# SEPARATING DETECTION FROM REPAIR IN LONG-HORIZON MANIPULATION

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

## ABSTRACT

A common architecture for long-horizon manipulation pairs a high-level planner with a low-level executor. The planner is expected to notice execution failures and put the task back on course. Research has focused on when to consult the planner, whether at subtask boundaries, on a schedule, or after signs of failure. We test their shared assumption that a planner consulted at the right moment can repair what it finds. Using a monitoring layer built for this study, we compare nine combinations of failure monitoring, planner consultation, and physical retry on identical scenes in RoboCasa365. We evaluate five types of failure signals and find that stalls, or prolonged periods without progress, provide a selective trigger for planner consultation. With ground-truth poses supplied as a diagnostic control, the stall monitor triggers consultations in most episodes that ultimately fail and rarely in those that succeed. Yet detecting trouble does not translate to successful repair. Through systematic experiments and analyses, we find no run-time intervention improves task completion over the executor alone. A review before execution provides a revealing contrast. It corrects known ordering defects and raises completion on some of them, showing that the planner can repair a faulty plan. The persistent stalls we analyze expose a different problem. In half of them, the executor never grasps the object, even though an applicable instruction usually exists. Consulting the planner at a stall does not detectably increase the rate at which the stalled segment resumes, and retrying the same skill rarely succeeds. In the perceptual configuration, the monitors trigger planner consultation in nearly every episode, including successful ones. These comparisons locate the bottleneck in the hands, not the plan. In this setting, the findings point toward training recovery skills for failure states, rather than further changes to when or how the planner is consulted.

## 1 INTRODUCTION

A robot may grasp an object, open a drawer, and place an item on a shelf, yet fail to perform these skills in sequence. Generalist policies have widened the repertoire of individual skills (Brohan et al., 2023; Kim et al., 2025; Black et al., 2025b;a; Ghosh et al., 2024), and composite benchmarks now ask for extended household activities whose subtasks must succeed in an order set by physical dependencies (Nasiriany et al., 2026; Li et al., 2023). Local competence does not guarantee completion. A slipped grasp, an object left behind, or an unclosed drawer invalidates every later step. The standard remedy is hierarchical. A high-level planner sits above the learned executor, selects its skills, watches execution, and, when something goes wrong, is expected to intervene and restore the task to its course (Ichter et al., 2023; Huang et al., 2023b; Guo et al., 2024; Liu et al., 2023b). The expectation is natural. The planner holds and can change the plan, so it should also repair execution.

Work in this design has therefore concentrated on *when* the planner should be consulted. Consulting it at subtask boundaries ties every intervention to the decomposition. A fixed schedule removes that dependence but spends calls without regard to evidence. Consulting it on detected trouble is the most natural, but inherits detection errors. Each rule determines when to consult the planner. Can consultation at the right moment, with the right evidence, repair what has gone wrong? The design thus bundles two capabilities, noticing that execution has gone wrong and fixing it. The first has been studied in its own right (Liu et al., 2023b; Agia et al., 2025; Xu et al., 2025; Gu et al., 2025; Pacaud et al., 2026; Navasardyan et al., 2026; Xia et al., 2026). This paper isolates the second, asking whether run-time deliberation improves long-horizon outcomes with a fixed executor.



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

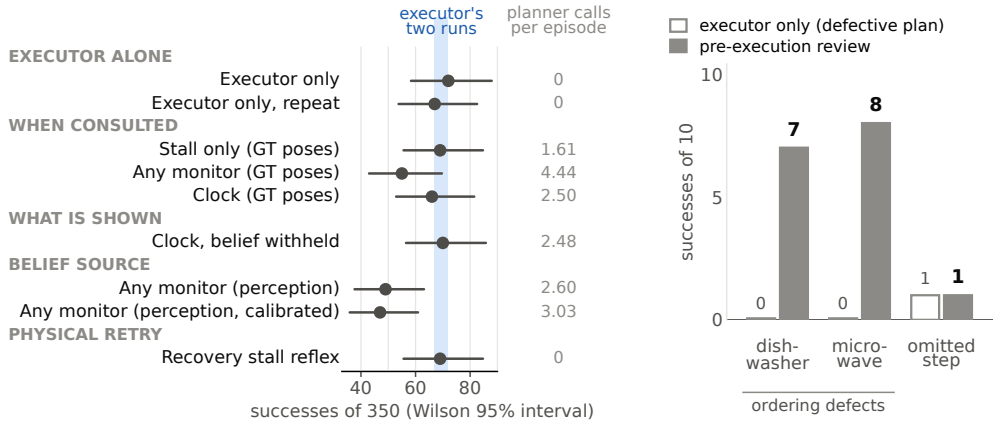


Figure 1: **Left.** Completions in 350 shared scenes, Wilson 95% intervals, and planner calls per episode for the executor, its repeat, and seven consultation or retry configurations. Shading spans both executor runs. None of five pre-registered monitored configurations differs from the executor after correction. **Right.** Successes out of ten before and after one pre-execution review, which repairs both shown order defects but not the omission.

We vary a fixed planner's interaction with a fixed executor through a monitoring layer whose explicit object belief makes alarms traceable and whose five switches select the belief source, consultation strategy, planner context, review timing, and response to a stall. Stalls prompt a planner edit to pending instructions or a scripted return to the segment's start to repeat its instruction. The executor uses its own observations and active instruction. The executor alone is the baseline.

We compare configurations on seven RoboCasa365 composite tasks, with 50 scenes per task. All configurations run the same 350 scenes, and the planner receives the task goal. *Noticing works*. A progress-based stall monitor flags most failing episodes while leaving almost every successful one uninterrupted when supplied with ground-truth poses as a diagnostic control. The broader belief monitors warn earlier but also interrupt most successes. *Fixing does not follow*. Consulting the planner on alarms, a schedule, or stalls during execution does not improve on the executor alone.

To tell a faulty plan from a failed execution, we test constructed plans with known ordering defects in separate ten-episode experiments. A review before execution finds and corrects the defect, raising success rate from 0% to 70% on loading the dishwasher and from 0% to 80% on steaming a vegetable. The planner can therefore fix what is wrong with a plan. The failures that occur in the main study are of a different kind. The executor never acquires the target object in half of the 193 episodes in which execution stalls and never resumes. An instruction in the executor's vocabulary applies to 73% of these persistent-stall episodes, and consulting the planner at a stall does not detectably raise the rate at which stalled segments resume (Section 5.2). The scripted retry completes the stalled segment in 38% of applications for contact stalls and only 5% for grasp stalls. Repair is bounded by the hands' skills in the failed state, whether or not the planner chooses well.

The picture gets worse in a specific way under noisy perception. The belief monitors raise alarms on nearly every episode, including absence alarms for objects that perception has never detected. Calibrating the monitors on healthy demonstrations makes them quiet and precise, with 40 alarms across the 350 episodes, most corresponding to real events. Yet completion remains below the Executor only baseline, and much of the loss lies in perception-based segment completion. The bottleneck is the hands rather than the plan. Monitoring offers awareness and economy, since stall signals alone would hand off most failures early with few false stops. Progress in repair lies in learned recovery skills.

**Contributions.** ❶ A controlled measurement of the separation between detection and repair on identical scenes. By varying five aspects of the planner–executor interaction, the comparison shows that selective failure warning does not translate into improved completion through run-time planner consultation. ❷ A localization of the remaining failures in the executor's competence on the failed skill rather than in the plan. Stalls persist even when the executor's vocabulary contains an instruction for the situation. Consulting the planner does not detectably change how often a stalled segment resumes. A scripted retry of the same skill rarely completes a stalled grasp. Tasks unseen

in the fine-tuning almost never succeed. The planner repairs imposed ordering defects, while natural failures persist.

## 2 Related Work

**Planner-in-the-loop repair.** Language models translate tasks into admissible action sequences (Huang et al., 2022), executable policies (Liang et al., 2023), grounded plans (Singh et al., 2023; Song et al., 2023), or continuous trajectory constraints (Huang et al., 2023a). In the planner-over-skills branch, SayCan grounds selection in learned affordances (Ichter et al., 2023), and Inner Monologue returns textual feedback (Huang et al., 2023b). DoReMi replans on constraint violations (Guo et al., 2024), and REFLECT explains failures to guide correction (Liu et al., 2023b). Visual feedback drives RePLan and ReplanVLM (Skreta et al., 2024; Mei et al., 2024), failed preconditions motivate CAPE (Raman et al., 2024), and execution feedback incentivizes STAR and BrainBody-LLM (Sakib & Sun, 2025; Bhat et al., 2025). ViReSkill diagnoses and revises the first failing step (Kagaya et al., 2025). From Inner Monologue to ViReSkill, replanning systems vary consultation timing and evidence. They share the premise that the planner can repair what it finds evidenced by end-to-end success. However, this measure combines noticing a failure, informing the planner, and repairing it, rarely compared with what the executor achieves unaided on the same failures. It is also rarely compared with the variation between repeated runs. We measure the repair step against both. In their evaluations, failures are often induced by adversarial interventions or injected disturbances (Huang et al., 2023b; Guo et al., 2024), whereas REFLECT synthesizes planning and execution failures (Liu et al., 2023b). Recovery comes from repeating a step the executor can perform, or from a corrected plan. A corrected plan may add a corrective action for an unmet precondition (Raman et al., 2024). The natural failures we observe differ, where the executor repeatedly fails the step itself. Pre-execution review repairs plan-level defects in our experiments as well. KnowNo calibrates requests for human help (Ren et al., 2023). Reflexion and Self-Refine study revision through feedback outside robotics (Shinn et al., 2023; Madaan et al., 2023).

**Detection as a separate measurement.** RoboFail provides a failure corpus (Liu et al., 2023b). Sentinel monitors action consistency and task progress (Agia et al., 2025). FAIL-Detect calibrates sequential out-of-distribution warnings from successful behavior (Xu et al., 2025), and SAFE learns cross-task signals from VLA features (Gu et al., 2025). These studies examine warning accuracy and timing. Guardian trains a vision-language verifier on synthesized failure reasoning (Pacaud et al., 2026), and FailBench benchmarks success judgments (Navasardyan et al., 2026). RoboProcess-Bench tests process-aware understanding (Xia et al., 2026), and StepEval proposes subgoal-level evaluation (ElMallah et al., 2025). Detection remains imperfect, with the strongest FailBench verifier reaching 0.77 mean balanced accuracy. A warning identifies a need for repair, which still requires an executable correction. True poses reduce observation error in the diagnostic control.

## 3 Experimental Instrument and Controlled Interventions

With the learned executor and multimodal planner held fixed, we vary state source, consultation strategy, planner context, review timing, and stall response to separate detection from repair.

### 3.1 Setting

A task is a chain $\mathcal{C} = (s_1, \ldots, s_J)$ of segment instructions. The composite-fine-tuned $\pi_{0.5}$ executor (Black et al., 2025a) maps the active instruction and current camera images to an action chunk (Section 4). The multimodal planner, Claude Opus 5, operates outside the low-level control loop and may edit the chain when consulted. We disable extended thinking, use medium effort and JSON-schema output, and allow at most 2,048 output tokens. The model exposes no temperature setting.

Table 1 separately compares two other executor checkpoints. The pretrained executor receives no task-specific fine-tuning. The executor fine-tuned on single skills uses individual-skill demonstrations rather than composite-task segments. Both run without planner consultation and are outside the nine configurations that share the composite-fine-tuned executor (Appendix B).

