ASTRA ON CALL: WHEN AND HOW GPT-6 SHOULD INTERVENE IN A VLA POLICY

Anonymous authors
Paper under double-blind review

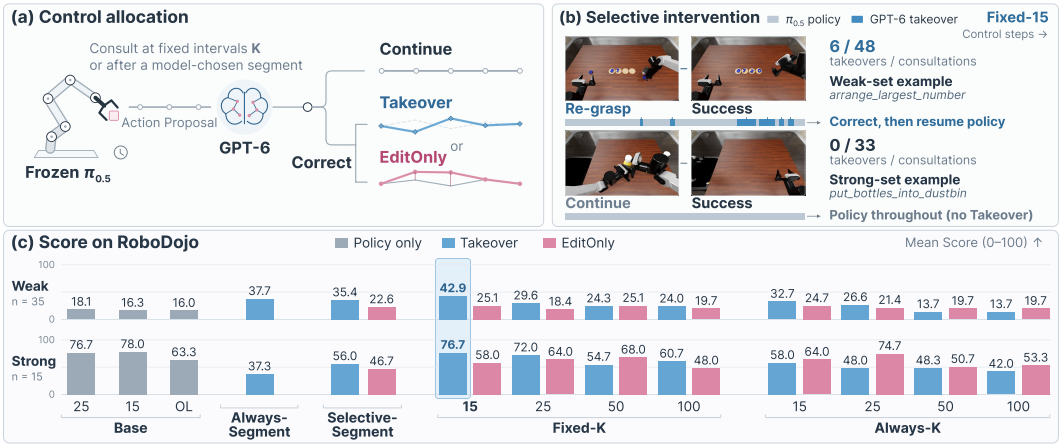


Figure 1: (a) GPT-6 is consulted at fixed or model-chosen intervals to accept the policy's proposal, take over an action segment, or apply bounded waypoint edits. (b) Representative Fixed-15 traces illustrate selective takeover and uninterrupted policy execution. (c) Comparison across controller variants using Score (mean partial credit, 0–100). The numbers above the bars represent the mean Score within each respective set.

ABSTRACT

A new generation of large models, exemplified by GPT-6, demonstrates strong multimodal understanding and embodied reasoning, enabling them to analyze robot behavior using visual scenes, task goals, and execution feedback. These capabilities offer opportunities to improve existing policies through intervention, but how to allocate control remains an open question: which actions should remain with the policy, when should the model intervene, and how should it make corrections? We investigate these questions in *Astra on Call*, an empirical study of how GPT-6 should share control with a frozen VLA policy, $\pi_{0.5}$. Using the Inspect-EEF harness, we compare consultation schedules, optional versus required takeover, and corrections expressed as absolute end-effector targets or bounded waypoint edits. Our results show that selective GPT-6 intervention can improve task outcomes while leaving most control steps to the base policy. On RoboDojo, Fixed-15, which consults GPT-6 every 15 policy steps with optional takeover, raises the weak-set Score from 18.1 to 42.9 while matching the policy's aggregate strong-set Score of 76.7. With GPT-6 inference taking approximately 16 s per call, Fixed-15 reduces wall-clock time per weak-set episode by 27.6% relative to full GPT-6 control. Our project webpage and code are available.

1 INTRODUCTION

A single misaligned grasp or off-target release can derail a manipulation task that is otherwise close to success. Vision–language–action (VLA) policies provide executable actions (Kim et al., 2024; Black et al., 2025), but reliable completion also requires recognizing and correcting errors during execution.

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.

A new generation of large models, exemplified by GPT-6, offers strong multimodal understanding and embodied reasoning: these models can examine visual observations, task goals, and execution feedback to identify mistakes and propose corrections. Their capabilities create an opportunity to improve an existing policy without retraining it. They also raise a question of control allocation: when the robotic policy already supplies useful actions, when should GPT-6 intervene, and how much should it change? We therefore investigate which actions should remain under VLA control, which require local corrections from the assisting model, and when taking over subsequent execution becomes necessary.

Existing test-time intervention methods use world-model rollouts, self-consistency, trajectory scoring, or online correction to select or refine policy proposals (Fan et al., 2026; Dai et al., 2026; Liang et al., 2026). We ask a central question: how should GPT-6 divide control with the policy whose execution it seeks to improve? A correction may recover a failing trajectory, but taking over can also replace actions that would have succeeded. Moreover, inspection frequency determines when the model receives observations and makes these decisions; even when GPT-6 accepts most policy proposals, each inspection still incurs inference overhead. Evaluating assistance therefore requires measuring gains on difficult tasks, performance on tasks the policy already handles, and the computational cost of deciding whether to intervene.

We study this problem in *Astra on Call*, an empirical investigation of GPT-6's test-time intervention in a frozen $\pi_{0.5}$ policy. The policy proposes executable motion, and GPT-6 uses scene observations and the proposed trajectory to decide whether to continue or correct execution. We organize the comparisons around three choices: *when to consult*, through fixed or model-chosen inspection schedules; *whether to intervene*, by allowing continuation or requiring takeover at each consultation; and *how to correct*, through absolute-target segment takeover or bounded edits to the policy's waypoints. We construct a set of controllers based on these choices and systematically compare different allocations of control, using policy-only execution and full GPT-6 control as the two reference endpoints.

The experiments show that GPT-6 can improve a frozen policy through selective intervention, while retaining much of the policy's existing capability and achieving lower episode time than full GPT-6 control. Four findings summarize the results: (1) *Selective intervention improves weak-set outcomes without lowering aggregate strong-set performance.* Fixed-15, which consults GPT-6 every 15 policy steps with optional takeover, raises the weak-set Score from 18.1 to 42.9 while matching the policy's aggregate strong-set Score of 76.7. Most weak-set control steps remain with the policy, and Fixed-15's advantage over full GPT-6 control comes mainly from preserving more of the policy's original successes. On ten single-arm RoboLab tasks, Fixed-15 also raises the success rate from 26.0% for policy-only Base-15 to 70.0%. (2) *Consultation timing and the option to continue the policy both matter.* Optional takeover achieves a higher weak-set Score than mandatory takeover at all four evaluated intervals. With model-chosen consultation timing, Selective-Segment scores below Fixed-15 on both the weak and strong sets. (3) *Bounded waypoint editing lowers the weak-set Score* at three of the four evaluated Fixed-K intervals and leads most controller variants to terminate more episodes with `give_up`. The correction interface therefore affects both task outcomes and the model's decision to keep trying. (4) *Selective intervention reduces overall latency.* On the weak set, Fixed-15 reduces wall-clock time per episode by 27.6% relative to full GPT-6 control. Although the base policy executes most actions, consultation overhead still accounts for approximately 75% of Fixed-15's total time.

Our contributions are threefold:

- **An empirical study of test-time control allocation.** We investigate how GPT-6 should share control with a frozen $\pi_{0.5}$ policy on RoboDojo, examining when and how intervention can improve task outcomes while preserving the policy's existing capabilities.
- **A unified framework for comparing interventions.** Building on the Inspect-EEF harness, we compare controller variants along three dimensions: consultation schedules, whether policy continuation is allowed, and correction through segment takeover or bounded waypoint editing. We evaluate task outcomes, intervention behavior, and computational cost.
- **Findings on intervention benefits and costs.** Our results show that effective assistance requires both correcting policy errors and preserving behavior that already works. The choice of correction method also affects task outcomes and whether the model continues trying. Selective intervention

can reduce episode time relative to full GPT-6 control, but consultation remains a major source of latency even when most actions come from the policy.

2 Related Work

Manipulation policies and action evaluation. VLA models and action-chunking policies learn to map observations to executable robot actions, providing the motor capabilities required for manipulation (Kim et al., 2024; Black et al., 2025; Chi et al., 2025). Additional models can assess whether execution is progressing correctly: SAFE predicts failures from VLA features, while AHA identifies and explains manipulation errors through vision–language reasoning (Gu et al., 2025; Duan et al., 2025). Sampling-based approaches evaluate alternative action chunks with verifiers (Kwok et al., 2025; 2026; Liu et al., 2026c; Wan et al., 2026), and world models predict physical consequences to support visual foresight, candidate ranking, and failure avoidance (Finn & Levine, 2017; Fan et al., 2026; Qi et al., 2026; Cui et al., 2026). These methods motivate examining both the quality of corrective actions and the conditions under which the original policy should continue.

Multimodal guidance and control allocation. A related line of work studies how multimodal models participate in the control loop and influence ongoing policy execution. The GPT 6 Astra as an Embodied Policy report describes an accept-or-correct loop in which GPT-6 either lets a motor policy continue or supplies end-effector commands. Other systems place a VLM agent above a frozen VLA to plan, monitor, and intervene during execution (Chen et al., 2026a; Li et al., 2026b; Hu et al., 2026a), or guide policy behavior through model-generated rewards and language feedback (Liu et al., 2026b; Jeong et al., 2026). Closest to our bounded waypoint edits are inference-time steering methods that adjust a frozen policy's trajectory with bounded, additive corrections instead of replacing it: from human points and sketches (Wang et al., 2025), from low-dimensional inputs applied in the action-token space (Chan & Kao, 2026), and from language corrections converted into additive action-magnitude adjustments (Kwon et al., 2026). Related studies examine when assistance should be requested and when control should be transferred, using calibration, intervention scheduling, or handoff mechanisms (Yuan et al., 2026; Tang & Zhi, 2026; Song et al., 2026). We construct a suite of controllers integrating GPT-6 with a frozen $\pi_{0.5}$ policy to systematically study control allocation between them by varying call schedules, takeover rules, and available correction tools. Our focus is how these choices affect task success, how much execution remains with the base policy, and how many model calls the assistance requires. Appendix B provides further discussion.

3 Experimental Setup

3.1 Test-Time Intervention on the Inspect-EEF Harness

We study a shared control loop in which a frozen VLA proposes executable actions and GPT-6 selects a tool to continue, correct, or terminate execution. The Fixed-K controller family consults GPT-6 periodically and leaves the intervention decision to the model. The remaining conditions modify its consultation schedule, available choices, or correction tools. This organization lets us study control allocation through a small number of design choices; the complete definitions appear in Table 1.

At each hybrid-controller consultation, the frozen policy proposes a 50-step chunk of 14-dimensional joint actions, and GPT-6 decides how execution should proceed. Let o_t, x_t, and ℓ denote the current images, robot state, and task instruction, respectively. The harness summarizes the policy proposal U_t as per-arm end-effector waypoints W_t at steps 10, 20, 30, 40, and 50, together with gripper openings. The proposal and intervention decision are written as

$$U_t = \pi_\theta(o_t, x_t, \ell), \quad (d_t, \eta_t) = G_{\text{Astra}}(o_t, x_t, \ell, h_t, W_t; \mathcal{D}_c), \tag{1}$$

where π_θ remains frozen, h_t contains execution history and feedback, and G_{Astra} denotes GPT-6's prompted tool-selection process. The model returns a tool d_t and its arguments η_t, with the available tool set $\mathcal{D}_c$ determined by the controller:

$$\begin{aligned} \mathcal{D}_{\text{takeover}} &= \{\texttt{continue_policy}, \texttt{move_eef}, \texttt{give_up}\}, \\ \mathcal{D}_{\text{edit}} &= \{\texttt{continue_policy}, \texttt{edit_proposal}, \texttt{give_up}\}. \end{aligned} \tag{2}$$

The Always-K variants remove `continue_policy`, requiring selection of the available correction tool at every decision. An editing call may nevertheless leave the proposal unchanged.

Table 1: Controller families. The Fixed-K and Always-K families, with and without +EditOnly, use $K \in \{15, 25, 50, 100\}$. C = `continue_policy`, M = `move_eef`, E = `edit_proposal`. All GPT-6 conditions also offer `give_up`.

