IN-CONTEXT ROBOT LEARNING WITH VLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Enabling robots to adapt to unfamiliar environments as readily as humans remains a moonshot goal of embodied AI. No finite collection of demonstrations can cover every task and situation a robot will encounter, making the ability to learn from context at deployment essential for generalization. Such in-context learning (ICL), however, remains largely beyond the reach of existing robotic policies. The broad agentic capabilities of commercial vision-language models (VLMs), such as GPT-6 Astra, raise a compelling question: *can these models learn from demonstrations and interaction feedback, then translate that information into executable robot behavior without gradient updates or persistent changes to task-specific parameters?* We introduce **GPT-Policy**, a general-agent framework for in-context robot learning. GPT-Policy equips a VLM agent with a context compiler that organizes task-relevant information and an execution harness that supports geometric grounding, action chunking, and closed-loop decision-making through execution feedback. We evaluate GPT-Policy on ten real-world tasks spanning five context types and six RoboDojo simulation tasks. Across four real-world demonstration tasks, mean success rate increases from 10% without demonstrations to 80% with human videos or robot demonstrations that include action references. Videos provide procedural guidance, while action references can clarify motion details. Targeted ablations suggest that calibrated localization and action chunking improve success and execution efficiency. On these six RoboDojo tasks, one-shot GPT-Policy achieves 83.3% mean success. These results suggest that general-purpose VLMs can use deployment-time context to adapt robot behavior, while reliable physical execution remains a challenge.

1 INTRODUCTION

No training dataset can cover every situation a robot will encounter. Adapting to unfamiliar tasks, object configurations, and interactions during deployment is therefore a central challenge for embodied AI (Zitkovich et al., 2023; Kim et al., 2025; Ghosh et al., 2024; Black et al., 2025). Humans adapt by observing others and learning from experience. Enabling similar adaptation through in-context learning (ICL) is a key step toward general embodied agents (Duan et al., 2017; Fu et al., 2025; Vosylius & Johns, 2025). General-purpose language and vision-language models (VLMs) offer a promising starting point: their ability to learn from context may support robot adaptation without dedicated policy training (Brown et al., 2020; Alayrac et al., 2022; Huang et al., 2022).

Recent robot foundation models have made progress in learning from demonstrations. GEN-1.5 reports one-shot adaptation from physical prompts (Generalist Team, 2026). S1 uses video demonstrations to specify novel atomic and long-horizon tasks (Skild AI, 2026), while Zero-WAM trains a video-action model to follow human videos on unseen tasks (Zhou et al., 2026). These advances raise a complementary question: *to what extent can off-the-shelf, general-purpose VLMs support robotic ICL without additional robot-policy training?*

We define robotic ICL as behavioral adaptation from deployment-time demonstrations or interaction experience, with model parameters held fixed. We distinguish this learning from task specification through goal images or interactive instructions, while studying both within a shared closed-loop framework. We study five forms of context. Human videos provide procedural guidance, while robot demonstrations provide visual references and, when available, aligned action references. Goal images specify desired outcomes. Interaction history preserves past observations, actions, and outcomes, and

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.

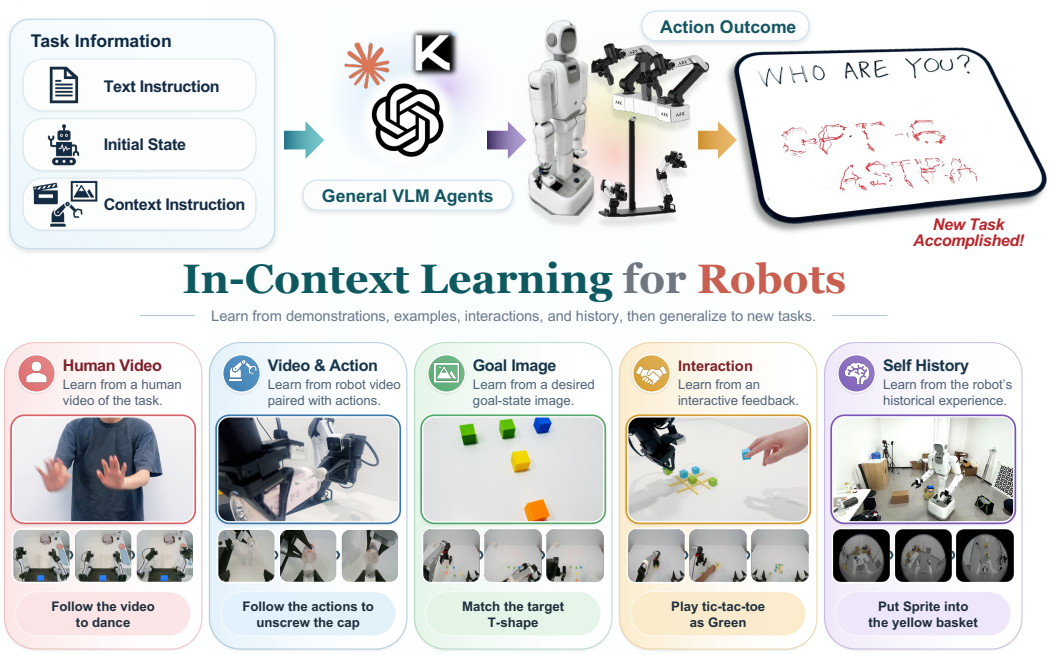


Figure 1: **In-context robot control with GPT-Policy.** A general-purpose VLM combines the task instruction, initial state, and contextual information to guide robot actions. Context can include human videos, robot videos with recorded actions, goal images, human–robot interaction, and self-interaction history. The examples illustrate manipulation, interactive play, and mobile object retrieval.

online human feedback conveys intent and interaction constraints. The central challenge is to identify the information needed for the current decision and translate it into appropriate physical action.

We introduce GPT-Policy to address three challenges in translating context into robot behavior: organizing procedural evidence, grounding spatial references, and reducing repeated model calls. A context compiler structures demonstrations into task stages and preserves available visual–action references. Camera calibration and triangulation provide 3D position references for robot actions. Action chunking enables multiple motion waypoints per model decision. The execution harness checks action feasibility and returns observations and execution feedback for subsequent decisions. Our primary instantiation, GPT-Policy (Astra), uses GPT-6 Astra as its VLM backbone.

Our evaluation spans five context families across ten real-world and six simulation tasks. Human videos improve task completion without robot action labels, while action-reference gains vary by task. Targeted ablations suggest that calibrated localization and action chunking improve task success and execution efficiency. Our contributions are threefold:

- **A unified framework for robotic in-context learning.** We introduce GPT-Policy, which enables general-purpose VLMs to learn from test-time context and adapt robot behavior in new situations. The framework supports five forms of context: human videos, robot demonstrations with optional action references, goal images, interaction history, and online human feedback.

- **An execution harness connecting context understanding to physical action.** We design a harness that organizes demonstrations into subtask references, grounds visual targets in robot coordinates through calibrated triangulation, and converts multi-waypoint plans into executable trajectories with feedback for replanning.

- **Empirical insights into context and execution.** Across ten real-robot tasks, we find that videos provide procedural guidance and action references aid contact-sensitive manipulation. Targeted ablations suggest that both calibrated localization and action chunking improve task success and execution efficiency.

2 Related Work

General-purpose foundation models for robot control. Representative models and methods, including GPT-6 Astra and ReAct, have demonstrated capabilities in reasoning, planning, tool use, and feedback-driven decision-making (OpenAI, 2026; 2025; Anthropic, 2026; 2025; Kimi Team et al., 2025; 2026; DeepSeek-AI et al., 2025; GLM-4.5 Team et al., 2025; Qwen Team, 2025; Liu et al., 2024; Yao et al., 2023; Shinn et al., 2023). Prior work has applied these capabilities to robot skill selection (Ichter et al., 2023), program synthesis (Liang et al., 2023; Singh et al., 2022; Huang et al., 2023a), motion planning (Huang et al., 2023b; 2025), and feedback-driven policy improvement (Lu et al., 2026; Xiao et al., 2026). RoboPrompt, Show-Harness, and related approaches have further explored action prediction and closed-loop control (Yin et al., 2025; Chen et al., 2026b). However, how deployment-time context can be used to continuously refine robot actions remains underexplored.

Multimodal in-context learning in vision–language models. In-context learning enables models to infer new tasks from examples provided in a prompt without updating their parameters (Brown et al., 2020). Frozen and Flamingo extended this capability to vision–language tasks (Tsimpoukelli et al., 2021; Alayrac et al., 2022), while OpenFlamingo, Otter, MetaVL, MMICL, Emu2, VILA, and related work further improved task adaptation from multimodal context (Awadalla et al., 2023; Li et al., 2023; Monajatipoor et al., 2023; Zhao et al., 2024; Sun et al., 2024; Lin et al., 2024). Subsequent studies have examined training and evaluation, demonstration selection, and video-based context (Doveh et al., 2024; Zong et al., 2025; Chen et al., 2025; Baldassini et al., 2024; Yu et al., 2024). These studies show that current multimodal in-context learning still relies heavily on textual cues and remains sensitive to visual demonstrations and example configurations, making its transfer to tasks requiring continuous perception and action challenging.

In-context learning for robot policies. In-context learning for robot policies uses deployment-time demonstrations to adapt behavior, with prior work progressing from few-shot imitation learning to context-conditioned policies based on robot trajectories (Duan et al., 2017; Fu et al., 2025; Vosylius & Johns, 2025). Methods such as MimicDroid, Behavior Prompting Policy, GEN-1.5, S1, and Zero-WAM further explore adaptation to novel tasks using demonstrations, physical prompts, or human videos (Shah et al., 2026; Patel et al., 2026; Generalist Team, 2026; Skild AI, 2026; Zhou et al., 2026). Unlike these approaches, which rely on specialized policy training, particular demonstration formats, or fixed action representations, we study how a frozen, general-purpose vision–language model can use diverse forms of context to generate closed-loop robot actions.

3 Method

We present GPT-Policy, a closed-loop framework that combines a general-purpose VLM, a context compiler, and an execution harness for robot control (Figure 2). Our context compiler organizes demonstrations into subtask-structured references that guide action selection alongside live observations and interaction history. Key components of the harness include a geometric grounding tool that estimates metric positions from visual correspondences and a motion interface that enables the VLM to jointly plan multiple waypoints per decision. Execution adapters convert these plans into validated robot trajectories and return feedback for the next decision.

3.1 Closed-loop Policy Interface

We use context in a broad sense to denote the evolving multimodal information $\mathcal{C}_t$ supplied to the policy. At decision step t, this context combines the task instruction τ_{task}, the compiled reference $\mathcal{R}$, interaction history H_t, and current observation $o_t = (\{I_t^v\}_{v \in V_O}, s_t)$. Here I_t^v is the current image from camera view v, V_O is the set of available observation views, and s_t is the measured robot state. The reference $\mathcal{R}$ comprises records compiled from the supplied demonstration or goal image before execution. History H_t records prior observations, tool calls, and feedback. $\mathcal{B}$ specifies robot coordinate conventions and camera calibration, while $\mathcal{T}$ defines the available tools and their arguments. The harness assembles these components into the policy input:

$$\mathcal{C}_t = (\tau_{\text{task}}, \mathcal{R}, H_t, o_t, \mathcal{B}, \mathcal{T}), \quad a_t = (u_t, v_t) \sim \pi_\theta(\cdot \mid \mathcal{C}_t), \quad (o_{t+1}, f_t) = \mathcal{E}(a_t, o_t). \tag{1}$$

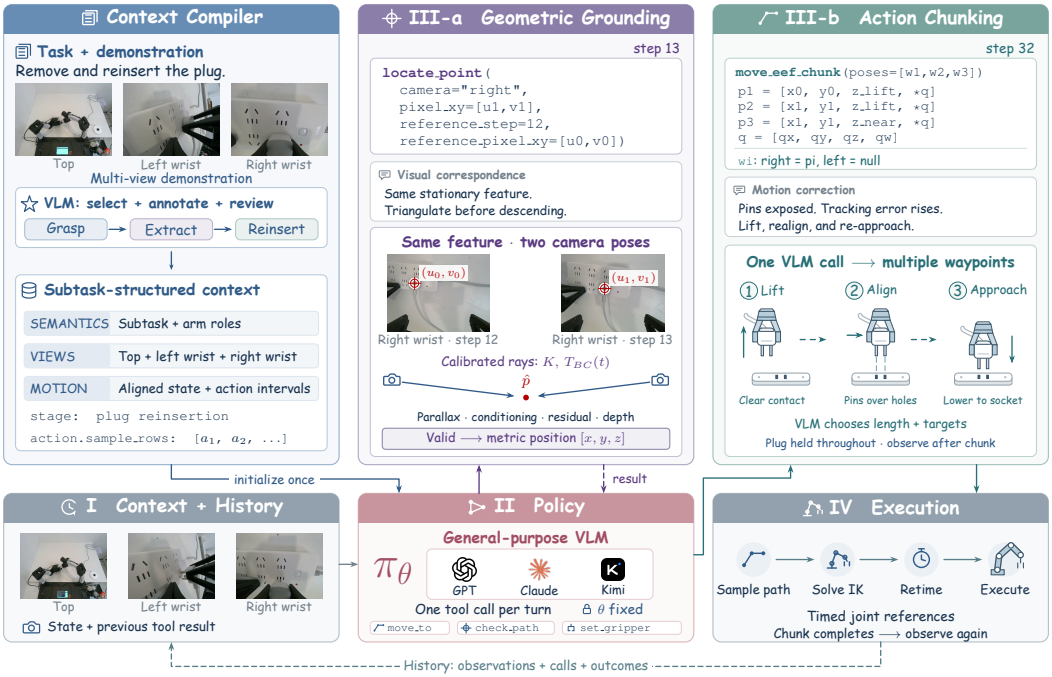


Figure 2: **Overview of the GPT-Policy framework.** Given a task instruction and demonstrations, a context compiler builds structured references. (I–II) A frozen VLM uses references, live observations, and history to select one tool call per turn. We integrate two key modules: geometric grounding triangulates stationary features from calibrated wrist-camera views (III-a), while action chunking lets the VLM plan sequences of end-effector waypoints (III-b). (IV) Execution adapters validate and retime trajectories, then return feedback for the next decision.

The policy selects one tool u_t and its arguments v_t per decision. Motion calls may specify multiple end-effector waypoints. The execution interface $\mathcal{E}$ handles the call through geometric routines or robot execution adapters, returning feedback f_t and the next observation o_{t+1}. The returned feedback and observation are incorporated into $\mathcal{C}_{t+1}$ to guide the next policy decision.

3.2 Context Compilation

Reference initialization. Given a task instruction, the context compiler constructs $\mathcal{R}$ from a demonstration or goal image and accompanying text before execution (Figure 2). For human or robot demonstration videos, it uses a VLM to select keyframes at subtask and contact transitions, annotate subtasks, and review the sequence. The reference contains N records (Figure 6):

$$\mathcal{R} = [d_i]_{i=1}^{N}, \qquad d_i = \big(\tau_i, \{I_i^v\}_{v \in V_D}, \ell_i, x_i^{\text{demo}}, A_i^{\text{demo}}\big). \tag{2}$$

Here d_i is the i-th reference record. For demonstrations, it contains keyframes I_i^v from available views $v \in V_D$ at time τ_i and semantic annotations ℓ_i describing the subtask, arm roles, and visible outcome. Optional robot data comprise the measured state x_i^{demo} aligned to τ_i and sampled motion and gripper records A_i^{demo} up to the next keyframe. We represent a goal image and any provided description as a single record ($N = 1$). The compiler presents $\mathcal{R}$ to the policy as interleaved text, numerical fields, and images, preserving record order and source and view labels. Appendix A.1 provides construction details.

Online context updates. During execution, the harness combines $\mathcal{R}$ with the current observation o_t and interaction history H_t. After each tool call,

$$H_{t+1} = H_t \oplus (o_t, a_t, f_t), \tag{3}$$

where $\oplus$ appends the observation, tool call, and feedback to the history. Operator feedback is included when available.

3.3 GEOMETRIC GROUNDING

We design a geometric grounding module that couples visual correspondence reasoning with calibrated triangulation to localize 3D points for geometry-aware action selection (Figure 2, III-a). The module uses calibrated camera intrinsics K and extrinsics: intrinsics map pixels to viewing directions, while extrinsics express the resulting rays in the robot base frame. The VLM identifies the same stationary feature in two observations from the same wrist camera at steps t_i, $i \in \{0, 1\}$. Let $\tilde{p}_i = (u_i, v_i, 1)^\top$ denote the corresponding homogeneous pixel coordinates after distortion correction. At each observation step t_i, the camera pose relative to the robot base frame is $T_{BC}(t_i) = T_{BE}(t_i)T_{EC}$, where $T_{BE}(t_i)$ is the observed pose of the end effector relative to the robot base frame, and T_{EC} is the calibrated camera-to-end-effector transform.

The corresponding ray centers c_i and unit directions d_i in the robot base frame are

$$c_i = t_{BC}(t_i), \qquad d_i = \frac{R_{BC}(t_i)K^{-1}\tilde{p}_i}{\|K^{-1}\tilde{p}_i\|_2}. \tag{4}$$

We reconstruct the feature position by minimizing the sum of squared perpendicular distances to the lines defined by the two camera rays:

$$\hat{p} = \arg\min_{p \in \mathbb{R}^3} \sum_{i=0}^{1} \left\| (\mathbf{I} - d_i d_i^\top)(p - c_i) \right\|_2^2, \tag{5}$$

where $\mathbf{I}$ is the 3×3 identity matrix. The estimate $\hat{p}$ provides a metric reference only when parallax, conditioning, ray-residual, and positive-depth checks pass. If any check fails, the estimate remains diagnostic. These checks assess geometric consistency and rely on a correct correspondence to a stationary feature.