The seven composite tasks in this study use 23 subgoal-completion conditions (Section 4). In configurations using ground-truth poses, the simulator checks after each control step whether the current

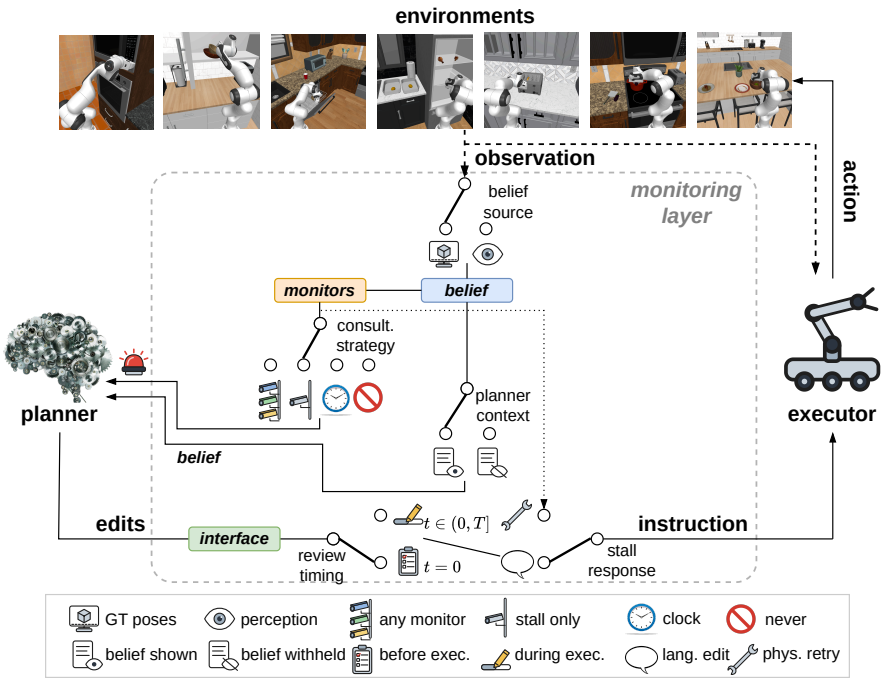


Figure 2: The instrument between a fixed planner and executor. Seven environments supply observations to the belief and separately to the executor. Five switches control the experiment. The belief-source switch selects visible-object GT poses with dead reckoning or open-vocabulary detection with depth. The consultation-strategy switch selects any enabled monitor, stall only, 4.3 evenly spaced clock check-ins per episode budget, or never. The planner-context switch shows or withholds the belief with an alarm. The review-timing switch selects one review at $t = 0$ or reviews during execution at $t \in (0, T]$, with budget $T$. The stall-response switch selects a planner language edit or monitor-triggered physical retry without the planner. A signal passes only through the contact its lever touches, and each outgoing line starts between the contacts. Both models stay fixed across configurations. The executor is fine-tuned once on composite segments before the study.

subgoal has been achieved. As soon as the check succeeds, the executor receives the next instruction. Perceptual configurations use no simulator completion signal. 17 of the 23 conditions can be evaluated from estimated object positions. These checks use a spatial tolerance widened by three estimated standard deviations ($3\sigma$) and require the completion condition to hold at two consecutive detection updates before switching instructions. The remaining six require fixture-state information that the perception stack does not estimate, such as whether a door is closed or a button is pressed. Those segments therefore advance only when their allotted control steps run out. Each of the seventeen position-based subgoals likewise waits for its segment budget to expire if its object has never been detected. Each segment also advances on timeout if its completion check has not passed. The chain ends at final completion or the episode budget.

### 3.2 Belief, monitors, and planner interface

The belief records each task object's position, uncertainty, observation age, and grasp state. This explicit state makes each alarm raised by a monitor traceable to a named quantity, with switchable state source and planner visibility (Appendix E). Monitors test this state to determine when the planner is consulted and what evidence it receives. Held-out re-observations verify tracking accuracy and support the monitors' inputs (Appendix J).

**Absence** reports an object missing when it should be visible. **Re-observation mismatch** reports a same-class detection that persistently disagrees with the predicted position and has no alternative gated track. **Grasp inconsistency** reports a held object's failure to move with the hand. **Unintended motion or regression** reports motion of a free object away from the hand or the loss of achieved

Table 1: The first nine rows use the 350 shared scenes and switch icons from Figure 2. A dash marks an inapplicable switch. The last columns give successful episodes and planner calls per episode. The last two rows separately vary executor training with no planner calls. *Thresholds calibrated on successful demonstrations, with a stall clock added. †The reported Any monitor (perception) rerun corrects a runner defect that dropped the planner's inserted and reordered steps (Appendix I).

| Configuration | belief source | consult. strategy | planner context | review timing | stall response | Success of 350 | Calls per episode |
|---|---|---|---|---|---|---|---|
| Executor only | – |  | – | – | – | 72 | 0 |
| Executor only, repeat | – |  | – | – | – | 67 | 0 |
| Stall only (GT poses) |  |  |  |  |  | 69 | 1.61 |
| Any monitor (GT poses) |  |  |  |  |  | 55 | 4.44 |
| Clock (GT poses) |  |  |  |  | – | 66 | 2.50 |
| Clock, belief withheld |  |  |  |  | – | 70 | 2.48 |
| Any monitor (perception)† |  |  |  |  | – | 49 | 2.60 |
| Any monitor (perception, calibrated)* |  |  |  |  |  | 47 | 3.03 |
| Recovery stall reflex |  |  | – | – |  | 69 | 0 |
| *Separate executor-training comparisons* | | | | | | | |
| Pretrained executor | – |  | – | – | – | 6 | 0 |
| Executor fine-tuned on single skills | – |  | – | – | – | 0 | 0 |

progress. **Stall** reports 300 control steps without a latched subgoal or relevant object or fixture displacement beyond 2 cm. Formal definitions and fixed persistence windows are in Appendix E.

Each consultation supplies three camera images and the task goal and identity. It also supplies segment status and attempts, remaining budget, and intervention history. Trigger type and operands are included, along with the belief when enabled. The planner can take six actions. It can *continue*, *rewrite* the active instruction, or *skip* a pending segment. It can also *insert* a recovery segment, *reorder* the pending suffix, or *rebind* an object reference. Edits are free text, affect pending steps, and cannot supply unlearned skills. We allow at most two retries per segment and ten planner consultations per episode, with a refractory period after each consultation.

### 3.3 Experimental comparisons

Executor only provides the reference for intervention outcomes, and its repeat measures variation without intervention (Table 1). We compare scheduled, stall-only, and broader monitor-triggered consultation to measure how consultation strategy affects task completion. At a fixed schedule, showing or withholding the belief tests its role in the planner's decisions while leaving its use by the monitors unchanged. On plans with known defects, reactive consultation and pre-execution review test whether the planner repairs the plan before or during execution (Table 2). Scripted physical retry tests whether repeating a skill resolves failures that persist under planner intervention.

The available monitor families depend on the state source, so switching from ground-truth poses to perception also changes which alarms can trigger consultation. In Any monitor (GT poses), the planner wakes in response to stall, re-observation mismatch, unintended motion or regression, and grasp inconsistency. Absence is evaluated and logged only. Under perception, the planner wakes in response to absence, re-observation mismatch, and grasp inconsistency. Stall and motion require fixture state that perception does not estimate. In perceptual configurations, detection runs every 30 control steps and monitors every 60, with a fire served at the next evaluation. Any monitor (perception, calibrated) uses thresholds calibrated on healthy demonstrations and adds a time-in-segment stall clock. Segment-completion rules also differ by state source (Section 3.1).

## 4 Experimental Protocol

**Benchmark and executor.** The study uses RoboCasa365 simulation with rendered depth and frozen open-vocabulary detection. The seven selected composite tasks cover steaming, dishwasher loading,

bowl stacking, lunch packing, toasting, stirring, and leftover storage (Nasiriany et al., 2026). Each task has three or four segments and generated scenes containing task objects and distractors. Success is determined by the benchmark's end-state checks. The $\pi_{0.5}$ executor is fine-tuned on instruction-bearing composite segments and fixed across configurations. The separate executor-training comparisons use the pretrained executor and the executor fine-tuned on single skills (Appendix B).

**Main study.** We evaluate nine configurations on the same 350 scenes, with 50 scenes per task (Table 1). Two runs use the executor alone, with the repeat measuring outcome variation without intervention. Six configurations consult the planner during execution. The remaining configuration applies a scripted physical retry without consulting the planner. All nine use the same executor fine-tuned on composite segments, so the comparisons measure interventions around a fixed executor.

**Planner-context comparison.** When consulted, the planner receives the benchmark's task instruction and a plain-language statement of the conditions required for success. The shared prompt's headings label the task, plan state, and object belief when provided. An earlier preliminary study gave six configurations independently drawn scenes and a planner that received task names without explicit goals or prompt headings. The context experiment repeats that setup with the goal information above and the shared template's headings restored. It measures the combined effect of explicit task information and prompt organization on the planner's edits and subsequent recovery. Appendix G reports both studies separately. Neither is pooled with the main study.

**Plan and retry controls.** Ordering and omission defects have known repairs, respectively restoring the correct order and inserting a missing step. For each defect, ten-episode cells compare Executor only, reactive consultation, and pre-execution review, with correct-chain controls and Fisher tests (Table 2). Choosing a repair and completing the chain are measured separately. Scripted physical retry uses the main study's scenes, executor, and GT poses. Retry conversion means segment completion within the granted budget, distinct from task success, and is reported by failure cause and attempt. A pre-specified conversion rate of 0.20 was required for a follow-on configuration in which the planner could choose the retry (Appendix H).

## 5 Results on Detection and Repair

We measure detection, repair, planner context, and consultation cost on identical scenes with goal-bearing context. Configurations using ground-truth poses are diagnostic controls, using true object pose while visible to a policy camera and dead reckoning otherwise. Preliminary-study and context-experiment results are labeled and never pooled with the main study.

### 5.1 Detection works selectively

Warning is the first monitor wake. Recall is the share of failing episodes warned, and false-stop rate the share of successful episodes with a wake. The latter measures interruption under stopping at the first wake, not observed loss from monitoring. With GT poses as a diagnostic control, Stall only has recall 0.66 and false-stop rate 0.07, against 0.97 and 0.76 for Any monitor. Perception wakes in 349 of 350 episodes, including 300 of 301 failures and every success. These include selective stall warning, sensitive but intrusive GT-pose warning, and near-indiscriminate perceptual warning (Figure 3).

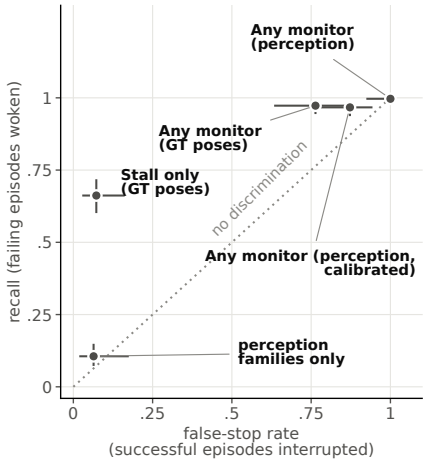


Figure 3: Warning trade-off of the monitors. The dotted diagonal denotes wakes unrelated to outcome.

