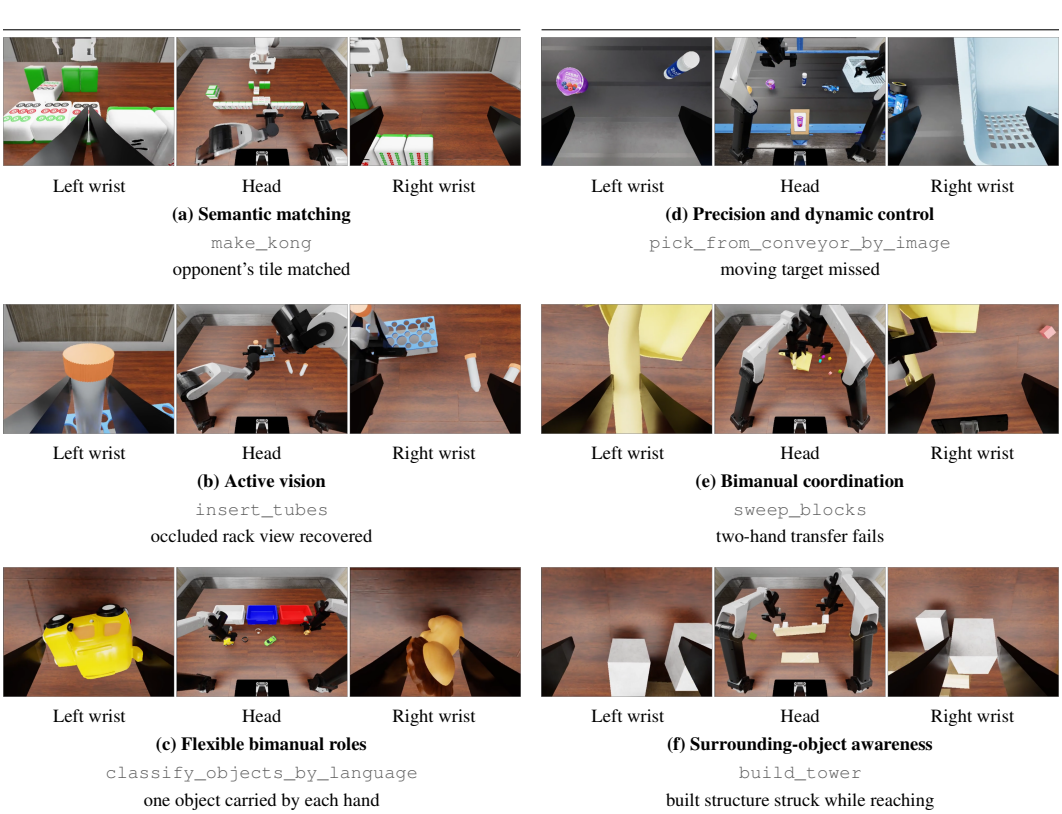


Figure 3: Capabilities and limits of Astra used as a policy. Each panel shows one episode in three camera views; the task and observed evidence appear below. Panels (a)–(c) show capabilities and panels (d)–(f) show limits. These examples establish occurrence, not frequency.

Table 3: Demonstration ICL on 340 matched task–layout pairs: the same 34 tasks and ten layouts per task under each condition. The two demonstration conditions are new runs of 340 episodes each; the zero-shot baseline is the official 50-episode Astra campaign restricted to those exact pairs.

Condition	Episodes	SR	vs zero-shot
Zero-shot (baseline)	340	78/340 = 22.9%	—
Image + end-effector demonstration	340	61/340 = 17.9%	−5.0
Text demonstration	340	44/340 = 12.9%	−10.0

Within-episode correction under perturbations. We next probe whether Astra revises actions in response to interaction feedback, without a demonstration. We apply six perturbations to eight `general_pickup` layouts. The perturbations alter visual observations, action execution, or the mapping of Cartesian coordinates to executed motion. The model weights and context policy remain fixed. No explicit explanation of the perturbation is added, but the negated-coordinate condition changes the bounds shown in the tool description. Appendix D defines this information boundary and the protocol.

Table 4 reports task success under these perturbations. Astra retains some success in every condition. Success alone does not distinguish existing robustness, repeated attempts, and adaptation from feedback. We therefore examine selected traces for changes in the model's actions and stated interpretation. The small probe does not establish the relative difficulty of the conditions.

Figure 4 shows what the model receives under three of these conditions, at an early and a later turn of the same episode. In row (a) the model writes down, between the two turns, why it moves the grasp to the other hand. That note documents a change in the model's stated arm assignment. Row (c) shows the same loop under pose jitter: an early command lands far to the right of the object,

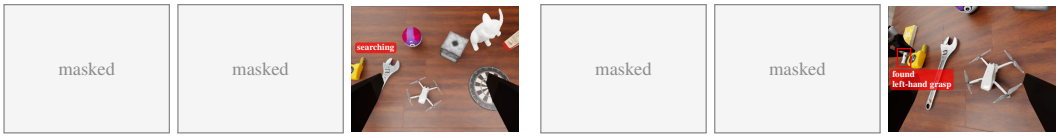
Table 4: Performance under perturbations on eight `general_pickup` layouts. Visual perturbations transform or remove camera views. Per-move pose jitter offsets each executed pose by a random displacement with mean magnitude 10 cm. Negated Cartesian axes change the coordinate convention and the advertised target bounds. The layouts are the subset Astra solves without perturbation, and each cell contains one episode per layout.

Perturbed component	Condition	Successes
LLM as policy (GPT-6 Astra)		
None	No perturbation	8/8
Visual observation	Top–bottom image flip	8/8
Visual observation	Left–right image mirror	6/8
Visual observation	No head-camera view	6/8
Visual observation	Right-wrist view only	3/8
Action execution	Per-move pose jitter	3/8
Coordinate mapping	Negated Cartesian axes	4/8



(a) Left–right image mirror — `general_pickup` layout 0, pick up the scissors

Reasoning written by the model between the two turns: *"The camera views show that the **right arm** is the one beside the scissors. I am restoring the idle **left arm** and preparing the **right hand** for a downward grasp."*



(b) Right-wrist view only — `general_pickup` layout 9

The model sweeps the right arm to search for the object with its wrist camera, then grasps with the left hand.



(c) Per-move pose jitter — `general_pickup` layout 0

Every executed move is perturbed, and the model corrects until the grasp succeeds.

Figure 4: The observations the model receives under three perturbations, defined in Appendix D.2. Each row is one condition, named under its frames; the left group of three views is an early turn and the right group a later turn of the same episode. The line under row (a) is the model's own reasoning from the episode, and the lines under rows (b) and (c) describe an observed behaviour instead. Row (a) shows real frames from the `flip_vision_lr` episode on layout 0. The mirror reverses apparent left–right positions without changing camera identities. Under (b) only the right wrist camera is available, and the masked cells mark the views the model does not receive. Row (b) comes from the supplementary rerun with 170 calls and 400 environment steps; the standard-budget run failed. Under (c) every executed move is offset by a random displacement of mean magnitude 10 cm. The model receives both images and motion feedback; their contributions are not isolated (Appendix D.4).

and later attempts lead to a successful grasp. Row (b) is a supplementary larger-budget rerun, not a standard-budget success in Table 4.

Under Negated Cartesian axes, one successful episode includes a note that the model's forward/back estimate was reversed. The note explicitly refers to the wrist view. Table 10 reproduces selected calls from that episode. These observations are consistent with within-episode adaptation, but do not identify which inputs caused the correction or establish complete identification of the coordinate mapping.

5 LLM AS POLICY ON REAL HARDWARE

We did not complete the official RoboDojo-Real protocol with GPT-6 Astra (RoboDojo Team, 2026). Testing was halted after the model repeatedly issued physically unreasonable or unsafe actions, including incidents that damaged hardware; no person was injured. The hardware material is therefore a selective sample rather than an official result, and it cannot establish Astra's general real-world reliability.