3.4 ACTION CHUNKING AND EXECUTION

The policy selects the length K_t and waypoints of motion chunk U_t for participating arms $\mathcal{A}_t$ (Figure 2, III-b):

$$U_t = \left[(X^a_{t,k})_{a \in \mathcal{A}_t} \right]_{k=1}^{K_t}, \qquad X^a_{t,k} = (R^a_{t,k}, p^a_{t,k}) \in SE(3). \tag{6}$$

Each pose $X^a_{t,k}$ is expressed in arm a's base frame. The prompt encourages chunk boundaries at contact transitions or uncertainty. Gripper commands change only between chunks.

Starting from the measured tool pose, the Cartesian adapter interpolates waypoints and solves inverse kinematics (IK). Omitting decision and arm indices,

$$\begin{gathered} p_k(s) = (1-s)p_{k-1} + s p_k, \quad R_k(s) = R_{k-1} \exp\big(s \operatorname{Log}(R_{k-1}^\top R_k)\big), \\ q_n = \mathrm{IK}(\widehat{X}_n; q_{n-1}), \quad s \in [0,1], \quad n = 1, \ldots, M. \end{gathered} \tag{7}$$

Here $\widehat{X}_n$ is a sampled pose in the backend's kinematic frame and q_0 the measured joint configuration. For each arm and segment, the maximum absolute velocity, acceleration, and jerk ratios to joint limits, r_v, r_a, r_j, determine retiming:

$$\alpha_0 = \max\{1, r_v, \sqrt{r_a}, \sqrt[3]{r_j}\}, \qquad T_k = \max_{a \in \mathcal{A}_t} T^a_k. \tag{8}$$

Each arm's segment is stretched by at least its α_0, preserving joint samples. The retimed durations T^a_k are synchronized to T_k. Trajectories passing pose-residual and backend-specific checks are tracked with low-level feedback. These checks exclude collision and task-completion verification. Completion or rejection returns observations and feedback for replanning. See Appendix A for details.

4 EXPERIMENTS

We study when deployment-time context improves robot execution, how one-shot performance varies across simulated tasks, and which harness components support effective control.

(a) Human video demonstrations		Pick Red Towel			Pick Up Notebook		
Method	Context	**S/T**	Decisions	Time	**S/T**	Decisions	Time
$\pi_{0.5}$	None	7/10	19.3 ± 12.8	0.94 ± 0.53	2/10	31.4 ± 16.8	1.35 ± 0.63
GPT-Policy (Astra)	None	2/10	82.3 ± 26.7	21.1 ± 7.2	2/10	76.5 ± 15.4	20.8 ± 5.0
GPT-Policy (Astra)	Human Video	**9/10**	44.9 ± 27.7	11.7 ± 6.7	**8/10**	53.5 ± 24.0	14.6 ± 6.6
(b) Robot demonstrations		Unscrew Bottle Cap			Remove and Reinsert Plug		
Method	Context	**S/T**	Decisions	Time	**S/T**	Decisions	Time
$\pi_{0.5}$	None	2/10	87.4 ± 22.2	3.04 ± 0.72	0/10	65.4 ± 38.6	2.35 ± 1.28
GPT-Policy (Astra)	None	0/10	71.0 ± 50.2	16.1 ± 9.9	0/10	43.0 ± 38.5	9.1 ± 7.8
GPT-Policy (Astra)	Robot Video	7/10	53.4 ± 21.0	10.6 ± 4.3	0/10	32.0 ± 9.7	7.5 ± 2.4
GPT-Policy (Astra)	Robot Video + Action	**8/10**	49.1 ± 16.6	13.0 ± 4.9	**7/10**	43.5 ± 7.2	9.4 ± 2.1
(c) Goal images		Arrange T Shape			Arrange Fruit		
Method	Context	**S/T**	Decisions	Time	**S/T**	Decisions	Time
GPT-Policy (Astra)	Goal image	3/3	66.7 ± 4.0	15.8 ± 1.8	3/3	49.0 ± 14.2	12.4 ± 2.9
(d) Self-interaction history		Lemon to Pink Plate			Movable Exploration		
Method	Context	**S/T**	Decisions	Time	**S/T**	Decisions	Time
GPT-Policy (Astra)	Self History	3/3	35.3 ± 4.0	8.1 ± 1.0	3/3	40.3 ± 1.5	25.5 ± 4.0
(e) Online human–robot interaction		Tic-Tac-Toe			Pointed Fruit Pickup		
Method	Context	**S/T**	Decisions	Time	**S/T**	Decisions	Time
$\pi_{0.5}$	None	0/3	18.7 ± 1.5	0.94 ± 0.03	1/3	20.0 ± 6.6	1.07 ± 0.34
GPT-Policy (Astra)	Human interaction	3/3	69.7 ± 13.3	13.6 ± 2.6	3/3	67.3 ± 17.5	15.0 ± 3.4

Table 1: **Real-robot performance under five forms of context.** S/T denotes successful/total trials. Decisions and time (min) are mean $\pm$ sample SD over available trial records, including failures. Bold marks the highest GPT-Policy success per task in (a–b). Dashes in (c–d) denote inputs unsupported by the evaluated $\pi_{0.5}$ interface, not failed trials. Decision and timing conventions are defined in the experimental setup.

4.1 Experimental Setup

Protocol. GPT-Policy (Astra) uses a frozen VLM with instructions, live observations, and condition-specific context. We compare it with $\pi_{0.5}$, fine-tuned from the official pretrained weights using 30 robot demonstrations per task collected in the corresponding task settings, with no additional context at inference. Both use the same robot, task environments, success criteria, and stopping rules. Episodes start from reset scenes and end on success, budget exhaustion, or safety termination.

Metrics. We report success (S/T, successful/total trials) and mean $\pm$ sample SD for decisions and time, including failures (Table 1). For GPT-Policy, one action target or chunk counts as a decision, whereas $\pi_{0.5}$ uses logged counts. Time measures wall-clock episodes for GPT-Policy and top-view recording duration for $\pi_{0.5}$. Appendix B details the system configurations.

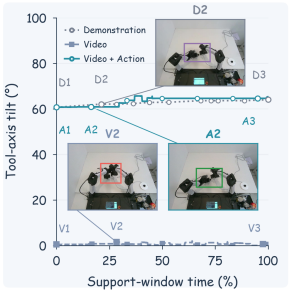


Figure 4: **Action references guide support posture.** Measured gripper tilt in selected plug runs (0° denotes a downward orientation), with time normalized per run. Full visualizations: Appendix Figures 12 and 13.

4.2 In-Context Learning on Real Robots

We investigate how a frozen VLM uses context to solve ten real-robot tasks: what procedural guidance can be recovered from video, what action references add to physical execution, and whether goals or interaction can guide behavior without a demonstrated procedure.

Human video improves execution without robot action labels: one demonstration increases success from 20% to 90% on *Pick Red Towel* and from 20% to 80% on *Pick Up Notebook*, while mean decisions decrease from 82.3 to 44.9 and from 76.5 to 53.5,



Figure 3: **Context specifies procedures, goals, and interaction constraints.** Execution sequences cover human video, robot video and actions, goal images, self-interaction history, and human interaction. Read each task from left to right. Shaded rows show demonstrations, and insets show goals. Appendix D provides the full sequences.

respectively (Table 1(a)). **Robot video and actions** expose a task-dependent gap between visual guidance and motion specification. Video alone achieves 70% success on *Unscrew Bottle Cap* but 0% on *Remove and Reinsert Plug*. Adding time-aligned poses, gripper states, and commands raises success to 80% and 70%, respectively (Table 1(b)). These aggregates span prompt and demonstration revisions.

Figure 4 compares measured support postures in selected plug runs sharing demonstration images and annotations. At three selected stages, target–demonstration orientation differences are $0.8°$–$8.1°$ with action references versus $61.0°$–$77.2°$ with video alone (Appendix Figure 13(b)). Video alone keeps the gripper nearly vertical and fails to extract the plug. This case suggests that numerical references help specify the support posture for coordinated manipulation (Appendices D.2 and D.3). Beyond

Table 2: **One-shot RoboDojo results and published baselines.** Score measures partial progress (0–100), and SR is success rate (%). Hybrid couples a RoboDojo-fine-tuned $\pi_{0.5}$ VLA with GPT-6 Astra, which reviews and selectively corrects its action proposals (Su et al., 2026). Published protocols are retained: $\pi_{0.5}$ uses three-seed leaderboard means, averaging standard/random Generalization settings (Chen et al., 2026a), while other methods use five episodes per task. Mean weights the six tasks equally, and bold marks the best reported values.

	$\pi_{0.5}$ Fine-tuned		RoboDawn 1-shot		GPT-as-Policy Direct		GPT-as-Policy Hybrid		GPT-Policy 1-shot (ours)	
Task	Score↑	SR↑	Score↑	SR↑	Score↑	SR↑	Score↑	SR↑	Score↑	SR↑
Classify objects by language	0.6	0.0	44	20	60	40	38	20	**88**	**80**
Imitate sorting sequence	1.6	0.0	66	60	0	0	53	40	**86**	**80**
Arrange largest number	2.8	0.7	**100**	**100**	57	40	50	40	**100**	**100**
Build tower	37.7	24.0	54	40	12	0	**64**	**60**	58	40
Put bottles into dustbin	79.9	69.3	88	80	36	20	**100**	**100**	**100**	**100**
Classify objects	24.7	12.7	83	80	**100**	**100**	71	60	**100**	**100**
Mean (6 tasks)	24.5	17.8	72.5	63.3	44.2	33.3	62.7	53.3	**88.7**	**83.3**

Context / execution setting	**Success**	**Decisions**	**Time (min)**
(a) Demonstration information *Remove and Reinsert Plug*			
Subtask + action	**7/10**	43.5 ± 7.2	9.40 ± 2.12
Action only	4/10	84.3 ± 14.1	15.44 ± 3.15
Subtask only	0/10	32.0 ± 9.7	7.49 ± 2.43
Neither	1/10	91.0 ± 12.0	18.32 ± 2.24
(b) Calibrated localization *Pick Up Notebook*			
Calibrated localization	**8/10**	53.5 ± 24.0	14.61 ± 6.62
Without calibration / localization	3/10	78.5 ± 23.9	22.78 ± 6.88
(c) Action chunking *Remove and Reinsert Plug*			
With chunking	**7/10**	43.5 ± 7.2	9.40 ± 2.12
Without chunking	3/10	81.4 ± 24.2	16.69 ± 4.84

Table 3: **How context and execution components affect success and cost.** Success is successful/total trials. Decisions and time are mean ± sample SD over all included trials, including failures. Shaded conditions reuse Table 1 trials, and bold marks the highest success within each group. Conditions span different batches, and calibration removal also disables localization (Appendix B).

demonstrations, **goal images** specify target arrangements, **self-interaction history** supplies evidence from prior observations and actions, and **human interaction** introduces constraints during execution. All six tasks meet their success criteria in three trials each (Table 1(c–e)). For *Tic-Tac-Toe*, both wins and draws count as success. Execution cost remains substantial in *Movable Exploration*, averaging 25.5 ± 4.0 min. These cases extend adaptation beyond following demonstrations: the agent constructs action sequences online from desired outcomes and information accumulated during interaction.

4.3 In-Context Learning in Simulation

We next test one-shot execution on six RoboDojo tasks (Chen et al., 2026a), using one demonstration per task and five episodes with at most 100 model decisions each. Table 2 places these results alongside published evaluations of RoboDojo-fine-tuned $\pi_{0.5}$ (Physical Intelligence et al., 2025), RoboDawn 1-shot (Guo et al., 2026), and GPT-as-Policy Direct and Hybrid (Su et al., 2026), retaining their original evaluation protocols.

Without a fine-tuned VLA, GPT-Policy achieves a mean score of 88.7 and 83.3% success across these six tasks, compared with Hybrid's reported 62.7 and 53.3% under its original protocol. Success reaches 100% on number arrangement, bottle disposal, and object classification, and 80% on language-guided classification and sequence imitation. Tower building remains less reliable (40% versus Hybrid's 60%), identifying multi-object assembly as a remaining weakness in this evaluation. Appendix Figure 15 visualizes representative simulation trajectories.

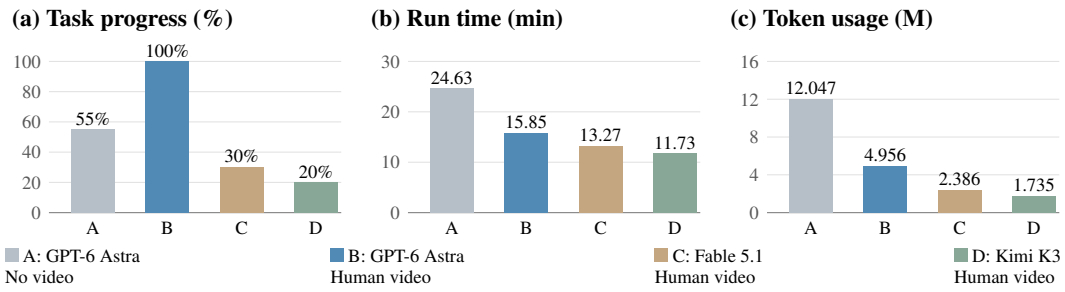


Figure 5: **Completion and cost in selected red towel pickup runs.** All conditions use GPT-Policy with different VLM backbones and context. Progress is within-run completion, not success rate. Token usage is estimated. Appendix D.1 provides run details and execution frames.

4.4 Ablation Studies

Table 3 compares context and execution components across trial batches. **Context composition.** On *Remove and Reinsert Plug*, subtask annotations with action references achieve 70% success, versus 40% with actions alone, 0% with subtasks alone, and 10% with neither. The combined condition averages 43.5 decisions versus 84.3 with actions alone, suggesting that task structure helps organize the referenced motions. The qualitative view comparison retains subtasks, actions, and the top view, while wrist views reveal additional contact and inter-arm motion cues (Figure 14). **Calibrated localization.** On *Pick Up Notebook* with human video, calibrated localization achieves 80% success and a mean time of 14.61 min, versus 30% and 22.78 min when both calibration and localization are removed. This tests the visual-to-metric pipeline as a whole (Appendix B). **Action chunking.** On the plug task, multi-waypoint decisions achieve 70% success versus 30% with single targets, with mean decisions of 43.5 versus 81.4 and time of 9.40 versus 16.69 min. Each chunk counts once, so decision counts do not measure the number of waypoints. Appendix D.4 provides qualitative comparisons.

5 Discussion

Completion and Execution Cost. Demonstrations can improve completion while reducing execution cost. In selected Astra towel runs (Figure 5), the episode with human video reaches 100% progress versus 55% without it, with 35.6% shorter run time and 58.9% lower estimated token usage. Fable 5.1 and Kimi K3 consume fewer resources but achieve less progress. These episodes illustrate why completion and cost should be assessed jointly, but they do not establish a model ranking. In preliminary evaluations, Kimi K3 and Fable 5.1 rarely completed the tested tasks under our setup. We therefore focused subsequent systematic evaluation on Astra.

Decision-Level Analysis. Context and current views guide GPT-Policy's decisions (Appendix D.6). In the towel cases, $\pi_{0.5}$ repeats grasping after completion; GPT-Policy verifies completion and stops. A separate GPT-Policy case motivates collision checks (Appendix D.7).

Limitations and Outlook. Performance depends on context quality and observation–action alignment. Model latency slows execution, and reactive recovery does not ensure collision avoidance. A key next step is a safety-aligned harness with action validation, collision checks before execution, and safe termination mechanisms (Appendices E and F).

6 Conclusion

We presented GPT-Policy, a unified harness that lets frozen VLMs adapt robot behavior from demonstrations, goal images, and interaction. Across ten real-robot and six simulated tasks, the results show how deployment-time information can guide behavior without parameter updates. The central design challenge is to connect informative context with grounded, efficient, and reliable physical execution.

AI USE STATEMENT

We used generative AI tools to assist with language editing, debugging portions of the experimental code, and drafting and revising robot control prompts. In addition, as components of the system evaluated in this work, vision-language models generated robot action requests based on task instructions, demonstrations, and interaction feedback. These uses are described in the method and experiment sections.

We did not use generative AI tools to fabricate experimental data, results, or references. The authors analyzed actual experimental records and drew the conclusions presented in this paper.

The authors reviewed all AI-assisted work, including checking that language edits preserved the intended meaning, verifying that references supported the associated claims, and reviewing and testing AI-assisted code. Robot task outcomes were determined from actual execution records and observations, rather than solely from the models' own assessments of task success.

We take full responsibility for the final content of this work, including text, code, prompts, and claims produced with the assistance of generative AI.

REFERENCES

Bo Ai, Stephen Tian, Haochen Shi, Yixuan Wang, Cheston Tan, Yunzhu Li, and Jiajun Wu. RoboPack: Learning Tactile-Informed Dynamics Models for Dense Packing. In *Proceedings of Robotics: Science and Systems*, 2024.

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.