Never-detected objects account for 38.3% of absence-wake slot mentions (350 of 915). In 82.0% of toaster wakes, no belief object has ever been detected (50 of 61). These counts and the missing fixture state locate the warning problem in coverage and fixture blindness. Calibration on successful demonstrations yields 40 fires in 350 episodes, with precision 0.75 for absence, 0.93 for mismatch, and 0.92 for grasp inconsistency. The perception families wake in 35 episodes, with recall 0.106, false-stop rate 0.064, and

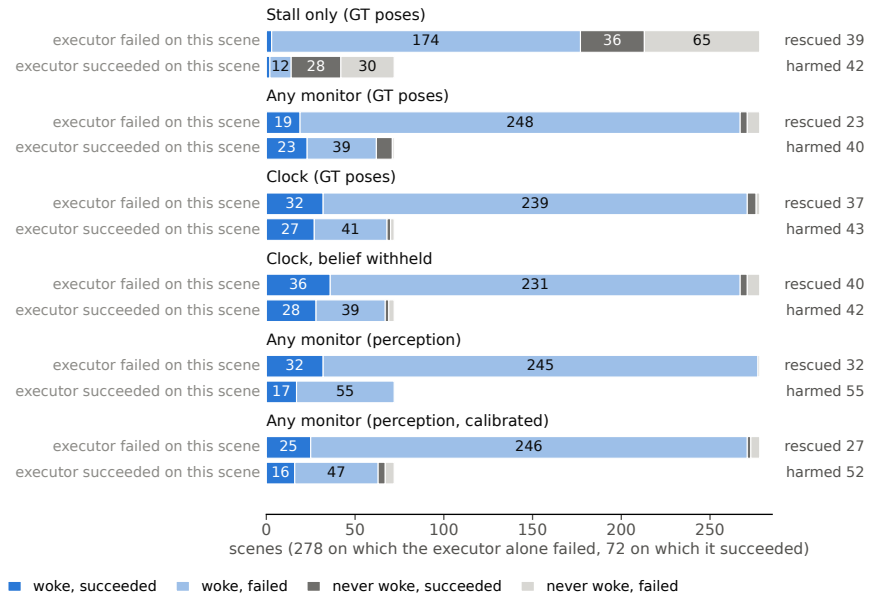


Figure 4: Outcomes split by whether the planner was consulted. Each configuration runs 350 shared scenes, grouped by the executor alone's outcome. Bars separate episodes with and without consultation and their eventual success or failure. Rescues and harms are changes from Executor only.

median 60% budget remaining. They are as selective as the stall monitor at about a sixth of its recall. The added stall clock supplies recall 0.97 on timeout-driven chains.

### 5.2 Repairing plans does not resolve execution failures

**Composite outcomes and scene-level discordance.** Executor only completes 72 of 350 episodes. Across nine configurations, completion ranges from 47 to 72, with no improvement over Executor only (Table 1). None of the five pre-registered monitored configurations differs after Holm correction. Any monitor (perception) is lowest in that family at 49. The repeat completes 67 and Recovery stall reflex 69, neither differing nominally. The separately pre-registered calibrated configuration completes 47. Interventions change outcomes on 63–87 scenes, compared with the repeat's 77 (22%), comprising 36 apparent rescues and 41 harms.

**Within-scene outcomes.** Within-scene outcomes are consistent with task-stratified independence for the replicate and five interventions, whose task-stratified Mantel–Haenszel intervals include 1. Any monitor (perception) has a common odds ratio of 0.97, and Any monitor (perception, calibrated) 1.90. Outcomes on a scene agree with the executor's more often than independence predicts in Any monitor (GT poses) and Recovery stall reflex, with odds ratios of 2.73 and 2.99, respectively.[1] Relative to that expectation, Any monitor (GT poses) has about eight fewer rescued scenes and eight fewer harmed scenes, while Recovery stall reflex has about ten fewer of each. Both-succeeded scenes are higher by the corresponding amount in each comparison. These rows establish an excess of concordance without identifying its source. Neither association is a repair signal. In Stall only (GT poses), the planner was never consulted in 36 of the 39 rescued scenes and in 30 of the 42 harmed scenes. Figure 4 separates outcomes with and without consultation. The aggregate null leaves smaller effects unresolved.

**Natural failures at the stall.** With ground-truth poses, 224 episodes contain a stall wake. Of these, 193 make no further progress and none succeeds. In this persistent-stall subset, the executor

[1]For executor $e$ and comparison $a$, let $p_{e,t}$ and $p_{a,t}$ be their success rates on task $t$. Under the baseline assumption of independence conditional on task, $D_{\mathrm{ind},a} = \sum_{t=1}^{7} \frac{n_t}{350}[p_{e,t}(1-p_{a,t}) + p_{a,t}(1-p_{e,t})]$, where $n_t = 50$. Rates are computed separately for each comparison, with a reference over all tasks of approximately 0.237, or 83 of 350 scenes. Conditioning on task accounts for the concentration of successes in four tasks. We use this expectation as a reference for observed discordance.

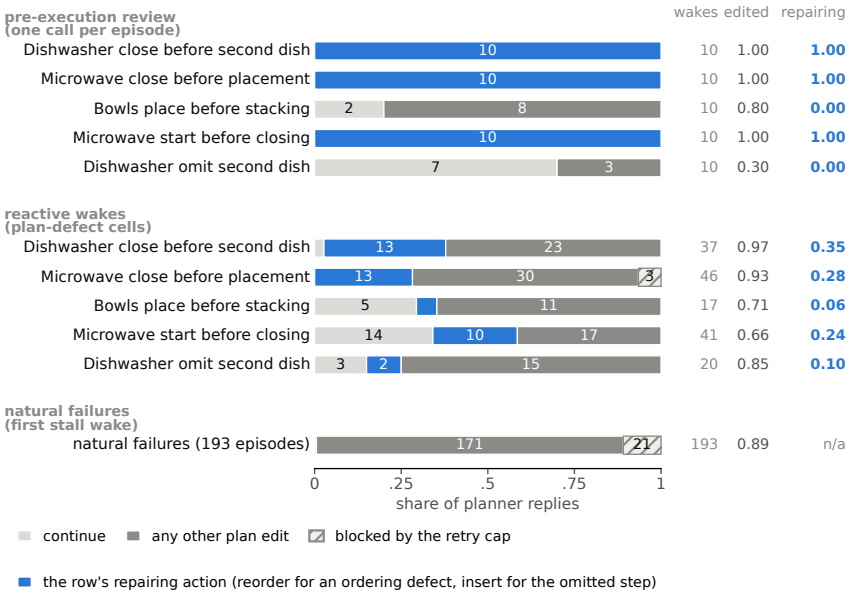


Figure 5: Planner replies from pre-execution review, reactive plan-defect wakes, and first stall wakes in 193 persistent natural failures. Bars split continue, repairing actions (reorder for order defects, insert for omission), other edits, and replies blocked by the retry cap. Columns report replies, editing share, and repairing-action share. No single operator defines repair for natural failures, hence n/a.

never grasps the target in 98 episodes (50.8%), while a trained instruction applies in 141 (73%). An available instruction therefore does not establish execution competence. Figure 5 reports the first-stall replies. The subset characterizes persistent stalls, not recovery across natural failures.

**Recovery after a stall, with and without the planner.** A pre-registered comparison, analyzed once, applies the stall rule offline to Executor only and uses the consultation configurations' first stall wakes. Without the planner, segments resume in 33/216 stalls (15.3%). They resume in 28/191 with Stall only (GT poses) (14.7%) and in 29/222 with Any monitor (GT poses) (13.1%). Episode successes are 9, 5, and 7. Neither slightly lower planner rate differs detectably. Stall only succeeds on 3 of the 177 scenes that Executor only failed and on which Stall only consulted the planner. The executor repeat succeeds on 8 of the same scenes. Reconstruction excludes 81 Executor-only episodes with unevaluable fixture-only first stalls. Completion-rate intervals appear in Appendix L.

The contrast extends beyond instruction wording. Reorders introduce no new strings and repair plan-visible defects, while scripted retry reissues a trained instruction and rarely converts grasp stalls. Free-text rewrites likewise show no composite-success gain. Training coverage supplies another boundary. Toast, Stir, and Store were absent from fine-tuning and account for 150 of 350 episodes per configuration. Across Executor only and the five pre-registered monitored configurations, they contribute five successes in 900 episodes, against 376 in 1,200 on the four fine-tuned tasks. The corrected aggregate test uses all 350 scenes, but roughly 200 carry most of the comparison. No additional subgroup significance test is reported.

**A physical second attempt.** Recovery stall reflex retries after a scripted retreat without a planner call. The primitive was validated separately (Appendix H). It converts 19 of 309 segments, with rates of 0.38 for contact stalls, 0.05 for grasp acquisition, and 0.02 on the second application. Composite success is 69 of 350 with no detectable gain, and conversion misses the pre-specified 0.20 gate for a follow-on configuration in which the planner could choose the retry. Returning the arm and repeating an instruction rarely resolves a failed grasp. Together with frequent grasp failures, this cause split locates the limitation in the skill and motivates training on its failure states.

**Plan-visible ordering defects.** Review restores three of four ordering defects, leaves bowls unordered, and inserts no omitted step (Table 2). Success rises from 0 to 7 of 10 for dishwasher and from 0 to 8 of 10 for microwave close-before-placement. For microwave start-before-closing, every reviewed plan is correct but only one episode succeeds. The planner's fix can be right while the hands fail. The two successful repairs saw the belief, but withholding it in replay returned identical plans in all twenty episodes.

Table 2: Plan-defect results with the primary planner and ten episodes per defective-chain cell. E is executor alone, R reactive consultation, and V one pre-execution review with ground-truth poses and no later consultation. V repair counts restored canonical orders or inserted omitted steps. Only the two yes cells show the belief, and no correct-chain control does. Correct-chain controls are shared across each task's defects and give review successes out of ten and executor-only successes out of fifty in the main study. Two-sided Fisher exact tests compare V and E on defective chains.

| | Defective chain | | | | | Correct chain | | |
|---|---|---|---|---|---|---|---|---|
| Task and defect | E | R | V | V repair | Belief in review | Review | Executor | Fisher $p$ |
| Dishwasher close before second dish | 0/10 | 0/10 | 7/10 | 10/10 | yes | 5/10 | 27/50 | 0.01 |
| Dishwasher omit second dish | 1/10 | 1/10 | 1/10 | 0/10 | no | 5/10 | 27/50 | 1.00 |
| Bowls place before stacking | 0/10 | 0/10 | 0/10 | 0/10 | no | 0/10 | 16/50 | 1.00 |
| Microwave close before placement | 0/10 | 0/10 | 8/10 | 10/10 | yes | 2/10 | 21/50 | 0.00 |
| Microwave start before closing | 0/10 | 2/10 | 1/10 | 10/10 | no | 2/10 | 21/50 | 1.00 |

Reactive repairing-action shares range from 6% to 35%, with completions of 0, 0, 0, 2, and 1 of 10 in the row order of Figure 5. Correct-chain review and executor controls are separate (Table 2). Overlapping Wilson intervals leave the repaired-versus-correct-review difference unresolved, and the zero-success bowls review control leaves that repair effect unresolved. Both significant benefits are ordering repairs, so the two-defect-class retention criterion is not met.

The second planner, Claude Sonnet 5, restores both orders, repairs no omission, and shows no detectable difference from Executor only on 105 paired scenes (Appendix K).

Initial review can correct a dependency violation before execution commits it. Reactive edits affect pending steps and cannot undo consequences outside the executor's vocabulary. Persistent natural failures instead occur when the action is already specified but cannot be reliably executed.

### 5.3 Context changes choices more than outcomes

**Belief for monitoring and planner context.** We measure the belief's role in monitoring by comparing warnings under GT poses and perception and checking restraint (Section 5.1). The representation is fixed, but the configurations differ in state source, wake families, and completion rules. Clock, belief withheld tests only planner context, retaining monitor use of the belief, GT poses, schedule, and completion rules. Clock (GT poses) completes 66 episodes against 70 with the belief withheld, a difference of $-4/350$ and no measurable benefit from the snapshot format as designed. The preliminary study separately reports 68 against 72 (Appendix G). Neither tests its value to monitors.