We keep the remaining trials as diagnostic material. They suggest that Astra interpreted the tasks more reliably than it executed them safely, which is a qualitative observation rather than a benchmark result. Appendix E reports the retained trials, their scores, the embodiments, and the separate joint-position deployments. Section 6 reports two further evaluations of Astra that also fall outside the official RoboDojo protocol.

6 BEYOND TABLETOP MANIPULATION

The official RoboDojo suite is tabletop manipulation. We also ran two evaluations of GPT-6 Astra outside that setting. Neither is part of the official RoboDojo protocol, and neither produces a Score that belongs on the Section 4 board. The humanoid recordings are qualitative; the piano runs include task-specific scores. Neither establishes a general performance level beyond the tested runs.

Mobile humanoid grasping. Astra was also deployed through a separate mobile-humanoid interface. The recordings show walking followed by grasping, and a longer sequence of walking, sitting, and picking while seated. No success rate or process score was computed for either. Appendix F records the action vocabulary, horizons, and clips. We do not use these recordings to establish compliance with the strict LLM-as-policy definition in Section 3.1.

Bimanual piano playing. A second evaluation asks Astra to play on an 88-key piano. Here it does not emit one motion target per turn. It writes a controller that outputs 45-D joint commands every 0.05 s for two Shadow hands. The LLM's weights remain fixed, but it refines its code through simulation trials without access to a reference solution. These runs use a piano-performance F1 metric, which is not comparable with a RoboDojo Score. Appendix G reports the isolation rules, practice runs, and whether the delivered program survives a change of music.

7 CONCLUSION

This study evaluates LLM as policy: a language model selects robot motion targets without a pretrained motor policy in the control path. Across 42 RoboDojo tasks, GPT-6 Astra achieves 28.97 Score and 22.48% average SR, ranking above the 40 public policies in our comparison. GPT-5.5 and DeepSeek-Flash perform substantially worse with the same interface, so the result does not generalize to LLMs as a class. Astra's lead on Open and Generalization coexists with large gaps in precision and coordinated manipulation. Unsafe actions halted our real-robot trials. Aggregate benchmark performance therefore does not establish reliable or safe physical control.

The context experiments suggest that learning from demonstrations and correcting actions during execution should be evaluated separately. A single demonstration from another layout reduces aggregate success in our matched trials. Yet under visual and action perturbations, selected episodes show Astra revising spatial judgments and later actions after seeing the outcome of a motion. These traces are preliminary evidence of within-episode adaptation. A next step is to test whether such corrections transfer across tasks and embodiments. Turning them into reliable, precise execution remains a central challenge for LLMs used as policies.

ETHICS STATEMENT

Real-robot evaluation was stopped for safety. During those trials the model repeatedly issued physically unreasonable or unsafe actions, and some incidents damaged equipment; no person was injured. These incidents show that the tested system was not sufficiently safe for continued hardware evaluation. They motivate independent safety checks and stopping mechanisms rather than reliance on the model's judgment alone. Any safety intervention should be documented when interpreting

controller performance. The retained 33 diagnostic trials are a selected sample, not a completed benchmark evaluation (Section 5, Appendix H).

The study evaluates commercial models on a public manipulation benchmark and involves no human subjects and no personal data.

USE OF AI TOOLS

The evaluated controllers are commercial large language models. We used them as the systems under study, through the documented API and tool interface. We also used generative AI tools to edit wording and to format LaTeX and tables. We did not use generative AI tools to generate datasets, proofs, or theoretical claims. Authors reviewed all AI-assisted text. We take responsibility for the final content of this work.

REFERENCES

Michael Ahn et al. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. In *Proc. CoRL*, volume 205 of *PMLR*, 2023. URL https://proceedings.mlr.press/v205/ichter23a.html.

Ruzena Bajcsy. Active Perception. *Proceedings of the IEEE*, 76(8):966–1005, 1988.

Peter W. Battaglia, Jessica B. Hamrick, and Joshua B. Tenenbaum. Simulation as an Engine of Physical Scene Understanding. *Proceedings of the National Academy of Sciences*, 110(45): 18327–18332, 2013.

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

Kevin Black et al. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan et al. RT-1: Robotics Transformer for Real-World Control at Scale. *arXiv preprint arXiv:2212.06817*, 2022. URL https://arxiv.org/abs/2212.06817.

Anthony Brohan et al. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In *Proc. CoRL*, volume 229 of *PMLR*, 2023. URL https://arxiv.org/abs/2307.15818.

Tom Brown et al. Language Models are Few-Shot Learners. In *Adv. Neural Inf. Process. Syst.*, 2020. URL https://arxiv.org/abs/2005.14165.

Guangyan Chen, Meiling Wang, Te Cui, Zichen Zhou, Qi Shao, Xiaofan Li, Hang Su, Ruyi Gan, Hao Wang, Mengyin Fu, Yi Yang, and Yufeng Yue. Robots Acquire Manipulation Skills in Seconds from a Single Human Video. *arXiv preprint arXiv:2607.20033*, 2026. URL https://arxiv.org/abs/2607.20033.

Cheng Chi et al. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Proc. RSS*, 2023. URL https://arxiv.org/abs/2303.04137.

XPolicyLab Community, Tianxing Chen, Yue Chen, Tian Nian, et al. XPolicyLab: A Unified Standard and Open Ecosystem for Robot Policy Evaluation and Deployment. *arXiv preprint arXiv:2608.09892*, 2026. URL https://arxiv.org/abs/2608.09892.

Danny Driess et al. PaLM-E: An Embodied Multimodal Language Model. In *Proc. ICML*, volume 202 of *PMLR*, 2023. URL https://arxiv.org/abs/2303.03378.

Figure AI. Helix: A Vision-Language-Action Model for Generalist Humanoid Control. Technical report, February 2025. URL https://www.figure.ai/news/helix.

Max Letian Fu, Huang Huang, Gaurav Datta, Lawrence Yunliang Chen, Will 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. GEN-1.5: Embodied Foundation Models are One-Shot Learners. Technical report, August 2026. URL https://generalistai.com/blog/gen-1.5.

Dibya Ghosh et al. Octo: An Open-Source Generalist Robot Policy. In *Proc. RSS*, 2024. URL https://roboticsproceedings.org/rss20/p090.html.

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 *Proc. CoRL*, volume 229 of *PMLR*, pp. 540–562, 2023a. URL https://proceedings.mlr.press/v229/huang23b.html.

Wenlong Huang et al. Inner Monologue: Embodied Reasoning through Planning with Language Models. In *Proc. CoRL*, volume 205 of *PMLR*, 2023b. URL https://proceedings.mlr.press/v205/huang23c.html.

Michael Ilie, C. Daniel Freeman, and Kevin K. Troy. Project Fetch: Phase Two. Anthropic Research, June 2026. URL https://www.anthropic.com/research/project-fetch-phase-two.

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.

Yunfan Jiang, Yevgen Chebotar, Ruijie Zheng, Fengyuan Hu, Yunhao Ge, Jimmy Wu, Tianyuan Dai, Scott Reed, Li Fei-Fei, Yuke Zhu, and Linxi Jim Fan. RoboTTT: Context Scaling for Robot Policies. *arXiv preprint arXiv:2607.15275*, 2026. URL https://arxiv.org/abs/2607.15275.

Daniel Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2011.

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.

Moo Jin Kim et al. OpenVLA: An Open-Source Vision-Language-Action Model. In *Proc. CoRL*, volume 270 of *PMLR*, 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Teyun Kwon, Norman Di Palo, and Edward Johns. Language Models as Zero-Shot Trajectory Generators. *IEEE Robotics and Automation Letters*, 2024. doi: 10.1109/LRA.2024.3410155. URL https://arxiv.org/abs/2310.11604.