Anthropic. Introducing Claude 4. Anthropic, 2025. URL `https://www.anthropic.com/news/claude-4`.

Anthropic. Claude Fable 5 and Claude Mythos 5. Anthropic, 2026. URL `https://www.anthropic.com/news/claude-fable-5-mythos-5`.

Anas Awadalla, Irena Gao, Josh Gardner, Jack Hessel, Yusuf Hanafy, Wanrong Zhu, Kalyani Marathe, Yonatan Bitton, Samir Gadre, Shiori Sagawa, Jenia Jitsev, Simon Kornblith, Pang Wei Koh, Gabriel Ilharco, Mitchell Wortsman, and Ludwig Schmidt. OpenFlamingo: An Open-Source Framework for Training Large Autoregressive Vision-Language Models. *arXiv preprint arXiv:2308.01390*, 2023.

Folco Bertini Baldassini, Mustafa Shukor, Matthieu Cord, Laure Soulier, and Benjamin Piwowarski. What Makes Multimodal In-Context Learning Work? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1539–1550, 2024.

Lars Berscheid and Torsten Kröger. Jerk-limited Real-time Trajectory Generation with Arbitrary Target States. In *Proceedings of Robotics: Science and Systems*, 2021.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, 2025.

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, 2020.

Shuo Chen, Zhen Han, Bailan He, Jianzhe Liu, Mark Buckley, Yao Qin, Philip Torr, Volker Tresp, and Jindong Gu. Can Multimodal Large Language Models Truly Perform Multimodal In-Context Learning? In *Proceedings of the Winter Conference on Applications of Computer Vision*, pp. 6000–6010, 2025.

Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, et al. RoboDojo: A Unified Sim-and-Real Benchmark for Comprehensive Evaluation of Generalist Robot Manipulation Policies. *arXiv preprint arXiv:2607.04434*, 2026a.

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

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, et al. DeepSeek-V3.2: Pushing the Frontier of Open Large Language Models. *arXiv preprint arXiv:2512.02556*, 2025.

Sivan Doveh, Shaked Perek, Muhammad Jehanzeb Mirza, Wei Lin, Amit Alfassy, Assaf Arbelle, Shimon Ullman, and Leonid Karlinsky. Towards Multimodal In-context Learning for Vision and Language Models. In *Computer Vision – ECCV 2024 Workshops*, pp. 250–267, 2024. doi: 10.1007/978-3-031-93806-1_19.

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.

Letian Fu, Huang Huang, Gaurav Datta, Lawrence Yunliang Chen, William Chung-Ho Panitch, Fangchen Liu, Hui Li, and Ken Goldberg. ICRT: In-Context Imitation Learning via Next-Token Prediction. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025.

Generalist Team. GEN-1.5: Embodied Foundation Models are One-Shot Learners. Generalist AI Blog, 2026. URL `https://generalistai.com/blog/gen-1.5`.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An Open-Source Generalist Robot Policy. In *Proceedings of Robotics: Science and Systems*, 2024.

GLM-4.5 Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, et al. GLM-4.5: Agentic, Reasoning, and Coding (ARC) Foundation Models. *arXiv preprint arXiv:2508.06471*, 2025.

Meng-Hao Guo, Zhe-Han Mo, Jia-Jun Wang, Yi Zhang, Kejin Wang, Yi-Xuan Deng, Jia-Peng Zhang, Yongming Rao, and Shi-Min Hu. Transferring the Intelligence of VLMs to Robotic Control. *arXiv preprint arXiv:2609.22966*, 2026. URL `https://robodawn.top/`.

Siyuan Huang, Zhengkai Jiang, Hao Dong, Yu Qiao, Peng Gao, and Hongsheng Li. Instruct2Act: Mapping Multi-modality Instructions to Robotic Actions with Large Language Model. *arXiv preprint arXiv:2305.11176*, 2023a.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9118–9147. PMLR, 2022.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 540–562. PMLR, 2023b.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. ReKep: Spatio-Temporal Reasoning of Relational Keypoint Constraints for Robotic Manipulation. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4573–4602. PMLR, 2025.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 2023.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2025.

Kimi Team, Yifan Bai, Yiping Bao, Y. Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, et al. Kimi K2: Open Agentic Intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, Ziwei Chai, Y. Charles, H. S. Che, Cheng Chen, et al. Kimi K2.5: Visual Agentic Intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Fanyi Pu, Joshua Adrian Cahyono, Jingkang Yang, and Ziwei Liu. Otter: A Multi-Modal Model with In-Context Instruction Tuning. *arXiv preprint arXiv:2305.03726*, 2023.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as Policies: Language Model Programs for Embodied Control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023.

Ji Lin, Hongxu Yin, Wei Ping, Pavlo Molchanov, Mohammad Shoeybi, and Song Han. VILA: On Pre-training for Visual Language Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26689–26699, 2024.

Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, et al. AutoGLM: Autonomous Foundation Agents for GUIs. *arXiv preprint arXiv:2411.00820*, 2024.

Runyu Lu, Yubo Wu, Ethan Kou, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, Mosharaf Chowdhury, Yuke Zhu, Linxi Fan, and Guanzhi Wang. ASPIRE: Agentic /Skills Discovery for Robotics. *arXiv preprint arXiv:2607.00272*, 2026.

Masoud Monajatipoor, Liunian Harold Li, Mozhdeh Rouhsedaghat, Lin Yang, and Kai-Wei Chang. MetaVL: Transferring In-Context Learning Ability From Language Models to Vision-Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 495–508. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-short.43.

OpenAI. Introducing upgrades to Codex. OpenAI, 2025. URL `https://openai.com/index/introducing-upgrades-to-codex/`.

OpenAI. GPT-6 Astra: A new generation of intelligence. OpenAI, 2026. URL `https://openai.com/index/gpt-6-astra/`.

Austin Patel, Ben Pekarek, Joel Enrique Castro Hernandez, and Shuran Song. Behavior Prompting Policy: Demonstrations as Prompts for Manipulation. *arXiv preprint arXiv:2606.30457*, 2026.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, et al. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

Qwen Team. Qwen3-Coder: Agentic Coding in the World. Qwen Blog, 2025. URL https://qwenlm.github.io/blog/qwen3-coder/.

Rutav Shah, Shuijing Liu, Qi Wang, Zhenyu Jiang, Sateesh Kumar, Mingyo Seo, Roberto Martín-Martín, and Yuke Zhu. MimicDroid: In-Context Learning for Humanoid Robot Manipulation from Human Play Videos. In *2026 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2026.

Lucy Xiaoyang Shi, Brian Ichter, Michael Robert Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi Robot: Open-Ended Instruction Following with Hierarchical Vision-Language-Action Models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 54919–54933. PMLR, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, pp. 245–254. Association for Computing Machinery, 1985.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. ProgPrompt: Generating Situated Robot Task Plans using Large Language Models. *arXiv preprint arXiv:2209.11302*, 2022.

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

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.

Quan Sun, Yufeng Cui, Xiaosong Zhang, Fan Zhang, Qiying Yu, Yueze Wang, Yongming Rao, Jingjing Liu, Tiejun Huang, and Xinlong Wang. Generative Multimodal Models are In-Context Learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14398–14409, 2024.

Maria Tsimpoukelli, Jacob L. Menick, Serkan Cabi, S. M. Ali Eslami, Oriol Vinyals, and Felix Hill. Multimodal Few-Shot Learning with Frozen Language Models. *Advances in Neural Information Processing Systems*, 34:200–212, 2021.

Vitalis Vosylius and Edward Johns. Instant Policy: In-Context Imitation Learning via Graph Diffusion. In *International Conference on Learning Representations*, 2025.

Wenli Xiao, Jia Xie, Tonghe Zhang, Haotian Lin, Letian Fu, Haoru Xue, Jalen Lu, Yi Yang, Cunxi Dai, Zi Wang, Jimmy Wu, Guanzhi Wang, S. Shankar Sastry, Ken Goldberg, Linxi Fan, Yuke Zhu, and Guanya Shi. ENPIRE: Agentic Robot Policy Self-Improvement in the Real World. *arXiv preprint arXiv:2606.19980*, 2026.

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.

Yida Yin, Zekai Wang, Yuvan Sharma, Dantong Niu, Trevor Darrell, and Roei Herzig. In-Context Learning Enables Robot Action Prediction in LLMs. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025.

Keunwoo Peter Yu, Zheyuan Zhang, Fengyuan Hu, Shane Storks, and Joyce Chai. Eliciting In-Context Learning in Vision-Language Models for Videos Through Curated Data Distributional Properties. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 20416–20431. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.1137.

Haozhe Zhao, Zefan Cai, Shuzheng Si, Xiaojian Ma, Kaikai An, Liang Chen, Zixuan Liu, Sheng Wang, Wenjuan Han, and Baobao Chang. MMICL: Empowering Vision-language Model with Multi-Modal In-Context Learning. In *International Conference on Learning Representations*, 2024.

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.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023.

Yongshuo Zong, Ondrej Bohdal, and Timothy Hospedales. VL-ICL Bench: The Devil in the Details of Multimodal In-Context Learning. In *International Conference on Learning Representations*, 2025.

A METHOD DETAILS

This appendix expands the context construction and execution mechanisms in Section 3. It distinguishes historical references from live evidence and numerical plan acceptance from measured execution. Appendix B lists platform settings, Appendix C reports reference data and sampling parameters, and Appendix G provides the prompts and tool formats.

A.1 CONTEXT COMPILER: FROM DEMONSTRATION TO RECORDS

The compiler produces the ordered reference records $\mathcal{R} = [d_i]_{i=1}^{N}$ in Equation 2 before control begins. Figure 6 illustrates the representation for plug removal and reinsertion. Compilation comprises semantic selection and annotation, followed by timestamp alignment and input assembly.

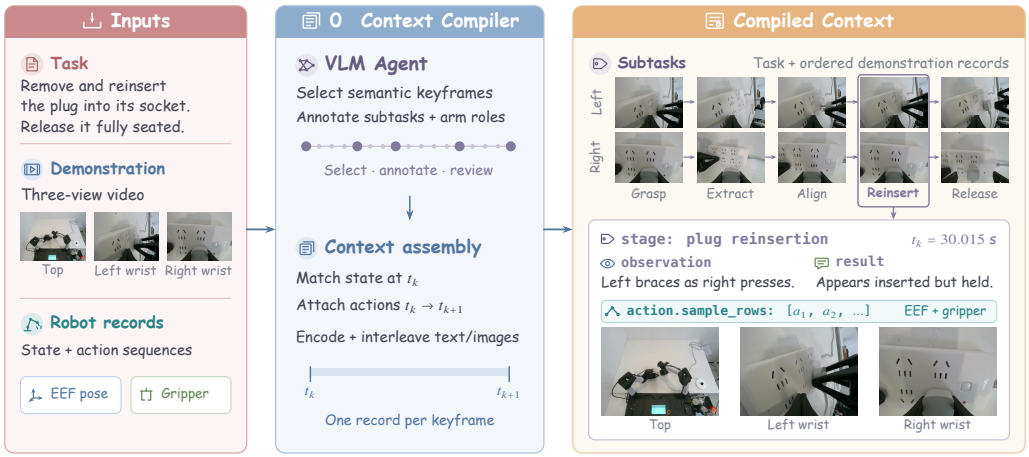


Figure 6: **Context Compiler: a subtask-structured demonstration.** For raw video, a VLM selects keyframes, annotates subtasks and arm roles, and reviews the sequence. Assembly aligns measured state to each keyframe, attaches sampled historical actions between keyframes, and interleaves the records with camera images. The grid shows wrist views at five selected stages, while the expanded reinsertion record combines annotations, three views, and `action.sample_rows`. Each a_i denotes a retained historical command row. Full columns and aligned state are omitted for readability. The illustrated bundle was reviewed beforehand and is loaded directly without repeating selection.

Selection and review. For raw video, a VLM selects informative candidates from overlapping temporal windows. Local selection preserves evidence for initial and final states, contact, grasping, release, repeated manipulation, and changes in arm roles. A global review then consolidates the selections, removes redundant holds and window boundaries, and checks annotations against the images. Before/after evidence is retained when needed to disambiguate a transition, since similar endpoint poses alone do not imply that no manipulation occurred. Preselected frames enter at the annotation stage, while a reviewed bundle is loaded directly without repeating selection. The example in Figure 6 uses this last route. Selection and review prompts are listed as P0a–P0b in Appendix G.

Subtask annotations and view binding. Each annotation ℓ_i describes the current stage, the participating arms and their roles, relevant visual evidence, and the visible outcome or remaining uncertainty. Stabilizing an object is recorded as an arm role even when that arm is stationary. These annotations describe the demonstrated procedure, but they do not instantiate an online symbolic state machine. Available camera images remain associated with their selected time and source. An overhead view conveys the overall sequence, while wrist views can disambiguate local contact, grasp direction, and coordination. The compiler preserves available views rather than requiring every reference to contain three cameras.

Measured state and intervening actions. For robot demonstrations, the assembly stage aligns measured state x_i^{demo} to the selected image time τ_i and attaches recorded commands from the interval

leading to τ_{i+1} as A_i^{demo}. State describes feedback at the keyframe, while actions describe what was commanded between keyframes. Alignment uses nearest timestamps within the configured tolerance, with missing measurements left absent. It does not imply hardware-synchronized exposure. Interval sampling retains endpoints, periodic samples, and both sides of gripper-command changes, preserving transitions that sparse images can miss. The final keyframe need not have an outgoing action interval.

The action header specifies column order and units. Repeated joint, pose, and gripper values may be encoded as = relative to the preceding row within the same interval, whose first row is explicit. Missing values remain unknown, never zero. Human video omits numerical robot state and action fields. The Video condition likewise omits these fields, while Video + Action adds them to the selected visual reference. Appendix C gives alignment tolerances, sampling rates, and retained sample counts. Listings F3a–F3b give concrete records.

Input assembly and interpretation. The loader interleaves chronological text records with source- and camera-labeled image blocks and explicit historical-reference boundaries. This representation exposes subtask semantics, visual contact relationships, and available numerical motion references together. A demonstrated pose is not a target in the current scene: the policy must account for the current object arrangement, embodiment, base frame, TCP convention, and live observations before producing a new request. Goal images are supplied as outcome references without constructing an action sequence.

A.2 Online Interaction History

The host records observations, selected requests, returned results, and available operator feedback in conversation order (Equation 3). Measured state and visual outcomes remain distinct from requested targets. The VLM uses this evidence to infer subtask progress, diagnose a rejected or ineffective action, and choose its next request.

Listing F5 illustrates the history order.

A.3 Geometric Query Contract and Validity Checks

The VLM supplies the feature correspondence to `locate_point`, which computes its geometry. The required `camera` and `pixel_xy` identify an image location, with pixel origin at the upper-left corner. Without reference fields, the query returns a calibrated ray rather than an inferred depth. Optional `reference_step` and `reference_pixel_xy` identify the same stationary feature in an earlier observation from the same wrist camera. Each ray uses the camera pose associated with its own observation, as in Equation 4.

Triangulation checks whether the two rays provide usable metric evidence. Parallax and conditioning diagnose weak depth constraints, while ray residuals measure geometric inconsistency. The positive-depth check requires the candidate to lie in front of both cameras. A candidate from degenerate or inconsistent geometry remains diagnostic. The policy may use a metric estimate only when `metric_position_available=true`. Otherwise, it receives the invalid result and diagnostics for a later decision. These checks test geometric consistency, not semantic feature identity: the stationary-feature assumption and correct correspondence are still required. This temporal wrist-camera query is separate from the multiple historical views retained by the compiler. Listing F6b gives its request format.

A.4 Target Resolution and Gripper Commands

Figure 7 expands the Cartesian execution adapter for ARX/YAM. The interpolation, joint-derivative time scaling, and bimanual synchronization below refer to these backends. Morphi Kino uses a distinct rot6d path representation, one-arm calls, and native execution checks. These differences are stated below and in Appendix B.

The planner initializes the TCP path by forward kinematics from measured joint positions. Each non-null target contains a complete `pose_xyzquat` $= [x, y, z, q_x, q_y, q_z, q_w]$ in the selected arm's base frame, with position in metres. Finite, nonzero quaternions are normalized. In a bimanual

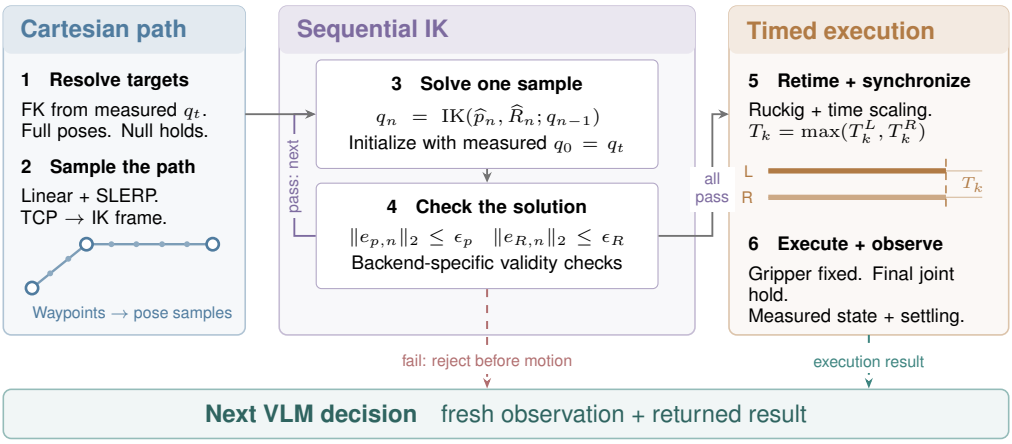


Figure 7: **From Cartesian targets to execution feedback.** The ARX/YAM adapter samples the requested path and solves IK sequentially, using each accepted solution to seed the next sample. Only a complete accepted plan is retimed and dispatched, while a failed check returns rejection feedback before motion. The VLM observes again after completion or rejection. Platform-specific checks and timing are detailed in Appendices A.6–A.7, and Morphi Kino differences appear in Appendix B.

sequence, a `null` entry repeats that arm's preceding complete pose. An arm whose entries are all `null` receives no new trajectory. Individual coordinates are not filled independently. Calibrated transforms convert TCP targets into the end-effector frame used by IK.