**A goal-bearing context.** The context experiment restores goals and headings, raising repairing-action choice from 7% to 23% and changing edited-episode recovery from 8% to 7% (Appendix G).

**Measured boundaries.** Perceptual completion is 49 and 47 episodes, with the corrected-family and separate tests above. Completion and warning selectivity are distinct measurements. On the microwave task, Any monitor (perception) completes 1 of 50 episodes against 21 of 50 for Executor only, and 144 of its 159 segment advances came from timeouts, since closing the door and starting the oven have no belief-evaluated completion. Preliminary microwave completion is 3/50 versus 13/50 (Appendix G). Appendix F reports first-edit timing.

## 6 Conclusion

We separate detection quality, consultation cost, and repair across nine configurations on shared scenes with a fixed executor. No run-time intervention improves completion despite selective warning. Review corrects three of four ordering defects but raises success on only two, separating plan correction from execution competence. Grasp acquisition, training coverage, and post-stall comparisons locate persistent failures in the executor. Consultation does not detectably increase segment resumption, and retry rarely converts grasp stalls. Goal context changes choices without increasing recovery. Calibration makes perceptual warnings selective without restoring completion. These findings point toward recovery skills trained on failure states.

### AI USE STATEMENT

In this work, we used generative AI tools for conducting part of human-designed experiments. We used generative AI to search for related work to help us understand how our work will sit in the broader research field. In our system, we employed Claude Opus 5 API to play the role as the high-level planner in the planner-in-loop manipulation. Details have been disclosed in 3.1. Finally, we used generative AI tools to help polish the writing and refine figure plotting. We have not used generative AI tools for directly synthesizing the manuscript. We also have not used generative AI to design the entire system. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by 2 authors. LLM (API)-generated planning instructions are standardized and valid. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

Sections 3 and 4 specify the experimental configurations, planner interface, shared scenes, and control comparisons. Appendix B gives the executor's training data, checkpoint, optimization settings, and held-out acceptance check. Appendix C defines progress scoring and documents the runner and logging corrections. Appendix D describes fixture-estimator training and validation, including the criterion that kept it out of the evaluated configurations. Appendix E defines the belief and monitors, while Appendix J reports the held-out tracking check. The preliminary study and context experiment are documented separately in Appendix G, and Appendix I records implementation checks and the corrected perceptual run. These specify alternative outcomes for plan review, the main study, physical retry, and perceptual calibration before the corresponding analyses. Each pre-registered comparison was analyzed once at its declared sample size, and the conclusions follow the declared outcomes.

## REFERENCES

Christopher Agia, Rohan Sinha, Jingyun Yang, Ziang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking Failure Modes of Generative Policies: Runtime Monitoring of Consistency and Progress. In *CoRL*, 2025.

Vaidehi Bagaria, Bijo Sebastian, and Nirav Kumar Patel. Recursive Belief Vision Language Action Models. *arXiv preprint arXiv:2602.20659*, 2026.

Vineet Bhat, Ali Umut Kaypak, Prashanth Krishnamurthy, Ramesh Karri, and Farshad Khorrami. Grounding Large Language Models for Robot Task Planning Using Closed-Loop State Feedback. *Advanced Robotics Research*, pp. e202500072, 2025. doi: 10.1002/adrr.202500072. URL https://doi.org/10.1002/adrr.202500072.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. In *CoRL*, 2025a.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. $\pi_0$: A Vision-Language-Action Flow Model for General Robot Control. In *RSS*, 2025b.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov,

Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In *CoRL*, 2023.

Yuanpei Chen, Chen Wang, Li Fei-Fei, and Karen Liu. Sequential Dexterity: Chaining Dexterous Policies for Long-Horizon Manipulation. In *CoRL*, 2023.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *RSS*, 2023.

Ramy ElMallah, Krish Chhajer, and Chi-Guhn Lee. Score the Steps, Not Just the Goal: VLM-Based Subgoal Evaluation for Robotic Manipulation. In *Workshop on Evaluation and Deployment Across the Robot Learning Lifecycle at CoRL*, 2025.

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

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. ManiSkill2: A Unified Benchmark for Generalizable Manipulation Skills. In *ICLR*, 2023.

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 *NeurIPS*, 2025.

Yanjiang Guo, Yen-Jen Wang, Lihan Zha, and Jianyu Chen. DoReMi: Grounding Language Model by Detecting and Recovering from Plan-Execution Misalignment. In *IROS*, 2024.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay Policy Learning: Solving Long-Horizon Tasks via Imitation and Reinforcement Learning. In *CoRL*, 2020.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. In *ICML*, 2022.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models. In *CoRL*, 2023a.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner Monologue: Embodied Reasoning through Planning with Language Models. In *CoRL*, 2023b.

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

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. RLBench: The Robot Learning Benchmark & Learning Environment. *RAL*, 5(2):3019–3026, 2020. doi: 10.1109/lra.2020.2974707.

Tomoyuki Kagaya, Subramanian Lakshmi, Anbang Ye, Thong Jing Yuan, Jayashree Karlekar, Sugiri Pranata, Natsuki Murakami, Akira Kinose, and Yang You. ViReSkill: Vision-Grounded Replanning with Skill Memory for LLM-Based Planning in Lifelong Robot Learning. *arXiv preprint arXiv:2509.24219*, 2025. URL https://arxiv.org/abs/2509.24219v1.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model. In *CoRL*, 2025.

Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. DART: Noise Injection for Robust Imitation Learning. In *CoRL*, 2017.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, Mona Anvari, Minjune Hwang, Manasi Sharma, Arman Aydin, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Silvio Savarese, Hyowon Gweon, Karen Liu, Jiajun Wu, and Li Fei-Fei. BEHAVIOR-1K: A Benchmark for Embodied AI with 1,000 Everyday Activities and Realistic Simulation. In *CoRL*, 2023.

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified Video Action Model. In *RSS*, 2025.

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

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. In *NeurIPS*, 2023a.

Zeyi Liu, Arpit Bahety, and Shuran Song. REFLECT: Summarizing Robot Experiences for Failure Explanation and Correction. In *CoRL*, 2023b.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative Refinement with Self-Feedback. In *NeurIPS*, 2023.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What Matters in Learning from Offline Human Demonstrations for Robot Manipulation. In *CoRL*, 2022.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. CALVIN: A Benchmark for Language-Conditioned Policy Learning for Long-Horizon Robot Manipulation Tasks. *RAL*, 7(3): 7327–7334, 2022. doi: 10.1109/lra.2022.3180108.

Aoran Mei, Guo-Niu Zhu, Huaxiang Zhang, and Zhongxue Gan. ReplanVLM: Replanning Robotic Tasks With Visual Language Models. *RAL*, 9(11):10201–10208, 2024. doi: 10.1109/LRA.2024.3471457. URL https://doi.org/10.1109/LRA.2024.3471457.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. RoboCasa365: A Large-Scale Simulation Framework for Training and Benchmarking Generalist Robots. In *ICLR*, 2026.

Zaruhi Navasardyan, Tatul Danielyan, and Hrant Davtyan. FailBench: How Reliable are VLMs at Judging Robot Task Success? *arXiv preprint arXiv:2609.03611*, 2026. URL https://arxiv.org/abs/2609.03611v1.

Paul Pacaud, Ricardo Garcia, Shizhe Chen, and Cordelia Schmid. Guardian: Detecting Robotic Manipulation Failures with Vision-Language Models. *arXiv preprint arXiv:2512.01946*, 2026. URL https://arxiv.org/abs/2512.01946v4. Version 4; accepted at WACV 2027.

Shreyas Sundara Raman, Vanya Cohen, Ifrah Idrees, Eric Rosen, Ray Mooney, Stefanie Tellex, and David Paulius. CAPE: Corrective Actions from Precondition Errors using Large Language Models. In *ICRA*, 2024.

Allen Z. Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, Zhenjia Xu, Dorsa Sadigh, Andy Zeng, and Anirudha Majumdar. Robots That Ask For Help: Uncertainty Alignment for Large Language Model Planners. In *CoRL*, 2023.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011.

Md Sadman Sakib and Yu Sun. STAR: A Foundation Model-Driven Framework for Robust Task Planning and Failure Recovery in Robotic Systems. *IJAIRR*, 2:2550007, 2025. doi: 10.1142/S2972335325500073.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. In *NeurIPS*, 2023.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. ProgPrompt: program generation for situated robot task planning using large language models. *Autonomous Robots*, 47(8):999–1012, 2023. doi: 10.1007/s10514-023-10135-3.

Marta Skreta, Zihan Zhou, Jia Lin Yuan, Kourosh Darvish, Alán Aspuru-Guzik, and Animesh Garg. RePLan: Robotic Replanning with Perception and Language Models. *arXiv preprint arXiv:2401.04157*, 2024. URL https://arxiv.org/abs/2401.04157v2.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. LLM-Planner: Few-Shot Grounded Planning for Embodied Agents with Large Language Models. In *ICCV*, 2023.

Xiaohao Sun, Hanxiao Jiang, Manolis Savva, and Angel X. Chang. OPDMulti: Openable part detection for multiple objects. In *3DV*, 2024.

Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E. Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. Recovery RL: Safe Reinforcement Learning With Learned Recovery Zones. *RAL*, 6(3):4915–4922, 2021. doi: 10.1109/lra.2021.3070252.

Dayu Xia, Yue Shi, Yao Mu, Huiting Ji, Chaofan Ma, Yingjie Zhou, Hua Chen, Yang Liu, Jiezhang Cao, and Guangtao Zhai. RoboProcessBench: Benchmarking Process-Aware Understanding in Vision-Language Robotic Manipulation. *arXiv preprint arXiv:2606.13040*, 2026. URL https://arxiv.org/abs/2606.13040v3.

Chen Xu, Tony Khuong Nguyen, Emma Dixon, Christopher Rodriguez, Patrick Miller, Robert Lee, Paarth Shah, Rares Andrei Ambrus, Haruki Nishimura, and Masha Itkina. Can We Detect Failures Without Failure Data? Uncertainty-Aware Runtime Failure Detection for Imitation Learning Policies. In *RSS*, 2025.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *RSS*, 2023.

# A Discussion and Limitations

**Detection and repair are different measurements.** Selective warning does not translate into a composite-success gain from reactive edits, scheduled consultation, planner-visible belief, or scripted retry. Initial review restores three ordering defects and raises success on two, separating a correctable plan error from a skill failure that persists under a correct instruction.

**Directions grounded in the measurements.** Recovery training must address failure states, since contact stalls convert more often than grasp stalls and second attempts have little return. Diagnosis must connect failed preconditions to executable corrections, since an applicable instruction alone does not establish competence. Clear goal context increases repairing-action choice without recovery, separating selection from execution. The simulated stall signal provides a measured hand-off basis at recall 0.66 and false-stop rate 0.07 (Figure 3).

**Scope.** The study covers one executor, a fixed edit interface, two planners, and seven tasks, without comparing the published systems in Related Work. GT-pose configurations are diagnostic controls. Perception also changes wake families and completion. Near-zero competence on three tasks limits power, and corrected nulls leave smaller effects unresolved. Retry tests one scripted primitive, not learned recovery. Plan-defect cells have ten episodes, with significant benefits confined to ordering. RoboCasa365 simulation supplies rendered depth and frozen detection, without measuring physical robot perception. We do not ablate object slots against learned latents or other beliefs, which could serve another executor better. Changing monitor representation alone supplies no new motor skill.