Jacky Liang et al. Code as Policies: Language Model Programs for Embodied Control. In *Proc. ICRA*, 2023. URL https://arxiv.org/abs/2209.07753.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. In *Adv. Neural Inf. Process. Syst.*, 2023. URL https://arxiv.org/abs/2306.03310.

Min Liu, Deepak Pathak, and Ananye Agarwal. LocoFormer: Generalist Locomotion via Long-context Adaptation. In *Proc. CoRL*, volume 305 of *PMLR*, pp. 532–546, 2025. URL https://proceedings.mlr.press/v305/liu25a.html.

Sewon Min et al. Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? In *Proc. EMNLP*, 2022. URL https://arxiv.org/abs/2202.12837.

Open X-Embodiment Collaboration et al. Open X-Embodiment: Robotic Learning Datasets and RT-X Models. In *Proc. ICRA*, 2024. URL https://arxiv.org/abs/2310.08864.

OpenAI. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023. URL https://arxiv.org/abs/2303.08774.

OpenAI. GPT-6 Astra: A New Generation of Intelligence. OpenAI, September 2026. URL https://openai.com/index/gpt-6-astra/.

Robocurve. GPT-6 Astra on Robotic Manipulation. Robocurve report, September 2026. URL https://openai.robocurve.org/gpt-6-astra/.

RoboDojo Team. 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.

Lucy Xiaoyang Shi et al. Hi Robot: Open-Ended Instruction Following with Hierarchical Vision-Language-Action Models. In *Proc. ICML*, volume 267 of *PMLR*, pp. 54919–54933, 2025. URL https://proceedings.mlr.press/v267/shi25d.html.

Skild AI. Introducing S1: In-Context Learning for Robotics. Technical report, August 2026. URL https://skild.ai/blogs/s1.

Brian Y. Tsui, Alan Y. Fang, and Tiffany J. Hwu. Demonstration-Free Robotic Control via LLM Agents. *arXiv preprint arXiv:2601.20334*, 2026. URL https://arxiv.org/abs/2601.20334.

Siyin Wang, Junhao Shi, Senyu Fei, Zhaoyang Fu, Li Ji, Jingjing Gong, and Xipeng Qiu. In-Context World Modeling for Robotic Control. *arXiv preprint arXiv:2606.26025*, 2026. URL https://arxiv.org/abs/2606.26025.

Seonghyeon Ye et al. World Action Models are Zero-shot Policies. *arXiv preprint arXiv:2602.15922*, 2026. URL https://arxiv.org/abs/2602.15922.

Yida Yin, Zekai Wang, Yuvan Sharma, Dantong Niu, Trevor Darrell, and Roei Herzig. In-Context Learning Enables Robot Action Prediction in LLMs. *arXiv preprint arXiv:2410.12782*, 2024. URL https://arxiv.org/abs/2410.12782.

Kevin Zakka et al. RoboPianist: Dexterous Piano Playing with Deep Reinforcement Learning. In *Proc. CoRL*, volume 229 of *PMLR*, pp. 2975–2994, 2023. URL https://proceedings.mlr.press/v229/zakka23a.html.

Andy Zeng and the Generalist Team. The Dark Matter of Robotics: Physical Commonsense. Generalist blog, January 2026. URL https://generalistai.com/blog/physical-commonsense.

Wenbo Zhang, Jianxiong Li, Shuai Yang, Sijin Chen, Jiajun Liu, Lingqiao Liu, and Xiao Ma. TTT-VLA: Test-Time Latent Prompt Optimization for Vision-Language-Action Models. *arXiv preprint arXiv:2606.03127*, 2026. URL https://arxiv.org/abs/2606.03127.

Tony Z. Zhao et al. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proc. RSS*, 2023. URL https://arxiv.org/abs/2304.13705.

A Prompt and tool interface

This appendix makes the model-facing interface explicit. An episode begins with one system message and one Goal message. Every control turn then appends one multimodal observation, one assistant tool call, and one tool result. All text remains in context. Images remain only on the two most recent observation turns.

System message. The header below is abridged only where marked. The omitted embodiment notes contain task-independent geometry and operating facts: world axes, arm bases, table height, reach, gripper polarity, active-view advice, contact-step advice, and the requirement to return both arms to their starting poses when the benchmark scores it.

```
You are controlling a real robot embodiment named 'robodojo-arx-x5'.
You receive RGB camera images, the current world-frame grasp-point
state of both arms in the same 14 dimensions move_eef takes, arm joint
angles as context you cannot command, and a task instruction.
Move with move_eef by naming only the world-frame dimensions you want
to change. Cartesian targets must be estimated from RGB; no depth or
world-coordinate query is available. Respond with exactly one tool
call per turn. After each motion the next observation reports how far
the grasp point ended up from what you asked for, so check it before
assuming a motion landed. Two budgets run down at once and whichever
empties first ends the episode: 100 LLM calls, one per turn, and the
environment's own step limit, reported with each observation as the
env steps remaining. A motion spends env steps in proportion to how
far it travels, so a small correction is cheap and a long reach is not.

Embodiment notes:
<task-independent robot geometry and operating notes>
```

Goal message. The following shows the task-level content for `general_pickup`, apart from the episode-specific instruction shown as a placeholder. All 42 tasks use the same template and a task-specific wiki recipe.

```
Goal: <official episode instruction>

TASK RECIPE:
# General Pickup

Official RoboDojo wiki capability dimension, Description, and
process-score ladder. The live Goal instruction is still authoritative
for instance-specific slots. RoboDojo's reward judges the episode;
these rows are the environment's partial-credit scores, not a
substitute for official success.

## Capability dimension
Open -- Open-ended or language/image-conditioned manipulation tasks.

## Description
There are multiple objects. The robot needs to understand the language
instruction, identify the target object, and pick it up.

## Scoring
0: The target object is not lifted high enough.
100: The target object is lifted at least 10 cm.
```

Interaction history. At turn t, the API receives the system and Goal messages followed by the alternating observations, assistant tool calls, and tool results from turns $1, \ldots, t-1$. The current observation is last. When `image_horizon` $= 2$, images in older observation messages are replaced by the text `[earlier camera image omitted to save context]`; their state text is unchanged.

Observation message. Values and camera pixels change each turn; the field structure does not. The 14 grasp-point values are absolute targets in the same coordinate system used by `move_eef`. Joint angles are included only as read-only proprioception.

```
Instruction: <official episode instruction>
World-frame grasp-point state (metres; degrees from straight down):
left_x=<v>  left_y=<v>  left_z=<v>
left_pitch_deg=<v>  left_roll_deg=<v>  left_yaw_deg=<v>
```

```
left_gripper=<v>
right_x=<v>  right_y=<v>  right_z=<v>
right_pitch_deg=<v>  right_roll_deg=<v>  right_yaw_deg=<v>
right_gripper=<v>
Arm joint angles, radians, for context; not commandable:
left_joint1=<v> ... left_joint6=<v>
right_joint1=<v> ... right_joint6=<v>
Arrival check for the previous move_eef target: <optional error>.
Env steps remaining before the episode ends: <n>
camera 'head' (step <t>): <RGB JPEG>
camera '<left wrist>' (step <t>): <RGB JPEG>
camera '<right wrist>' (step <t>): <RGB JPEG>
```

Tools. The model must return exactly one of the following calls. The `targets` object must be nonempty, but may contain any subset of the 14 dimensions. Unnamed dimensions hold their observed values.

```
move_eef({
  "targets": {"<dimension>": <number>, ...},
  "note": "<current observation and reason for this motion>"
})

give_up({
  "reason": "<why the episode is unrecoverable>",
  "hindsight": "<what would have been needed>"
})
```

Position units are metres. The position bounds are `left_x`$\in [-1.1, 0.5]$, `right_x`$\in [-0.5, 1.1]$, `left/right_y`$\in [-1.25, 0.35]$, and `left/right_z`$\in [0.7, 1.565]$. For either arm, `pitch_deg`$\in [-180, 180]$, `roll_deg`$\in [-90, 90]$, `yaw_deg`$\in [-180, 180]$, and `gripper`$\in [0, 1]$, where 0 is closed and 1 is open. Zero orientation denotes a straight-down grasp. An invalid or unreachable request returns an error as the tool result and does not move the robot.