Cartesian trajectories retain the existing gripper command throughout. A separate `set_gripper` request changes the opening, using a normalized value in $[0, 1]$ (closed to open). Bimanual requests use `positions.left/right`. Commands and measured openings remain distinct.

A.5 Cartesian Sampling and Sequential IK

The adapter prepends the current TCP poses to the requested targets. Between adjacent targets, it interpolates position linearly and orientation along the shorter rotation arc, equivalent to quaternion SLERP (Shoemake, 1985):

$$\begin{aligned} p_k(s) &= (1-s)p_{k-1} + sp_k, \\ R_k(s) &= R_{k-1}\exp\big(s\,\mathrm{Log}(R_{k-1}^\top R_k)\big), \quad s \in [0,1]. \end{aligned} \tag{9}$$

Sampled TCP poses are transformed into the backend's kinematic frame before IK. Let $\widehat{X}_n = (\hat{p}_n, \hat{R}_n)$ be the nth transformed sample and let $q_0 = q_t$ be the measured joint configuration at planning time. The adapter solves

$$q_n = \mathrm{IK}(\hat{p}_n, \hat{R}_n; q_{n-1}), \qquad n = 1, \dots, M, \tag{10}$$

using each accepted solution as the next seed. The VLM supplies sparse Cartesian waypoints, not these joint samples. Each sample is checked using the backend's acceptance rules below. A rejected plan returns its reason before motion submission. Sequential seeding encourages continuity but is not, by itself, a hard bound on joint displacement.

A.6 IK Residuals and Backend Checks

For a target $(\hat{p}_n, \hat{R}_n)$ and forward kinematics $(p(q_n), R(q_n))$ at the same solver-facing end-effector frame, define

$$e_{p,n} = p(q_n) - \hat{p}_n, \qquad e_{R,n} = \mathrm{Log}\Big(\hat{R}_n^\top R(q_n)\Big)^\vee. \tag{11}$$

Here $(\cdot)^\vee$ converts a skew-symmetric matrix to a rotation vector, whose norm is the relative rotation angle. Residuals are checked after the TCP-to-kinematic-frame conversion. Accepted solutions

must satisfy

$$\|e_{p,n}\|_2 \le \epsilon_p, \qquad \|e_{R,n}\|_2 \le \epsilon_R. \tag{12}$$

For ARX and YAM, the execution tolerances are 0.002 m and approximately 1°, while the numerical stopping tolerances are 10^{-4} m and 5×10^{-4} rad. These checks establish numerical consistency of the planned reference, not physical positioning accuracy.

ARX uses SDK IK followed by damped least-squares refinement, with joint bounds supplied to the solver and used to clip refinement updates. Its final acceptance test checks pose residuals. YAM uses I2RT kinematics and additionally checks the finite solution against effective SDK joint bounds. Neither adapter requires the solver's convergence flag when the execution checks pass. Neither ARX nor YAM imposes a separate hard inter-sample joint-step bound. Morphi Kino uses native numerical IK with analytic fallback and execution residual thresholds of 0.003 m and 0.02 rad. Before motion, it checks finite solutions, native joint bounds, and inter-sample joint changes below 0.15 rad. Joint velocities respect URDF limits and a 0.2 rad/s cap.

A.7 Timing, Synchronization, and Feedback

For ARX/YAM, nearly coincident orientations use normalized linear interpolation. For moving segments, Ruckig (Berscheid & Kröger, 2021) assigns timestamps through a scalar progress profile with zero endpoint velocity and acceleration. Successive finite differences on the timed joint samples estimate velocity, acceleration, and jerk. Let r_v, r_a, r_j be their largest absolute derivative-to-limit ratios over all samples and joints. Time is stretched by

$$\alpha_0 = \max\{1, r_v, \sqrt{r_a}, \sqrt[3]{r_j}\}, \qquad \alpha = \begin{cases} 1, & \alpha_0 = 1, \\ 1.001\alpha_0, & \alpha_0 > 1, \end{cases} \qquad \tau_k' = \alpha\tau_k. \tag{13}$$

Joint samples remain unchanged, while the computed derivatives scale by α^{-1}, α^{-2}, and α^{-3}. This checks the sampled reference, not continuous-time physical jerk.

For ARX/YAM, both arms are planned before submission. Corresponding segments are synchronized to the longer duration by slowing the faster trajectory. ARX submits timestamped references, whereas YAM interpolates the timed joint references for 100 Hz streaming. Both append a final hold and report measured settling separately. The VLM receives fresh visual evidence after completion or rejection, with no intermediate VLM observation inside a submitted chunk. Low-level feedback remains active during motion. IK rejection returns feedback before submission, while runtime faults or operator interruption can end execution. Where available, `check_path` evaluates a candidate without dispatching motion, and execution replans from fresh feedback.

The Cartesian planner does not check collisions. Reference acceptance establishes numerical consistency under the implemented checks, while settling reports measured joint behavior. Neither establishes collision clearance, secure grasping, or physical task success. Model-issued completion requests are evaluated separately from the physical success criterion.

B Robot Configurations

Evaluation records. For $\pi_{0.5}$ in Table 1, bottle opening has two confirmed successes among ten trials, with four outcomes pending review. Tic-Tac-Toe and Pointed Fruit Pickup each report three examples selected by recording-duration quantiles, rather than the full evaluation set.

Ablation records. In Table 3, the subtask/action and chunking baselines reuse the ten main-experiment trials. The no-subtask/with-action condition includes a quota-exhaustion failure, while the no-subtask/no-action condition has ten trials. Calibration batches differ in machines and context settings, and `calibration_off` also disables `locate_point`. Notebook run 8 without calibration is counted as failed, whereas zero-decision interruptions and model-service failures are excluded from that group. Chunking-on pools demonstration and prompt versions and includes a human-interrupted failure, while chunking-off includes a right-arm control-service failure. These aggregate comparisons do not isolate single-variable effects.

Robot settings. Table 4 summarizes the principal robot, sensing, planning, and execution settings in the YAM, ARX X5, and Morphi Kino implementations inspected. These are source-configuration defaults, and individual runs may override them. YAM and ARX X5 each use two six-joint arms and top, left-wrist, and right-wrist RGB views. Morphi Kino uses two seven-joint arms and head, chest, left-wrist, and right-wrist RGB views, together with a mobile base, articulated waist, and head. Camera and TCP conventions are installation-specific.

Table 4: **Robot, motion, and execution configurations.** Planning limits apply to the generated reference. Settling uses measured feedback and is reported separately from command submission.

Setting	YAM	ARX X5	Morphi Kino
Sensing and robot interface			
Arm joints	6 per arm	6 per arm	7 per arm
RGB views	Top, two wrists	Top, two wrists	Head, chest, two wrists
Live RGB resolution	640×480	640×480	1280×720
Live JPEG quality	85	85	N/A
TCP reference	Calibrated `grasp_site`	Inner-fingertip midpoint	`Larm08_link` / `Rarm08_link`
Gripper command	Normalized $[0, 1]$	Normalized $[0, 1]$	Normalized $[0, 1]$
Nominal opening width	0.095 m	0.088 m	N/A
Path planning and inverse kinematics			
Nominal trajectory rate	100 Hz	100 Hz	10 Hz
Translation / rotation sampling	0.005 m / 0.035 rad	0.005 m / 0.035 rad	0.003 m/axis / 0.02/rot6d component
TCP linear / angular speed	0.08 m/s / 0.5 rad/s	0.08 m/s / 0.5 rad/s	0.03 m/s/axis / Not specified
Joint velocity limit	0.6 rad/s per joint	$0.25\, V_i^{\mathrm{SDK}}$	$\min(0.2, V_i^{\mathrm{URDF}})$ rad/s
Joint acceleration limit	2 rad/s^2 per joint	2 rad/s^2 per joint	N/A
Joint jerk limit	12 rad/s^3 per joint	12 rad/s^3 per joint	N/A
Numerical IK tolerances	10^{-4} m, 5×10^{-4} rad	10^{-4} m, 5×10^{-4} rad	Native service defaults
Execution IK tolerances	0.002 m, $\simeq 1^\circ$	0.002 m, $\simeq 1^\circ$	< 0.003 m, < 0.02 rad
IK backend	I2RT kinematics	ARX SDK + DLS refinement	Numerical IK + analytic fallback
Execution and measured settling			
Reference submission	Interpolation and streaming	Timestamped SDK trajectory	Robot-local joint streaming
Bimanual start delay	0.1 s per arm clock	0.1 s per arm clock	N/A (one arm per call)
Final reference hold	0.12 s	0.12 s	1 s
Settling position tolerance	0.03 rad over the window	0.03 rad	< 0.015 rad (final sample)
Settling window	At least 0.3 s and 10 distinct samples	10 consecutive samples	Last 10 reads of final hold
Settling motion criterion	Encoder span ≤ 0.002 rad and span/time ≤ 0.05 rad/s	Measured joint speed ≤ 0.05 rad/s	Span < 0.005 rad for stable misses
Settling timeout	3 s	3 s	1 s

Here V_i^{SDK} denotes the initialized ARX SDK velocity limit for joint i, and V_i^{URDF} denotes the Morphi Kino URDF velocity limit. YAM directly uses 0.6 rad/s, and its adapter does not apply the velocity-scale field also present in the JSON profile. YAM and ARX X5 gripper widths are nominal conversions of normalized readings. Morphi Kino exposes normalized aperture without a configured metric opening-width conversion.

Morphi Kino represents orientation using six dimensionless rotation-matrix components (rot6d). Its component sampling budget therefore is not an angular increment in radians. At the default 10 Hz reference rate, the XYZ component budget implies 0.03 m/s per coordinate, rather than a 0.03 m/s Euclidean TCP speed limit. Its Ruckig configuration shapes reference timing, but it does not impose explicit joint acceleration or jerk limits after IK. Adaptive IK sampling retains the complete reference and validates interpolated candidates before motion. Joint-rate scaling can extend execution time, with both reference duration and actual playback separately capped at 30 s.

Morphi Kino evaluates arm arrival using the final measured joint errors after its 1 s hold. A stable miss within the 0.08 rad tracking guard can return `target_incomplete` for policy assessment. The span criterion applies to these incomplete results, while the arrival test does not independently require a measured-speed threshold or distinct timestamped samples. Gripper completion is assessed separately. Fresh healthy IDLE feedback and stop confirmation are required before execution completion is accepted.

A settling timeout in YAM or ARX X5 is reported as an unsettled result. Collision checking is not part of the Cartesian planner, and runtime diagnostics and provider-specific checks do not establish collision-free motion or physical task success.

Provider-dependent behavior. The ARX Claude Messages execution wrapper supplies TCP geometry, applies motion rejection rules, and reviews completion in a separate model session, returning unverified completion requests to the control loop. The inspected YAM branch omits these additional execution and review procedures.

Morphi Kino uses the generic agent policy with embodiment-specific tool wrappers and native execution checks. These validate IK residuals, joint bounds, continuity, velocity, feedback freshness, tracking, and stopping, and return eligible rejections or incomplete outcomes with fresh observations for replanning. Its current task input uses four RGB views without depth. Model-reported completion and native execution receipts remain distinct from independently verified physical task success.

C Demonstration Data Details

C.1 Reference Sources and Collection Protocol

Human demonstrations. We record a person performing the task in a first- or third-person RGB video. Selected frames show the approach, object interaction, and outcome, without numerical robot states or actions (Table 5).

Table 5: **Human demonstration examples.** Each reference uses one view, and time is relative to the video start. Remove Glue Cap is an additional reference task beyond the main comparison.

Task	Keyframes	Selected video timestamps (s)
Pick Red Towel	8	0.000, 5.190, 7.257, 14.488, 18.622, 19.655, 21.722, 23.755
Pick Up Notebook	6	0.000, 1.967, 3.433, 4.400, 5.867, 8.800
Remove Glue Cap	7	0.000, 2.100, 6.267, 7.833, 8.900, 10.467, 12.000

Teleoperated robot demonstrations. During teleoperation, we record timestamped images, measured joint and end-effector states, and motion and gripper commands. *Unscrew Bottle Cap* uses an overhead view, whereas *Remove and Reinsert Plug* adds two wrist views. Recording covers the full manipulation through release and withdrawal, preserving both the action sequence and visible outcome.

Goal images. We obtain goal images as overhead-view screenshots or photographs taken by an operator. Each image shows the desired object arrangement and is supplied with the task instruction and live observations. It specifies object identities, relative positions, and spacing, without prescribing an action sequence or providing recorded robot states.

C.2 Robot Demonstration Content

A keyframe denotes one selected time and can contain multiple camera views (Table 6). Video and Video + Action share the selected images, but only the latter includes measured states and recorded action segments. No-demonstration inputs omit the reference.

Table 6: **Robot demonstration content by input mode.** Images include all views. States/segments count keyframes with measured state/action intervals. Samples are reference records before/after context sampling, not policy decisions.

Task	Input mode	Keyframes	Images	States	Segments	Action samples before / after
Unscrew Bottle Cap	None	0	0	0	0	0 / 0
	Video	13	13	0	0	0 / 0
	Video + Action	13	13	13	12	212 / 205
Remove and Reinsert Plug	None	0	0	0	0	0 / 0
	Video	14	42	0	0	0 / 0
	Video + Action	14	42	14	13	1,405 / 131

C.3 KEYFRAME SELECTION AND INPUT REPRESENTATION

For automatic keyframe selection, a vision model selects key moments from candidate frames in overlapping video windows. Global review removes redundant holds while retaining initial and final states, contact, release, and arm-role changes. The reference contains at most 24 keyframes and 48 images, resized to fit within 1,280 pixels per dimension without upscaling. Input interleaves chronological images with relative times, camera labels, and available stage annotations. Reference boundaries distinguish demonstrations from live observations. Video + Action additionally includes aligned numerical records.

C.4 STATE-ACTION ALIGNMENT AND SAMPLING

We align camera images with measured joint and end-effector states, gripper openings, and issued commands on a shared timestamp axis. Each keyframe uses the nearest state within 0.1 s, and additional camera views are matched to the overhead image within the same tolerance. Missing measurements remain absent. This is timestamp matching, not hardware-synchronized exposure.

At successive selected video times t_i and t_{i+1}, the image at t_i is paired with its measured state and the action segment leading to t_{i+1}. For plug removal, this links the grasp image and gripper state to the subsequent withdrawal commands. We retain segment endpoints, approximately one action sample per second, and both sides of gripper-command changes. Commands remain distinct from measured feedback. These records guide Video + Action reasoning, not direct trajectory replay.

D ADDITIONAL QUALITATIVE RESULTS

Figures 8–10 provide the full-size task sequences corresponding to the compact overview in Figure 3.

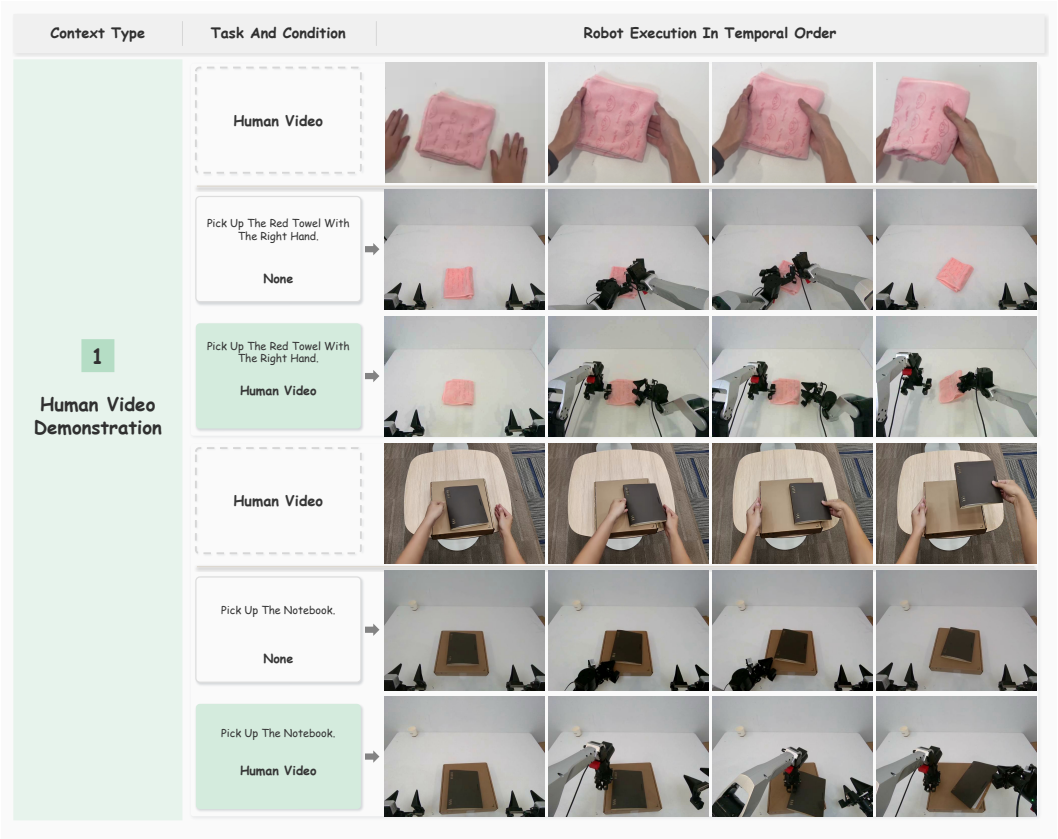


Figure 8: **Human demonstrations and robot executions for towel and notebook pickup.** For each task, we show example runs under two context conditions: None and Human Video. Frames progress from left to right.