# B Executor training and composite-segment exposure

**Training recipe.** The executor is $\pi_{0.5}$, initialized from the RoboCasa365 multitask pretraining checkpoint at step 75,000, the checkpoint behind the benchmark's published $\pi_{0.5}$ entry (Nasiriany et al., 2026). We fine-tune both transformers with low-rank adapters, keeping the base weights of the Gemma-2B vision–language backbone and the 300M-parameter action expert frozen. The image encoder and the action, state and time projection layers are updated in full. The fine-tuning corpus contains 8,235 composite-segment episodes, comprising 2,010,224 frames at 20 Hz and 49 distinct instructions. We first set aside a seeded 10% of each task's demonstrations for the held-out split, then cut the composite demonstrations at subgoal boundaries. The corpus covers six composite tasks. SteamInMicrowave contributes 1,840 segments, PackIdenticalLunches 1,802, LoadDishwasher 1,353, WashLettuce 1,346, StackBowlsCabinet 968 and PrepareCoffee 926. Four of these are evaluation tasks. GetToastedBread, StirVegetables and StoreLeftoversInBowl are absent. Neither their segments nor their segment instructions occur in the corpus. Table 3 quotes the optimization settings from the training configuration. We serve the final checkpoint of one uninterrupted 30,000-step run, numbered 0–29,999, initialized from the pretrained weights without resuming an earlier run. The policy predicts chunks of 50 actions. The acceptance check below and the main study execute the first 5 actions before querying the policy again. The preliminary study executes the first 8.

Table 3: Executor fine-tuning settings, quoted from the training configuration.

| Setting | Value |
|---|---|
| Training steps | 30,000 |
| Batch size | 24 |
| Learning-rate schedule | Cosine decay from $10^{-4}$ to $10^{-5}$ over 30,000 steps |
| Warmup | 1,000 steps |
| Optimizer | AdamW, gradient-norm clipping at 1.0, no weight averaging |
| Action horizon | 50, the model default without an override |

**Executor acceptance check.** Before any configuration was evaluated, the fine-tuned executor had to pass a pre-committed acceptance check on the nineteen subgoals of its six training tasks. For each subgoal, the simulator is restored to the start state of a held-out demonstration segment, and the executor receives that subgoal's frozen instruction verbatim. An episode succeeds if the subgoal's predicate holds, after debouncing, before a timeout of 1.5 times the 95th-percentile demonstration duration. Each subgoal has 20 starts, except StackBowlsCabinet's second bowl placement, whose

held-out split contains 16. All three executors are evaluated on the same 376 starts. The segment-trained executor reaches a mean of 68.6% over the nineteen subgoals (pooled 257/376 = 68.4%, Wilson 95% CI [63.5, 72.8]). The official pretrained checkpoint reaches 19.6% (pooled 73/376 = 19.4%, [15.7, 23.7]), and an atomic-trained executor reaches 18.7% (pooled 69/376 = 18.4%, [14.8, 22.6]). The atomic-trained executor starts from the same checkpoint and uses the same adapter recipe as the segment-trained one, but is fine-tuned on 6,608 episodes of 13 single-skill (atomic) tasks. The acceptance criterion required the mean over subgoals to exceed the better reference and the Wilson intervals to be non-overlapping. Those intervals are computed on the pooled rates.

As a consistency check on the evaluation setup, we also ran the official pretrained checkpoint on six whole composite tasks, with 50 episodes each. It received the whole-task instruction and was scored by the benchmark's success check. It completed 22/300 episodes (7.3%, Wilson 95% CI [4.9, 10.9]) in the benchmark's target kitchens under the horizons of release 1.0.1. The benchmark's published figure for the same checkpoint is 7.1% on its 16-task Composite-Seen split, measured in the pretraining kitchens under release 1.0.0. The protocols differ in kitchens, horizons and task set. Their agreement is therefore a consistency check rather than a reproduction.

**Per-segment comparison.** We compared the three executors segment by segment in the preliminary study, using logs from the same runner with 50 episodes per task and executor. A segment counts as completed only when its predicate advances the plan, rather than its time budget expiring. Each later segment is scored over episodes in which the immediately preceding segment completed by its predicate, regardless of how earlier segments ended. An earlier segment may have timed out.

- The atomic-trained executor fails in two different ways. On five of the seven tasks, first-segment completion from the initial state is low, at 3/50 for LoadDishwasher, 2/50 for StoreLeftoversInBowl, 1/50 for StackBowlsCabinet, 8/50 for PackIdenticalLunches and 6/50 for GetToastedBread. On the other two tasks, it completes the first segment more often but rarely completes the next. SteamInMicrowave reaches 33/50 on the first segment, then 1/33 when placing the bowl in the microwave. StirVegetables reaches 18/50, then 0/18 on the second vegetable.

- Its training instructions overlap little with the evaluation chains. Of the 23 segment instructions across the seven tasks, 3 exactly match an atomic training instruction. These are pressing the microwave's start button and StackBowlsCabinet's two bowl placements. Another 16 match partially, using the same primitive with an unseen destination or fixture, and 4 have no atomic counterpart.

- StackBowlsCabinet's second bowl placement is an exact instruction match. Conditional on the first placement completing, the atomic-trained executor completes it in 2/13 episodes, the official pretrained checkpoint in 27/32 and the segment-trained executor in 46/46. The earlier stacking segment timed out in 12 of those 13 atomic-trained episodes, 29 of the 32 pretrained episodes and 3 of the 46 segment-trained episodes. The rates therefore describe different starting states. With only 13 eligible atomic-trained episodes, we report this as an observation rather than a finding.

- On the three tasks absent from its corpus, the segment-trained executor completes the first segment less often than the official pretrained checkpoint. The respective counts are 3/50 against 30/50 for GetToastedBread, 24/50 against 29/50 for StirVegetables, and 10/50 against 40/50 for StoreLeftoversInBowl.

The comparison does not isolate a mechanism. The training data differ in several confounded ways. These include handoff states unseen in atomic training, instruction mismatch, the scene and state coverage of the atomic tasks, and possible narrowing through fine-tuning.

**Separate executor-training comparisons.** Table 1 includes two separate training comparisons. The official pretrained checkpoint completes 6/350 episodes, with 1/50 in SteamInMicrowave and 5/50 in StirVegetables. Every other task has 0/50. The atomic-trained executor completes 0/350, with 0/50 on every task. Both used the preliminary study's Executor only protocol, running the same static chains and budgets. Their scenes were drawn through sequential environment resets, so each scene depends on the preceding episodes. The main study instead seeds each scene from its task and episode index. The two training comparisons therefore use independent scene draws, are not paired with the main study's configurations, and enter no significance test.

# C PROGRESS METRIC AND LOGGING CORRECTION

This appendix concerns the preliminary study and changes no composite-success count.

**Definition.** The second pre-registration defines route-agnostic end-state progress. For a task chain with $K$ subgoal predicates, end-state progress is the fraction true at the episode's final tick. It does not depend on the order in which subgoals were reached or on how the plan advanced. $K$ is 4 for SteamInMicrowave and PackIdenticalLunches, 3 for LoadDishwasher, StackBowlsCabinet, StirVegetables and StoreLeftoversInBowl, and 2 for GetToastedBread. The toaster's lever-up predicate is excluded because it held before lever-down in all 200 episodes checked. The secondary metric, maximum progress, is the largest fraction of simultaneously true predicates during the episode. It is descriptive and untested. The pre-registered family compares four configurations with Executor only on macro end-state progress using a paired permutation test and Holm correction. Per-task contrasts are exploratory and use Benjamini–Hochberg correction. Episodes are paired by task and episode index. Because the preliminary study drew scenes by sequential resets, paired episodes generally do not share a scene. The analysis script uses a two-sided sign-flip permutation test with 20,000 resamples and percentile bootstrap intervals from 5,000 resamples. These implementation details were not specified in the pre-registration.

**The first analysis and its logging asymmetry.** The first pre-registered analysis instead used latch progress. It counts subgoals advanced because their predicate held, as recorded in each episode's advance list, and divides by $K$. It excludes timeout advances and planner-inserted subgoals. The preliminary study's configurations were run by two generations of the episode runner. The earlier runner executed Executor only and Any monitor (perception). The later runner executed Clock (GT poses), Any monitor (GT poses), Stall only (GT poses) and Clock, belief withheld. The two runners log the same event differently. When a pending subgoal's predicate becomes true, the later runner marks it as latched. The plan pointer then skips it without activating it or recording it in the advance list. The earlier runner checks only the active subgoal's predicate. If the predicate already holds when the subgoal becomes active, the next tick advances and logs it.

The first analysis therefore compared later-runner configurations with an earlier-runner baseline. It read latch progress of 0.435, 0.420, 0.427 and 0.458 for Clock (GT poses), Any monitor (GT poses), Stall only (GT poses) and Clock, belief withheld, against 0.485 for Executor only. These means use each configuration's logged episodes. Executor only has 349 because its log lacks StoreLeftoversInBowl episode 49. The pre-registered paired differences are computed within each task over matched episodes, then averaged over the seven tasks. They are $-0.048$, $-0.064$, $-0.059$ and $-0.026$. Rerunning Executor only on the later runner exposed the asymmetry. Its latch progress fell to 0.425, although a pre-registered consistency check had predicted about 0.047 above the original. In the rerun, 66 subgoals were true at episode end but absent from the advance lists. LoadDishwasher contributed 17, PackIdenticalLunches 40, StirVegetables 5 and StoreLeftoversInBowl 4. Crediting them raises the rerun's latch progress to 0.478, within 0.007 of the original 0.485. On the same runner, the four configurations' latch-progress differences from Executor only are $+0.010$, $-0.005$, $+0.002$ and $+0.033$, so the deficits of the first analysis came from the comparison across runners. On the same runner, replacing latch progress with end-state progress shifts the differences by a further $+0.016$, $+0.007$, $+0.016$ and $+0.004$. This remaining discrepancy attributable to the advance-list metric is at most 0.02.

**The corrected pre-registered analysis.** The second pre-registration replaced latch progress with end-state progress and compared the four later-runner configurations with Executor only rerun on the later runner. The rerun used the same seed sequence and executor checkpoint across seven tasks with 50 episodes each. It had no monitors or planner, and every episode recorded zero wakes. The analysis was run once, on 9 September 2026. The rerun completed 57/350 episodes, against 58/350 originally. The per-task counts were 17, 24, 8, 8, 0, 0 and 0, against 13, 25, 12, 8, 0, 0 and 0. The observed count falls between the materiality thresholds applied by the analysis, 45 and 72 successes, whose provenance is detailed below. No per-task difference survived Holm correction, with all adjusted $p = 1.000$. Table 4 gives the result. None of the four progress differences survives Holm correction. Clock, belief withheld has the smallest raw $p$, at 0.056, or 0.224 after correction. None of the 28 per-task comparisons (four configurations $\times$ seven tasks) is significant after Benjamini–Hochberg correction. The results therefore follow the pre-registered null interpretation, with no measurable progress difference in either direction after correcting the logging artifacts.