Controller family	Control allocation	GPT-6 consultation	Tools
Policy-only baselines			
Base-25 / Base-15	Policy only	None; policy replans every 25 / 15 steps	—
Base-OL	Policy only, open loop	None; each 50-step policy chunk executes open loop	—
Fixed-interval consultation			
Fixed-K	Policy + optional takeover	Every K policy steps	C, M
Always-K	Policy + forced takeover	As Fixed-K; policy resumes after takeover	M
Random-15	Policy + forced takeover	Every 15 policy steps with probability $p \approx$ Fixed-15's takeover selection rate	M
Fixed-K+EditOnly	Policy + optional edit	As Fixed-K	C, E
Always-K+EditOnly	Policy + required edit call	As Always-K; policy resumes after the edited segment	E
Segment-level consultation			
Selective-Segment	Policy + optional takeover, model-timed	After each segment; model selects accepted policy length $k \leq 50$	C, M
Selective-Segment+EditOnly	Policy + optional edit, model-timed	As Selective-Segment	C, E
Always-Segment	GPT-6 only	After each GPT-6 action segment; no policy execution	M

Periodic Inspection and Optional Takeover Fixed-K consults GPT-6 initially and after every K executed policy steps, with $K \in \{15, 25, 50, 100\}$. The model chooses `continue_policy` to retain policy execution, `move_eef` to specify absolute grasp-point targets for a replacement segment, or `give_up` to terminate. The takeover executor converts targets into joint commands, executes the segment open loop, and returns fresh observations and feedback on the executed endpoint; the policy then resumes. Thus, K counts policy steps between calls, excluding inserted takeover steps. A *consultation* is a model call, an *intervention* changes executable actions. We use *takeover* specifically for `move_eef`. Accepting a proposal still incurs a consultation but adds no corrective action. The policy's nominal replanning interval is 25 steps, whereas K determines when GPT-6 inspects execution. Inspection obtains a fresh proposal, so comparisons across K evaluate the resulting control protocols, including their interaction with replanning. Policy-only Base-15 and Base-OL runs examine policy-only replanning at 15 and 50 steps (Appendix F.1).

Bounded Waypoint Editing The +EditOnly conditions replace absolute-target takeover (`move_eef`) with bounded modifications of the policy's proposed waypoints (`edit_proposal`). For each arm, let $\mathbf{p}_{t,i}$ denote the world-frame position of displayed waypoint i. GPT-6 specifies a translation offset $\Delta\mathbf{p}_{t,i}$ and a world-frame rotation offset represented by the rotation vector $\Delta\boldsymbol{\omega}_{t,i}$, subject to

$$\widetilde{\mathbf{p}}_{t,i} = \mathbf{p}_{t,i} + \Delta\mathbf{p}_{t,i}, \quad \|\Delta\mathbf{p}_{t,i}\|_\infty \leq \epsilon_p, \quad \|\Delta\boldsymbol{\omega}_{t,i}\|_2 \leq \epsilon_R, \tag{3}$$

where $\epsilon_p = 0.05\,\mathrm{m}$ and $\epsilon_R = \pi/9$ radians. The translation bound applies independently to each coordinate, while the rotation bound limits the rotation-vector magnitude. These constraints keep the corrected waypoint poses close to the policy proposal. GPT-6 may also modify the gripper command. Unmodified arms replay the original policy joint commands, while moved waypoints are reached through closed-loop inverse kinematics from the measured state. Execution returns to the policy proposal at the next unchanged waypoint. An edit without translation, rotation, or gripper changes is equivalent to continuation. The edited segment follows the continuation duration of its controller; waypoint indexing, validation, and execution details are provided in Appendix E. Comparing a controller with its +EditOnly counterpart holds the consultation schedule fixed and exchanges the correction mechanism: absolute-target segment replacement or bounded modification of the policy proposal. The comparison covers representation and execution together: the edit branch follows the proposal using closed-loop inverse kinematics, whereas `move_eef` executes an open-loop chunk.

Comparisons and Their Roles Fixed-K versus Always-K tests the value of allowing continuation under a shared consultation rule. Fixed versus model-chosen schedules examine consultation timing, while each controller and its +EditOnly counterpart compare correction mechanisms. Base-25 and Always-Segment provide policy-only and GPT-6-only reference endpoints. Full definitions appear in Table 1 and Appendix C.

3.2 Benchmark and Evaluation Sets

We evaluate on RoboDojo's ARX X5 embodiment at a simulated control frequency of 25 Hz using its official frozen $\pi_{0.5}$ checkpoint (Chen et al., 2026b; Black et al., 2025). All GPT-6 conditions use persistent Codex threads with ***medium reasoning effort***, and the simulator pauses during inference. Each episode follows the benchmark's task-specific step limit and permits up to 100 consultations (120 for `imitate_sorting_sequence`). Appendix C provides checkpoint and budget details. We select ten tasks based on the official $\pi_{0.5}$ success rates, emphasizing tasks with low success rates while covering all five RoboDojo capability dimensions (Chen et al., 2026b). We then divide them into weak and strong sets according to Base-25's success on the evaluated layouts. The weak set comprises seven tasks with at most one success in five episodes per task (35 episodes total), while the strong set comprises three tasks with at least three successes in five episodes per task (15 episodes total). These sets assess failure recovery and preservation of the policy's existing capabilities, respectively. We also evaluate Base-25, Fixed-15, Fixed-15+EditOnly, and Always-Segment on six additional RoboDojo tasks outside these sets (30 episodes), and report the results separately.

3.3 Outcomes, Behavior, and Cost

We report two outcome measures over n evaluated episodes: the benchmark's per-episode partial-credit *Score*, and success rate (*SR*), each with a descriptive standard error ($s/\sqrt{n}$ and $\sqrt{\hat{p}(1-\hat{p})/n}$, where s is the sample standard deviation of episode scores and $\hat{p}$ the observed success rate). Score awards intermediate credit for completed sub-goals according to each task's rubric (for example, 15 and 40 points for one and two correctly sorted categories in `classify_objects`, and 100 for success), distinguishing controllers by how far they progress in episodes that do not succeed. Behavior is characterized by consultations per episode; takeover selection and execution rates, the fractions of decisions other than `give_up` that call `move_eef` and whose call the planner executes (Appendix D); `give_up` counts; and the fraction of executed control steps supplied by the policy. Cost includes model calls, GPT-6 time per episode, and wall-clock time per episode. GPT-6 time per episode counts 16 s of model latency per GPT-6 call. Wall-clock time adds 0.3 s of simulator time per control step and 3 s of local processing per consultation (Appendices D and G).

4 Empirical Results

4.1 Can GPT-6 Improve a Policy While Preserving Its Strengths?

Selective intervention improves the weak-set Score without reducing aggregate performance on the strong set. Table 2 compares policy-only Base-25, full GPT-6 control with Always-Segment, and Fixed-15. Fixed-15 is the highest-scoring setting in the evaluated Fixed-K sweep on the same set. On the weak set, Fixed-15 raises the Score from 18.1 (SR: 5.7%) for Base-25 to 42.9 (SR: 22.9%), while Always-Segment reaches 37.7 (SR: 17.1%). On the strong set, Fixed-15 keeps the policy's aggregate Score (76.7 for both; SR: 73.3% for both), whereas Always-Segment falls to 37.3 (SR: 20.0%).

Table 2: Score (mean partial credit, 0–100) and success rate (%) under three allocations of control. Weak, strong, and combined results use 35, 15, and 50 recorded episodes, respectively.

		Score			Success (%)		
Controller	Allocation of control	Weak	Strong	Combined	Weak	Strong	Combined
Base-25	Policy only	18.1 ± 5.1	$\mathbf{76.7} \pm 10.5$	35.7 ± 6.1	5.7 ± 3.9	$\mathbf{73.3} \pm 11.4$	$\underline{26.0} \pm 6.2$
Always-Segment	GPT-6 only	$\underline{37.7} \pm 6.0$	$\underline{37.3} \pm 8.8$	$\underline{37.6} \pm 4.9$	$\underline{17.1} \pm 6.4$	$\underline{20.0} \pm 10.3$	18.0 ± 5.4
Fixed-15	Policy + optional takeover	$\mathbf{42.9} \pm 6.4$	$\mathbf{76.7} \pm 10.5$	$\mathbf{53.0} \pm 5.8$	$\mathbf{22.9} \pm 7.1$	$\mathbf{73.3} \pm 11.4$	$\mathbf{38.0} \pm 6.9$

Recovering failures and preserving successes. Fixed-15's advantage over full GPT-6 control comes mainly from preserving the base policy's original successes. Across matched episodes from both evaluation sets (Figure 2), Fixed-15 and Always-Segment recover similar numbers of Base-25 failures (8 and 7 of 37, respectively), while preserving 11 and 2 of its 13 successes. Fixed-15's unchanged aggregate success rate on the strong set reflects two recovered failures offsetting two lost successes. Together, these results highlight that effective assistance depends on recognizing when to

retain the policy's behavior as well as when to correct it. Appendix D details the matching protocol and metric definitions, and Appendix F.1 reports per-task outcomes.

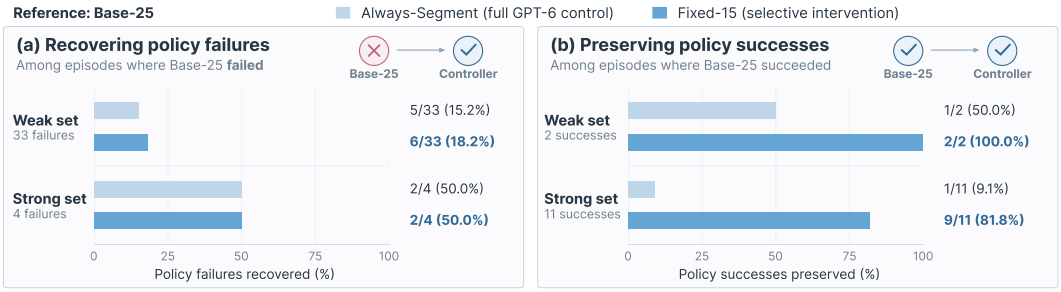


Figure 2: Failure recovery and success preservation under full GPT-6 control (Always-Segment) and selective intervention (Fixed-15), relative to policy-only control (Base-25). (a) Recovery: the fraction of Base-25 failures that succeed under each evaluated controller. (b) Preservation: the fraction of Base-25 successes that remain successful under each evaluated controller. Bars show these conditional rates; labels report counts, denominators, and percentages.

How much execution remains with the policy? Fixed-15 leaves approximately 66% of executed weak-set control steps to the base policy and selects a takeover at 25% of its non-`give_up` decisions (19% executed). On the strong set, it executes only 14 takeovers over 526 such decisions (2.7%): in 10 of the 15 episodes it never takes over and the policy alone succeeds in 9 of them, while the 5 episodes with at least one takeover yield 2 successes. The controller therefore exhibits different intervention behavior across the two sets, with substantially less takeover on tasks where the base policy is more successful. Figure 3 illustrates a weak-set recovery and a strong-set success without takeover.



Figure 3: Selective takeover and policy continuation under Fixed-15. Frames are tagged with their control step and, at consultations, GPT-6's decision; in (a) the highlighted region is enlarged under each frame. (a) Weak-set task `arrange_largest_number`: the policy's grasp closes beside the 2 (step 285); GPT-6 takes over for 8 steps to reopen and shift the hand, and the policy completes the re-grasp on its own (steps 323–383). An 18-step takeover at step 398 re-orients the 2 in transit; once all digits are placed the 2 still lies tilted (step 656), and four takeovers of 167 steps in total turn it upright (step 910). The policy alone (Base-25) leaves this layout unfinished at the 1050-step limit. (b) Strong-set task `put_bottles_into_dustbin`: GPT-6 accepts every proposal and the policy completes the task; Base-25 also succeeds. More trajectories appear in Appendices F.6 and F.7.

4.2 When Should GPT-6 Be Consulted, and Should It Always Take Over?

The option to allow the policy to continue execution matters at a fixed consultation interval. Fixed-K and Always-K use the same consultation rule but differ in whether GPT-6 can accept the policy proposal. Across all four shared intervals, optional takeover achieves a higher weak-set Score and success than mandatory takeover (Table 3): the Score margins are 10.2, 3.0, 10.6, and 10.3 points at $K = 15$, 25, 50, and 100. At $K = 25$, for example, allowing continuation raises weak-set success from 11.4% to 17.1% and strong-set success from 40.0% to 66.7%. The larger difference on the strong set highlights the value of leaving useful policy behavior intact. These results support a simple design choice: each consultation should give GPT-6 an opportunity to correct the policy, while retaining the option to let it proceed.