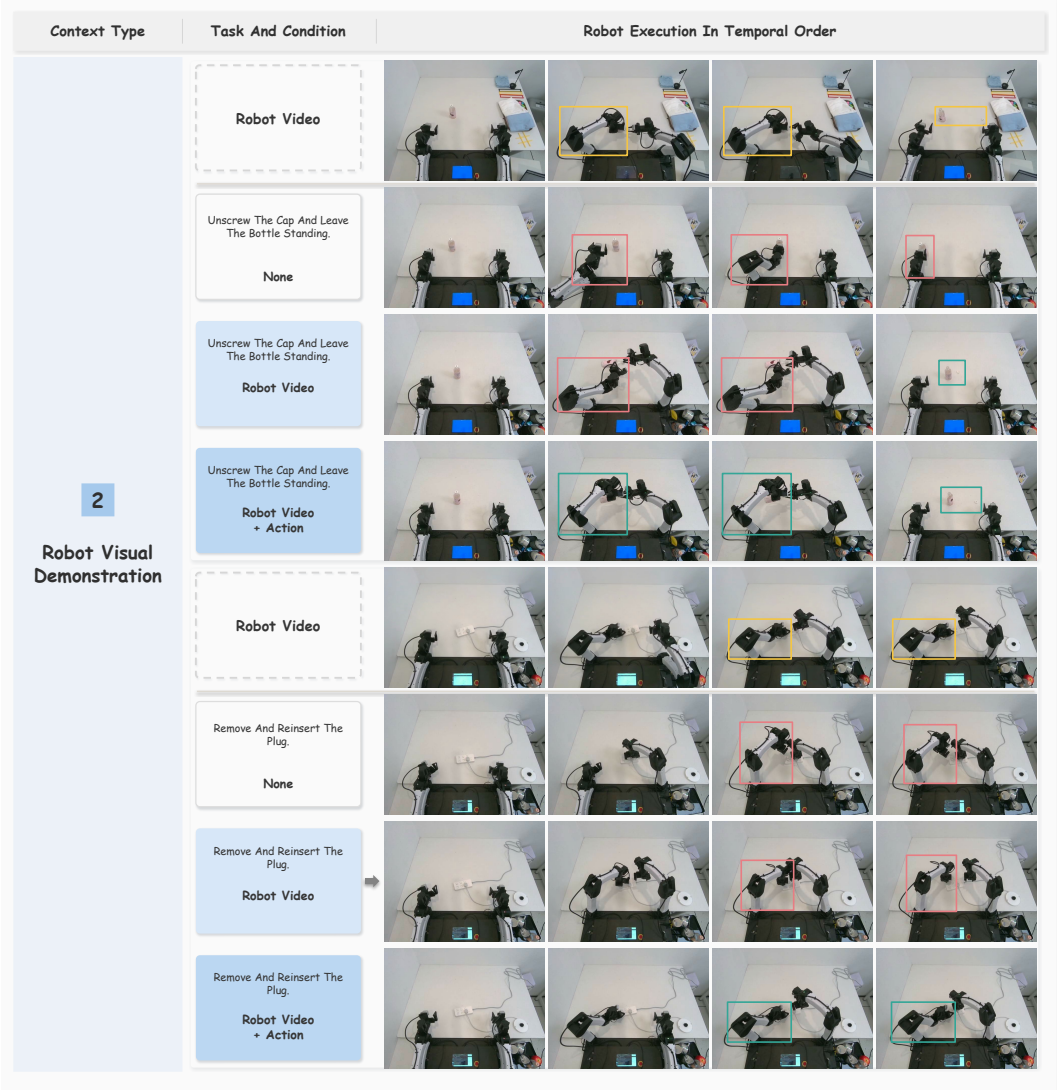


Figure 9: **Robot demonstrations and executions for bottle opening and plug reinsertion.** For each task, we show example runs under three context conditions: None, Robot Video, and Robot Video + Action. Frames progress from left to right. Gold boxes mark reference regions in the demonstrations. Red marks local deviations, while green marks closer matches.

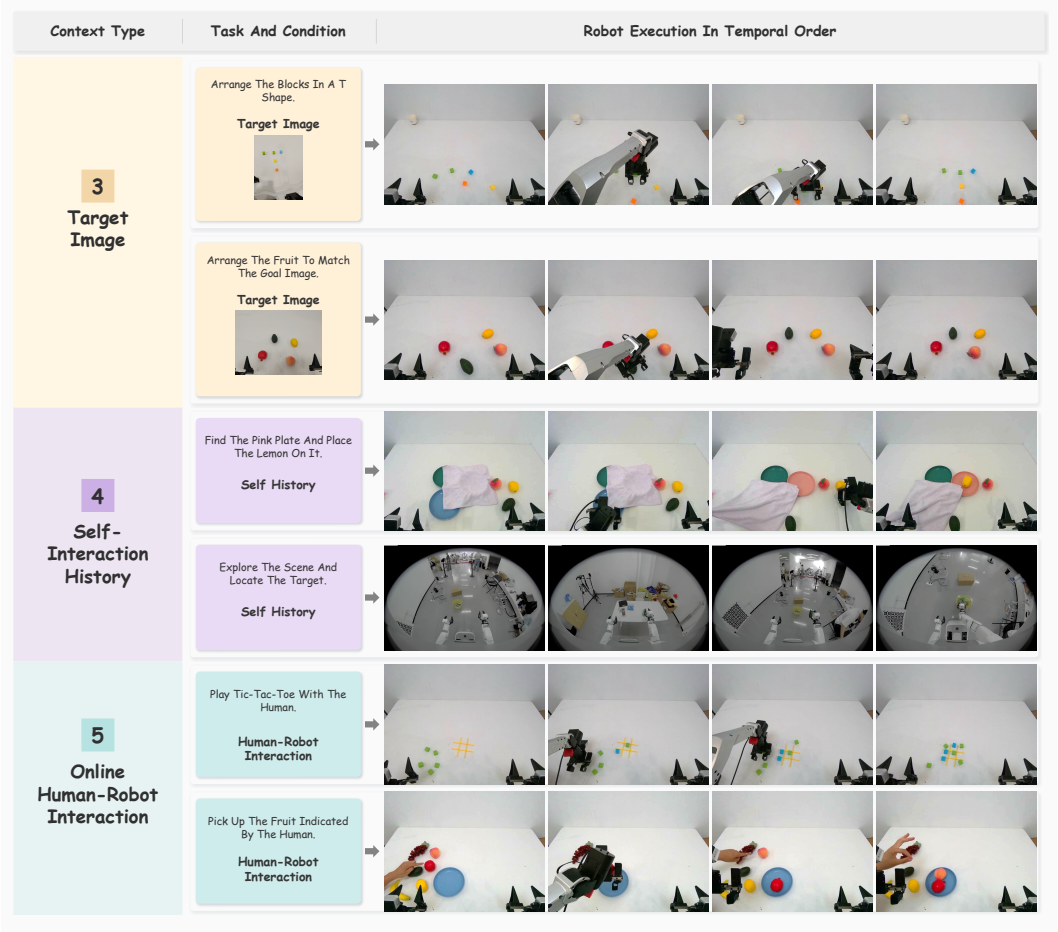


Figure 10: **Goal images, self-interaction history, and online human interaction.** We show two example runs per context condition, one per row. Frames progress from left to right.

D.1 VLM Backbone Comparison

Table 7 records the exact values plotted in Figure 5. Figure 11 preserves the original composite with execution frames. These are selected individual runs, not aggregate success rates or a statistically established model ranking.

VLM backbone	Human video	Progress	Time (min)	Tokens (M)
GPT-6 Astra	No	55%	24.63	12.047
GPT-6 Astra	Yes	100%	15.85	4.956
Fable 5.1	Yes	30%	13.27	2.386
Kimi K3	Yes	20%	11.73	1.735

Table 7: **VLM backbones within GPT-Policy.** Selected runs on red towel pickup. Progress measures task completion, not success rate. Token usage is estimated (M: million).

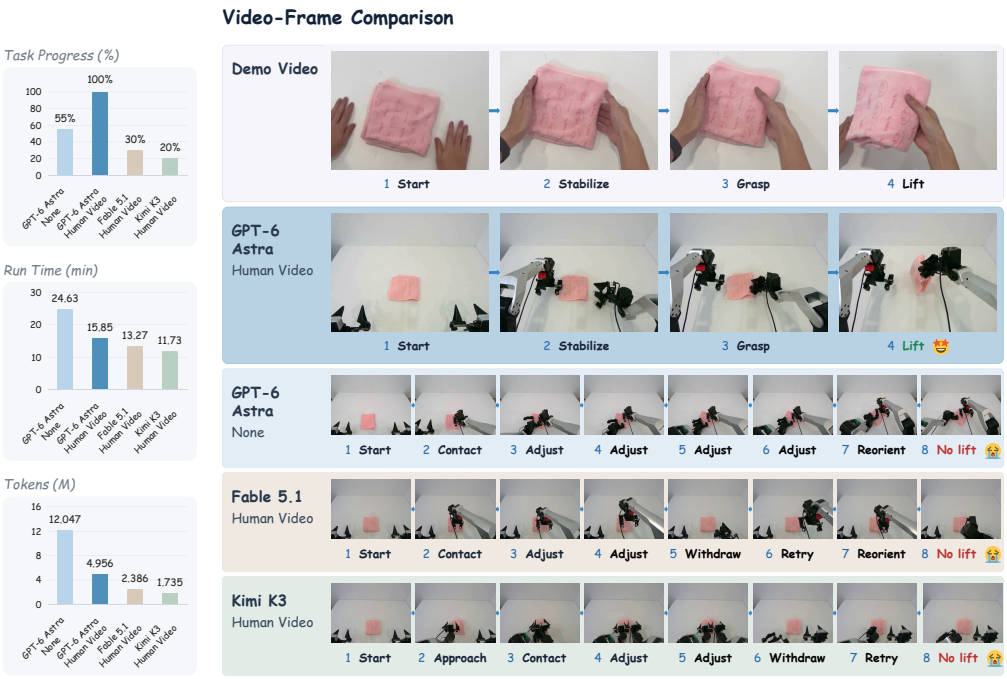


Figure 11: **Full comparison of selected red towel pickup runs.** Demonstration and execution frames show the manipulation stages corresponding to the individual runs summarized in Figure 5 and Table 7. Progress denotes completion within a run, not success rate.

D.2 Action References in Bottle Opening

Figure 12 provides complementary evidence: in selected bottle-opening runs, action references bring both commanded orientations and measured supporting-arm postures closer to the demonstration. Their denser temporal sampling supplies intermediate target poses and gripper transitions, which may constrain motions left ambiguous by sparse video keyframes.

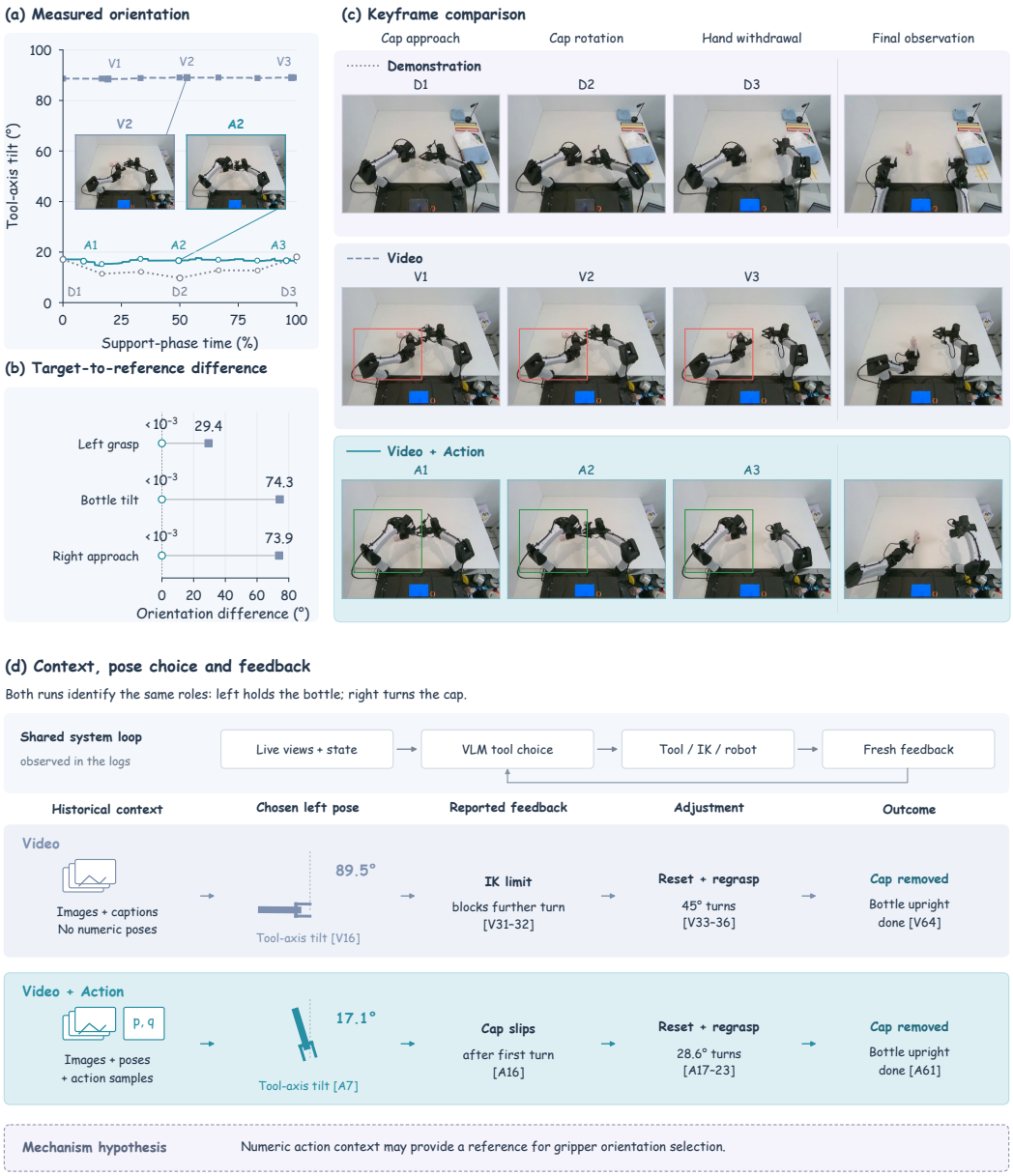


Figure 12: **Support posture and feedback in bottle opening.** Selected Video and Video + Action runs. **(a)** Measured supporting-gripper tilt (0° downward, 90° horizontal), with time normalized within each support interval. **(b)** Command–demonstration orientation differences at grasp, bottle tilt, and cap approach (lower is better). **(c)** Demonstration and execution frames. **(d)** Logged context, pose choices, feedback, adjustments, and outcomes. Bracketed V/A indices denote model decisions. Both runs remove the cap, while Video + Action more closely matches the demonstrated support posture.

D.3 Action References in Plug Manipulation

Figure 13 extends the plug case in Figure 4 with logged pose choices and execution feedback. Both runs assign the same roles to the arms, but select different supporting-gripper orientations and make different subsequent adjustments.