Tool result and next observation. A valid call first adds a result of the following form to the textual history:

```
Accepted: playing <N> waypoints (<seconds>s).
The next observation reports how close the grasp point landed.
```

An invalid target instead returns a specific error, such as `target for 'left_z' is outside [0.7, 1.565]`, and leaves the robot stationary while the model chooses another target. After an accepted chunk is executed, RoboDojo supplies the next three RGB images, robot state, and remaining step budget. The interface formats these as the next observation message and adds an arrival check such as `left 20 mm and 0.0 deg away`. Thus the tool result reports whether the request was executable, while the next observation reports what physically happened.

B Full simulation boards

Table 5 is the complete 43-row RoboDojo-Sim board that Table 1 excerpts. Table 6 is the official Per-Task board plus the two LLM-controller columns from the 50-episode single-seed dump and the DeepSeek-Flash † column from the 10-episode campaign; Generalization cells are the mean of Gen-Std and Gen-Rand. The complementary task subsets are reported in Table 2 in the main text.

DeepSeek-Flash records successes on four tasks: `general_pickup` (30% SR), `stack_blocks` (20%), `align_blocks` (20%), and `match_and_pick_from_conveyor` (10%). These SR values differ from the process-reward Scores in Table 6.

C Details of one-shot in-context learning

This appendix records the protocol, condition definitions, and task-level evidence for the one-shot demonstration-conditioned ICL experiment in Section 4.5 and Table 3.

Table 5: RoboDojo-Sim board in the official Score/SR% cell format: the 40 public policies plus GPT-6 Astra, DeepSeek-Flash, and GPT-5.5, sorted by Average Score and ranked 1–43 together. Ranking the three LLM controllers alongside the board makes the comparison legible; the public submission itself contains only the 40 policy rows, where the same order gives DM0.5 rank 1. † DeepSeek-Flash is 1 seed $\times$ 10 episodes per task, not 50. Snapshot of the policy rows taken 2026-09-10 from the public leaderboard.

Rank	Model	Average	Gen.	Prec.	Long-Hor.	Memory	Open
1	**GPT-6 Astra**	28.97/22.48	33.36/30.50	12.65/4.00	21.45/8.25	43.04/38.67	34.36/31.00
2	DM0.5	24.90/19.34	15.77/10.95	24.82/16.75	33.70/19.50	47.74/47.44	2.43/2.08
3	GalaxeaVLA (G0.5)	20.23/14.88	18.46/12.83	28.25/20.42	44.12/32.25	8.61/7.33	1.73/1.58
4	Xiaomi-Robotics-1	20.07/13.93	23.54/17.00	26.69/18.83	38.39/23.67	7.81/6.56	3.94/3.58
5	OpenWAM-α	17.18/11.92	20.71/14.83	18.45/9.25	34.93/25.33	10.41/9.11	1.41/1.08
6	Meituan-Robotics-0	14.95/9.53	13.75/8.17	16.77/7.75	29.61/18.58	10.06/8.89	4.54/4.25
7	Hy-Embodied-0.5-VLA	13.07/8.80	11.78/8.39	13.81/8.00	25.74/14.92	13.37/12.11	0.65/0.58
8	Spatial Forcing	12.38/8.04	14.12/9.34	17.32/10.58	23.26/14.58	5.43/4.11	1.78/1.58
9	Pi-05	11.41/6.91	13.38/8.17	12.40/5.50	23.54/14.67	5.78/4.56	1.98/1.67
10	InternVLA-A1.5	11.15/7.14	10.35/6.83	15.23/10.17	23.80/13.75	4.93/3.56	1.43/1.42
11	StarVLA-PI_v3	10.81/7.51	11.22/8.05	17.77/12.50	18.46/11.00	4.59/4.00	2.03/2.00
12	VLAct	10.65/7.58	9.54/6.28	20.57/15.17	20.12/13.67	0.66/0.56	2.37/2.25
13	X-VLA	10.13/6.52	10.47/6.78	18.32/12.00	16.53/9.75	4.76/3.56	0.55/0.50
14	MolmoAct2	8.99/5.03	8.53/4.67	10.92/4.50	15.20/7.00	5.53/4.56	4.75/4.42
15	StarVLA-GR00T	8.25/4.67	7.96/5.28	9.56/4.67	16.18/7.17	5.60/4.33	1.95/1.92
16	StarVLA-OFT	8.18/4.82	7.41/4.78	16.08/10.17	14.28/6.17	1.74/1.67	1.41/1.33
17	X-WAM	7.69/3.83	7.39/3.33	6.72/1.83	17.47/9.08	6.32/4.67	0.57/0.25
18	Xiaomi-Robotics-0	6.93/4.18	7.43/5.55	8.42/4.58	13.51/6.92	5.07/3.67	0.22/0.17
19	StarVLA	6.40/3.24	3.94/2.33	9.90/4.33	14.15/6.50	3.34/2.44	0.67/0.58
20	GigaWorld-Policy-0	6.20/3.27	5.34/2.89	6.15/1.83	15.51/8.92	3.46/2.22	0.54/0.50
21	GalaxeaVLA (G0)	5.82/2.96	4.54/2.83	8.10/3.83	12.60/5.58	3.17/1.89	0.70/0.67
22	LingBot-VLA	5.50/2.96	6.71/4.28	5.33/1.83	10.89/5.25	3.82/2.78	0.72/0.67
23	EventVLA	4.97/2.81	3.95/1.94	10.13/5.75	5.05/0.83	4.92/4.78	0.80/0.75
24	AHA-WAM	4.82/2.39	5.79/3.27	5.86/2.42	8.61/2.67	2.97/2.78	0.88/0.83
25	ABot-M0	3.67/1.73	5.73/3.50	5.50/1.75	3.96/0.50	2.44/2.22	0.72/0.67
26	Fast-WAM	3.48/2.03	2.33/1.11	1.96/0.00	9.14/5.17	3.55/3.44	0.42/0.42
27	Pi-0	3.48/1.53	3.94/2.55	3.56/0.75	6.19/2.00	3.47/2.11	0.25/0.25
28	**DeepSeek-Flash†**	2.99/1.92	2.42/1.67	1.75/0.00	2.12/0.00	2.17/1.67	6.50/6.25
29	GROOT-N1.7	2.85/1.31	2.16/1.22	2.54/0.67	8.30/3.58	1.06/0.89	0.18/0.17
30	InternVLA-A1	2.48/1.08	2.86/1.83	3.00/0.92	4.79/1.17	1.58/1.33	0.17/0.17
31	SmolVLA (Single Task)	1.82/0.85	1.68/1.22	2.87/0.33	1.22/0.25	3.35/2.44	0.00/0.00
32	LDA-1B	1.66/0.54	0.81/0.11	3.33/0.67	1.98/0.08	2.11/1.78	0.08/0.08
33	**GPT-5.5**	1.13/0.88	0.19/0.00	0.42/0.00	0.51/0.00	1.67/1.67	2.87/2.75
34	ACT (Single Task)	1.08/0.47	0.69/0.56	0.85/0.00	1.73/0.92	2.15/0.89	0.00/0.00
35	GO-1	0.99/0.53	1.58/1.22	1.45/0.42	1.13/0.25	0.70/0.67	0.08/0.08
36	H-RDT	0.67/0.12	0.49/0.22	0.41/0.00	2.23/0.17	0.12/0.11	0.08/0.08
37	RDT	0.51/0.13	0.55/0.33	0.38/0.00	1.13/0.00	0.49/0.33	0.00/0.00
38	DM0	0.45/0.05	0.49/0.06	0.61/0.00	0.97/0.08	0.20/0.11	0.00/0.00
39	Dexora-1B	0.38/0.02	0.49/0.11	0.49/0.00	0.82/0.00	0.12/0.00	0.01/0.00
40	A1	0.28/0.02	0.16/0.00	0.09/0.00	1.07/0.00	0.00/0.00	0.08/0.08
41	Spirit v1.5	0.23/0.14	0.80/0.50	0.03/0.00	0.11/0.00	0.22/0.22	0.00/0.00
42	TinyVLA	0.22/0.07	0.03/0.00	0.05/0.00	0.67/0.00	0.11/0.11	0.25/0.25
43	OpenVLA-OFT	0.21/0.02	0.04/0.00	0.20/0.00	0.70/0.00	0.00/0.00	0.08/0.08

Table 6: Per-task Score on RoboDojo-Sim (process-reward mean $\times$ 100). Public columns copied from the official Per-Task board (2026-09-10); only the strongest four public policies are shown here. Astra and GPT-5.5: 1 seed, 50 episodes per task. † DeepSeek-Flash: 1 seed, 10 episodes per task. Generalization cells are the mean of Gen-Std and Gen-Rand. Best value per row in bold.