Table 4: Corrected pre-registered progress analysis in the preliminary study, with 350 episodes per configuration and one analysis. Differences are computed within task over matched episodes against Executor only rerun on the later runner, then averaged across tasks. Brackets give 95% bootstrap intervals. Two-sided sign-flip permutation tests use Holm correction across the four comparisons.

| Configuration | End-state progress | Difference [95% CI] | Raw $p$ | Holm $p$ |
|---|---|---|---|---|
| Executor only (rerun) | 0.431 | — | — | — |
| Clock (GT poses) | 0.457 | $+0.026$ $[-0.014, +0.066]$ | 0.206 | 0.617 |
| Any monitor (GT poses) | 0.434 | $+0.002$ $[-0.036, +0.041]$ | 0.911 | 0.911 |
| Stall only (GT poses) | 0.450 | $+0.018$ $[-0.021, +0.058]$ | 0.372 | 0.745 |
| Clock, belief withheld | 0.469 | $+0.037$ $[-0.000, +0.075]$ | 0.056 | 0.224 |

Any monitor (perception) is not in the family. It ran on the earlier runner, whose per-tick record stores only the predicates of the active subgoal and of those before it, so whether later subgoals held at episode end was never observed and its end-state progress cannot be computed from its records. An additional rerun on the later runner was considered and not undertaken. Its first-analysis latch progress (0.170) is not comparable either, because that configuration advances on belief-evaluated predicates rather than ground-truth ones.

## D ARTICULATED-FIXTURE STATE ESTIMATION

**Estimator.** The estimator tracks how far an articulated fixture is open from RGB images. It processes a fridge, cabinet, dishwasher or microwave door in three stages.

1. *Part localization.* The pretrained articulated-part detector and segmenter released with OPD-Multi (Sun et al., 2024), using its RGB motion-state checkpoint, runs on two external cameras and the wrist camera every 10 ticks. At each observation tick, we retain the highest-scoring part instance across cameras. This stage only localizes and identifies the part. The detector's built-in joint-state estimate serves as a baseline below and is not used by the estimator.
2. *Readout.* Closed-form ridge regression with $\lambda = 1$ maps 261 standardized features to the normalized opening $q$. The features are the part's 256-dimensional decoder embedding, four box coordinates and detection score, standardized on the training split. The opening is the joint position divided by its range, with 0 closed and 1 fully open. The segmentation mask is not an input.
3. *Filter.* Each joint has a scalar Kalman filter with a constant-position model. Its variance starts at 0.25 at the first reading. Each observation step adds process noise $0.001 + 0.01\, m_t$, where $m_t = \min(|\Delta\hat{q}|/0.01, 1)$ measures recent motion. The measurement variance is $\sigma_0^2$. Here $\sigma_0$ is the standard deviation of training residuals, at 0.186 for the fridge, 0.381 for the cabinet, 0.212 for the dishwasher and 0.266 for the microwave. The filter discards a reading if its innovation exceeds twice the predicted innovation standard deviation.

The selected configuration combines this ridge readout and filter. A $4 \times 4 \times 4$ grid search on 25 fridge development episodes selected the filter constants to maximize progress-event F1, defined below. F1 was 0.568, against 0.438 with the initially specified constants. The selected constants were then frozen for all four fixtures. For each fixture, only the standardization, the ridge weights and $\sigma_0$ were refit on its training episodes. A learned-residual readout variant reached development F1 of 0.610, below the pre-committed requirement of 0.618, and was not selected. These episodes were collected separately from the evaluation episodes and contain scripted and executor-driven openings and closings. The fridge has 36 training and 18 held-out episodes. The cabinet, dishwasher and microwave each have 48 training and 38 held-out episodes.

**Pre-registered criteria.** Four descriptive measures characterize state estimation. The acceptance criteria below additionally set thresholds for crossing timing and uncertainty calibration.

- *State error* is the mean absolute error in normalized opening over observation ticks, averaged within each episode and then across held-out episodes. We report the mean $\pm$ 1.96 SD$/\sqrt{n}$.

- *Crossing timing* is the lead or lag between estimated and true crossings of half-open, $q = 0.5$, for each opening or closing motion.
- *Error by visibility* separates state error when the fixture is detected, visible but not detected, or occluded. Occlusions are further divided by robot contact. This measure was not computed for the selected configuration.
- *Calibration* describes the estimator's stated uncertainty and is reported through the coverage criterion below.

Four measures have pre-registered acceptance thresholds.

- *Progress-event F1* must reach 0.75. A progress event occurs when the running maximum of $|q(t) - q(t_0)|$ enters a new 0.01 bin at an observation step. Estimated events are matched one-to-one to true events within $\pm 1$ observation step, or $\pm 10$ ticks. F1 is computed per held-out episode and averaged. This uses the same running-maximum progress signal as the stall monitor, although its reset rule differs in detail.
- *Motion-onset F1* must reach 0.70. It matches rising edges of an indicator for changes in $q$ of at least 0.01 per step against true edges within $\pm 5$ ticks.
- *Crossing lag* is the median absolute lag at the half-open crossing. It must be at most 20 ticks for scripted motions and 40 ticks for executor-driven motions.
- *Uncertainty coverage* measures the share of ticks within each error bound, averaged across episodes. The share with $|\hat{q} - q| \leq \sigma_0$ must be between 0.55 and 0.80. The share with $|\hat{q} - q| \leq 2\sigma_0$ must be at least 0.90.

Before computing any replay results, three further acceptance thresholds were set for the stall monitor that consumes the estimator's output. We replay the monitor offline on each held-out episode twice, feeding it true joint state in one replay and estimated state in the other. Its clock fires after 300 ticks without progress. Each new 0.01 advance of the running maximum resets the clock, as does a fire. These episodes contain no subgoal completions. The fires from the true-state feed are the reference.

- *Event recall*, pre-registered as sensitivity, must reach 0.8. It is the share of true-state fires with at least one estimate-fed fire within $\pm 150$ ticks.
- *Event precision*, pre-registered as specificity, must reach 0.9. It is the share of estimate-fed fires with at least one true-state fire within $\pm 150$ ticks.
- *Onset* must be at most 150 ticks, half the stall window. It is the median distance to the nearest estimate-fed fire, computed over matched true-state fires.

Matching is evaluated separately in each direction. One fire can match several, so the two matched counts can differ. A fixture could be connected to an evaluated configuration only if it met all three monitor criteria. Otherwise, the pre-registration allowed no deployment, retuning or second fit.

**Per-fixture results.** Table 5 gives the results. No fixture reached the progress-event F1 threshold, and only the fridge met both crossing-lag thresholds. The cabinet and microwave met both coverage criteria. The cabinet's large training-residual standard deviation, $\sigma_0 = 0.381$, makes those criteria easier to satisfy. The detector's built-in joint-state estimate is scored without filtering on the same held-out episodes and detected part instances. It reaches progress-event F1 of 0.212 on the fridge and 0.029 on the cabinet. Normalizing by the detector's predicted maximum rather than the joint range gives 0.205 and 0.032. The estimator's F1 is at least 2.5 times the baseline's on the fridge and 4.5 times on the cabinet. The baseline's state error is 0.567 on the fridge and 0.453 on the cabinet, or 0.554 and 0.454 under the alternative normalization. These are about 2.7–2.8 times the estimator's error on the fridge and 1.3 times on the cabinet. Only the estimator's output is filtered. The baseline is scored raw.

**Timing model and exclusion.** Under the original clock, both fridge and cabinet event recall was 0.741, despite state errors of 0.205 and 0.346. On both fixtures, most missed reference fires had an estimator fire 150–300 ticks away. Noise-induced progress resets suppressed most of the remaining fires. Both effects arise when the clock consumes a noisier signal and are distinct from state accuracy. We then froze four alternative stall clocks before scoring them, and evaluated each once on the

Table 5: Articulated-fixture estimation on held-out episodes, analyzed once per fixture. Parentheses give pre-registered thresholds and bands. State error and progress-event F1 are means with 95% half-widths across episodes. Crossing lags and onset are medians in ticks. Event recall and precision give the fractions and counts from stall-monitor replay. The original clock was replayed only on the fridge and cabinet. The soft clock was selected subsequently and replayed on all four fixtures. Onset is the same under both clocks for the fridge and cabinet. Motion-onset F1 was not computed for the other three fixtures. [a]The pre-registered $\pm 5$-tick window permits only same-step matches on the 10-tick observation grid. The value originally recorded, 0.221, used a $\pm 10$-tick window. [b]Three executor-driven motions. [c]Openings only. The held-out set contains two scripted closings.

| | Fridge | Cabinet | Dishwasher | Microwave |
|---|---|---|---|---|
| Held-out episodes | 18 | 38 | 38 | 38 |
| State error | $0.205 \pm 0.045$ | $0.346 \pm 0.013$ | $0.184 \pm 0.023$ | $0.206 \pm 0.015$ |
| Progress-event F1 ($\geq$0.75) | $0.533 \pm 0.069$ | $0.145 \pm 0.041$ | $0.361 \pm 0.060$ | $0.209 \pm 0.042$ |
| Motion-onset F1 ($\geq$0.70) | 0.126[a] | not computed | not computed | not computed |
| Crossing lag, scripted ($\leq$20) | 20 | 70 | 80[c] | 100 |
| Crossing lag, executor ($\leq$40) | 10[b] | 370 | 10 | 30 |
| Coverage $\leq \sigma_0$ (0.55–0.80) | 0.608 | 0.579 | 0.683 | 0.703 |
| Coverage $\leq 2\sigma_0$ ($\geq$0.90) | 0.853 | 0.985 | 0.885 | 0.940 |
| *Stall-monitor replay, original clock* | | | | |
| Event recall ($\geq$0.8) | 0.741 (63/85) | 0.741 (374/505) | — | — |
| Event precision ($\geq$0.9) | 0.926 (63/68) | 0.984 (366/372) | — | — |
| *Stall-monitor replay, soft clock* | | | | |
| Event recall ($\geq$0.8) | 0.816 (71/87) | 0.832 (421/506) | 0.730 (54/74) | 0.764 (123/161) |
| Event precision ($\geq$0.9) | 0.959 (70/73) | 0.979 (418/427) | 0.898 (53/59) | 0.938 (120/128) |
| Onset, median ($\leq$150) | 100 | 70 | 110 | 100 |

fridge and cabinet. Three met all three monitor criteria on both fixtures. We selected the uncertainty-weighted soft clock, which uses the estimator's own confidence and adds no per-fixture constant. Each advance $\delta$ of the running maximum adds $c_t\,\delta$ to an accumulator. Here $c_t = \sigma_0^2/(P_t + \sigma_0^2)$ is the filter's confidence and $P_t$ its posterior variance, with $c_t = 1$ for the true-state feed. The clock resets when the accumulator reaches 0.05. The soft clock met all three criteria on the fixtures used to select it, with event recall of 0.816 on the fridge and 0.832 on the cabinet. The fridge and the cabinet had already failed their pre-registered replays under the original clock and stayed unconnected. The wiring pre-registration that followed covered only the dishwasher and the microwave, for LoadDishwasher and SteamInMicrowave, with the soft clock frozen. On the held-out fixtures, its single evaluation gave event recall of 0.730 for the dishwasher and 0.764 for the microwave, both below 0.8. Dishwasher event precision was 0.898, also below its threshold. Following the pre-registration, the estimator is connected to no evaluated configuration, without retuning or a second fit. Any monitor (perception) runs without the stall and unintended-motion monitors, which require fixture state. Any monitor (perception, calibrated) adds a stall clock based on segment timing and completion rather than on the fixture estimator.