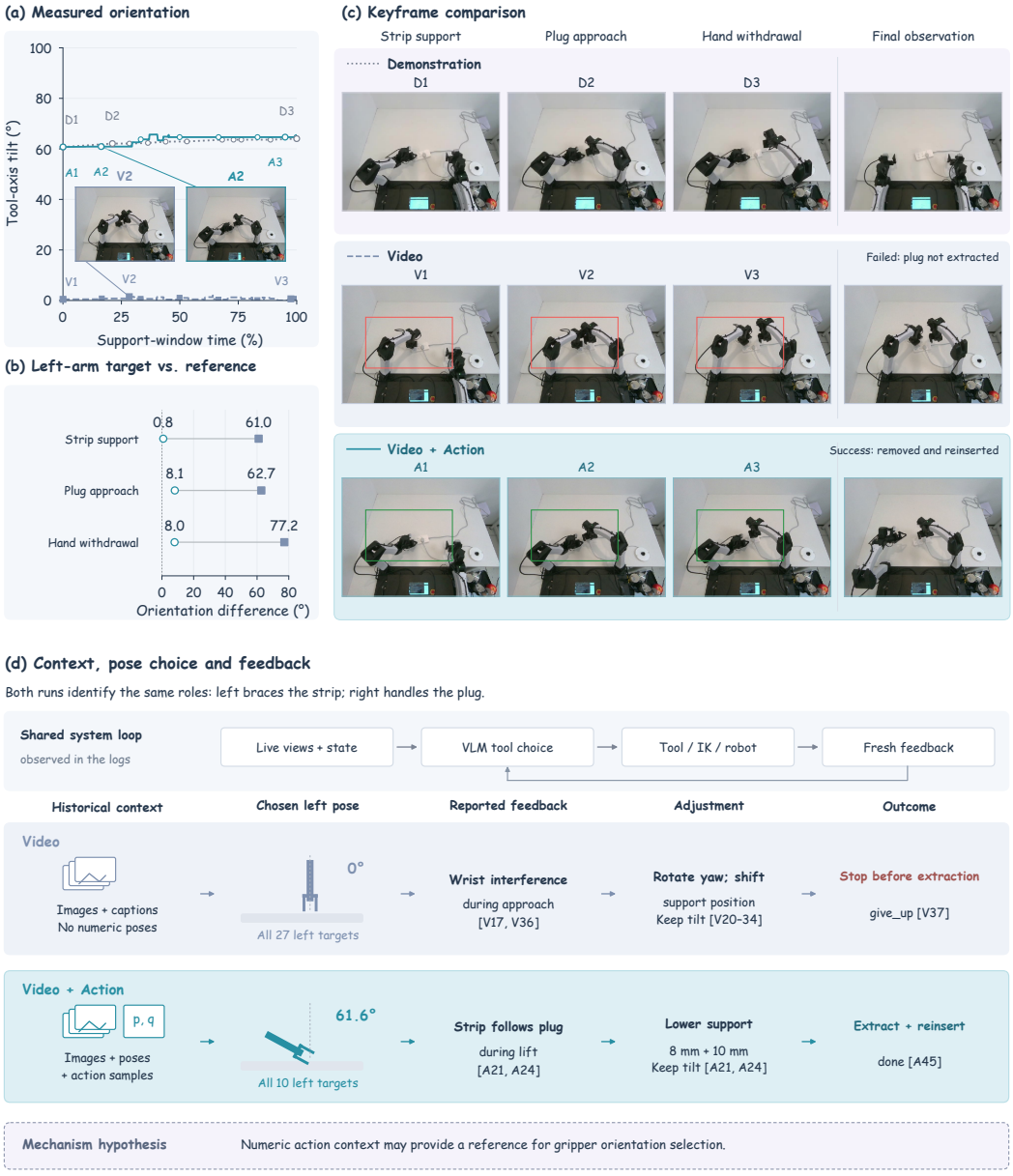


Figure 13: **Support posture and feedback in plug manipulation.** **(a)** Measured supporting-gripper tilt (0° downward, 90° horizontal), with time normalized independently within each support window. **(b)** Angular differences between commanded left-gripper orientations and the demonstration at three selected observations (lower is better). **(c)** Demonstration and execution frames. **(d)** Logged context, pose choices, feedback, adjustments, and outcomes. Bracketed V/A indices denote model decisions. Video stops before extraction, whereas Video + Action completes removal and reinsertion.

D.4 Qualitative Ablations

Figure 14 shows representative execution sequences for the context and execution ablations in Table 3.

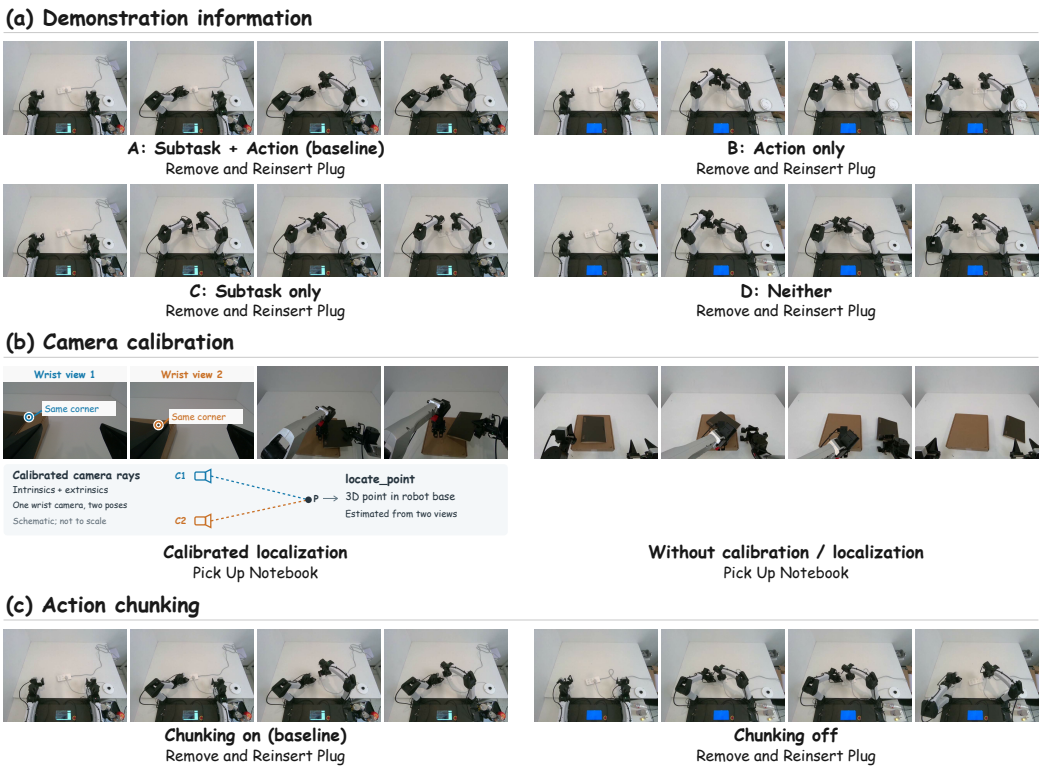


Figure 14: **Qualitative ablation results.** Representative execution frames progress from left to right. **(a)** Demonstration information, **(b)** camera calibration, and **(c)** action chunking follow the condition order in Table 3.

D.5 Simulation Execution Sequences

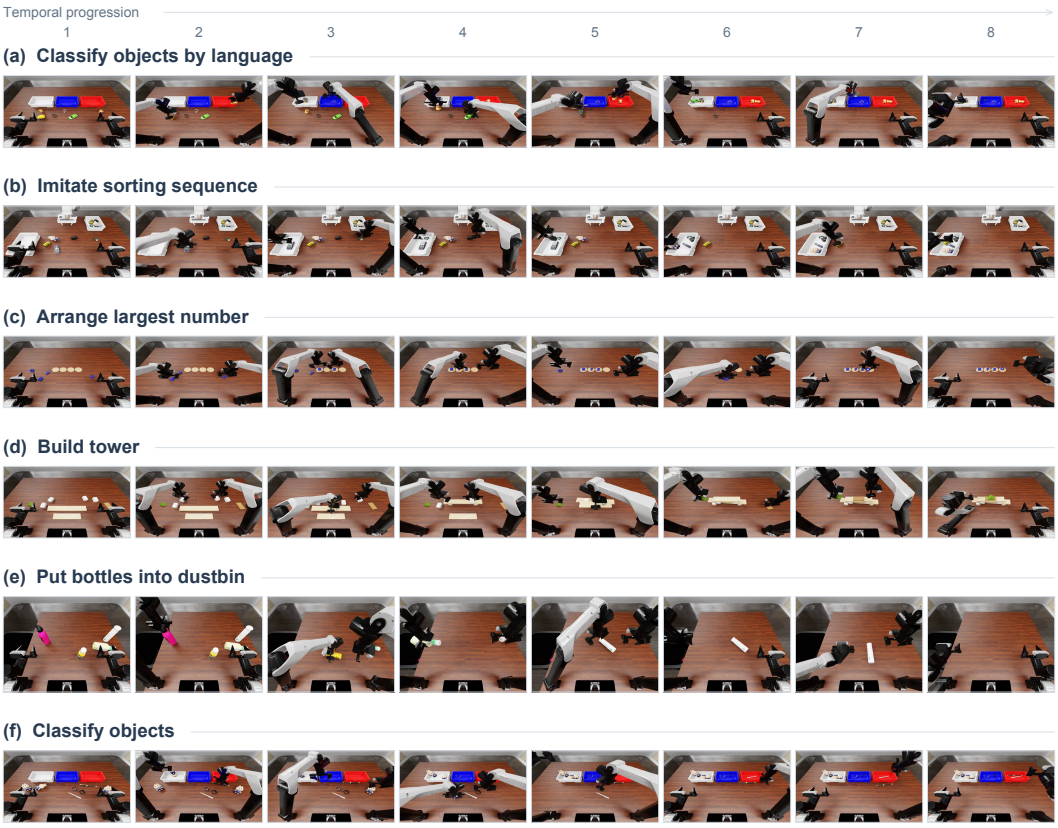
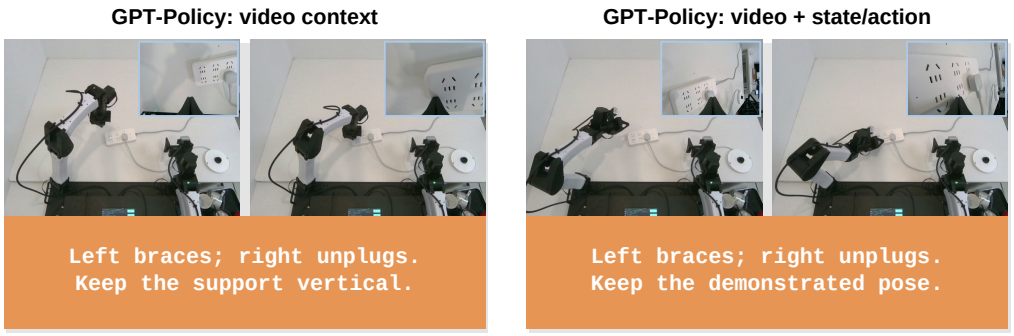


Figure 15: **Qualitative execution sequences in simulation.** Each row shows eight frames in chronological order from left to right, following the task order in Table 2. In (b), frames show robot execution after the observed sorting demonstration.

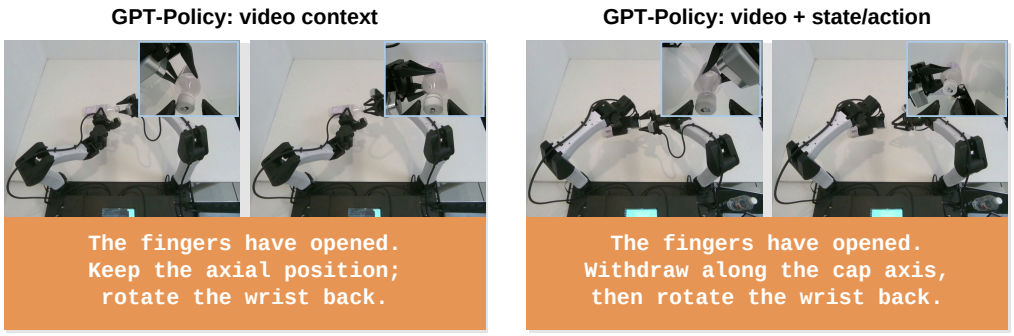
D.6 Context, Geometry, and Active Perception

Shared task semantics leave contact geometry underspecified. Across contexts, GPT-Policy preserves arm roles but changes support orientation and wrist-reset motion (Figure 16a–b). The decisions concern pose selection and contact release. The video-only bottle run later adopts withdrawal too; both runs succeed.

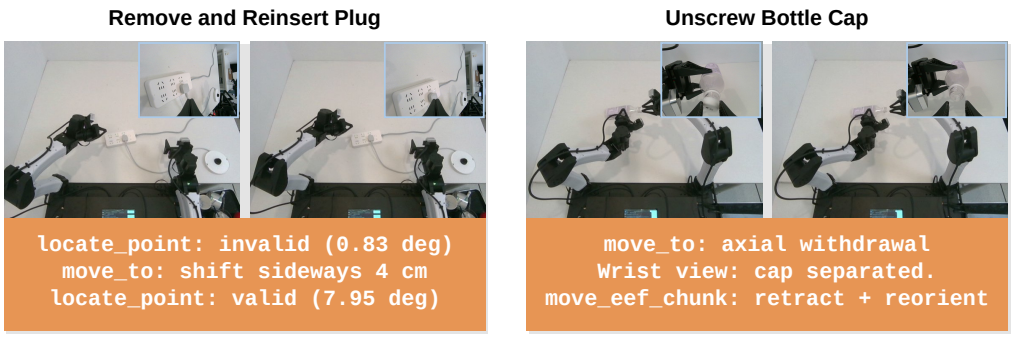
Current views inform whether to act or observe again (Figure 16c). When localization fails, GPT-Policy shifts laterally and retries; during cap removal, wrist imagery confirms separation before retraction. The observable loop is geometry selection, feedback inspection, and action revision; these traces do not identify internal reasoning mechanisms.



(a) Remove and Reinsert Plug: support orientation



(b) Unscrew Bottle Cap: wrist-reset strategy



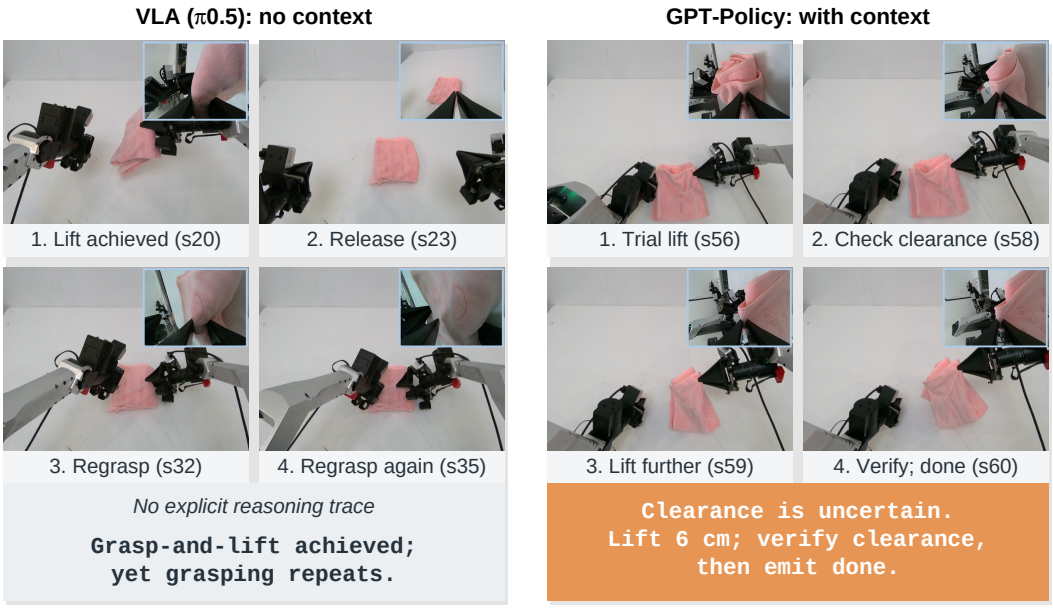
(c) Current views: depth estimation and separation verification

Figure 16: **Where context and current views enter decisions.** (a) Support orientation; (b) the first wrist reset; (c) online localization and separation verification. Paired runs share demonstration pixels (42 plug images; 13 bottle images) and control prompts; state/action context adds numerical states, poses, and trajectories. Initial conditions are unmatched, so these cases do not isolate context effects or test demonstration viewpoints. Orange bars summarize GPT-Policy's logged rationale; tool calls are named explicitly. The parallax values are tool feedback, and accepted triangulation does not establish ground-truth accuracy.

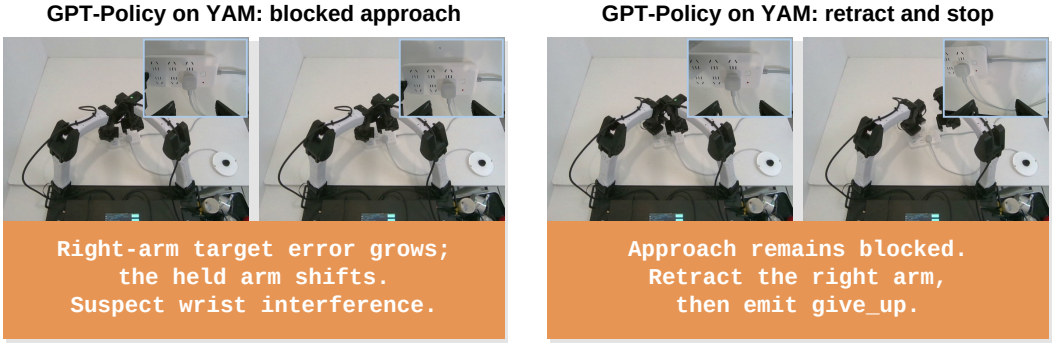
D.7 Completion, Repetition, and Bimanual Safety

Achieving a grasp and deciding to stop are distinct requirements. In *Pick Red Towel*, $\pi_{0.5}$ lifts the towel, releases it, and makes at least two further grasps; a separate GPT-Policy episode checks clearance before emitting `done` (Figure 17a). Here, $\pi_{0.5}$ receives current images, state, and the instruction, but no demonstration or interaction history; GPT-Policy receives both. The action-only interface exposes no explicit reasoning trace or semantic stop output; repetition therefore concerns termination in this deployment, without establishing memorization or a causal context effect.