Task	GPT-6 Astra	GPT-5.5	DS-Flash †	DM0.5	Galaxea G0.5	Xiaomi-1	OpenWAM-α
Generalization							
stack_bowls	**66.5**	0.3	3.0	30.5	42.2	56.4	49.9
push_T	**60.0**	0.0	0.0	0.0	0.0	0.7	0.0
pack_objects_into_box	11.6	0.2	2.0	15.1	18.2	19.9	**22.5**
fold_clothes	**76.4**	1.2	2.0	32.5	36.3	47.3	57.5
hang_mugs	0.9	0.0	0.0	9.0	8.6	15.7	**16.3**
sweep_blocks	**4.0**	0.0	0.0	4.0	2.0	2.7	1.3
pour_liquid_into_cup	4.0	0.0	0.0	34.7	35.3	**42.0**	34.7
make_toast	0.5	0.0	0.0	4.5	12.0	**12.7**	12.5
arrange_largest_number	**68.7**	0.0	0.5	9.6	4.7	10.4	4.9
sort_nesting_dolls_by_size	**18.0**	0.0	0.0	10.7	7.3	13.3	4.7
store_laptop_and_headphones	1.6	0.0	0.0	16.9	28.3	**44.4**	30.5
stack_blocks	**88.1**	0.6	21.5	21.8	26.5	17.1	13.7
Precision							
fasten_screws	36.4	0.0	2.0	**40.2**	30.0	22.9	24.7
insert_tubes	6.0	0.0	0.0	**71.7**	58.5	50.0	26.3
plug_in_charger	0.0	0.0	0.0	2.7	0.7	**4.0**	4.0
pour_balls_into_vase	4.0	0.0	0.0	9.3	28.0	**46.0**	17.3
play_Xylophone	**8.0**	0.0	0.0	0.7	0.0	0.0	0.0
deposit_coin	16.0	1.6	4.0	13.9	10.9	**24.7**	9.1
insert_key	14.4	1.8	6.0	4.9	**14.9**	13.3	13.7
build_tower	16.4	0.0	2.0	55.2	**82.9**	52.6	52.5
Long-Horizon							
put_bottles_into_dustbin	35.3	1.3	2.0	81.7	96.3	**97.7**	94.0
play_tic_tac_toe	10.7	0.8	5.0	35.8	**65.2**	36.8	64.1
classify_objects	**69.0**	0.3	3.0	26.8	10.3	17.2	5.5
fill_pen_holder	5.0	0.0	0.0	22.1	41.3	**44.7**	15.4
fill_egg_holder	5.5	0.2	2.0	1.4	3.0	**11.7**	5.8
organize_table	39.5	1.5	5.0	44.0	46.3	57.7	**62.5**
play_stacking_toy	**6.6**	0.0	0.0	1.2	0.5	0.0	0.1
make_kong	0.0	0.0	0.0	56.7	**90.0**	41.3	32.0
Memory							
cover_blocks	49.3	0.0	3.0	**100.0**	20.7	17.7	22.2
match_and_pick_from_conveyor	62.0	10.0	10.0	**70.7**	29.3	26.7	36.7
swap_T	**18.0**	0.0	0.0	0.0	0.0	0.0	0.0
press_by_number	70.0	0.0	0.0	**95.3**	0.0	0.0	0.0
imitate_sorting_sequence	**58.9**	0.0	0.0	1.8	1.7	2.5	2.9
swap_blocks	0.0	0.0	0.0	**18.7**	0.0	0.0	0.7
Open							
align_blocks	**50.0**	0.0	20.0	0.0	0.0	0.0	0.0
solve_equation	**40.0**	2.0	0.0	0.0	0.0	0.0	0.0
stack_blocks_by_language	**48.0**	0.8	2.0	4.8	0.1	1.3	0.9
general_pickup	**84.0**	20.0	30.0	14.0	12.7	28.0	8.7
classify_objects_by_language	**46.0**	0.2	0.0	0.5	1.1	2.0	1.3
pick_from_conveyor_by_image	**4.0**	0.0	0.0	0.0	0.0	0.0	0.0
store_tools_in_toolbox	**2.5**	0.0	0.0	0.2	0.0	0.2	0.3
pour_by_language	**0.4**	0.0	0.0	0.0	0.0	0.0	0.0

C.1 PROTOCOL

Every run uses the interface of Section 3: the same `move_eef` and `give_up` tools, the same bounding and interpolation of an accepted target, and the same observation contract. The demonstration is the intended difference from the zero-shot prompt.

The design is matched. Each condition covers the same 34 tasks and the same ten layouts per task, so each condition is 340 task–layout pairs at one episode per pair on a single seed. The two demonstration conditions are 340 new episodes each. The zero-shot column is not a new run: it is the official 50-episode Astra campaign restricted to those exact 340 task–layout pairs. Because the pairs are identical across conditions, the zero-shot column is the control for the other two, and a gap between columns cannot come from a difference in task or layout difficulty.

The zero-shot cell of Table 3 is a micro success rate over these 340 pairs, $78/340 = 22.9\%$. It is close to the $472/2100 = 22.48\%$ micro rate of the official board (Section 4.1), but it is computed over a different task set and a different number of episodes per task, so only the within-table comparison carries weight.

C.2 WHAT EACH CONDITION PUTS IN THE PROMPT

The demonstration always comes from a *different layout of the same task*. It is therefore a correct solution to the task the model faces, recorded in a different scene. The experiment tests transfer from that example, not replay of a solution for the current layout.

The image condition supplies a sequence of camera images together with the end-effector pose recorded at each of them. The poses are in the same dimensions `move_eef` takes, so the model is shown, in its own action vocabulary, a sequence of targets that solved the task once. The text condition describes the same trajectory in prose and supplies no images. Neither condition provides a demonstration recorded in the current layout.

C.3 TASK-LEVEL EVIDENCE

Table 7: Where the demonstration conditions lose episodes. The 34 tasks split into those the zero-shot controller never solves and those it solves at least once. The first row is the subset count reported in Section 4.5; the second row is the total of Table 3 minus the first row. No cell is a new measurement.

Task subset	Pairs	Zero-shot	Image demo	Text demo
Zero-shot 0/10 (15 tasks)	150	0/150 = 0.0%	2/150 = 1.3%	0/150 = 0.0%
At least one success (19 tasks)	190	78/190 = 41.1%	59/190 = 31.1%	44/190 = 23.2%
All tasks (34 tasks)	340	78/340 = 22.9%	61/340 = 17.9%	44/340 = 12.9%

Table 7 splits the 34 tasks by what the zero-shot controller already does with them. Fifteen tasks are 0/10 zero-shot; the reported subset counts cover 150 pairs per condition, which is those fifteen tasks at ten layouts each. The remaining nineteen tasks hold every zero-shot success, since the first subset contributes none by construction.