Choosing when to intervene matters for failure recovery. Random-15 follows Fixed-15's schedule but takes over at random with $p = 0.25$, Fixed-15's weak-set takeover selection rate (Appendix A). It selects as many takeovers as Fixed-15 (434 versus 431) and executes more (384 versus 327), yet scores only 20.1 on the weak set, close to Base-25's 18.1 and well below Fixed-15's 42.9. Of Base-25's 33 weak-set failures, Random-15 recovers one and Fixed-15 recovers six, supporting the value of selecting when to take over (Appendix F.2).

Table 3: Inspection interval and takeover rule on the *weak set*. Score and success (%) are over 35 episodes. Calls and takeover rates describe Fixed-K; Always-K calls `move_eef` at each consultation that does not end the episode. Full behavioral and timing measurements are in Table 4.

	Score		Success (%)		Fixed-K behavior		
K	Fixed-K	Always-K	Fixed-K	Always-K	Calls/episode	Selection (%)	Execution (%)
15	**42.9**	**32.7**	**22.9**	**20.0**	50.0	25	19
25	<u>29.6</u>	<u>26.6</u>	<u>17.1</u>	<u>11.4</u>	31.9	34	25
50	24.3	13.7	14.3	2.9	18.0	39	30
100	24.0	13.7	14.3	5.7	10.0	41	30

Periodic consultation can improve outcomes while reducing model use. On the weak set, Fixed-15 reaches a higher Score than Always-Segment (42.9 versus 37.7) while using one third fewer model calls per episode (50.0 versus 75.0). Its cumulative GPT-6 time correspondingly decreases from 1,201 s to 800 s (Appendix G). This joint improvement shows that giving GPT-6 control over every action segment is unnecessary for obtaining better task outcomes. Longer consultation intervals reduce model use further and lower the Score monotonically (29.6, 24.3, and 24.0 at $K = 25$, 50, and 100): Fixed-50 requires only 18.0 calls per episode (Score 24.3). Within the evaluated settings, Fixed-15 therefore offers an effective division of work: the policy supplies most of the motion, while periodic GPT-6 decisions improve completion with less inference than full GPT-6 control.

Model-chosen scheduling uses slightly fewer calls on the weak set but performs worse than Fixed-15 on both evaluation sets. Selective-Segment reaches a weak-set Score of 35.4 (SR: 20.0%), compared with 42.9 (SR: 22.9%) for Fixed-15, while reducing consultations per episode from 50.0 to 44.1. The performance gap is larger on the strong set, where Selective-Segment reaches a Score of 56.0 (SR: 46.7%) versus 76.7 (SR: 73.3%) for Fixed-15. These results highlight the importance of evaluating consultation schedules by both their gains on difficult tasks and their ability to maintain strong-set performance, alongside model-call cost.

4.3 Local Editing versus Segment Takeover

We compare two ways for GPT-6 to correct the policy: bounded editing of the waypoints proposed by the policy, and taking over execution with new absolute targets (Figure 4). Each controller and its +EditOnly variant share the same consultation schedule but differ in both correction tools and execution: editing uses closed-loop inverse kinematics, whereas takeover executes an open-loop action chunk (Section 3.1).

Replacing takeover with bounded editing lowers the weak-set Score and success at three of the four evaluated Fixed-K intervals. In the paired comparisons in Figure 4, editing reduces the weak-set Score by 17.8, 11.2, and 4.3 points at $K = 15$, 25, and 100, respectively. Success rates also fall at these intervals, from 22.9% to 14.3%, 17.1% to 8.6%, and 14.3% to 11.4%, alongside

declines in both strong-set Score and success. The sole exception is $K = 50$, where editing raises the weak-set Score from 24.3 to 25.1 and the strong-set Score from 54.7 to 68.0, although both remain lower than those of Fixed-15.

Under bounded editing, GPT-6 tends to judge further attempts unproductive and stop earlier, even when substantial execution budget remains. On the weak set, eight of the nine EditOnly variants terminate more episodes with `give_up` than their takeover counterparts. Aggregating results across controllers, EditOnly give-ups leave a median 33% of the step budget unused, compared with 6% under takeover. GPT-6's stated reasons help explain this difference: 80% of EditOnly give-ups primarily attribute failure to bounded edits being unable to redirect the policy or reach the target, whereas 64% of takeover give-ups cite insufficient remaining steps. The editing rationales describe proposals that fail to approach the target, target choices that cannot be corrected within the allowed offsets, and corrections whose effects are lost when execution rejoins the unedited proposal. Where neither controller gives up, both approaches achieved the same number of successes, so the Score gap between them arises in the episodes that editing abandons (Appendix F.4).

Semantic errors remain difficult for both correction methods. Manual annotation identifies 26 of Base-25's 33 weak-set failures as semantic errors, including wrong objects, incorrect action order, skipped sub-goals, and false completion, with the remaining seven classified as geometric errors (Appendix F.8). Pooled over the four Fixed-K schedules, takeover and editing each recover 17.9% of geometric failures, while their recovery rates for semantic failures are 11.5% and 8.7%, respectively (Appendix F.4). Although takeover recovers slightly more semantic failures, these errors account for most policy failures and remain challenging under both methods.

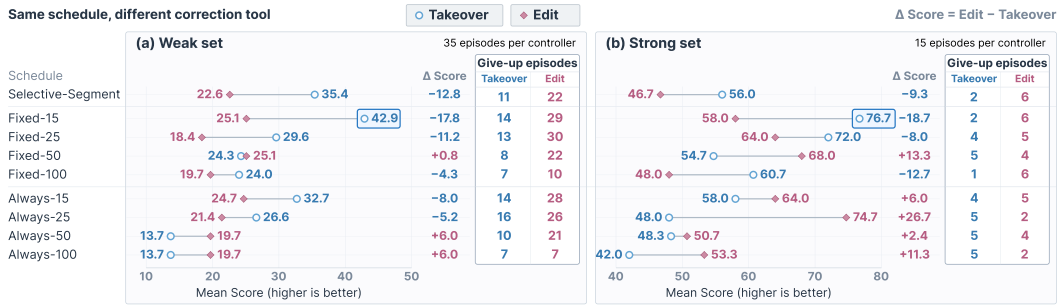


Figure 4: Takeover versus bounded editing. Connected points compare mean Score for Takeover (blue circles) and Edit (pink diamonds) under the same schedule on (a) the weak set ($N = 35$) and (b) the strong set ($N = 15$). ΔScore denotes Edit minus Takeover. The 'Give-up episodes' columns report the number of episodes terminated by `give_up`.

4.4 Model Latency and Episode Time

Consultation remains costly even when most actions come from the policy. The simulator pauses while GPT-6 responds, so each consultation adds approximately 16 s, including decisions to continue the policy. Wall-clock time averages 1,269 s per weak-set episode for Fixed-15, 27.6% less than Always-Segment's 1,753 s (Table 4). Model inference and local processing together account for approximately 75% of Fixed-15's total, despite most control steps being executed by the policy. Reducing takeover frequency alone therefore does not remove the main source of delay; consultation scheduling must also balance the expected benefit of inspection against its waiting cost.

4.5 Do the Findings Hold on Additional Tasks and a Second Benchmark?

On **six additional RoboDojo tasks** with five layouts per task, Fixed-15 raises the Score from 37.3 for Base-25 to 51.5 and the success rate from 33.3% to 50.0%. It also achieves higher success than Always-Segment (46.7%) while using fewer than half as many consultations per episode (21.7 versus 46.6). These results extend the benefits of selective takeover to additional tasks, whereas Fixed-15+EditOnly remains less effective, with a Score of 33.5.

Table 4: Consultation behavior and episode time. Calls are means per weak-set episode (maximum in parentheses). Selection is the fraction of non-`give_up` decisions that call `move_eef`, Execution the fraction the planner executed; the gap is refused, unreachable targets.

	Consultation				Score		Success (%)		Time per episode (s)	
Controller	Calls (max)	Selection (%)	Execution (%)	Give-ups	Weak	Strong	Weak	Strong	GPT-6	Wall-clock
Base-25	0	—	—	—	18.1	**76.7**	5.7	**73.3**	0	≈300
Selective-Segment	44.1 (86)	66	58	11	35.4	56.0	20.0	46.7	705	1,159
Fixed-15	50.0 (84)	25	19	14	**42.9**	**76.7**	**22.9**	**73.3**	800	1,269
Fixed-25	31.9 (55)	34	25	13	29.6	72.0	17.1	66.7	510	927
Fixed-50	18.0 (29)	39	30	8	24.3	54.7	14.3	46.7	288	673
Fixed-100	10.0 (17)	41	30	7	24.0	60.7	14.3	53.3	160	507
Always-15	33.7 (50)	100	90	14	32.7	58.0	20.0	46.7	539	959
Always-25	26.3 (39)	100	90	16	26.6	48.0	11.4	40.0	421	831
Always-50	16.9 (24)	100	87	10	13.7	48.3	2.9	40.0	270	650
Always-100	9.8 (14)	100	88	7	13.7	42.0	5.7	26.7	156	513
Always-Segment	75 (100)	100	92	15	37.7	37.3	17.1	20.0	1,201	1,753

Selective takeover also improves performance on RoboLab. We evaluate ten tasks with five episodes each on this single-arm benchmark (Yang et al., 2026), using a Franka Panda and the public DROID joint-position $\pi_{0.5}$ checkpoint (Base-15, which replans every 15 control steps). Fixed-15 raises the Score from 36.5 to 80.8 and the success rate from 26.0% to 70.0%, compared with 55.1 and 44.0% for Always-Segment. Bounded editing performs better here than on the additional RoboDojo tasks: Fixed-15+EditOnly improves over the base policy and approaches Fixed-15, reaching a Score of 76.5. Fixed-15 achieves its gains with GPT-6 executing 8.5% of control steps and 54 consultations per episode, about one-third of Always-Segment's 151. Together, these evaluations support the benefits of selective takeover across tasks and benchmarks. Details are in Appendices C and F.5.

5 Conclusion

In *Astra on Call*, we studied when and how GPT-6 should intervene in a frozen VLA policy. On RoboDojo, Fixed-15 improves weak-set outcomes without reducing aggregate strong-set performance, while leaving most control steps to the base policy. Its advantage over full GPT-6 control comes primarily from preserving more existing policy successes while recovering a similar number of failures. Effective assistance therefore requires recognizing when the policy should continue, and should be evaluated on both difficult tasks and tasks the policy already handles well. A random gate with the same clock and takeover selection rate stays at the policy's level on the weak set, so Fixed-15's weak-set gain rests on GPT-6's choice of when to intervene. Bounded waypoint editing underperforms segment takeover at three of the four evaluated fixed intervals and is associated with earlier give-ups, suggesting that the correction method influences both task execution and the model's decision to keep trying. Keeping corrections close to the policy proposal thus does not by itself ensure better outcomes. Finally, selective intervention can reduce overall execution time relative to full GPT-6 control, but consultation remains costly even when GPT-6 accepts the policy's proposal. Further efficiency gains therefore require reducing unnecessary consultations and shortening model response times, alongside choosing when and how to correct the policy.

6 Limitations and Future Work

Our findings on when and how GPT-6 should intervene are based on a frozen VLA policy evaluated in simulated manipulation tasks. Future work could learn adaptive consultation mechanisms that determine when to invoke GPT-6 based on task progress and the expected benefits of intervention, while dynamically choosing among policy continuation, local editing, and segment takeover according to the execution state. Successful interventions could also provide training supervision for policies with more efficient inference, turning temporary assistance at test time into lasting improvements in robot capabilities. More broadly, GPT-6 could support task planning, execution monitoring, and skill learning, helping robots perform longer and more diverse tasks.

AI USE STATEMENT

GPT-6 Astra is the intervention model evaluated in this study. Generative AI tools were also used for language polishing. The authors reviewed all AI-assisted content and take full responsibility for the final manuscript, experiments, reported results, and their interpretation.

ETHICS STATEMENT

The experiments are conducted in RoboDojo and RoboLab simulations and involve no human subjects, personal data, or physical-robot deployment. The simulated intervention interface is constrained by action bounds and execution checks. Physical deployment would require validating the controller and its timing under the intended hardware and operating conditions.