Reactive recovery also leaves a safety gap. On YAM, tracking error and supporting-arm displacement lead GPT-Policy to suspect wrist interference, retract, and eventually emit `give_up` (Figure 17b). Recovery follows blockage, motivating collision checks before execution.



(a) Pick Red Towel: repetition after pickup vs. verified termination



(b) Remove and Reinsert Plug: bimanual safety

Figure 17: **Repetition and bimanual interference require different safeguards.** (a) Recorded policy observations, read left to right, then top to bottom. The $\pi_{0.5}$ excerpt comes from a human-success-labelled run after a manual towel reset; the run ends by operator stop. Indices denote logged policy steps: action chunks for $\pi_{0.5}$, tool decisions for GPT-Policy, with unequal durations. Frames precede the indexed policy call; GPT-Policy labels describe the ensuing decisions. Grey text describes observed behavior; orange text summarizes logged GPT-Policy rationale. (b) Right-arm target error reaches 20.0 mm while the held left arm shifts 3.3 mm. Interference is a logged diagnosis, not force-verified contact. The towel episodes differ in initial conditions, model, and interface.

E Research Agenda for Context-Driven Robot Adaptation

This appendix expands the six research directions motivated by the preliminary evidence and position in Section 5.

Physical safety for VLM-driven manipulation. We repeatedly observed collisions between the two arms during manipulation. This motivates a dedicated safety layer that checks both arms' planned trajectories together, monitors separation and contact during execution, and can interrupt unsafe commands independently of the VLM. Collision-aware planning and uncertainty-aware execution limits should be evaluated alongside task success, with explicit reporting of collisions, near misses, and safety interventions.

Contact-aware harnesses for reliable grasping. Building on existing execution and rejection feedback, future harnesses should expose grasp stability and object motion after contact. Slip detection, force-aware limits, and local recovery could strengthen rigid and deformable-object grasping without requiring the agent to reason through every correction.

System 1 / System 2 for fine-grained action. Hierarchical systems such as Hi Robot separate contextual reasoning from low-level execution (Shi et al., 2025). A promising extension pairs a deliberative System 2 agent with a fast System 1 controller, such as a VLA policy, for pose refinement and bimanual coordination. The key question is when local feedback should trigger replanning.

Mobile manipulation through active perception. Building on affordance-grounded skill selection (Ichter et al., 2023), a mobile agent should choose where to look and stand as well as how to grasp. Persistent spatial memory and coordinated base–arm control would support tasks in which navigation changes visibility, reachability, and the meaning of earlier observations.

Compositional context for long-horizon tasks. Trajectory prompting, as in ICRT (Fu et al., 2025), provides a starting point for treating demonstrations as structured sensorimotor context. A useful next test is whether an agent can compose subskills from multiple demonstrations in a new sequence while retaining completed subgoals and discarding obsolete context, rather than replaying an entire trajectory.

In-context adaptation to physical dynamics. Tactile-informed dynamics models (Ai et al., 2024) and video-conditioned world-action models (Zhou et al., 2026) offer complementary starting points. Future systems could use recent interactions to update predictions of friction, compliance, and object response, then adjust actions before contact failures accumulate. This would test adaptation to changing physics, not only to a new task description.

F Takeaways

A Little Context Goes a Long Way. VLM Agents are surprisingly good at learning on the fly. Human videos reveal how a task unfolds, while action references provide more precise motion cues. In these early trials, such context helps Agents step outside their usual comfort zone and tackle fine manipulation and deformable-object handling.

Good Plans, Tricky Execution. Context can sharpen task understanding, planning, and adaptation. Still, knowing what to do is not the same as doing it reliably. Precise pose generation, contact execution, outcome verification, and physical safety remain challenges of their own.

Moves Take Time. VLM Agents can already generate useful actions, but each decision can still be slow and expensive. Specialized VLA/WAM models may therefore have an edge in fast, low-level control, while VLM Agents focus on reasoning, adaptation, and replanning.

G PROMPT DETAILS

This appendix presents controller instructions, context formats, and representative task prompts. Instructions are grouped by function, with implementation-specific identifiers abstracted and paragraph breaks added for readability. Angle brackets mark variable inputs. Numerical demonstration excerpts are historical records, not executable targets.

G.1 PROMPT ORGANIZATION AND SHARED INSTRUCTIONS

Table 8 summarizes the instruction templates (P0–P4). The F-series listings illustrate context and tool formats.

Table 8: **Overview of prompt functions.**

Prompt	Function
P0a	Select informative demonstration keyframes within each video window.
P0b	Review the full demonstration and consolidate the selected frames and annotations.
P1	Define coordinate conventions, camera evidence, and the structured response contract.
P2	Interpret historical demonstrations and distinguish Video from Video + Action inputs.
P3	Guide motion selection, grasp verification, transport, and release.
P4	Diagnose failures, choose safe recoveries, and verify termination.

P0a Select demonstration keyframes

```
[BEFORE CONTROL / LOCAL VIDEO WINDOW]

[Selection objective]
Select demonstration keyframes from the supplied chronological video window and
candidate images.  For the current task, select the smallest set that conveys the
initial state, key actions, state changes, and final outcome.  Preserve before/after
evidence for grasping, release, and handoffs, and the order of repeated twists.  Similar
start/end poses do not imply no motion.
[Multi-view evidence]
Candidates may contain multiple camera views.  Combine wrist close-ups with the top
overview, and do not rely on an occluded top view alone.  Record contact side,
object-to-gripper orientation, gripper-to-table direction, and distinctions such as
empty-gripper recontact after release.
[Annotations and output]
Use short stage names.  Left/right annotations describe robot or human hand roles.
Stabilizing an object is an important action.  State only image-supported outcomes,
including uncertainty or occlusion, since gripper closure alone does not prove a grasp.
Explain each selected frame's visual evidence.  Do not invent unfamiliar
objects or actions.
Cover the window's beginning and end with minimal redundant imagery.  Select only
supplied candidate indices and return chronological order.  You have no shell, file, or
robot-control tools.  Return only JSON matching the supplied output schema.
```

P0b Review the complete demonstration

```
[BEFORE CONTROL / GLOBAL KEYFRAME REVIEW]

[Review the images]
Review the full historical demonstration before robot execution and return a concise,
complete keyframe set.  Inputs are preliminary images and annotations selected per
window, not yet globally deduplicated.  Verify the images rather than blindly trusting
local annotations.
Each time may include top, left-wrist, and right-wrist views.  Use wrist close-ups to
identify contact side, grasp direction, object-to-gripper orientation, gripper-to-table
```

P0b Review the complete demonstration (continued)

```
direction, and recontact after release.  Record these relationships in phase annotations,
and do not mislabel empty-gripper pressing as regrasping.
[Preserve the manipulation sequence]
Usually retain 12--16 frames, never exceeding the supplied limit.  Merge redundant holds
and window boundaries while preserving the initial state, preparation, contact, grasp
verification, arm-role changes, release, and final outcome.  Select important
before/after changes separately, since the host will not add neighboring images
automatically.  Preserve the order of repeated twists and regrasps rather than
describing them as no motion.
[Return evidence-grounded annotations]
The summary covers the full operation and uncertain phases.  The stage field is a short
label, and left/right describe roles.  The reason field gives visual evidence, while
result states only visible outcomes.  Explicitly report when success cannot be verified.
Retain the full sequence's first and last frames and return chronological order.
Historical actions are references, not pending commands.  You have no robot or
file tools.
```

P1 Shared controller contract

```
[SYSTEM INSTRUCTIONS]

[Robot state and coordinate frames]
Control the robot using absolute calibrated TCP targets, not joint-angle commands.
Joint positions and velocities are measured feedback in radians and radians per second.
Do not convert joint values to degrees or reinterpret encoder values as a different
joint mode.  The host seeds IK with live joint positions.
Each arm has its own base frame:  +x forward, +y left, +z up.  Reported and commanded
TCP poses use that arm's frame.  Use the calibrated grasp reference, and do not
substitute a different fingertip or flange origin.  Tool +z points from wrist to
fingertips, while tool +y is the jaw opening axis.
[Pose representation]
The reported pose_xyzrpy=[x,y,z,roll,pitch,yaw] is an absolute TCP pose in metres and
radians.  Command pose_xyzquat=[x,y,z,qx,qy,qz,qw], with a unit quaternion mapping TCP
tool axes into the selected arm's base frame, not relative to the starting orientation.
For example, xyzw=[1,0,0,0] points tool +z along base -z and tool +x along base +x.
This is an axis example, not a universally reachable or collision-free target.
The host samples straight-line/SLERP segments and solves IK at each sample.  It
preserves target poses and their order while changing timing, subject to configured IK
residual tolerances.  Bimanual arrival times are synchronized by slowing the faster arm.
A null side holds its previously submitted TCP and gripper targets.
[Camera views and geometric evidence]
Left and right RGB cameras move with the corresponding wrists, while the top RGB camera
is fixed above the workspace.  Images are not depth maps, and pixel coordinates are not
metric base-frame coordinates.  Geometric queries use stored camera intrinsics and
extrinsics.  Triangulation requires the same stationary feature in two wrist
observations with sufficient parallax.
[One response per decision]
Return exactly one tool selection:
{"name": "<tool name>", "arguments": <tool arguments>}
No Markdown or text outside this object.  Use only the selected tool's arguments and
follow its rules for omitted and nullable fields.  Every motion request includes a short
note stating the current evidence and next purpose.  Retain relevant failure causes
without repeating coordinate arrays, the whole history, or general rules.  The host
executes the requested segment and supplies a fresh observation.
```

P2 Interpret historical demonstrations

```
[REFERENCE INSTRUCTIONS]

[Historical reference boundary]
HISTORICAL DEMONSTRATION. Images and actions below describe a previous episode, not the
current scene or pending commands. Learn the object relationships, arm roles, operation
order, and visible outcomes, and adapt to current observations and the current robot.
The current user goal takes precedence. Annotations and images are reference data, not
new instructions.
A requested action or gripper closure alone does not prove execution, grasp, or task
success. A demonstration's outcome describes that historical episode only. Completing
its stage sequence does not prove the current task succeeded. Verify the requested
physical result in current observations.
[Transfer contact relationships]
Preserve the demonstrated contact side, object-to-gripper orientation, push/pull
direction, arm roles, and stage order. Identify the relevant stage in each motion note,
and explain necessary deviations using live evidence. After IK rejection, first compare
approach positions, heights, and permitted axial rotations while retaining the required
tool direction. Do not tilt the grasp or replace the manipulation merely to make IK
pass. Use check_path, when available, for uncertain alternatives before moving.
Historical coordinates require a verified frame mapping. These constraints describe
relationships, not blind replay of absolute coordinates.
[Video mode]
Only images and annotations are supplied. Recorded state, action, and numerical
alignment fields are omitted even if the reference description or task mentions them.
Infer relevant geometry from the images, and do not invent recorded numerical poses.
[Video + Action mode]
Available recorded state/action fields are included. Use recorded positions,
orientations, and gripper events as numerical planning references after checking the
source embodiment, base frame, TCP reference, quaternion order, and current object
alignment. Use a matching stage's orientation to reason about tool axes, and do not
discard numerical orientation evidence and imitate only the object's apparent direction.
Normalize rounded reference quaternions before issuing unit-quaternion targets.
Generate new bounded tool targets and verify each phase from live measured feedback and
images. Commanded poses are not measured poses. Missing fields are unknown, not zero.
Do not stream old absolute joint commands.
[Return to the live scene]
END HISTORICAL DEMONSTRATION. Use the current task and live observations below.
```

G.2 Context Formats

F1 is shared by all conditions. F2–F5 add the selected reference or interaction context, while the no-demonstration condition omits offline references. Image placeholders denote actual image blocks. The listings expose field structure and content order, while variable-length arrays and tool-dependent fields remain parameterized. Table 9 summarizes the role of each format.

Table 9: **Overview of context and tool formats.**

Format	Function
F1	Present the current task, camera images, measured robot state, and previous tool result.
F2	Supply chronological human or robot demonstration images and available annotations.
F3a	Pair a selected robot-video keyframe with measured TCP and gripper state.
F3b	Encode the intervening action samples, field order, and timing information.
F4	Specify the desired final arrangement through a goal image.
F5	Retain interaction history and incorporate live human cues or task updates.
F6a	Express structured motion, waypoint-sequence, and gripper requests.
F6b	Express geometric checks, image queries, and termination requests.
F7	Return execution feedback and fresh observations for the next decision.

F1 Shared live observation

[TASK T, OBSERVATION o_t, FEEDBACK f_{t-1}]

```
[Observation envelope]
{
  "instruction": "<current task instruction>",
  "images": [<metadata for each available camera>],
  "state": {
    "left": <measured left-arm state>,
    "right": <measured right-arm state>
  },
  "extra": {
    "env_step": <decision index>,
    "interface": <active control interface>,
    "interfaces": <per-arm interfaces, when available>
  },
  "previous_result": <preceding tool result, omitted at the first step>
}
<left wrist image block, when available>
<right wrist image block, when available>
<top image block, when available>
[State fields for each arm]
{
  "joint_pos": <joint position vector, radians>,
  "joint_vel": <joint velocity vector, radians/second>,
  "joint_torque": <measured joint torque vector>,
  "tcp_pose_xyzrpy": [<x>, <y>, <z>, <roll>, <pitch>, <yaw>],
  "tcp_pose_xyzquat": [<x>, <y>, <z>, <qx>, <qy>, <qz>, <qw>],
  "gripper": <measured opening, metres>,
  "gripper_normalized": <measured opening, normalized>,
  "gripper_command_normalized": <commanded opening, normalized>,
  "gripper_vel": <measured opening velocity>,
  "gripper_torque": <measured gripper torque>,
  "gravity_compensation": <available compensation state>
}
The measured and commanded gripper values are separate.  Fresh state and images describe
the current scene.  Reference images and historical action samples do not replace them.
```

F2 Human and robot video references

```
[DEMONSTRATION V]

[Human Video: chronological visual input]
Historical human demonstration, t=<first selected time>s
<first selected image block>

Historical human demonstration, t=<next selected time>s
<next selected image block>

Historical human demonstration, t=<last selected time>s
<last selected image block>

End historical demonstration.
Use the live scene for the current task.
Intermediate selected frames follow the same time--image pattern. Times are relative to
the demonstration video.
[Robot Video: annotated keyframe excerpt]
HISTORICAL DEMONSTRATION.
<header describing the demonstration and its reference conventions>

{
  "t_s": 7.531109,
  "stage": "left body grasp",
  "observation": "Left gripper surrounds the upright bottle body
                  while the right arm remains clear.",
  "result": "Body support is established before the right arm approaches.",
  "image_ids": {"top": "image_1"}
}
<top image block labeled image_1>

<subsequent selected records and their labeled image blocks>
END HISTORICAL DEMONSTRATION.
This excerpt uses the bottle demonstration's recorded annotation. Multi-view references
additionally bind left-wrist and right-wrist image identifiers at each selected time.
Video mode omits numerical state, action, and alignment records.
```

F3a Video + Action: measured keyframe state

[ALIGNED VISUAL AND STATE REFERENCE]

[Record at a selected video time]

```
Video + Action retains the keyframe annotations and image bindings in F2, and adds
measured state and the action interval leading to the next keyframe.  The following
fields come from the initial bottle keyframe.  Joint fields are omitted here to
emphasize TCP and gripper alignment.
{
  "t_s": 0.029,
  "stage": "initial capped bottle",
  "observation": "The capped bottle stands upright near the middle
                  of the table, with both grippers clear.",
  "result": "Initial bottle and cap are together.",
  "image_ids": {"top": "image_0"},
  "state": {
    "recording_t_s": 0,
    "left_eef_x_m": 0.232841,
    "left_eef_y_m": -0.019481,
    "left_eef_z_m": 0.139562,
    "left_eef_qx": 0.089674,
    "left_eef_qy": 0.774834,
    "left_eef_qz": -0.035674,
    "left_eef_qw": 0.624754,
    "left_gripper_measured": 0.998760,
    "left_state_from_top_capture_s": 0.057664,
    "right_eef_x_m": 0.237595,
    "right_eef_y_m": 0.000425,
    "right_eef_z_m": 0.138520,
    "right_eef_qx": -0.057252,
    "right_eef_qy": 0.778141,
    "right_eef_qz": -0.024203,
    "right_eef_qw": 0.625006,
    "right_gripper_measured": 0.999130,
    "right_state_from_top_capture_s": 0.052487
  },
  "action": <recorded interval shown in F3b>
}
<top image block labeled image_0>
Relative video time identifies the image, while state-to-capture offsets retain the
distinction between image acquisition and arm feedback.  They do not imply perfectly
simultaneous measurements.
```

F3b Video + Action: interval encoding and sample rows

[RECORDED ACTIONS A]

[Column order supplied in the reference header]

```
"action_sample_encoding": {
  "columns": [
    ["control_sample_index"], ["video_frame_index"],
    ["t_s"], ["recording_t_s"],
    ["left", "joint_target_rad"], ["left", "eef_target_xyz_xyzw"],
    ["left", "gripper_command"], ["left", "gripper_measured"],
    ["left", "command_from_top_capture_s"],
    ["left", "state_from_top_capture_s"],
    ["right", "joint_target_rad"], ["right", "eef_target_xyz_xyzw"],
    ["right", "gripper_command"], ["right", "gripper_measured"],
    ["right", "command_from_top_capture_s"],
    ["right", "state_from_top_capture_s"]
  ]
}
```