On the fifteen tasks with no matched zero-shot success, the image condition succeeds in 2 of 150 episodes and the text condition in none. On the remaining nineteen tasks, success drops from 41.1% to 31.1% with images and to 23.2% with text. These are net losses of 19 and 34 successes within that subset. The two additional image-condition successes in the first subset reduce its overall deficit to 17 episodes. These counts describe outcomes under this protocol, not whether a task is fundamentally beyond the model's capability.

Paired outcome changes. Against the zero-shot column, an image demonstration turns 31 failures into successes and 48 successes into failures. A text demonstration turns 18 up and 52 down. A demonstration therefore changes 79 of the 340 pairs in the image condition and 70 in the text condition. More pairs change from success to failure than in the opposite direction. The net losses above are the difference between the two directions, not the number of pairs a demonstration changes. The losses all fall on the nineteen tasks with at least one zero-shot success, since the other fifteen have no success to lose: 29 of the 31 image-condition gains and all 48 of its losses sit in that subset. Counted by task instead of by episode, images help 7 of the 34 tasks and hurt 12, and text helps one.

C.4 Possible explanations for the decline

The experiment measures the effect of adding these demonstrations, but does not isolate why aggregate success decreases. When the model already interprets an instruction correctly, another example may not resolve an execution-precision limitation. Transferring layout-specific contact geometry may also lead to errors. Neither explanation is established by the aggregate results. Demonstration format, context length, and stochastic variation remain alternative explanations.

Table 8: An illustrative task from the demonstration campaign. Each condition contains ten matched task–layout pairs. Higher success on this task does not identify the cause of the difference.

Task	Zero-shot	With demonstration	Condition
`press_by_number`	5/10	7/10	image
`press_by_number`	5/10	8/10	text

Table 8 illustrates a task with higher success under both demonstration conditions, `press_by_number`. Its 5/10, 7/10, and 8/10 counts do not establish whether the difference comes from task interpretation or execution. The 70.0 Score in Table 6 is a different metric from the full 50-episode campaign and should not be compared directly with these counts.

C.5 What this design does not control

Every cell is one episode per task–layout pair on a single seed. Small per-task differences may reflect stochastic variation; even the larger subset totals do not identify the cause of the decline. The experiment covers the one-demonstration setting only, so it does not establish what several demonstrations, or a demonstration from the current layout, would do. The 34 tasks are a subset of the official 42, so the 22.9% baseline should not be read as a restatement of the official Average. Table 8 is illustrative rather than a full task-level breakdown.

D Perturbation protocol and episode traces

This appendix records the protocol, the perturbation definitions, and the episode-level evidence behind Section 4.5 and Table 4.

D.1 Protocol

Every run uses the `move_eef` and `give_up` loop of Section 3, with the condition-specific changes described below. The task is `general_pickup`. The layouts are 0, 1, 2, 4, 7, 8, 9, and 10, with one episode per layout on a single seed. The context policy is `image_horizon` $= 2$, so the two most recent observations keep their images while all earlier text remains. The budget is 100 model calls per episode unless a row states otherwise, and RoboDojo's own step limit runs down in parallel.

These eight layouts are the ones the unperturbed controller solves. Unperturbed, the eight episodes are 8/8 at process score 1.00, against 42/50 for the same task under the official 50-episode protocol. The baseline row therefore sits at the ceiling by construction, and a perturbation can only move the count down.

The perturbation is active from the first turn. No explicit explanation of the perturbation is added to the system or Goal message. However, the `negate_xyz` tool description exposes transformed coordinate bounds: for example, the advertised z range is $[-1.565, -0.7]$ rather than $[0.7, 1.565]$. The model-visible interface is therefore not identical across conditions, and these bounds may provide a cue to the transformation. Model weights remain fixed and no memory is carried between episodes. Table 4 includes no retries after failure. A separate larger-budget rerun is reported below and is excluded from that table.

D.2 What each perturbation changes

Four probes change the visual observation. *Top–bottom image flip* (`flip_vision_ud`) flips every camera image before the model receives it. *Left–right image mirror* (`flip_vision_lr`) mirrors every camera image, reversing apparent left–right positions without changing camera identities. *No*

head-camera view masks `cam_head` and leaves the two wrist views. *Right-wrist view only* masks the head and left-wrist cameras.

One probe changes the coordinate mapping. *Negated Cartesian axes* (`negate_xyz`) uses sign-reversed Cartesian target coordinates, with correspondingly transformed bounds in the tool description. This changes the coordinate convention rather than simply reversing a displacement from the current pose. One probe changes action execution. *Per-move pose jitter* adds a random offset to every commanded pose. The displacement has a mean magnitude of 10 cm and is resampled on each move.

The model receives state text, action feedback, and the available camera views. These traces do not isolate the contribution of each feedback channel to recovery.

D.3 The conditions without a figure

Figure 4 in Section 4.5 shows Left–right image mirror, Right-wrist view only, and Per-move pose jitter. The three remaining perturbation rows of Table 4 are recorded here.

All eight layouts succeed under Top–bottom image flip. Six succeed without the head-camera view, compared with three under Right-wrist view only. Selected camera-masking failures include prolonged visual search (Appendix D.7); the counts alone do not establish why the conditions differ.

Four layouts succeed under Negated Cartesian axes. In the layout 0 trace, the model revises its forward/back estimate after observing the wrist view. Appendix D.6 reproduces selected calls. This is evidence of a spatial correction, not proof that the model identified the full coordinate transformation.

D.4 Episode-level evidence

Table 9: Selected episodes from the perturbation sweep. Calls counts the model calls spent; a failed episode reports the calls used before the budget or the episode ended. All rows are `general_pickup` at a 100-call budget except the last, which repeats layout 9 at 170 calls and 400 environment steps.

Condition	Layout	Outcome	Calls
Top–bottom image flip	0	success	14
Left–right image mirror	0	success	15
Negated Cartesian axes	0	success	19
Negated Cartesian axes	4	failure	22 of 100
No head-camera view	0	success	16
Per-move pose jitter	0	success	14
Per-move pose jitter	4	failure	18 of 100
Right-wrist view only	9	failure	25 of 100
Right-wrist view only, larger budget	9	success	38

Table 9 lists selected episodes from the sweep. The diagnoses are in the model's own notes. Under Negated Cartesian axes on layout 0, the fifth call states that its forward/back estimate was reversed; that episode is excerpted in Table 10. Under Per-move pose jitter, the second call reads: "The rotation reached 45 degrees, but the position missed by 129 mm." Under Left–right image mirror (`flip_vision_lr`), the correction is an arm reassignment: "The camera views show that the right arm is the one beside the scissors. I am restoring the idle left arm and preparing the right hand for a downward grasp." That note is the one quoted in Figure 4, between the two turns shown there. The notes refer to both visual observations and motion feedback; they do not establish a text-only recovery mechanism.

Right-wrist view only includes a supplementary example of one arm observing while the other grasps. Layout 9 fails at the standard budget after spending fifteen calls rejecting a quadcopter, a wrench, and a tape measure before locating the corkscrew. Re-run at 170 calls and 400 environment steps, the same layout succeeds in 38 calls: the right arm sweeps, tilts, and rolls in order to look ("Only the right wrist camera is available and the corkscrew is not yet visible"), and the other hand performs the grasp ("I bring the left hand above the handle while the right wrist watches the approach"). One arm becomes a camera and the other a manipulator. This follows general advice in the system prompt to use an idle wrist camera when the working view is insufficient. It does not establish that the model discovered the role split without guidance.

D.5 SUCCESS AND OBSERVABLE CORRECTION

An episode that succeeds without a visible change in behaviour may reflect existing robustness rather than adaptation. Other episodes include notes about an unexpected outcome followed by a change in actions. These provide evidence of observable correction, but the notes alone do not establish the internal mechanism or distinguish adaptation from ordinary closed-loop control. Table 4 reports final outcomes; Table 10 illustrates a sequence of corrections. Neither measures the causal benefit of retaining history.