REPRODUCIBILITY STATEMENT

Appendices A–E describe controller definitions, evaluation conditions, metrics, and the `edit_proposal` tool. Appendices F and G report detailed outcomes and timing; Appendix D explains their statistical interpretation. Our project webpage and code are available.

REFERENCES

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025. URL `https://proceedings.mlr.press/v305/black25a.html`.

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

Jason Chan and Jonathan C Kao. Steering autoregressive vision-language-action policies via action token intervention. *arXiv preprint arXiv:2606.15021*, 2026.

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

Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, Weijie Wan, Baijun Chen, Songling Liu, Haowen Yan, et al. Robodojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026b.

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

Hanchen Cui, Sergio Arnaud, Arjun Majumdar, Daniel Dugas, Elie Aljalbout, Karthik Desingh, Krishna Murthy Jatavallabhula, and Franziska Meier. Dreamsteer: Latent world models can steer vla policies during deployment without any finetuning. *arXiv preprint arXiv:2607.02865*, 2026.

Yinwei Dai, Zhuofu Chen, Lijie Yang, and Ravi Netravali. Geometry guided self-consistency for physical ai. *arXiv preprint arXiv:2605.08638*, 2026.

Jadelynn Dao, Milan Ganai, Yasmina Abukhadra, Ajay Sridhar, Mozhgan Nasr Azadani, Katie Luo, Clark Barrett, Jiajun Wu, Chelsea Finn, and Marco Pavone. Direct: When and where should you allocate test-time compute in embodied planners? *arXiv preprint arXiv:2606.12402*, 2026.

Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 45493–45517, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/70a06501001e1820fd1eb9ee821302d2-Paper-Conference.pdf.

Xianzhe Fan, Yuxiang Lu, Shenyuan Gao, Xiaoyang Wu, Ruihua Han, Manling Li, and Hengshuang Zhao. Dreamavoid: Critical-phase test-time dreaming to avoid failures in vla policies. *arXiv preprint arXiv:2605.11750*, 2026.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2786–2793. IEEE, 2017.

Shenyuan Gao, William Liang, Kaiyuan Zheng, Ayaan Malik, Seonghyeon Ye, Sihyun Yu, Wei-Cheng Tseng, Yuzhu Dong, Kaichun Mo, Chen-Hsuan Lin, Qianli Ma, Seungjun Nah, Loic Magne, Jiannan Xiang, Yuqi Xie, Ruijie Zheng, Dantong Niu, You Liang Tan, K. R. Zentner, George Kurian, Suneel Indupuru, Pooya Jannaty, Jinwei Gu, Jun Zhang, Jitendra Malik, Pieter Abbeel, Ming-Yu Liu, Yuke Zhu, Joel Jang, and Linxi "Jim" Fan. Dreamdojo: A generalist robot world model from large-scale human videos, 2026. URL https://arxiv.org/abs/2602.06949.

Qiao Gu, Yuanliang Ju, Shengxiang Sun, Igor Gilitschenski, Haruki Nishimura, Masha Itkina, and Florian Shkurti. Safe: Multitask failure detection for vision-language-action models. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 40041–40076. Curran Associates, Inc., 2025. doi: 10.52202/085713-1337. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/392d0d05e2f514063e6ce6f8b370834c-Paper-Conference.pdf.

Yanjiang Guo, Yen-Jen Wang, Lihan Zha, and Jianyu Chen. Doremi: Grounding language model by detecting and recovering from plan-execution misalignment. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 12124–12131. IEEE, 2024.

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

Zheyuan Hu, Robyn Wu, Naveen Enock, Jasmine Li, Riya Kadakia, Zackory Erickson, and Aviral Kumar. Rac: Robot learning for long-horizon tasks by scaling recovery and correction. *IEEE Transactions on Robotics*, 2026b.

Hyun Joe Jeong, Gokul Swamy, and Andrea Bajcsy. Learning what to say to your vla: Mostly harmless vision language action model steering. *arXiv preprint arXiv:2606.12299*, 2026.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Jacky Kwok, Christopher Agia, Rohan Sinha, Matt Foutter, Shulu Li, Ion Stoica, Azalia Mirhoseini, and Marco Pavone. Robomonkey: Scaling test-time sampling and verification for vision-language-action models. In *9th Annual Conference on Robot Learning*, 2025. URL https://openreview.net/forum?id=QPTZfaxATs.

Jacky Kwok, Xilun Zhang, Mengdi Xu, Yuejiang Liu, Azalia Mirhoseini, Chelsea Finn, and Marco Pavone. Scaling verification can be more effective than scaling policy learning for vision-language-action alignment. In *European Conference on Computer Vision*, pp. 37–55. Springer, 2026.

Owen Kwon, Pablo Ortega-Kral, Arthur Bucker, and Jean Oh. Training-free action correction for vla model failures via language feedback. *arXiv preprint arXiv:2608.29967*, 2026.

Andrew Zou Li, Gokul Swamy, Yonatan Bisk, and Andrea Bajcsy. Elastic: Efficiently learning to adaptively scale test-time compute for generative control policies. *arXiv preprint arXiv:2606.31132*, 2026a.

Zhuoran Li, Zhiyang Li, Kaijun Zhou, and Jinyu Gu. Roboharness: A memory-augmented policy harness for vision-language-action model robustness via in-context adaptation, 2026b. URL https://arxiv.org/abs/2603.24060.

Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S Huang, Luke Zettlemoyer, Dieter Fox, et al. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. *arXiv preprint arXiv:2603.02115*, 2026.

Shuaijun Liu, Qifu Wen, Shuyang Hao, Qi Luo, Chenglong Zhang, Feiyang You, Chengyu Wu, and Ningxin Su. Cowam: Coordination contracts for selective policy intervention with wams. *arXiv preprint arXiv:2608.02578*, 2026a.

Shuo Liu, Ishneet Sukhvinder Singh, Yiqing Xu, Jiafei Duan, and Ranjay Krishna. Vls: Steering pretrained robot policies via vision-language models. *arXiv preprint arXiv:2602.03973*, 2026b.

Yushan Liu, Peibo Sun, Xintao Chao, Zhenyang Yang, Yifan Xie, Lingfeng Zhang, Shoujie Li, Chenyu Tang, Fang Chen, Xiao-Ping Zhang, et al. Checkvla: Execution-time verification with action-conditioned world model for long-horizon mobile manipulation. *arXiv preprint arXiv:2607.26789*, 2026c.

Zeyi Liu, Arpit Bahety, and Shuran Song. Reflect: Summarizing robot experiences for failure explanation and correction. *arXiv preprint arXiv:2306.15724*, 2023.

Chenyang Ma, Kai Lu, Guangyu Yang, Jiuming Liu, Shitong Xu, Bill Byrne, Ioannis Havoutis, Niki Trigoni, and Andrew Markham. Cyclevla: Proactive self-correcting vision-language-action models via subtask backtracking and minimum bayes risk decoding. *arXiv preprint arXiv:2601.02295*, 2026.

Dhia Naouali, Minghan Wu, Claudia Wong, Abhinav Puthran, and Omar G Younis. Vlcp: Vision language control policy closed-loop code replanning for robot manipulation. *arXiv preprint arXiv:2608.16978*, 2026.

Han Qi, Haocheng Yin, Aris Zhu, Yilun Du, and Heng Yang. Inference-time enhancement of generative robot policies via predictive world modeling. *IEEE Robotics and Automation Letters*, 2026.

Pinhao Song, Ze Fu, Yutong Hu, and Renaud Detry. Assistron: Bayesian shared autonomy with off-the-shelf vision-language-action models. *arXiv preprint arXiv:2606.23147*, 2026.

Jinhe Tang and Weiming Zhi. Autointervene: Calibrated intervention for action-chunking imitation learning policies. *arXiv preprint arXiv:2608.07065*, 2026.

Yang Wan, Xihang Yue, Zhirui Liu, Ziyuan Chu, Shuxun Wang, Yuhan Chen, Xiaonan Jiang, Xukun Zhu, Yubo Dong, and Linchao Zhu. No free checker: A survey of verifiers for robot policies. *arXiv preprint arXiv:2609.09250*, 2026.

Yanwei Wang, Lirui Wang, Yilun Du, Balakumar Sundaralingam, Xuning Yang, Yu-Wei Chao, Claudia Pérez-D'Arpino, Dieter Fox, and Julie Shah. Inference-time policy steering through human interactions. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15626–15633. IEEE, 2025.

Weichen Xu, Zhenhua Liu, Lin Luo, Yaobo Liang, Chengtang Yao, Qingyu Mei, Jian Cao, Xixin Cao, Xing Zhang, Jiaolong Yang, et al. Continue or replan? bernoulli-continuation policy learning for adaptive horizon execution. *arXiv preprint arXiv:2608.03483*, 2026.

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

Jessie Yuan, Yilin Wu, and Andrea Bajcsy. When to act, ask, or learn: Uncertainty-aware policy steering. *arXiv preprint arXiv:2602.22474*, 2026.

Ganlong Zhao, Zijia Tang, Xingping Chen, Zhanghui Kuang, Ye Tian, and Guanbin Li. Flare: A failure-aware framework for autonomous correction and recovery in visual-language robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22391–22401, 2026.

A Controller Definitions and Evaluation Coverage

The controller names encode three choices: the consultation schedule, whether GPT-6 may leave execution with the policy, and which correction tool is offered. Table 1 groups the evaluated conditions by these choices. Base-25 is the policy-only reference; Base-15 and Base-OL are supplementary execution controls. Fixed-K is the central controller family, Always-K removes its continue option, and +EditOnly replaces the takeover tool with bounded editing. Selective-Segment lets GPT-6 choose the length of an accepted policy segment, while Always-Segment assigns every action segment to GPT-6.

Shared tool semantics. `continue_policy` leaves action generation to the frozen policy. `move_eef` specifies absolute grasp-point targets, executed as one open-loop interpolated chunk with the configured 0.05 rad waypoint increment. Fixed-K and Always-K then resume the policy for K steps before the next consultation; Selective-Segment and Always-Segment consult GPT-6 again immediately after the segment, so the next segment may again be a takeover. A `move_eef` target the planner cannot reach is refused: the call consumes budget, the robot does not move, and GPT-6 is re-asked once on the same observation; a second refusal holds the pose for 25 steps before the next consultation. Every executed segment is followed by a fresh observation and landing feedback. In +EditOnly conditions, `edit_proposal` replaces `move_eef` and applies bounded offsets to the policy proposal, as detailed in Appendix E. A call to `give_up` terminates the episode.

Random-15 control. Random-15 follows Fixed-15's 15-step schedule but selects consultations through scene-independent random draws, using a fixed seed per lane. At each scheduled check, GPT-6 is consulted with probability p and must either take over (`move_eef`) or terminate the episode (`give_up`); otherwise, the policy executes another 15 steps without consuming the call budget. We set p from Fixed-15's observed takeover rates: $p = 0.25$ on the weak set, its selection rate (431 of 1,737 decisions; execution rate 18.8%), and $p = 0.027$ on the strong set, its execution rate (14 of 526; selection rate 3.0%), and specify the schedule in the system prompt. This comparison assesses the value of using scene information to decide when to intervene (Appendix F.2).

B Extended Related Work

Learned manipulation and recovery. *Generalist policies.* RT-1 trained a transformer on 130k real-robot episodes and showed that policy capability scales with data (Brohan et al., 2022); OpenVLA released a 7B open VLA fine-tuned from a vision–language backbone (Kim et al., 2024); $\pi_{0.5}$ co-trains on heterogeneous robot and web data for open-world generalization and is the frozen policy of this paper (Black et al., 2025). *Failure detection and explanation.* SAFE trains a multitask failure detector on VLA internal features (Gu et al., 2025); AHA fine-tunes a VLM to detect and explain manipulation failures (Duan et al., 2025); REFLECT summarizes robot experience so that an LLM can explain a failure and plan a correction (Liu et al., 2023). Language-grounded monitoring of plan–execution misalignment follows the same idea (Guo et al., 2024). *Recovery by training or by a wrapped corrector.* RaC fine-tunes a policy on human recovery-and-correction rollouts and reports large gains on long-horizon tasks (Hu et al., 2026b); CycleVLA backtracks to the previous subtask when a VLM predicts failure and retries with better decoding (Ma et al., 2026); FLARE lets a multimodal monitor switch between task execution and learned reset skills (Zhao et al., 2026).