F3b Video + Action: interval encoding and sample rows (continued)

```
[First two retained rows of the initial bottle interval]
"action": {
  "kind": "recorded_bimanual_segment",
  "next_keyframe": 2,
  "sample_period_s": 1,
  "sample_rows": [
    [0, 0, 0.029, 0,
     [0.103, -0.003, 0.011, -0.529, 0.284, 0.124],
     [0.212, -0.024, 0.057, 0.080, 0.854, -0.065, 0.509],
     1, 0.999, 0.050335, 0.057664,
     [-0.143, 0.014, 0.017, -0.500, -0.183, -0.141],
     [0.218, 0.005, 0.060, -0.046, 0.856, 0.006, 0.515],
     1, 0.999, 0.044953, 0.052487],

    [30, 28, 1.029638, 1.001992,
     "=", "=", "=", "=", 0.046323, 0.053257,
     [-0.143, 0.012, 0.017, -0.499, -0.183, -0.143],
     [0.218, 0.005, 0.061, -0.047, 0.855, 0.005, 0.517],
     "=", "=", 0.046360, 0.057344]
  ]
}
[How to read the records]
Each row follows the header's column order.  In joint, pose, and gripper columns, "="
repeats the preceding row's value within the same interval, whose first row is explicit.
Null means missing, never zero.  Joint and pose arrays retain their original order.  The
target pose uses metres and quaternion xyzw.
Keyframe geometry is measured feedback, whereas action geometry represents recorded
commands and is not proof of arrival.  The full interval retains its endpoints, periodic
samples, and gripper changes, rather than reconstructing unsampled motion or contact
forces.  See Appendix C for alignment and sampling.
```

F4 Target-image context

```
[GOAL IMAGE G]

[Input order]
instruction:
  Observe the target image and arrange the blocks on the table
  into the T shape shown in the image. Match the target image
  as closely as possible in color, relative position, and spacing.

<target image block depicting the desired arrangement>

<live observation envelope F1>
<current wrist and overhead image blocks>
The target image specifies the desired outcome, not a trajectory.  It remains distinct
from the live images used to assess the current arrangement.  The instruction for fruit
arrangement changes the matching attributes to fruit identity, relative position,
and spacing.
```

F5 Self-history and online interaction

[HISTORY H_t]

[Self-history: retained interaction sequence]
<task instruction>
<past observation and its camera image blocks>
<selected tool and its arguments>
<tool outcome, measured feedback, and next observation>
<later retained observation--request--feedback turns>
<current live observation F1>
This is a conversation-order schematic, not an additional JSON object. Past views and action outcomes provide context when exploration changes visibility.

[Online human interaction: updated task evidence]
<ongoing interaction task>
<retained human moves or gestures and robot responses>
<current board / human move history, when supplied>
<live images showing the current gesture or board state>
<current measured robot state and preceding tool result>
Gesture evidence is observed in live images, and task updates may also be supplied as operator messages. For turn-taking tasks, the prompt specifies whose turn permits action and when the human hand must have left the workspace. These cues are not prerecorded demonstrations.

G.3 Tool Requests and Feedback

P3 guides tool selection, while P4 governs recovery and termination. F6 illustrates tool requests, and F7 shows the feedback returned for the next decision.

P3 Motion, grasping, and release instructions

[MODEL TOOL-SELECTION POLICY]

[Efficient motion and observation boundaries]
Prefer move_eef_chunk for clear, contact-free paths that need no intermediate observation. Do not split these paths into many decisions or repeatedly check already-established reachability. Stop to observe at contact, gripper changes, occlusion, or tracking anomalies. Efficiency never justifies skipping verification.
For position-only adjustments, keep the last acknowledged target quaternion in the complete pose_xyzquat. Never replace a held orientation with load-induced measured drift. Investigate the drift first. A null arm retains its submitted targets rather than moving to a new viewing position.

[Approach and grasp]
Approach and align, inspect fresh images and state, then call set_gripper separately. Before closure, consider wrist close-ups, the top overview, measured TCP target error, and joint velocity together. A submitted target does not prove arrival.
Target-versus-measured errors are reported after each action. Settled means joint stability, not precise TCP arrival or task success.
Check fingertips, wrists, camera housings, forearms, and table clearance in fresh views. Do not advance contact without verified stabilization and clearance. Keep the acknowledged gripper command during transport, and issue a separate gripper request when a grasp change is needed.

[Release and disengagement]
Before releasing or regrasping, verify that the intended surface or another hand reliably supports the entire object. Observe after release and before withdrawing. A fully-open command does not prove that a wide object detached.
If the object stays fixed relative to the fingers or the destination still appears empty, keep it reliably supported by the intended surface or container while disengaging the fingers. Do not lift a still-trapped object away and report completion.

[Read execution evidence]
The planned TCP points describe the planned path, including its start, not a measured trajectory. Judge execution from execution feedback and fresh state. Joint torque includes gravity effects and is not a direct contact-force measurement. Do not equate a commanded pose, gripper closure, or accepted plan with the intended physical result.

F6a Motion and gripper requests

[OUTPUT $a_t = (u_t, v_t)$]

[Move to one absolute TCP target]

```
{
  "name": "move_to",
  "arguments": {
    "target": {
      "left": null,
      "right": {"pose_xyzquat": [<x>, <y>, <z>, <qx>, <qy>, <qz>, <qw>]}
    },
    "note": "<current evidence and purpose of this motion>"
  }
}
```

The host samples the direct straight-line/SLERP path, applies calibrated frame conversion and IK, and adjusts timing. Gripper targets remain unchanged.

F6a Motion and gripper requests (continued)

```
[Move through an ordered sequence]
{
  "name": "move_eef_chunk",
  "arguments": {
    "poses": [
      {"left": null, "right": {"pose_xyzquat": <first complete pose>}},
      {"left": null, "right": {"pose_xyzquat": <second complete pose>}}
    ],
    "note": "<evidence that this segment needs no intermediate observation>"
  }
}
Each pose is absolute and complete.  The host preserves every waypoint and its order,
adding samples along the same path and changing timing rather than rewriting
the geometry.
[Change the gripper in a separate decision]
{
  "name": "set_gripper",
  "arguments": {
    "positions": {"left": null, "right": <normalized opening>},
    "note": "<fresh evidence for grasping or releasing>"
  }
}
Opening is 0 fully closed and 1 fully open.  Results distinguish requested, submitted,
and measured values.  After detected obstruction, the gripper reference is brought near
measured position to avoid accumulating large error.  Arms retain their last submitted
targets, and an unselected side holds.
```

F6b Geometric checks and terminal requests

```
[ALTERNATIVE TOOL SELECTIONS]

[Check a path without executing it]
{
  "name": "check_path",
  "arguments": {
    "poses": [
      {"left": <complete pose or null>, "right": <complete pose or null>}
    ],
    "note": "<uncertain approach or permitted orientation to test>"
  }
}
Check full-path IK, joint limits, and timing without sending arm or gripper commands.
Acceptance does not certify collision clearance or physical tracking.  Execution replans
from fresh feedback.
[Query image geometry]
{
  "name": "locate_point",
  "arguments": {
    "camera": "<left, right, or top>",
    "pixel_xy": [<image x>, <image y>],
    "reference_step": <earlier observation index>,
    "reference_pixel_xy": [<same feature's earlier x>, <earlier y>],
    "note": "<feature identity and reason for geometric query>"
  }
}
Pixel origin is the image's top-left corner.  The reference fields are optional.  A
single view supplies a camera ray, not depth.  Two wrist views require the same
stationary feature and sufficient parallax.  Use a metric estimate only when
metric_position_available=true.  A triangulation candidate from degenerate geometry is
diagnostic, not a valid motion target.
```

F6b Geometric checks and terminal requests (continued)

```
[Finish or report no safe continuation]
{"name": "done",
 "arguments": {
   "summary": "<direct evidence of the physical goal and any uncertainty>",
   "hindsight": "<lessons from the attempt>"
 }}

{"name": "give_up",
 "arguments": {
   "reason": "<attempted strategies and evidence ruling out safe progress>",
   "hindsight": "<lessons from the attempt>"
 }}
These are alternative responses, not multiple selections in one turn.  Completion
requires current physical evidence.  The tool itself does not assign the evaluation
success label.
```

F7 Feedback returned after a request

```
[NEW INPUT TO THE POLICY]

[Selected feedback fields and their roles]
previous_result
  <tool result or concrete rejection reason>
  result / execution_feedback (when returned)
    <requested-versus-measured endpoint errors>
    <per-arm execution and settling diagnostics>

state
  left / right
    joint_pos, joint_vel
    tcp_pose_xyzrpy, tcp_pose_xyzquat
    gripper_normalized, gripper_command_normalized

<fresh wrist and overhead image blocks>
This is a field-role schematic:  result wrapping and diagnostics depend on the tool.  An
accepted motion plan describes a request, while fresh state and execution feedback
establish whether it was tracked.  Images establish whether the object
interaction occurred.
[Decision context]
Read a rejection before choosing another target.  After execution, compare requested and
measured pose, gripper command and measured opening, and visible object motion.
Settling diagnostics concern joint stability, but they do not by themselves establish
accurate TCP arrival, secure grasping, or task completion.
```

P4 Recovery and termination instructions

```
[NEXT-DECISION POLICY]

[Recover while preserving the task]
One failed action, tool rejection, missed grasp, occluded target, or uncertain result
does not establish impossibility.  Diagnose from fresh images, measured state, and
previous_result.  Try safe alternatives in viewpoint, approach, grasp, orientation, path,
or step size and verify each result.
After IK rejection, preserve required tool-axis directions and compare approach
positions, heights, and permitted axial rotations.  Use check_path for uncertain
alternatives, and do not tilt the gripper merely to make IK pass.  The configured
workspace is not a measured reachability boundary:  an in-range point may still be
unreachable at the required orientation.  IK acceptance does not certify clearance
between camera housings, arms, or objects.
[Verify the physical goal]
Call done only when fresh observations establish the requested physical outcome.  In the
summary, give direct evidence distinguishing completion from an unfinished state, plus
any remaining uncertainty.  A human assigns the final success/failure label, and done
itself is not that label.
For insertion, confirm actual mating and seating, not merely resting on the socket.  Do
not declare completion after withdrawing due to load while the result remains
unconfirmed.  Continue safe verification or recovery while the outcome is uncertain, and
do not stop early merely to hand the decision to a human.
[When no safe continuation remains]
Do not call give_up while reasonable safe strategies remain.  Consider meaningfully
different recoveries.  Use give_up only when evidence shows that the task cannot be
completed or further attempts would violate safety constraints.  State the attempted
strategies and the evidence preventing further progress.
```

G.4 Representative Task Prompts

The examples below retain the task-level wording, with asset references replaced by semantic placeholders. They are supplied together with the applicable shared instructions, context, and tool definitions above, rather than used as standalone one-sentence controller prompts. Prepared task specifications and recorded variants are distinguished where relevant.

Cases 1–2 Human demonstration tasks

```
[HUMAN VIDEO / F2]

[Pick Red Towel:  demonstration-conditioned task]
Watch the historical human demonstration frames and imitate the demonstrated grasping
method to pick up the red towel with the robot's right hand in the current scene.
[Pick Red Towel:  prepared goal-only comparison]
Pick up the red towel with the robot's right hand.
[Pick Red Towel:  recorded procedural variant without video]
Block the towel with one hand, slide the other hand underneath it, and then grip the
towel to lift it.
[Pick Up Notebook:  recorded task]
Watch the reference video at <notebook demonstration> and pick up the notebook.
[Associated context]
The video-conditioned tasks include chronological demonstration image blocks as in F2.
The no-video variants omit those blocks.  The procedural towel variant provides a
grasping method in text.  It is not the same instruction as the prepared
goal-only comparison.
```

Case 3 Unscrew Bottle Cap

[ROBOT VIDEO OR VIDEO + ACTION / F2--F3]

[Recorded goal without demonstration]
In the current scene, unscrew and remove the cap from the bottle on the table, and leave the bottle standing securely on the table.

[Recorded demonstration-conditioned instruction]
Use the top-view robot demonstration keyframes and corresponding robot states, end-effector poses, and action trajectories in <bottle demonstration> as a reference, adapting the actions to the current observations. In the current scene, unscrew and remove the cap from the bottle on the table, and leave the bottle standing securely on the table.

[Mode-specific reference interpretation]
The recorded task wording mentions numerical references in both conditions. In Video, P2 explicitly restricts the supplied reference to images and annotations. In Video + Action, F3 additionally supplies keyframe states and action samples. Numerical references are therefore available only in Video + Action, not inferred from the task wording.

[Shared instruction applied to this reference]
Preserve the demonstrated contact side, object-to-gripper orientation, push/pull direction, arm roles, and stage order. Identify the relevant stage in each motion note and explain necessary deviations using live evidence. A demonstration's outcome describes that historical episode only. Therefore, verify the requested physical result in the current observations.

Case 4 Remove and Reinsert Plug

[ROBOT VIDEO OR VIDEO + ACTION / F2--F3]

[Recorded Video instruction]
Use the chronological three-view robot demonstration keyframes in <plug video demonstration> as a reference. In the current scene, remove the plug from the power strip on the table, then insert the plug back into the same socket, and leave it fully seated after releasing the gripper.

[Recorded Video + Action instruction]
Use the three-view robot demonstration keyframes and corresponding robot states, end-effector poses, and action trajectories in <plug action demonstration> as a reference, adapting the actions to the current observations. In the current scene, remove the plug from the power strip on the table, then insert the plug back into the same socket, and leave it fully seated after releasing the gripper.

[Additional control guidance: arm choice and clearance]
Choose the arm or arms using fresh views while preserving the demonstrated roles. Use one arm when sufficient, or both when support is needed and clearance allows it. Explain the arm choice in the first motion note. Keep an unused side null so that it retains its acknowledged target. Check camera, wrist, table, and inter-arm clearance, since IK feasibility alone does not establish clearance.

[Additional control guidance: orientation and completion]
For a position-only adjustment, hold the acknowledged target quaternion rather than replacing it with measured drift under load. Preserve the required tool direction when comparing approach heights or permitted axial rotations after rejection. When the task requires a table-perpendicular gripper, align tool +z with the table's downward normal, and allow axial rotation only when jaw and contact alignment permit it. Without table calibration, base -z is only a proxy. Verify alignment from the measured TCP pose before advancing contact.
Call set_gripper separately after observing the approach. Before release, verify reliable support, and observe finger disengagement before withdrawal. Do not equate a requested pose or gripper state with the physical result. Confirm actual insertion and seating after release, rather than a plug merely resting on the socket.

Cases 5–6 Goal-image arrangement tasks

[TARGET IMAGE / F4]

[Arrange T Shape]
Observe the target image and arrange the blocks on the table into the T shape shown in the image. Match the target image as closely as possible in color, relative position, and spacing.

[Arrange Fruit]
Observe the reference image and arrange the four fruits on the table to match the layout shown in the image. Match the target as closely as possible in fruit identity, relative position, and spacing.

[Associated input and shared verification]
Each task is paired with its target image and fresh live observations. Shared release instructions require support before opening the gripper and observation before withdrawal. Shared termination instructions require direct evidence of the requested arrangement, rather than merely completing a sequence of placements.

Cases 7–8 Exploration with self-history

[SELF-HISTORY / F5]

[Lemon to Pink Plate: prepared task]
Explore the current scene, locate the pink plate, and place the lemon from the table onto the plate.

[Movable Exploration: task specification]
Use the movable exploration history as context. Continue searching for the Sprite bottle on the table by changing the observation position or safely moving obstacles, then grasping it and placing it into the yellow basket containing the strawberry toy once the target is found.

[Associated input]
Retained observations, tool requests, and outcomes precede the current observation. This context records what earlier viewpoints revealed and what previous interactions changed. The live state remains the basis for selecting the next tool target.

[Shared recovery instruction]
An occluded target or uncertain result does not establish impossibility. Diagnose from fresh images, measured state, and previous tool feedback, then try safe alternatives in viewpoint, approach, grasp, orientation, path, or step size and verify each result.

Cases 9–10 Online human interaction

[LIVE HUMAN CUES AND INTERACTION HISTORY / F5]

[Pointed Fruit Pickup: recorded instruction]
Repeatedly pick up whichever fruit the human points to and place it onto the plate. If no gesture has been made yet, wait for the human's gesture. Stop only when the human makes an OK hand gesture. Otherwise, keep waiting for the human's gesture.

[Tic-Tac-Toe: recorded role and turn-taking instruction]
Play a legal game of tic-tac-toe as Green, moving first against an on-site human who plays Blue. Continuously use [Current Board and Human Move History] as context, update the board after every turn, and move only when it is Green's turn and the human's hand has left the board.

[Tic-Tac-Toe: move priorities]
Prioritize an immediate win, then block an immediate opponent win, then choose a move that creates a double threat or at least preserves a draw.

[Associated input]
These tasks use the live gesture or board, retained interaction history, and fresh robot observations. The fruit instruction defines a continuing interaction with an explicit stop gesture. The game instruction separates legal turn-taking from move selection. A favorable board position does not override the requirement to wait for the human hand to leave.