D.6 SELECTED CALLS FROM A SUCCESSFUL EPISODE

Table 10: Selected verbatim notes from a successful `negate_xyz` episode on `general_pickup` layout 0, 19 calls, success. The note is the text the model attached to its own `move_eef` call, followed by the dimensions that call requested. Call 5 refers to the wrist view and revises the forward/back estimate; its requested motion is rejected as unreachable. Targets use the transformed bounds described in Appendix D.2.

Call	Model's own note, and the targets it requested
4	"The scissors are ahead and slightly to the right in the wrist view. I will move toward their hinge while staying above the tabletop." `right_x` -0.40, `right_y` 0.38, `right_z` -1.07
5	"The wrist view now shows the near table edge, so the forward/back estimate was reversed. I will move back across the table toward the scissors." `right_y` 0.02, `right_z` -1.18
12	"The closer view places the hinge slightly right and ahead of the jaw center. I will make a small lateral correction over the metal hinge." `right_x` -0.389, `right_y` 0.025
Intervening calls omitted.	
17	"The gripper has closed against the scissors rather than fully shutting. I will lift a few millimetres first to confirm they are secure." `right_z` -0.78
19	"The scissors are securely held and clear of the table. I will lift them to more than 10 cm above their starting height." `right_z` -0.89

Table 10 reproduces selected notes and requested targets from the `negate_xyz` layout 0 episode. After call 4, call 5 refers to the wrist view and revises the forward/back estimate. That call is rejected as unreachable; later calls make further corrections and the episode eventually succeeds. The displayed targets use the transformed coordinate bounds, not the normal bounds in Appendix A. The trace does not establish a one-call adaptation cost or equivalence to an unperturbed trajectory.

The same episode also shows the limit of this evidence. The recovery is visible because the model wrote it down. Nothing in the trace establishes that the model could not have recovered without stating the diagnosis, and nothing establishes that a pretrained policy could not recover in some other way.

D.7 FAILURE MODES

Selected failures include extended visual search under camera masking and repeated grasp corrections under pose jitter. The Negated Cartesian axes run on layout 4 ends after 22 calls without success. Under Right-wrist view only on layout 0, the model reports that the object slipped during lifting and then attempts further grasps. These observations suggest multiple failure modes, but do not isolate their causes or account for every failed layout.

An excluded run. On layout 4 the prompt stated that a lift may be shared between both hands, and the controller produced a two-handed pickup in 23 calls. It opened both grippers at opposite ends of a toy car, rotated both jaws by 75 degrees, seated each hand over a separate axle, and lifted in synchronized 5 mm steps. That run adds explicit task guidance rather than only perturbing the interface, so it is not a row of Table 4.

D.8 What this design does not control

Each cell of Table 4 contains only eight binary trials on one seed, so small differences should not be treated as reliable condition rankings. The layouts are matched across conditions but selected for unperturbed success, which limits generalization to other layouts. There is no history ablation or pretrained-policy comparison under these perturbations. Thus the probe does not establish that correction depends on retained history or is distinctive of LLMs. The transformed tool bounds also limit claims about discovering an entirely unannounced coordinate mapping.

E Real-robot diagnostic material

These tables preserve the detailed material behind Section 5. The official RoboDojo-Real protocol was not completed with GPT-6 Astra. The retained samples and separate joint-position deployments are diagnostic material, not official results. Table 11 reports the 12-task, 33-trial material retained around the safety stop, with variable $n = 1$–4 per task across ARX X5 (9 clips), Piper (21), and Piper X (3); trial scores come directly from each task's `scores.json`. `stack_bowls trial_8` is the only full success; its trials 4, 5, and 6 and `store_in_safe trial_1` receive partial credit, and zero-score clips often show approach, grasp, or transport attempts with the terminal predicate unmet. The official complete-protocol rows are context only and are not directly comparable.

Table 12 collects the qualitative notes from the joint-position deployments on the Franka and the two-arm humanoid. These use a different interface from the evaluated `move_eef` controller and carry no comparable Score or SR. The notes do not establish a controlled comparison between the embodiments or identify a cause for differences in behaviour. This appendix does not include images of those deployments. The retained operator notes also record that the model did not adopt large rotations to improve contact on either body and sometimes described dynamic corrections too late to execute them.

Table 11: RoboDojo-Real diagnostic sample. Diagnostic rows report Astra's retained 12-task, 33-trial material; Score is the mean process score × 100 and SR is the full-success rate. The official 18-task rows are complete-protocol references and are not directly comparable.

Type	Entry	n	Raw trial scores	Score	SR
ARX X5	`cover_blocks`	2	0 / 0	0	0%
ARX X5	`insert_tubes`	3	0 / 0 / 0	0	0%
ARX X5	`make_bread`	1	0	0	0%
ARX X5	`make_food`	2	0 / 0	0	0%
ARX X5	`store_in_safe`	1	0.4	40	0%
Piper	`fill_pen_holder`	3	0 / 0 / 0	0	0%
Piper	`insert_charger`	3	0 / 0 / 0	0	0%
Piper	`put_objects_into_basket`	3	0 / 0 / 0	0	0%
Piper	`stack_and_cover_blocks`	4	0 / 0 / 0 / 0	0	0%
Piper	`stack_bowls`	4	0.3 / 0.3 / 0.3 / 1.0	47.5	25%
Piper	`stand_up_bottles`	4	0 / 0 / 0 / 0	0	0%
Piper X	`sweep_blocks`	3	0 / 0 / 0	0	0%
Diagnostic	Pooled retained clips	33	—	6.97	3.03%
Official 18-task	OpenWAM-α	18 tasks	—	37.60	24.40%
Official 18-task	Pi-05	18 tasks	—	22.90	12.80%
Official 18-task	InternVLA-A1	18 tasks	—	12.00	7.20%

F Mobile humanoid demonstrations

These two recordings show a mobile humanoid performing walking and grasping sequences commanded by Astra through a separate interface. They are qualitative demonstrations, with no success rate or process score. The action vocabulary includes locomotion commands rather than only joint-position targets. The recordings alone do not establish how each command is implemented by the low-level controller, so we do not classify them under the strict definition in Section 3.1. Section 6 introduces the recordings; this appendix records their contents.

Table 12: Qualitative notes from separate joint-position deployments, not an official Score/SR.

Body	Task	Observation
Franka	Pour water	initial miss; later flow reaches cup, then stops
Franka	Cup on shelf	placement and mid-air drop observed
Humanoid	Pick and place	rigid and soft objects manipulated
Humanoid	Make breakfast	toast-into-slot fails
Humanoid	Pour water	high pour, some spill
Humanoid	Fold a shirt	corners first; neatness stalls
Humanoid	Draw	color OK; later uses idle wrist cam

Pickup. The model walks to a target object and grasps it. The action vocabulary is `Walk`, `turn`, `hand cm`, `wrist R/P/Y`, and `close`; the overlay in the recording is the request. The task is `target RED`, actions 004–026, with `close 1.00` at action 022. The recording runs 23 s across five views: two external, plus ego, left wrist, and right wrist. The left hand performs the grasp.

Long-horizon sit-and-pick. The model walks, sits down, and grasps while seated in a longer recorded sequence. The task is `target WHITE`, 117 s, actions 001–038, with `close 1.00` at action 037.

G When the model writes the controller instead of acting

In the piano runs, GPT-6 Astra writes a controller rather than selecting targets turn by turn: a `play.py` that emits 45-D joint targets every 0.05 s for two Shadow hands on an 88-key piano in RoboPianist (Zakka et al., 2023). The LLM weights remain fixed, but the model refines the program through simulation trials without access to a reference solution. This is outside the LLM-as-policy condition of Section 3 and outside the RoboDojo protocol. It examines code-based controller design rather than turn-by-turn action selection. Section 6 introduces the evaluation; this appendix keeps the isolation rules, runs, and transfer tests.