Test-time computation and verification. *Sampling and verifying action candidates.* RoboMonkey samples many action chunks from a VLA and lets a learned verifier choose (Kwok et al., 2025); CoVer shows that scaling a contrastive verifier can beat scaling the policy itself (Kwok et al., 2026); CheckVLA verifies committed chunks with a world model under a conformally calibrated risk threshold (Liu et al., 2026c). *World models for foresight and imagination.* Deep visual foresight planned robot motion by predicting future images (Finn & Levine, 2017); DreamAvoid dreams ahead at critical phases so that a VLA can avoid failures (Fan et al., 2026); DreamDojo learns a generalist robot world model from 44k hours of egocentric human video (Gao et al., 2026). Predictive world modeling also enhances generative policies at inference time (Qi et al., 2026). *Allocating assistance and compute.* DIRECT routes test-time compute for embodied VLM planners by scene context (Dao et al., 2026); ELASTIC learns how many denoising steps and samples a frozen generative policy

should spend per state (Li et al., 2026a). A continue-or-replan head turns the execution horizon itself into a learned decision (Xu et al., 2026).

Multimodal intervention in a motor policy. *A multimodal agent in charge of a frozen VLA.* The GPT-6 Astra as an Embodied Policy report runs an accept-or-correct loop in which the model either lets $\pi_{0.5}$ act or issues end-effector targets. VoLo treats a VLA as an interruptible tool that a VLM plans around, monitors, and steers mid-rollout (Chen et al., 2026a). RoboHarness wraps frozen π_0, $\pi_{0.5}$, and SmolVLA in a memory-augmented harness with tool interventions and no fine-tuning (Li et al., 2026b). A systematic study of such hierarchies isolates the planner–controller switching mechanism as a design axis (Hu et al., 2026a). *Steering a frozen policy's actions.* VLS steers the sampler of a frozen diffusion or flow policy with VLM-synthesized rewards (Liu et al., 2026b); DreamSteer ranks candidate action chunks with a latent world model and a language-conditioned value model (Cui et al., 2026); Learning What to Say searches for language feedback that raises closed-loop success and withholds it when it would hurt (Jeong et al., 2026). Bounded additive steering is the closest form to our `edit_proposal`: ITPS biases the sampling of a frozen generative policy toward human points, sketches, or physical corrections (Wang et al., 2025); Token Steering intervenes on the action tokens of an autoregressive VLA from low-dimensional user inputs that guide rather than override the policy (Chan & Kao, 2026); CorrectVLA converts a language correction into an additive action-magnitude adjustment applied to a frozen VLA (Kwon et al., 2026). Our edits are authored by GPT-6 on the displayed waypoints of the policy's own proposal, bounded per waypoint, and rejoin the proposal at the next unchanged waypoint. At the policy-free extreme, VLCP has the VLM write and rewrite the controller as code every K steps (Naouali et al., 2026). *Deciding when to intervene.* Uncertainty-aware policy steering calibrates a VLM verifier over policy samples and chooses per step to act, ask, or request a human action (Yuan et al., 2026); AutoIntervene scores proposed action chunks against a support memory to hand control to an operator and back (Tang & Zhi, 2026); Assistron solicits human intervention at contact-rich failure points of an off-the-shelf VLA (Song et al., 2026). Coordination contracts provide a related gate for bimanual policies (Liu et al., 2026a). Building on the Inspect-EEF harness, we study GPT-6's intervention in a frozen VLA policy by varying consultation schedules, whether policy continuation is allowed, and the available correction tools. We evaluate how these choices affect task outcomes, measured by Score and success rate, as well as intervention behavior and consultation cost.

C Evaluation Protocol

Benchmark and policy. RoboDojo uses the ARX X5 embodiment in Isaac Sim 5.1.0 with a simulated control frequency of 25 Hz, corresponding to 0.04 s of simulated time per control step. The benchmark's normalized reward criterion determines success, and its per-episode partial-credit Score is the continuous outcome we report as primary. All policy-based conditions use the official checkpoint `RoboDojo-sim-arx_x5-joint-0/59999`. It is openpi's `pi05` model fine-tuned from `pi05_base` (Black et al., 2025) on the benchmark's LeRobot v3.0 training set, `RoboDojo_sim_arx-x5_v30`, with all training task directories combined into one multi-task checkpoint (batch size 256, 60,000 training steps). It predicts a 50-step horizon of 14-dimensional joint actions and remains frozen throughout evaluation.

Harness. All GPT-6 conditions build on the L3 Inspect-EEF harness from the RoboProbe codebase[1]. In that harness, GPT-6 is the only source of actions: at every turn it either calls `move_eef` with world-frame grasp-point targets or calls `give_up`, and a non-learned conversion module validates each target, plans a joint-space path, and returns either an error without motion or the executed state with its arrival error. We retain this tool interface, the target validation and path planning, and the published system prompt. We add the frozen policy's 50-step proposal and its waypoint summary to the observation, the `continue_policy` and `edit_proposal` tools, and the consultation schedules of Table 1.

Task selection. We select ten task families by dividing the 0–72% range of official $\pi_{0.5}$ success rates into four equal bands and choosing six, two, one, and one

[1] `https://github.com/RoboProbe/RoboProbe`

families from the lowest to the highest band. These families cover all five RoboDojo capability dimensions (Chen et al., 2026b): Long-Horizon (`organize_table`, `classify_objects`, `make_kong`, `put_bottles_into_dustbin`), Generalization (`arrange_largest_number`, `pack_objects_into_box`, `fold_clothes`), Open (`classify_objects_by_language`), Memory (`imitate_sorting_sequence`), and Precision (`build_tower`). Following the official report, we use five episodes per family, split between standard and randomized configurations for the `arrange`, `pack`, and `fold` families. We divide these families into weak and strong sets based on Base-25's success on the evaluated layouts. The weak set assesses failure recovery and comprises seven families with at most one success in five episodes each. Base-25 succeeds once each on `classify_objects` and `make_kong` and in none of the episodes on the remaining five families, totaling 2/35. The strong set assesses preservation of the policy's existing capabilities and comprises three families with at least three successes in five episodes each: `build_tower` (3/5), `fold_clothes` (3/5), and `put_bottles_into_dustbin` (5/5), totaling 11/15.

Task subsets and budgets. The recorded evaluation comprises 50 episodes across ten task families and 13 task–scene configurations: 35 weak-set episodes from seven families and nine configurations, and 15 strong-set episodes from three families and four configurations. Each task family has five episodes; the `arrange`, `pack`, and `fold` families divide theirs between two configurations (two and three episodes), so a configuration has five, three, or two episodes. Table 5 gives the step limits. GPT-6 conditions additionally have a 100-call budget, increased to 120 for `imitate_sorting_sequence`. Episode termination may therefore reflect success, the task step limit, `give_up`, or exhaustion of the call budget.

Table 5: Per-episode control-step budgets. Both configurations are included where a task family is split.

Task family	Step limit
`fold_clothes`	500
`make_kong`	600
`put_bottles_into_dustbin`	700
`organize_table`	1000
`arrange_largest_number`, `build_tower`	1050
`classify_objects`, `classify_objects_by_language`	1100
`pack_objects_into_box`	1300
`imitate_sorting_sequence`	1600

RoboLab. The RoboLab experiments use the public RoboLab benchmark (v0.2.1) (Yang et al., 2026) on Isaac Lab 2.2: a Franka Panda with a Robotiq 2F-85 gripper controlled at 15 Hz by absolute joint targets, and the public DROID joint-position $\pi_{0.5}$ checkpoint as the frozen base policy (Base-15: it replans every 15 control steps). We evaluate ten tasks (`BlocksInBin`, `BlockStackingSpecifiedOrder`, `ButterAboveRaisin`, `CannedFoodInBin`, `ClutterPumpkin`, `LargerObjectRaisinBoxInBin`, `ReorientRedMug`, `RubiksCube`, `SauceBottlesCrate`, `YogurtInBowl`) with the benchmark's default instruction, five single-shot episodes per task (run indices 0–4, deterministic initial layouts, no retries), the task-defined step limit, and the benchmark's termination condition as success; Score is the benchmark's partial-credit subtask score with a successful episode scored 100. The step-level harness mirrors the Inspect-EEF harness at RoboLab's clock: at every consultation the policy proposes a 15-step chunk, which GPT-6 sees as an end-effector trajectory (forward kinematics of every row) next to the current and previous-boundary images (main and wrist, 504×896), the measured pose, and an explicit history of its earlier assessments. GPT-6 answers one strict-schema JSON decision: continue the policy for $k \leq 15$ steps, take over with an absolute end-effector target within 0.05 m and 0.35 rad of the current pose tracked for one to five control steps by damped least-squares inverse kinematics, edit the proposal by shifting its trajectory by at most 0.05 m and 0.35 rad over one to five steps (+EditOnly replaces the absolute-target tool by this edit), or stop. Fixed-15 consults every 15 policy steps; Always-Segment runs no policy and is consulted after every one-to-five-step segment, with stop holding the pose until the episode ends. Out-of-bound targets are projected onto the bounds, and there is no per-episode call budget beyond the step limit. GPT-6 uses medium reasoning effort.

Observations and model settings. GPT-6 (`openai/gpt-6-astra`) uses medium reasoning effort and receives the head and both wrist images. Hybrid controllers additionally receive the policy's next 50 steps summarized at steps 10, 20, 30, 40, and 50 as per-arm end-effector poses and gripper openings, together with the actions executed since the preceding consultation. The published system prompt is retained, with one paragraph describing the hybrid tools appended. The assisted-controller comparisons use the same Codex-thread invocation setting, model, and executor; the hybrid controllers additionally share the policy-proposal observation format. Base-25 uses neither GPT-6 nor the correction executor.

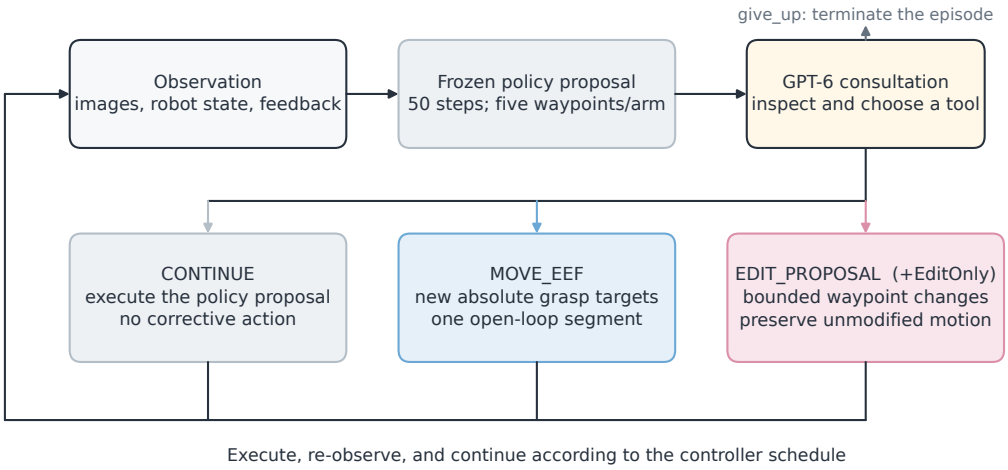


Figure 5: The shared decision loop. The frozen policy supplies an action proposal and GPT-6 receives the current observation at a consultation. It may continue the policy, specify absolute targets with `move_eef`, or, in +EditOnly conditions, modify proposal waypoints with `edit_proposal` instead of issuing absolute targets. The selected segment is executed before obtaining a fresh observation. `give_up` terminates the episode; the controller's consultation schedule determines the next decision.

Consultation and replanning. The consultation interval K counts base-policy control steps; takeover chunks insert additional execution before the policy resumes. Fixed-K and Always-K therefore share an assistance schedule, while differing in the responses available at each consultation. The policy's nominal replanning interval is 25 steps. Without takeovers or early termination, $K \in \{15, 25, 50, 100\}$ corresponds to 6.7, 4, 2, and 1 consultations per 100 policy steps, respectively, apart from the initial consultation. Selective-Segment instead allows GPT-6 to request an accepted policy segment of at most 50 steps.