## E Belief Definitions, Constants, and Calibration

Object $i$ has position $\hat{\mathbf{x}}_{i,t}$, covariance $\sigma_{i,t}^2 I$, observation age $u_{i,t}$, and grasp log-odds $\ell_{i,t}$, with $g_{i,t} = \mathbf{1}[\ell_{i,t} > 0]$. Given hand position $\mathbf{p}_t$, prediction and typed uncertainty are

$$\hat{\mathbf{x}}_{i,t+1}^{-} = \hat{\mathbf{x}}_{i,t} + g_{i,t}(\mathbf{p}_{t+1} - \mathbf{p}_t). \tag{1}$$

$$\sigma_{i,t+1}^{2,-} = \sigma_{i,t}^2 + q_g g_{i,t} + (q_0 + q_d d_{i,t})(1 - g_{i,t}). \tag{2}$$

For detection $\mathbf{z}_k$, write $\mathbf{e}_{ik,t} = \mathbf{z}_k - \hat{\mathbf{x}}_{i,t}^{-}$ and $S_{i,t} = \sigma_{i,t}^{2,-} + r^2$. The nearest same-class detection inside the gate is updated by

$$\begin{aligned} \delta_{ik,t}^2 = \|\mathbf{e}_{ik,t}\|^2/S_{i,t} < \gamma, \quad K_{i,t} = \sigma_{i,t}^{2,-}/S_{i,t}, \\ \hat{\mathbf{x}}_{i,t} = \hat{\mathbf{x}}_{i,t}^{-} + K_{i,t}\mathbf{e}_{ik,t}, \qquad \sigma_{i,t}^2 = (1 - K_{i,t})\sigma_{i,t}^{2,-}. \end{aligned} \tag{3}$$

Grasp evidence and calibration on re-observations $\mathcal{D}_{\text{cal}}$ are

$$\ell_{i,t+1} = \text{clip}\left(\ell_{i,t} + \beta_c c_{i,t} - \beta_o o_t + \beta_m \left(1 - \frac{\|\Delta \mathbf{z}_i - \Delta \mathbf{p}\|}{\tau_m}\right) h_{i,t}, -L, L\right). \tag{4}$$

$$\max_{\theta} \sum_{(i,k,t)\in\mathcal{D}_{\text{cal}}} \log \mathcal{N}\left(\mathbf{z}_k; \hat{\mathbf{x}}_{i,t}^{-}, S_{i,t} I\right). \tag{5}$$

We define the indicators, constants, fitting procedure, and coverage region below.

**Formal monitor definitions.** Each monitor tests a specified condition over a persistence window. A logged fire wakes the planner only if that family belongs to the configuration's wake set (Section 3.3). Thresholds and windows are fixed before evaluation (Appendix E).

**Absence.** Let $f_{i,t}$ indicate projection inside a camera image with margin, and $o_{i,t}^{\text{arm}}$ indicate known arm occlusion. If $\mathcal{F}_t$ is the union of closed fixture interiors, expected visibility is

$$v_{i,t} = f_{i,t}(1 - g_{i,t})(1 - o_{i,t}^{\text{arm}})\mathbf{1}[\hat{\mathbf{x}}_{i,t} \notin \mathcal{F}_t]. \tag{6}$$

Absence fires when $v_{i,t} = 1$ and $u_{i,t} \geq k_a$. Expected invisibility does not itself trigger consultation. Known arm occlusion suppresses the absence alarm. These tests depend on available visibility and fixture information, whose perceptual limitations are measured in Section 5.3.

**Re-observation mismatch.** A same-class detection satisfies $\delta_{ik,t}^2 \geq \gamma$, has no alternative gated track, and persists for $k_m$ frames.

**Grasp inconsistency.** The object is believed held, $g_{i,t} = 1$, but its motion differs from the hand's by $\|\Delta \mathbf{z}_i - \Delta \mathbf{p}\| > \tau_m$ over $k_g$ observed frames.

**Unintended motion or regression.** An object has a gated innovation whose distance exceeds $\tau_u$ while it is believed free ($\ell_{i,t} < -1$) and the hand is distant. Alternatively, a previously satisfied segment predicate becomes false for $k_r$ frames. These conditions detect unexpected motion and loss of achieved progress.

**Stall.** In simulation, the active segment has made no progress for $W_C = 300$ control steps (15 s) when no subgoal has latched and neither the segment's object nor its relevant fixture has moved beyond $\epsilon_d = 0.02$ m since the window began.

**State and indicators.** Each position is three-dimensional and each covariance is isotropic. Superscript $-$ denotes the prediction before observation. The hand position is $\mathbf{p}_t$, and $g_{i,t} = \mathbf{1}[\ell_{i,t} > 0]$ is the propagation state. The disturbance indicator $d_{i,t}$ is one when the hand approaches within $r_d$ of the object or a containing fixture is actuated during its observation gap. The increments $q_g$, $q_0$, and $q_d$ parameterize transport, quiet drift, and disturbance, with $q_d \gg q_0$. Position is set to $\mathbf{p}_{t+1}$ and variance to $\sigma_{\text{rel}}^2$ when an opening event changes the grasp state from held to free.

**Observations and association.** We obtain class labels and back-projected centroids $\mathbf{z}_k$ from frozen open-vocabulary segmentation and metric depth. The centroid covariance is $r^2 I$. Depth is rendered in simulation. The innovation variance is $S_{i,t} = \sigma_{i,t}^{2,-} + r^2$. Equation 3 uses the dimensionless squared innovation and selects the nearest eligible same-class detection. Association resets observation age to zero. Otherwise the prediction is retained and age increments by one. An out-of-gate same-class detection remains available to the mismatch monitor.

**Grasp evidence.** The indicators $c_{i,t}$ and $o_t$ denote closure near object $i$ and gripper opening. The indicator $h_{i,t}$ requires two object observations, with object and hand displacements measured over the same interval. Otherwise, the motion term is omitted. The weights are $\beta_c$, $\beta_o$, and $\beta_m$, the motion scale is $\tau_m$, and clipping is to $[-L, L]$. Values $-1 < \ell_{i,t} < 1$ are reported to the planner as ambiguous, while propagation uses the binary $g_{i,t}$.

**Scheduled allocation and review interpretation.** The clock's allocation was matched to the mean triggered call count, so the comparison varies timing at a matched allocation. Table 1 separately reports realized calls. Reviewing a plan before any dependency is violated tests intervention before irreversible consequences, but does not alone isolate the cause of failed reactive repair. The preliminary study's snapshot assignments are documented separately in Appendix G.

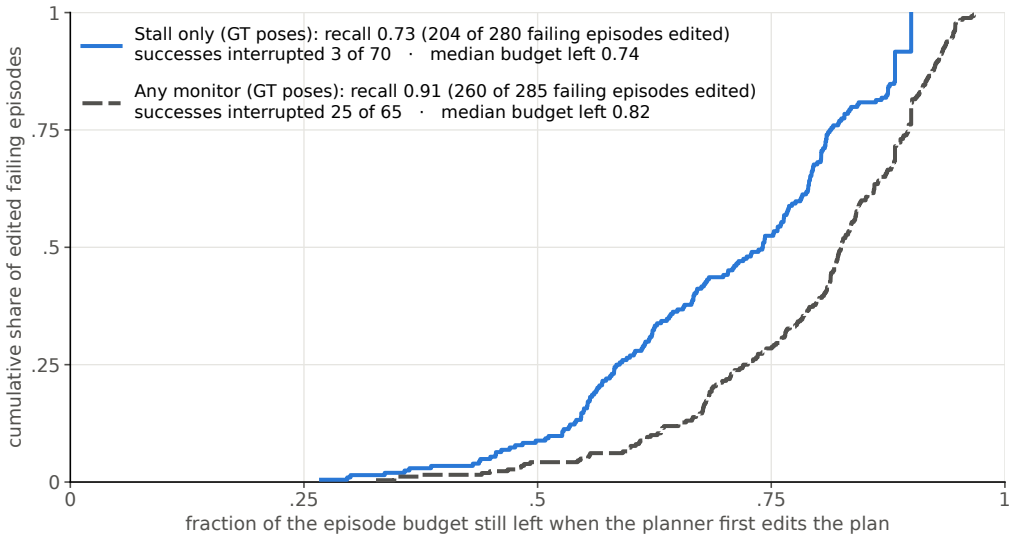


Figure 6: Budget remaining in the preliminary study at the first non-*continue* edit among episodes that later fail. The median fractions left unspent are 0.74 for Stall only (GT poses) and 0.82 for Any monitor (GT poses). Stopping then would also interrupt 3 of 207 and 25 of 285 edited episodes that succeed. The budget fractions describe timing and do not measure execution-time savings.

## F Retrospective Hand-Off at the First Edit

This analysis of the preliminary study uses the first non-*continue* planner edit as its hand-off point. It therefore measures a different decision from the first monitor wake used to define warning in Section 5.1.

A stopping rule at the first edit would flag 207 episodes in Stall only (GT poses). This configuration feeds the belief true pose while visible to a policy camera and dead-reckons it otherwise. Of these, 204 subsequently fail and three succeed. The rule covers 204 of 280 failures and would interrupt three of 70 successful episodes, leaving 67 successes uninterrupted by this rule. Among the flagged failures, the median episode budget remaining at the first edit is 74%. The 76 other failures draw no edit. For the Any monitor (GT poses) configuration, 285 episodes contain an edit, of which 260 fail and 25 succeed. This covers 260 of 285 failures and would interrupt 25 of 65 successes, with a median 82% of the budget remaining among flagged failures (Figure 6).

These are counterfactual outcomes of stopping at an edit, not outcomes of a deployed hand-off rule. Budget remaining is a retrospective timing quantity, and failed episodes can end before their budgets expire. The counts and budget fractions here must not be substituted for first-wake warning metrics.

## G Preliminary Study and Context Experiment

The preliminary study was pre-specified, with fifty episodes per task and independently drawn scenes across configurations. Its task/index alignment is not scene pairing. The planner received benchmark task names without explicit goals, and the prompt loader removed context headings. The context experiment restored those prompt fields within the same setup. The main study additionally used shared scenes. The two studies use different scene draws for the same seven tasks and are never pooled. The preliminary study's runner executes the first 8 actions of each 50-action chunk before querying the policy again, against 5 in the main study.

The preliminary study's Executor only outcomes were recovered from a run log after its per-episode files were lost in a documented output-path collision. The log lines printed a wrong configuration name, and the counts were confirmed against a separate record. The run provenance identifies the Executor only configuration.

Table 6: Preliminary-study outcomes, 350 episodes per configuration on independently drawn scenes. Costs are replies received. The perceptual configuration issued 2.34 wakes per episode but received 2.31 replies. Nine of 818 wakes failed after provider-overload retries and were treated as no action. All nine episodes failed. These results are from the preliminary study and are separate from the main study. The preliminary study's perceptual configuration had the same defect in applying the planner's edits as the main study's first perceptual run, so inserted and reordered steps were not applied.

| Configuration | Success/350 | Replies/episode |
|---|---|---|
| Executor only | 58 | 0 |
| Clock (GT poses) | 68 | 2.43 |
| Any monitor (GT poses) | 65 | 4.29 |
| Stall only (GT poses) | 70 | 1.82 |
| Clock, belief withheld | 72 | 2.33 |
| Any monitor (perception) | 62 | 2.31 |

The perceptual configuration woke in 349/350 episodes. On microwave it completed 3/50 episodes against 13/50 for Executor only. In the original scheduled comparison, withholding the belief changed aggregate success by four episodes, with task-level shifts of $+9$, $-5$, and $-3$. These task counts belong to the preliminary study and are not substituted for the main study's per-task outcomes.

In the held-out ground-truth-pose absence test, the monitor remained quiet in fifty episodes with an occluded but present object. Absence was logged but never woke the planner using ground-truth poses. In the original Any monitor (GT poses) configuration, 395 of 445 stall consultations identified a segment that ultimately failed to complete. These event-level checks differ from the first-wake operating points in the main text.