The controller is written by GPT-6 Astra, driven through the Codex agent scaffold at reasoning effort `high`; the model under test is the same one as in the main experiments, and Codex is only the shell that runs it. There are no motion primitives: every finger, wrist, and slide target is code the model wrote. The two-hand Twinkle RL reference scores F1 0.8863 when replayed through our evaluation setup. We use this as a reference point for the recorded piano runs, not as a comparison of training efficiency. F1 and precision/recall (P/R) below denote the piano evaluator's reported metrics, rather than a RoboDojo Score or success rate.

What the model may read. The score, meaning pitch, onset and offset, the left and right staff split, and pedal marks; the position and size of all 88 keys; its own joint angles and a forward-kinematics call; and after each attempt, which keys sounded when, the resulting F1, and rendered frames.

What is withheld. Human fingering annotations, which ship with the RoboPianist data and were removed; the RL reference actions and code; the simulator source and model files; contact forces and other internal state; network access; and any memory of an earlier session. The protocol excludes fitting a neural policy, reinforcement learning, and behaviour cloning. Code revision, search, optimization, and trial and error are permitted. Thus the runs are not practice-free, even though the LLM weights remain fixed.

Isolation and audit. The simulator runs in a separate process behind a socket, the rules are stated in the task file the model reads, and every shell command and tool call is audited afterwards. Across the three runs, the audit recorded 201 shell commands, six image views, and zero network calls; it flagged no violations. The host had user namespaces disabled, so no system-level sandbox was available. Access restrictions therefore relied on the stated rules and audit, not on enforced system-level isolation.

Table 13 reports both practice and final verification. One-hand Twinkle reaches F1 0.907 after 22 total episodes, with precision 1.00. Two-hand Twinkle reaches a similar F1 of 0.902 after 33 total episodes; all 35 sounded notes are assigned to the hand specified by the score. The 117-note nocturne

Table 13: Piano-performance F1 from the final verification episode of each run. Tries counts practice episodes plus that verification. Two-hand Twinkle reaches 0.902; the RL reference replay scores 0.886 in this setup.

Score	Notes	F1	P / R	Tries	Wall
Twinkle, one hand	14	0.907	1.00 / 0.91	22	31 min
Twinkle, two hands	34	0.902	0.99 / 0.89	33	34 min
Chopin nocturne excerpt	117	0.599	0.85 / 0.58	150	40 min

excerpt reaches F1 0.599 after 150 total episodes, with precision 0.85 and recall 0.58. The model stopped below its target of 0.9. Lower recall is consistent with missed or insufficiently sustained notes, but the metrics alone do not isolate reachability, timing, and coordination as causes.

G.1 Does the delivered program survive a change of music?

We also test the delivered controller on different music. We re-ran the two-hand Twinkle `play.py` with no re-optimization on transposed and time-stretched versions of its own score, and on three scores it had never seen. The program, the 0.05 s control loop, and the piano evaluation metric are identical across rows.

Table 14: The delivered two-hand Twinkle program re-run without modification. Notes is the score length; Correct hand counts sounded notes assigned to the staff the score specifies. Time factors multiply note timestamps, not tempo. P/R and F1 are reported by the piano evaluator. One episode per row.

Score	Change	Notes	F1	P / R	Correct hand
Twinkle, two hands	as delivered	34	0.902	0.99 / 0.89	35/35
Twinkle, two hands	time $\times 0.85$	34	0.883	0.98 / 0.87	32/32
Twinkle, two hands	-7 semitones, time $\times 1.25$	34	0.837	0.94 / 0.82	33/38
Twinkle, two hands	+5 semitones, time $\times 1.1$	34	0.805	0.93 / 0.79	30/33
Twinkle, two hands	+2 semitones	34	0.683	0.89 / 0.65	32/36
Twinkle, two hands	-3 semitones	34	0.655	0.85 / 0.63	31/37
C major scale	unseen score	30	0.903	0.95 / 0.93	30/30
C major chords	unseen score	16	0.654	0.98 / 0.56	12/12
D major scale	unseen score	30	0.556	0.81 / 0.56	23/34

Transfer varies across the tested changes. Reducing note times by a factor of 0.85, equivalent to a 17.6% faster tempo, lowers F1 by 0.019. All sounded notes remain assigned to the correct hand. Pure transpositions of $+2$ and -3 semitones lower F1 by approximately 0.22–0.25. The two transposed rows above 0.80 also change timing, so they do not isolate transposition. On unseen music, the C major scale reaches 0.903 with 30 of 30 sounded notes assigned to the correct hand. The D major scale reaches 0.556, with 23 of 34 sounded notes assigned correctly. For C major chords, precision is 0.98 but only 12 notes sound for a 16-note score. These outcomes suggest limitations in transfer, but do not isolate black-key geometry, timing, or simultaneous pressing as causes.

Beyond writing controllers, the model also produced the performance itself as a fixed sequence of joint targets, refining each trajectory through repeated simulation trials and submitting 161 keyframes at 50 ms intervals. Open-loop replay achieved F1 scores of 0.901 on one-hand Twinkle and 0.920 on two-hand Twinkle.

H Limitations

This appendix states the limits of the evaluation. Observed model failure modes are described in Section 4.4.

The detailed analysis concerns one model. The qualitative capability analysis and Section 4.5 focus on Astra. The benchmark and ICL probes use one evaluation seed, without repeated trials to estimate variability. Three models cannot establish a general capability threshold for LLMs. DeepSeek-Flash is also evaluated for 10 episodes per task rather than 50, and the eight perturbation layouts were selected as those the unperturbed model solves, which puts that baseline at the ceiling by construction

and leaves Table 4 too small for a reliable quantitative comparison.

The action and observation spaces define a measurement boundary. `move_eef` is the only motion tool. A task requiring continuous contact or non-quasi-static motion may fail because this action representation cannot express the behaviour, rather than because the model cannot conceive it. Visual observations are RGB-only: no metric depth is exposed, so part of the precision ceiling may come from the inputs rather than from the controller. Finally, `image_horizon` $= 2$ means the Memory axis measures the model together with this context policy. The scores cannot separate these interface causes from model failure.

Comparability. The LLM controllers receive a wiki-derived task description and process-score ladder in addition to the official instruction. The public policy cells were not re-run with equivalent text. The comparison therefore shares the evaluator and outcome metric, but not the input information. Those cells are also a leaderboard snapshot and were not re-run alongside the LLM-controller columns. The 33 real-robot trials retained around a safety stop are a selected sample supporting no estimate of general real-world reliability. For a closed-weight model, contamination by benchmark-related pretraining material is unquantified rather than ruled out.

Cost and time scale. The per-episode call budget is a condition, not a neutral setting. An episode that ends without success contributes zero to SR, whether it exhausted its budget or failed for another reason; partial-credit Scores can still differ. Inference latency and action timing may also limit deployment. This evaluation does not isolate their contribution to failures from limitations in perception or physical reasoning.

The ICL probes are incomplete. The demonstration-conditioned probe is a single insertion recipe on one set of tasks and one choice of example. A different place in the message sequence, a different difficulty slice, or a different demonstration could change the result. This evaluation therefore cannot conclude that few-shot ICL does not work. It shows only that one demonstration did not help under this protocol. The perturbation probe has no history ablation and cannot separate learned adaptation from existing robustness or ordinary feedback control. The negated-coordinate condition also exposes transformed tool bounds.

The main conclusions come from one simulator. The ranking and the capability split come from RoboDojo-Sim. No comparably broad manipulation benchmark was evaluated in another simulator. The RoboDojo-Real campaign was diagnostic and stopped for safety, so the official real protocol was not completed. The mobile-humanoid recordings and RoboPianist scores use separate settings and do not enter the official board. This evaluation therefore does not show that the ranking or the capability split transfers to another simulator or to hardware under the official real protocol. Other simulators and real-robot experiments need further investigation.