D Metric Definitions and Statistical Interpretation

Paired gains and losses. Pairing requires identical task, configuration, layout, and initialization conditions for Base-25 and the evaluated controller. Base-25, Fixed-15, and Always-Segment each recorded 35 weak-set and 15 strong-set episodes. Let F_0 and S_0 denote the numbers of Base-25 failures and successes among the N pairs, with $N = F_0 + S_0$. For recovered failures R, lost successes L, and preserved successes P, the corresponding rates are R/F_0, L/S_0, and P/S_0, respectively; $P + L = S_0$. The paired success-rate difference is $100(R - L)/N$ percentage points.

Intervention behavior. Calls per episode count every model call; a refused `move_eef` is re-asked once on the same observation within the consultation, so calls exceed consultations (Fixed-15, weak set: 1,751 calls over 1,668 consultations). Takeover selection rate is the fraction of logged decisions other than `give_up` that call `move_eef`; takeover execution rate is the fraction whose `move_eef` the planner executed; the two differ by the refused calls, all of which named an unreachable target.

For +EditOnly conditions, the correction rate is the fraction selecting `edit_proposal`, the only correction tool; no `edit_proposal` call was refused. Give-ups count episodes terminated by `give_up`. Policy step share is the fraction of executed control steps attributed to the base policy in the logs. For the takeover controllers, calls equal executed takeovers plus refused calls plus continues plus give-ups in every episode, so the totals in Tables 8 and 4 reconcile with the call counts.

E THE EDIT_PROPOSAL TOOL

Arguments and bounds. The +EditOnly conditions offer `edit_proposal` in place of absolute-target takeover. Its arguments index the five displayed proposal rows, `w10` through `w50`, separately for each arm. Translation offsets (`dx`, `dy`, `dz`) are expressed in metres in the world frame and bounded by ± 0.05 m per axis. A world-frame rotation vector (`drx`, `dry`, `drz`) has magnitude at most 20°. An optional gripper command selects `open`, `close`, or the proposal's own opening. Missing or null fields leave the corresponding component unchanged. Edits refer to the grasp point, 0.150 m along the tool axis from the flange; the executor adjusts the flange so that rotation preserves the requested grasp-point location.

Execution and feedback. For an edited segment of length $S \leq 50$, the executor uses displayed waypoints up to S plus the endpoint at S; offsets assigned beyond S are ignored and identified in the tool response. Fixed-K and Always-K use $S = K$ within this range. A translation exceeding the 0.3 cm dead band, or any rotation, activates closed-loop inverse kinematics for that arm toward the edited waypoint. Unchanged arms replay the policy's joint commands, and an edited arm returns to the proposal at the next unchanged waypoint. A gripper-only edit preserves the joint commands and ramps the opening over its sub-segment. A call with no translation, rotation, or gripper change is an identity edit. Out-of-bounds values, non-numeric arguments, and unknown keys are refused: the call consumes budget and returns an error without moving the robot. After execution, the next observation reports each arm's measured landing relative to its requested endpoint.

Execution precision does not separate the two tools. Across 6,930 pose-named arm moves, the open-loop `move_eef` chunk lands with a median error of 0.03 cm against the requested pose (2.8% beyond 2 cm). Across 2,017 edited arm-waypoints, the closed-loop edit branch reaches the edited waypoint with a median error of 0.00 cm (7.3% beyond 2 cm, concentrated in the two fastest-moving tasks). In addition, none of the 310 weak-set give-up rationales mentions execution. The tools differ instead in what a correction may express and whether it outlives the segment (Section 4.3).

F COMPLETE OUTCOMES AND BEHAVIORAL MEASUREMENTS

F.1 PER-CONFIGURATION OUTCOMES

Tables 6 and 7 give the complete task breakdown underlying the main results. The weak-set table retains all 23 evaluated controllers, including supplementary policy cadences and the forced +EditOnly conditions; Score is the mean partial credit over the same 35 episodes. The strong-set table contains the conditions evaluated on the strong set and also reports the combined 50-episode success rate.

Supplementary policy cadences. Base-15 records 8.6% weak-set and 73.3% strong-set success, while Base-OL records 5.7% and 53.3%. Their Scores are 16.3 and 78.0 for Base-15 and 16.0 and 63.3 for Base-OL, against 18.1 and 76.7 for Base-25. Base-OL executes each 50-step chunk open loop and is taken from the native open-loop evaluation on the Base-25 layout IDs. These runs document sensitivity to policy execution cadence; they do not expand the main intervention family. Changing Base-25 to Base-15 adds one weak-set success in this sample, compared with the six-success difference between Base-25 and Fixed-15 in the full weak-set aggregates.

F.2 RANDOM GATING CONTROL

Fixed-K beats Always-K at every interval (Table 3), but the two differ both in who chooses the moments of intervention and in how often intervention happens. Random-15 (Appendix A) separates the two: it consults GPT-6 on the Fixed-15 clock at a random subset of the ticks, with p set to Fixed-15's takeover selection rate, and GPT-6 must call `move_eef` whenever it is consulted. Table 8

Table 6: Weak-set mean Score, success rate (%) over the 35 episodes, and mean Score by configuration. The nine configuration columns give the mean Score over that configuration's episodes; "random" denotes the randomized variant of a configuration. Bold marks the best value in each column and underline the second best. The Random-15 control (Appendix F.2) is listed for reference and excluded from the marks.

Controller	Score	Success (%)	Organize table	Arrange number	Arrange random	Classify by language	Imitate sorting	Pack box	Pack random	Classify objects	Make kong
Base-25	18.1	5.7	50.0	15.0	0.0	0.0	1.0	25.0	20.0	28.0	20.0
Base-15	16.3	8.6	25.0	15.0	1.7	2.0	0.0	30.0	33.3	28.0	20.0
Base-OL	16.0	5.7	35.0	10.0	0.0	0.0	0.0	17.5	11.7	19.0	40.0
Selective-Segment	35.4	20.0	30.0	65.0	23.3	20.0	13.0	50.0	10.0	39.0	**80.0**
Fixed-15	**42.9**	**22.9**	55.0	**100.0**	13.3	**38.0**	23.0	17.5	**50.0**	59.0	40.0
Fixed-25	29.6	17.1	35.0	65.0	15.0	4.0	29.0	37.5	10.0	63.0	20.0
Fixed-50	24.3	14.3	35.0	57.5	1.7	2.0	3.0	5.0	41.7	19.0	60.0
Fixed-100	24.0	14.3	40.0	65.0	1.7	4.0	2.0	**55.0**	16.7	43.0	20.0
Selective-Segment+EditOnly	22.6	8.6	45.0	52.5	0.0	10.0	7.0	17.5	20.0	36.0	20.0
Fixed-15+EditOnly	25.1	14.3	55.0	15.0	0.0	0.0	0.0	**55.0**	41.7	28.0	40.0
Fixed-25+EditOnly	18.4	8.6	40.0	22.5	0.0	0.0	1.0	**55.0**	15.0	28.0	20.0
Fixed-50+EditOnly	25.1	20.0	25.0	17.5	6.7	4.0	1.0	50.0	15.0	46.0	60.0
Fixed-100+EditOnly	19.7	11.4	15.0	17.5	0.0	0.0	3.0	10.0	33.3	29.0	60.0
Always-15+EditOnly	24.7	11.4	**60.0**	15.0	1.7	20.0	0.0	12.5	16.7	51.0	20.0
Always-25+EditOnly	21.4	11.4	25.0	10.0	0.0	4.0	0.0	30.0	28.3	68.0	20.0
Always-50+EditOnly	19.7	8.6	35.0	15.0	1.7	8.0	8.0	17.5	11.7	46.0	20.0
Always-100+EditOnly	19.7	8.6	45.0	10.0	0.0	0.0	1.0	12.5	25.0	28.0	40.0
Always-15	32.7	20.0	45.0	**100.0**	11.7	12.0	15.0	10.0	25.0	**71.0**	20.0
Always-25	26.6	11.4	55.0	**100.0**	3.3	16.0	5.0	25.0	6.7	54.0	0.0
Always-50	13.7	2.9	35.0	15.0	1.7	4.0	2.0	12.5	20.0	31.0	0.0
Always-100	13.7	5.7	30.0	15.0	1.7	0.0	2.0	5.0	25.0	40.0	0.0
Always-Segment	37.7	17.1	55.0	57.5	**76.7**	32.0	**52.0**	17.5	11.7	42.0	0.0
Random-15 (control)	20.1	8.6	40.0	7.5	3.3	0.0	3.0	37.5	20.0	46.0	20.0

Table 7: Mean Score on the strong set and over all 50 episodes, success rate (%) on the strong set (15 episodes), the weak set (35), and combined (50), and mean Score by strong-set task. Tower: build tower; fold/fold-r: folding / randomized folding; bottles: put bottles into dustbin. Bold marks the best value in each column and underline the second best. The Random-15 control (Appendix F.2) is listed for reference and excluded from the marks.

Controller	Score strong	Score all	Success (%) strong	Success (%) weak	Success (%) all	Score Tower	Score Fold	Score Fold-r	Score Bottles
Base-25	76.7	35.7	**73.3**	5.7	26.0	66.0	**100.0**	40.0	**100.0**
Base-15	**78.0**	34.8	**73.3**	8.6	28.0	**82.0**	**100.0**	40.0	88.0
Base-OL	63.3	30.2	53.3	5.7	20.0	54.0	**100.0**	13.3	88.0
Selective-Segment	56.0	41.6	46.7	20.0	28.0	36.0	60.0	13.3	**100.0**
Fixed-15	76.7	**53.0**	**73.3**	**22.9**	**38.0**	50.0	**100.0**	66.7	**100.0**
Fixed-25	72.0	42.3	66.7	17.1	32.0	64.0	60.0	46.7	**100.0**
Fixed-50	54.7	33.4	46.7	14.3	24.0	28.0	60.0	40.0	88.0
Fixed-100	60.7	35.0	53.3	14.3	26.0	50.0	20.0	40.0	**100.0**
Selective-Segment+EditOnly	46.7	29.8	33.3	8.6	16.0	32.0	20.0	20.0	88.0
Fixed-15+EditOnly	58.0	35.0	53.3	14.3	26.0	26.0	60.0	40.0	**100.0**
Fixed-25+EditOnly	64.0	32.1	60.0	8.6	24.0	44.0	60.0	40.0	**100.0**
Fixed-50+EditOnly	68.0	38.0	60.0	20.0	32.0	64.0	**100.0**	20.0	88.0
Fixed-100+EditOnly	48.0	28.2	40.0	11.4	20.0	28.0	60.0	6.7	88.0
Always-15+EditOnly	64.0	36.5	60.0	11.4	26.0	44.0	60.0	40.0	**100.0**
Always-25+EditOnly	74.7	37.4	66.7	11.4	28.0	72.0	60.0	46.7	**100.0**
Always-50+EditOnly	50.7	29.0	40.0	8.6	18.0	32.0	60.0	13.3	88.0
Always-100+EditOnly	53.3	29.8	46.7	8.6	20.0	28.0	**100.0**	6.7	88.0
Always-15	58.0	40.3	46.7	20.0	28.0	62.0	60.0	40.0	64.0
Always-25	48.0	33.0	40.0	11.4	20.0	4.0	60.0	46.7	88.0
Always-50	48.3	24.1	40.0	2.9	14.0	8.0	**100.0**	40.0	73.0
Always-100	42.0	22.2	26.7	5.7	12.0	22.0	60.0	6.7	76.0
Always-Segment	37.3	37.6	20.0	17.1	18.0	26.0	50.0	**73.3**	22.0
Random-15 (control)	83.3	39.1	80.0	8.6	30.0	82.0	100.0	46.7	100.0

pairs it with Fixed-15 and the references layout by layout. On the weak set Random-15 selects as many takeovers as Fixed-15 (434 versus 431 over the 35 episodes) and executes more (384 versus

327; 50 versus 104 targets refused; realised consult share 0.23), gives up less often (5 versus 14 episodes), yet reaches a Score of 20.1 (SR: 8.6%), the level of the policy alone: Base-25 scores 2.0 ± 3.6 points lower over the 35 pairs. Random-15 recovers 1 of the 33 Base-25 failures; Fixed-15 recovers 6. Fixed-15 scores 22.7 ± 5.7 points higher than Random-15 over the 35 layouts. On the strong set the coin fires 14 times in 15 episodes, every takeover executes, and Random-15 stays at the policy's level (Score 83.3; SR: 80.0%), as does Fixed-15 (76.7; SR: 73.3%): at a takeover execution rate of 2.7% the strong set does not separate chosen from random moments. The weak-set result therefore attributes the Fixed-K gain to the choice of when to intervene rather than to intervening less.