**Persistent stalls and plan edits in the preliminary study.** The preliminary study's failure analysis examines consultations in Any monitor (GT poses) in loading, steaming, and stirring. It selects 62 episodes in which a stall wake was followed by no further progress on the active segment. None completed. An instruction in the trained vocabulary addressed the encountered state in 61 cases, so an available instruction did not establish that the executor could perform it. Thirty of the 62 failures involved grasp acquisition. This selected subset characterizes persistent stalls, rather than estimating recovery over all natural failures. In the preliminary study's plan-defect cells, reactive consultation selected the repairing operator in 14 of 197 wakes (7%).

**Context experiment.** The context experiment uses the preliminary study's setup as its reference. Restoring explicit goals and context headings raises repairing-action selection from 7% to 23% (35 of 153 reactive wakes). Edited-episode recovery changes from 8% to 7% (7 of 103 in the restored-context condition). These recovery rates use a broader denominator than the 62 persistent-stall episodes. The planner more often chooses a relevant edit when the intended outcome is explicit, while successful recovery does not increase.

The preliminary study remains the reference for the context experiment because the main study also changes scene pairing.

## H Physical Retry and Execution Checks

At a stall, a ground-truth grasp-state check admits the primitive only when nothing is held. Before the main study, the primitive was validated to execute as follows. It records the arm's path in the mobile base frame, including the segment activation pose and contact force. On an eligible stall, a force-guided release from contact precedes reverse replay through previously visited waypoints, with loops removed. The arm returns relative to the base's current location, without resetting the scene or base. A verified return must be within 2 cm and 5° of the activation pose. A successful return grants one additional segment budget, resets the stall clock, and reissues the original instruction. Blocked attempts grant no additional budget and still count against the limits of two applications per segment and four per episode. Section 5.2 reports conversion and composite success.

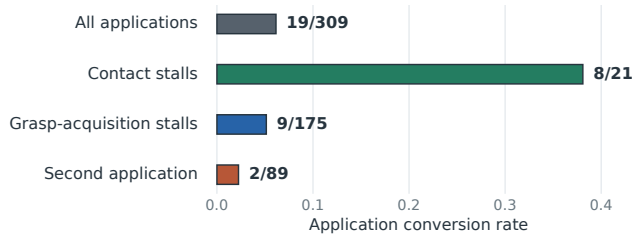


Figure 7: Segment conversion after scripted physical retry. Cause-specific rates distinguish contact stalls from grasp-acquisition stalls, while the second-application rate pools causes. The categories therefore are not a partition.

## I Implementation and Provenance Checks

The toaster's `lever_up` segment advanced vacuously in every episode of every configuration because a declared precondition was not applied at rollout time. Toast is interpreted only as an all-configurations-zero task, so this issue changes no composite success count.

The first run of Any monitor (perception) applied only the planner's rewrites, because a field-name mismatch in its runner dropped inserted, reordered, and rebound steps. It completed 67 of 350 episodes with 294 of 809 replies applied. It was rerun on the same scenes after correcting how the planner's edits were applied, and the rerun, 49 of 350, is the one reported.

The implementation audit checked `counter.py` against reported outcomes and traced declared `AFTER` invariants through rollout logic. It found that a declared invariant was not consumed during execution. Appendix G documents the preliminary study's log recovery and task/index alignment.

The response schema asks the planner whether the belief agrees with the images, and it answered yes in all sixty replay replies from contexts without the belief.

## J Tracking Accuracy as an Instrument Check

The check measures position and binding errors in the state supplied to the monitors. Predictions are evaluated against ground-truth position at 10,349 re-observation events from 2,301 held-out episodes, separate from the intervention comparisons. Table 7 reports the object-slot belief, its grasp-coupling ablation, and a constant-velocity Kalman tracker on the same observations. During gaps containing a grasp, their median errors are 0.059, 0.161, and 0.208 m, with binding rates 0.97, 0.55, and 0.49. In gaps without a grasp, the errors are 0.043, 0.043, and 0.058 m. Figure 8 reports these errors by gap duration.

Table 7: Median position error at re-observation and correct track binding, split by whether the object was grasped during the observation gap. All estimators use the same observations.

| Estimator | Error (m) | Binding |
|---|---|---|
| *Grasped during gap* ($n = 4{,}133$) | | |
| Constant-velocity Kalman | 0.208 | .49 |
| Belief without grasp coupling | 0.161 | .55 |
| Object-slot belief | 0.059 | .97 |
| *Not grasped* ($n = 6{,}216$) | | |
| Constant-velocity Kalman | 0.058 | .85 |
| Belief without grasp coupling | 0.043 | .96 |
| Object-slot belief | 0.043 | .96 |

A separate detection-only comparison uses a tracker that retains the last observation. Position error is 0.076 m for the object-slot belief and 0.114 m for that reference. Under long occlusion, the position

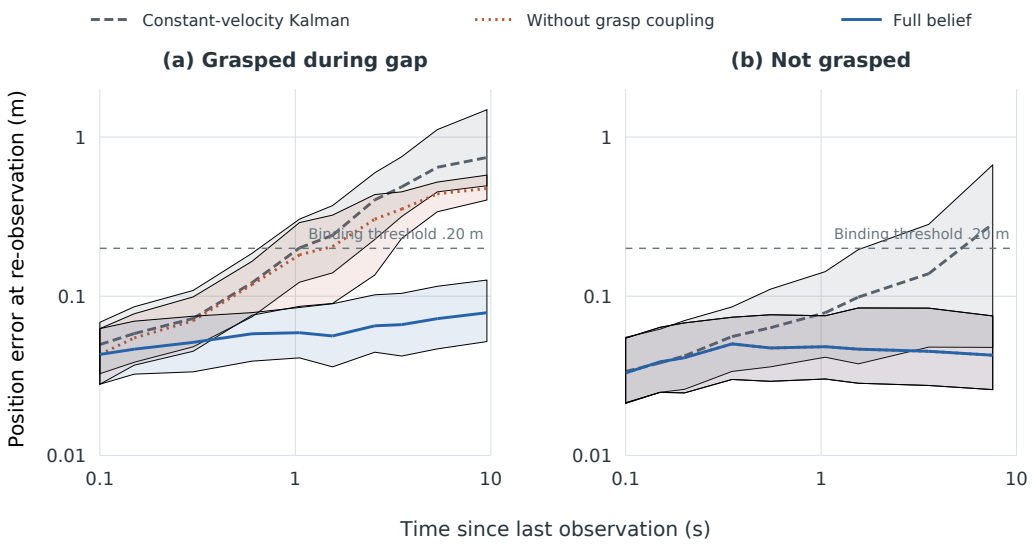


Figure 8: Instrument check on 10,349 held-out re-observation events. Lines show median position error and bands the interquartile range in gap-duration bins, split by whether the object was grasped. Axes are logarithmic. The dashed line is the 0.20 m binding threshold. The coupled and uncoupled variants coincide in gaps without a grasp.

errors are 0.170 and 0.353 m. Binding after long gaps is 0.85 versus 0.50. These aggregates and the stratified medians in Table 7 are separate summaries.

Fitted uncertainty constants yield a held-out predictive log-likelihood 9.7 higher than constants set from kinematics, with a reported interval of [2.1, 18.2]. Appendix E records the fitting and coverage definitions. These checks quantify the instrument's state-estimation error alongside the monitor checks in Section 5.1.

## K Second-Planner Ablation

We reran the plan-defect cells and the Stall only (GT poses) configuration with Claude Sonnet 5 in place of Claude Opus 5. The two models belong to different tiers. We held seeds, scenes, runners, action set, and goal-bearing context fixed. We also verified that all 402 second-planner calls were served by Sonnet 5. We pre-specified that little would change in repairing-action selection, omission handling, reactive repair, or composite success. We also predicted a similar editing mix. These comparisons are reported separately from the main study's Holm-corrected family.

Table 8: Plan-defect completion under two planners, successes out of ten. Executor only receives the defective chain. Review makes one call before execution and none later. For both order-defect cells, the models select the same canonical reorder in all ten review episodes and pass identical plans to the executor on the same scenes.

| Injected defect | Executor only | Opus 5 review | Sonnet 5 review |
|---|---|---|---|
| Dishwasher close before second dish | 0/10 | 7/10 | 7/10 |
| Microwave close before placement | 0/10 | 8/10 | 2/10 |
| Dishwasher omit second dish | 1/10 | 1/10 | 0/10 |

Both models restore the canonical order in the two order-defect cells, while neither repairs the omission (Table 8). Identical reviewed plans yield different microwave completion counts, eight of ten with the primary planner and two of ten with the second. The observed completion gap therefore does not reflect a difference in the plans selected by review.

Reactive consultation produces no repair under either model in this ablation. On the paired prefix of 105 scenes, Stall only (GT poses) completes 19 episodes with either planner, against 24 for Executor only. Relative to Executor only, the second-planner configuration changes nine failures to successes and fourteen successes to failures (two-sided exact sign test, $p = .40$). Section 5.2 reports the executor's repeat as the reference for apparent rescues and harms.

The second planner skips a pending step in 30% of its edits, compared with 10% for the primary, rewrites in 27% versus 45%, and inserts in 35% versus 24%. Its rewritten strings fall within the executor's training vocabulary in 17.3% of cases, compared with 2.3% for the primary. These editing frequencies differ despite equal observed composite-completion counts in the paired stall-only comparison. The pre-specified pattern holds for the plan choices, omission outcome, and reactive null. The editing mix departs from the prediction of similarity. The paired comparison with Executor only detects no success difference and does not establish equivalence.

## L Post-Stall Completion Intervals

Section 5.2 reports the registered comparison of segment resumption after a stall, together with its procedure, counts, tests, and exclusions. The corresponding completion-rate intervals are [11.1, 20.7]% for Executor only, [10.3, 20.4]% for Stall only (GT poses), and [9.3, 18.1]% for Any monitor (GT poses).

## M Extended Related Work

**Modifying the executor, rather than the layer above it.** A complementary direction changes the behavior available to the executor. Imitation learning has long studied errors that shift execution away from the demonstration distribution. DAgger collects supervision on learner-visited states (Ross et al., 2011), and DART elicits corrective behavior by perturbing demonstrations (Laskey et al., 2017). Relay Policy Learning combines hierarchical imitation with reinforcement learning (Gupta et al., 2020). Sequential Dexterity trains policies for feasible transitions between skills (Chen et al., 2023), and Recovery RL learns a separate policy for returning toward safety (Thananjeyan et al., 2021). These approaches change the behavior the executor learns. Action Chunking with Transformers and Diffusion Policy model action sequences directly (Zhao et al., 2023; Chi et al., 2023), while Unified Video Action Model jointly learns video and action prediction (Li et al., 2025). RB-VLA conditions a diffusion policy on a fixed-size recurrent belief trained with world-model objectives and a single initial vision-language intent query, inferring task phases from the belief's evolution rather than from a re-invoked planner (Bagaria et al., 2026). Its belief is validated by predictive consistency, and its authors propose integration into pretrained VLAs as a fine-tuning layer. These executor-modifying approaches are complementary to our analysis. They change learned control, whereas we measure run-time interventions above a fixed executor. Our failure localization points toward training the missing skill and recovery behavior, without evaluating these alternatives.

**Benchmarks and evaluation.** Manipulation benchmarks measure different forms of competence. RLBench and ManiSkill2 provide diverse task families (James et al., 2020; Gu et al., 2023), CALVIN evaluates language-conditioned skill sequences (Mees et al., 2022), and LIBERO studies knowledge transfer in lifelong manipulation (Liu et al., 2023a). BEHAVIOR-1K and RoboCasa365 extend evaluation to household activities (Li et al., 2023; Nasiriany et al., 2026). Studies of offline demonstrations also show that measured competence depends materially on data quality, policy design, and evaluation choices (Mandlekar et al., 2022).