Table 8: Random-15 against Fixed-15 and the references, paired by layout. Weak set: 35 layouts. ΔScore is the mean paired Score difference to Random-15, with its standard error, over the 35 layouts. Recovered / Preserved count the Base-25 failures recovered and the Base-25 successes kept over the same 35 pairs. Takeovers are executed `move_eef` calls, with refused calls in parentheses; Give-ups are totals over the 35 weak-set episodes. Strong set: 15 layouts.

	Weak set						Strong set	
Controller	Score	Success (%)	ΔScore	Recovered / Preserved	Takeovers	Give-ups	Score	Success (%)
Random-15	20.1	8.6	—	1/33 / 2/2	384 (50)	5	83.3	80.0
Fixed-15	42.9	22.9	$+22.7 \pm 5.7$	6/33 / 2/2	327 (104)	14	76.7	73.3
Always-Segment	37.7	17.1	$+17.6 \pm 8.0$	5/33 / 1/2	2,408 (205)	15	37.3	20.0
Base-25	18.1	5.7	-2.0 ± 3.6	—	0	—	76.7	73.3
Base-15	16.3	8.6	-3.9 ± 4.1	—	0	—	78.0	73.3

F.3 Tool Use in the +EditOnly Conditions

On the weak set, +EditOnly controllers that retain `continue_policy` select `edit_proposal` in 21%–44% of decisions other than `give_up`, and otherwise continue the policy. Always-K+EditOnly requires selection of the editing tool at every non-terminal consultation, but permits the proposed waypoints to remain unchanged. Compared with their takeover counterparts, eight of the nine +EditOnly variants terminate more episodes with `give_up`; the remaining pair ties. For example, Fixed-25+EditOnly increases these terminations from 13 to 30 of 35 episodes. Table 9 reports the detailed counts.

Table 9: Logged behavior of the +EditOnly conditions on the weak-set episodes. Success is the success rate (%) over the 35 episodes; Calls and GPT-6 time (16 s per call) are means per episode; Edits and Continues are totals over decisions other than `give_up`; Give-ups are episodes ended by `give_up`. Bold marks the best Score and Success and underline the second best. Always-K+EditOnly has no continue option, so its Continues cell is "—".

Controller	Score	Success (%)	Calls	Edits	Continues	Give-ups	GPT-6 time (s)
Selective-Segment+EditOnly	22.6	8.6	18.4	217	404	22	295
Fixed-15+EditOnly	**25.1**	<u>14.3</u>	39.8	291	1,074	29	637
Fixed-25+EditOnly	18.4	8.6	26.5	235	663	30	425
Fixed-50+EditOnly	**25.1**	**20.0**	16.9	250	320	22	271
Fixed-100+EditOnly	19.7	11.4	8.9	132	168	10	142
Always-15+EditOnly	<u>24.7</u>	11.4	42.9	1,472	—	28	686
Always-25+EditOnly	21.4	11.4	28.8	981	—	26	460
Always-50+EditOnly	19.7	8.6	17.3	582	—	21	276
Always-100+EditOnly	19.7	8.6	9.3	319	—	7	149

F.4 Give-Up Behavior and Recovery by Error Type

This section supports the discussion of `give_up` in Section 4.3 with the weak-set record. Every GPT-6 controller can end an episode with `give_up`; across the 19 non-random GPT-6 controllers the weak set contains 310 such decisions, 115 under controllers that correct by absolute-target takeover (the Fixed-K, Always-K, Selective-Segment and Always-Segment conditions) and 195 under the +EditOnly conditions. For each give-up we record the fraction of the task's step limit (Table 5) that was still available, the episode's final Score, and whether the paired controller with the same schedule and the other correction tool succeeded on the same layout. Table 10 shows that editing

controllers stop much earlier: their give-ups leave a median 33% of the step budget unused and 64% of them leave more than a quarter, against 6% and 17% under takeover, and the gap appears at every one of the nine schedules. The share of give-ups on layouts that the paired controller solved is similar between the two tools (19 of 195 versus 10 of 100). The tool difference is confined to the episodes that editing abandons: over the nine schedule pairs, on the 91 layouts where neither controller gave up the outcomes are the same (26 of 91 successes under each tool; Score 36.8 versus 36.5), and under Fixed-15 the 17.8-point Score gap decomposes into 19.9 points contributed by the 29 episodes that editing abandoned (mean Score 36.7 under takeover against 12.8 under editing, a within-subset difference of 24.0 points) and -2.1 points contributed by the other 6 (72.5 against 85.0, -12.5 points); each contribution is the subset's summed difference divided by all 35 episodes. None of the 310 rationales cites an execution or tracking error.

Table 10: Weak-set `give_up` decisions by schedule and correction tool. Budget left is the median fraction of the task's step limit (Table 5; for `imitate_sorting_sequence` the limit includes the 600-step demonstration) remaining at the give-up step; Early is the share of give-ups with more than a quarter of the step limit remaining; Score is the mean final Score of the abandoned episodes. Always-Segment has no editing counterpart; the All row sums every takeover and every editing give-up.

	Takeover				Edit			
Schedule	Give-ups	Budget left (%)	Early (%)	Score	Give-ups	Budget left (%)	Early (%)	Score
Fixed-15	14	8	14	29.3	29	43	83	12.8
Fixed-25	13	6	23	13.5	30	40	77	9.8
Fixed-50	8	13	25	8.8	22	14	36	4.1
Fixed-100	7	13	43	7.1	10	31	80	6.0
Always-15	14	7	0	17.9	28	42	79	13.4
Always-25	16	11	25	18.4	26	33	65	12.7
Always-50	10	20	30	13.5	21	25	48	11.7
Always-100	7	12	43	22.1	7	38	57	23.6
Selective-Segment	11	2	0	19.1	22	19	41	13.2
Always-Segment	15	3	0	24.7	—	—	—	—
All	115	6	17	18.4	195	33	64	11.4

GPT-6 states a reason with every `give_up`. We classified the 310 rationales into five categories with a fixed rubric, using the text alone: the correction tool cannot redirect the policy or reach the target (bounded edits have not redirected it, targets refused or unreachable); the policy keeps repeating the same error without a tool limit being cited; insufficient remaining steps; an irrecoverable physical state (object dropped or knocked over, rule violated); and other. Two independent labelling passes agreed on 297 of the 310 records; the remainder were adjudicated in a third pass. Table 11 gives the result: 156 of the 195 editing give-ups (80%) attribute the impasse to what bounded edits can do, and 81% mention a limit of the tool somewhere in the rationale, whereas takeover give-ups mostly cite insufficient remaining steps (74 of 115) and cite a tool limit in 23% of cases.

Table 11: Primary reason GPT-6 gives for `give_up` on the weak set, by correction tool. Counts with the share of each tool's give-ups in parentheses; the last row counts rationales that mention any limit of the correction tool, independently of the primary category.

Primary reason given by GPT-6	Takeover (n=115)	Edit (n=195)
correction tool cannot redirect or reach	22 (19%)	156 (80%)
policy keeps repeating the same error	2 (2%)	4 (2%)
insufficient remaining steps	74 (64%)	30 (15%)
irrecoverable physical state	11 (10%)	3 (2%)
other	6 (5%)	2 (1%)
Mentions a limit of the correction tool	26 (23%)	157 (81%)

Finally, Table 12 relates recovery to the type of the policy's error. The 33 Base-25 weak-set failures carry the manual labels of Appendix F.8 (7 geometric, 26 semantic), and each controller's outcome on the same layout is paired as in Figure 2. Pooled over the four Fixed-K schedules, geometric failures are recovered in 5 of 28 attempts under both tools, while semantic failures are recovered in 12 of 104 attempts with takeover and 9 of 104 with editing; at least one schedule recovers 4 and 3 of the 7

geometric failures and 7 and 5 of the 26 semantic ones. The error type therefore sets the recovery rate for both tools, and the tool difference is confined to the semantic majority.

Table 12: Recovery of Base-25 weak-set failures by failure type and correction tool at the four Fixed-K schedules. Entries are recovered failures over labelled failures; Pooled counts every (failure, schedule) attempt.

	Geometric (7 failures)		Semantic (26 failures)	
Schedule	Takeover	Edit	Takeover	Edit
Fixed-15	2/7	1/7	4/26	3/26
Fixed-25	0/7	0/7	4/26	2/26
Fixed-50	3/7	2/7	1/26	3/26
Fixed-100	0/7	2/7	3/26	1/26
Pooled over K	5/28	5/28	12/104	9/104
Recovered by at least one K	4/7	3/7	7/26	5/26

F.5 Evaluation on Additional Tasks

Table 13 reports success counts on the six additional tasks. Fixed-15 achieves 15 successes in 30 episodes, compared with 10 for Base-25 and 14 for Always-Segment. The effects vary across tasks: relative to Base-25, Fixed-15 increases successes from 0/5 to 5/5 on `cover_blocks` and from 1/5 to 3/5 on `stack_blocks`, but decreases them from 5/5 to 4/5 on `stack_bowls` and from 2/5 to 1/5 on `pour_liquid_into_cup`. Its success counts are unchanged on the remaining two tasks. Fixed-15+EditOnly achieves 9 successes in 30 episodes, compared with 15 for Fixed-15, despite similar consultation counts (22.2 versus 21.7 per episode). It also terminates more episodes with `give_up`: 14, compared with 8 for Fixed-15 and 10 for Always-Segment.

Table 13: Per-task success counts and aggregate results on the additional task set. Each controller is evaluated on five layouts per task, totaling 30 episodes. Score and consultations are means over episodes; give-ups count episodes ending with `give_up`.

Task	Base-25	Fixed-15	Fixed-15+EditOnly	Always-Segment
`stack_bowls`	5/5	4/5	2/5	5/5
`pour_liquid_into_cup`	2/5	1/5	2/5	0/5
`stack_blocks`	1/5	3/5	2/5	4/5
`cover_blocks`	0/5	5/5	2/5	2/5
`general_pickup`	2/5	2/5	1/5	3/5
`press_by_number`	0/5	0/5	0/5	0/5
Success rate (%)	33.3	**50.0**	30.0	46.7
Score	37.3	**51.5**	33.5	47.2
Consultations per episode	0	21.7	22.2	46.6
Give-ups	0	8	14	10

Table 14 summarizes the RoboLab outcomes and Table 15 reports the per-task success counts. Fixed-15 achieves 35 successes in 50 episodes, compared with 13 for Base-15, 32 for Fixed-15+EditOnly and 22 for Always-Segment. Always-Segment completes none of the 15 episodes of the three long-horizon tasks (`BlocksInBin`, `BlockStackingSpecifiedOrder`, `ClutterPumpkin`), where its episodes run to the step limit with 270–450 calls each, and it matches Fixed-15 only on `ReorientRedMug` and `RubiksCube`. Fixed-15+EditOnly exceeds Fixed-15 on `SauceBottlesCrate`, `CannedFoodInBin` and `BlockStackingSpecifiedOrder` and trails it on `LargerObjectRaisinBoxInBin` and `ClutterPumpkin`.

F.6 A Weak-Set Episode with a Hold

Figure 6 shows how Fixed-15 responds to unreachable correction targets on `classify_objects_by_language`, layout 0. GPT-6 intervenes when the policy carries watches and wooden toys toward the wrong baskets. Attempts to place a wooden toy directly into the right basket are rejected because the target is unreachable for the left arm, triggering a hold when the harness's retry limit is reached. GPT-6 then places the toy at the table centre for transfer to the right arm, which deposits it in the correct basket. Base-25 fails on the same layout, leaving

Table 14: RoboLab: Score (mean partial credit, 0–100) and success rate (%) under the same allocations of control, ten tasks × five episodes (50 recorded episodes per controller). Calls are GPT-6 calls per episode; takeover is the share of GPT-6 decisions that replace or edit the policy's segment.

Controller	Allocation of control	Score	SR (%)	Calls/ep.	Takeover (%)
Base-15	Policy only	36.5 ± 6.2	26.0 ± 6.2	–	–
Always-Segment	GPT-6 only	55.1 ± 6.3	44.0 ± 7.0	151	100
Fixed-15	Policy + optional takeover	$\mathbf{80.8} \pm 4.9$	$\mathbf{70.0} \pm 6.5$	54	18.4
Fixed-15+EditOnly	Policy + optional edit	$\underline{76.5} \pm 5.1$	$\underline{64.0} \pm 6.8$	85	37.9

Table 15: Per-task success counts and aggregate results on RoboLab. Each controller is evaluated on five episodes per task, totaling 50 episodes. Score and consultations are means over episodes; give-ups count episodes ending with `stop`.

Task	Base-15	Fixed-15	Fixed-15+EditOnly	Always-Segment
`BlocksInBin`	0/5	2/5	1/5	0/5
`BlockStackingSpecifiedOrder`	0/5	2/5	3/5	0/5
`ButterAboveRaisin`	1/5	5/5	4/5	4/5
`CannedFoodInBin`	3/5	4/5	5/5	3/5
`ClutterPumpkin`	0/5	2/5	0/5	0/5
`LargerObjectRaisinBoxInBin`	0/5	5/5	3/5	2/5
`ReorientRedMug`	1/5	3/5	2/5	3/5
`RubiksCube`	5/5	5/5	5/5	5/5
`SauceBottlesCrate`	2/5	3/5	5/5	2/5
`YogurtInBowl`	1/5	4/5	4/5	3/5
Success rate (%)	26.0	**70.0**	<u>64.0</u>	44.0
Score	36.5	**80.8**	<u>76.5</u>	55.1
Consultations per episode	0	54.0	85.4	151.4
Give-ups	0	0	4	1

wooden toys in the middle basket and a car on the table. The hold is imposed by the harness rather than selected by GPT-6.

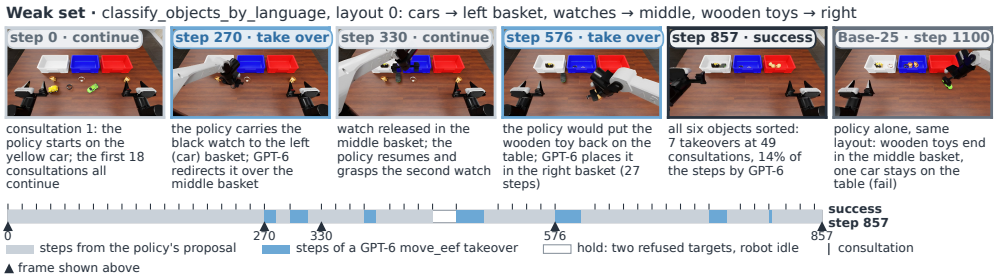


Figure 6: Fixed-15 on `classify_objects_by_language`, layout 0. Frames show the control step and GPT-6's decision at the nearest consultation; the bar shows which controller supplies each executed step, with ticks at consultations. The unfilled segment is a 25-step hold after two refused targets (steps 447–472), during which the robot stays idle.

F.7 Episodes with Other Outcomes

Figures 7 and 8 present additional trajectories with different outcomes and correction interfaces. Each row ends with a summary of the controller, outcome, Score, and call and step counts. Unfilled timeline segments indicate inactivity during a harness-triggered hold or after termination by `give_up` or call-budget exhaustion.

Figure 7 illustrates three unsuccessful outcomes. (a) On `classify_objects_by_language`, layout 4, Fixed-15 redirects the peppers and chocolate bars toward the correct baskets, but leaves the wooden toys unsorted and reaches the step limit (Score 40). (b) On `organize_table`, layout 1, the policy completes object placement with a brief grasp correction from GPT-6. GPT-6 then takes

over to open the drawer, but its attempts fail and it calls `give_up` (Score 75). (c) On the same layout, Always-Segment repeatedly retries grasps and adjusts object placement, but completes only the mouse and clock placements before exhausting its call budget (Score 25).

Figure 8 illustrates outcomes under bounded editing and full GPT-6 control. (a) On `classify_objects`, layout 0, Fixed-50+EditOnly lowers the policy's proposed grasp of the white pen through a single edit, after which the policy completes the task. (b) On `imitate_sorting_sequence`, layout 0, Fixed-50+EditOnly repeatedly edits the proposal to grasp the remote after the demonstration. The remote is lifted but falls outside the basket, and GPT-6 eventually calls `give_up` (Score 0). (c) On `build_tower`, layout 0 of the strong set, Always-Segment restores the lower layer after a board slips, but gives up before building the upper layers (Score 30).

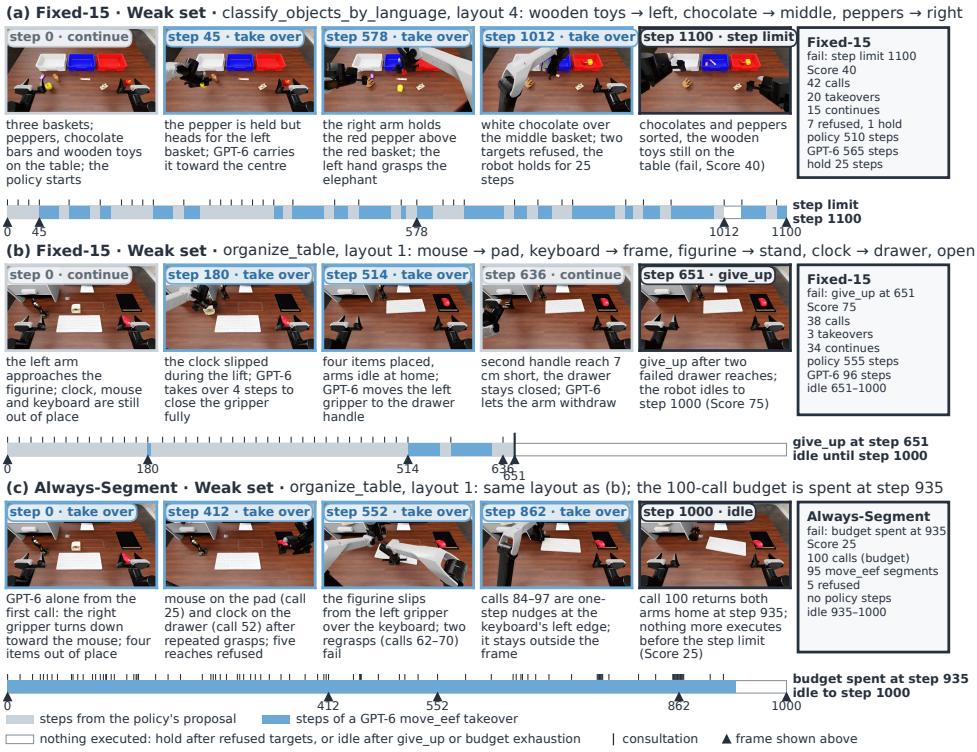


Figure 7: Three unsuccessful episode endings: (a) Fixed-15 reaches the step limit on `classify_objects_by_language`, layout 4; (b) Fixed-15 gives up on `organize_table`, layout 1; and (c) Always-Segment exhausts its call budget on the same layout. Frames show control steps and nearby GPT-6 decisions; timeline colors identify action sources, and ticks mark consultations.

F.8 Failure-Type Annotation

The policy-only weak-set runs (Base-25 and Base-15) were manually labeled from ten head-camera frames per episode, supplemented by wrist views where needed. The label families are geometric (missed grasp, misplacement, stall, collision) and semantic (wrong object, wrong order, skipped sub-goal, false completion). Table 16 gives the Base-25 counts: 26 of its 33 failures carry a semantic label. On 31 shared failing layouts, Base-25 and Base-15 agree on the label family in 23 cases.

G Consultation Cost and Execution Time

Section 4.4 and Table 4 report consultation behavior and execution time. This appendix provides additional behavioral details and the invocation setting.

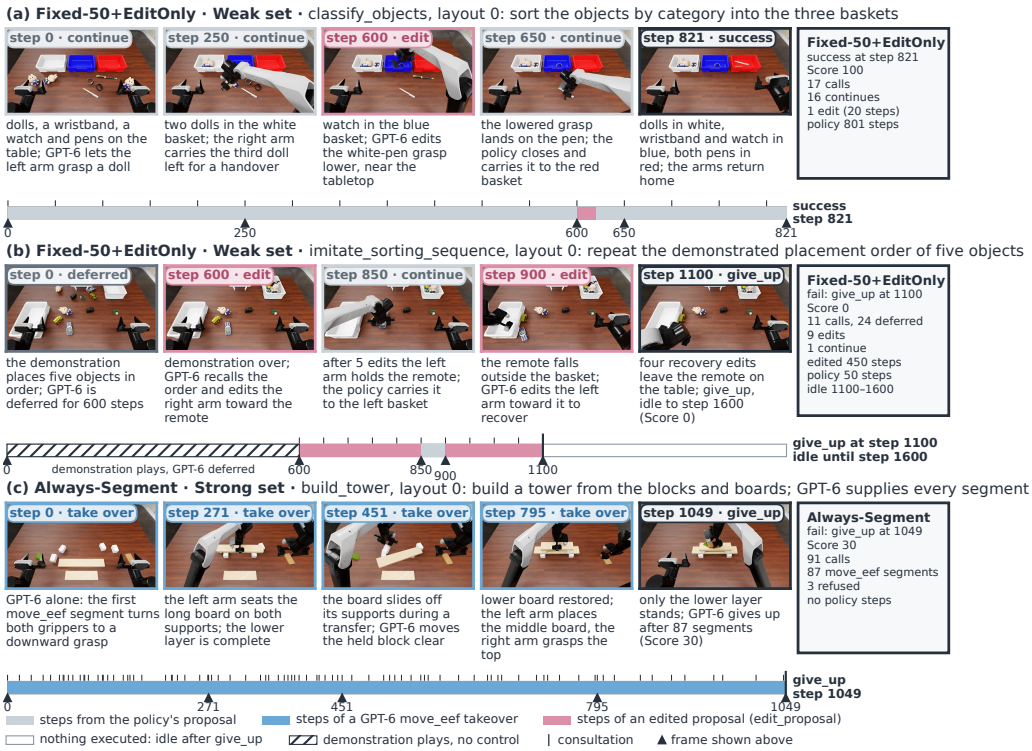


Figure 8: Bounded editing and full GPT-6 control: (a) a single grasp edit followed by policy completion; (b) repeated edits followed by give-up; and (c) partial tower construction followed by give-up under Always-Segment. Pink timeline segments indicate execution of edited proposals. The hatched segment in (b) marks the demonstration phase, during which the harness defers consultations.

Table 16: Base-25 weak-set failure labels (33 failed episodes out of 35).

Family	Subtype	Episodes
Geometric	Missed grasp	1
	Misplacement	1
	Stall	3
	Collision	2
Semantic	Wrong object	9
	Wrong order	6
	Skipped sub-goal	5
	False completion	6
Total		33

Behavior beyond aggregate success. On the strong set, takeover execution rates are 3.7% (11 of 299 decisions) and 11.7% (19 of 163) for Fixed-25 and Fixed-50, with selection rates of 4.0% and 12.3%. Selective-Segment averages 20 consultations per episode on the strong set, with a 43% selection rate and a 39% execution rate (129 and 117 of 302 decisions); its execution rates on tower building and randomized folding are 52% (99 of 192) and 37% (14 of 38).

Invocation setting. The Codex-thread runs use a login-node relay and Codex CLI 0.154.0 with model `gpt-6-astra`; the per-call model latency is 16 s. In the Always-Segment run, each turn re-sends the conversation, with a median 2.3 million input tokens (mostly cache-served) and approximately 6400 output tokens under the forced output schema.