FIT-MAC: FAILURE-AWARE INFERENCE-TIME MULTI-AGENT CONTROL FOR ROBUST VLA MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models have achieved strong performance on standard robotic manipulation benchmarks. However, their reliability can degrade substantially when task conditions deviate from the nominal setting. Perturbations in object positions and task instructions may lead to complete task failure. Existing VLA policies generally lack an explicit runtime mechanism for distinguishing these failure modes and selecting an appropriate recovery strategy. To systematically mitigate this issue, we propose FIT-MAC, a failure-aware inference-time multi-agent control framework for robust VLA manipulation. FIT-MAC augments a frozen VLA policy with an inference-time control layer. First, the Task Planner Agent decomposes the language instruction into an ordered sequence of manipulation subtasks, and the Grounder Agent resolves each instruction-level reference to a visual target. Next, the Monitor-Diagnosis Agent tracks phase-specific execution progress and distinguishes target-binding failures from local execution failures. Finally, FIT-MAC routes each detected failure to a corresponding recovery mechanism, target rebinding for target failures and bounded grasping or placement recovery for execution failures. When no failure is detected, the original VLA remains the default action generator. Extensive experiments on LIBERO-PRO demonstrate that FIT-MAC achieves state-of-the-art performance under diverse perturbations and consistently improves success rate across multiple VLA backbones. The largest gains occur under position, task, and environment perturbations, while high success rates are maintained under object and semantic perturbations. Code is available at `https://anonymous.4open.science/r/research-artifact-B431`.

1 INTRODUCTION

Vision-language-action (VLA) models integrate language understanding, visual perception, and action generation for general-purpose robotic manipulation. OpenVLA provides an open-source foundation for generalist robot control (Kim et al., 2024), while π_0 uses flow matching to generate continuous actions (Black et al., 2025). Its successor, $\pi_{0.5}$, further improves open-world generalization through heterogeneous co-training (Physical Intelligence et al., 2025). Recent models pursue complementary directions. NORA reduces the computational cost of generalist VLA policies (Hung et al., 2025), while X-VLA targets scalable cross-embodiment transfer (Zheng et al., 2025). MolmoAct incorporates structured spatial reasoning between perception and low-level control (Lee et al., 2025). Other methods improve predictive modeling and long-horizon execution through world-knowledge modeling, future-goal prediction, and phase-aware policies (Zhang et al., 2025b;c; Fan et al., 2025). These advances highlight the potential of VLAs as generalist robotic policies.

However, strong benchmark performance does not necessarily translate to reliable execution under distribution shift. AGNOSTOS reveals limited zero-shot generalization across tasks, while SAFE highlights frequent failures on novel tasks, underscoring the need for runtime failure detection (Zhou et al., 2025; Gu et al., 2025). Broader evaluations reveal substantial sensitivity to perturbations in actions, instructions, observations, and environments, including controlled changes to objects, their positions, instructions, and scene layouts (Guo et al., 2026; Zhou et al., 2026). VLATest reports similar robustness gaps under changes in clutter, lighting, viewpoints, and instructions, as well as when encountering unseen objects (Wang et al., 2025b). Adversarial evaluations reveal additional vulnerabilities to digital and physical perturbations (Wang et al., 2025a). Robust deployment therefore

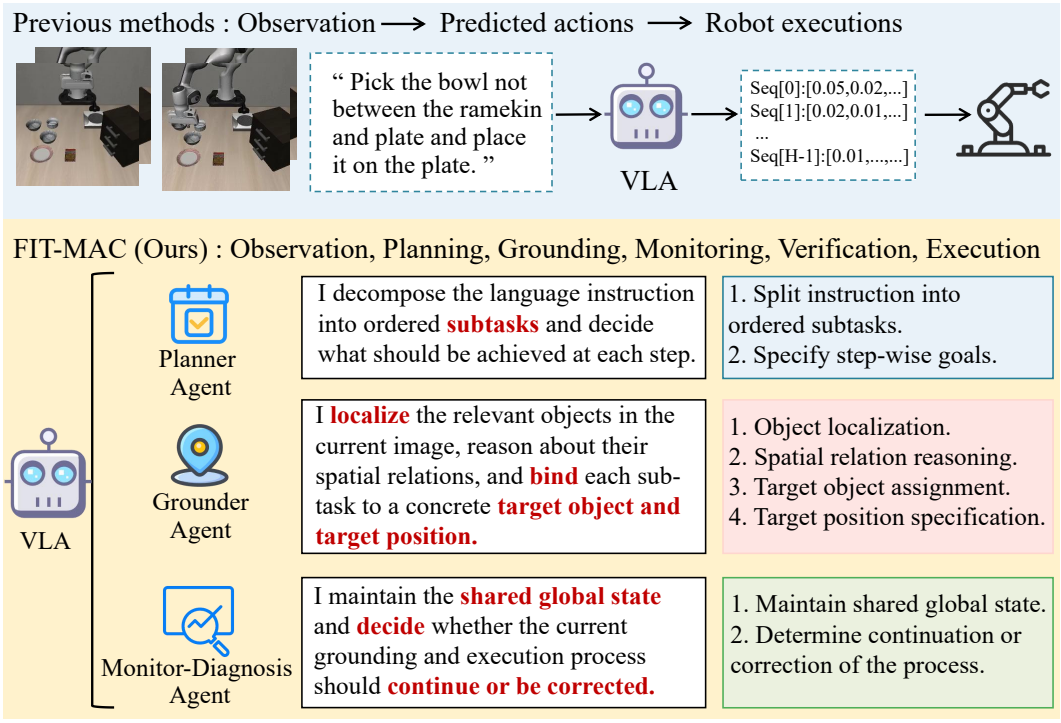

Figure 1: Comparison of standard VLA execution and FIT-MAC. Top: a VLA policy predicts actions from visual observations and a language instruction. Bottom: FIT-MAC augments the pretrained policy with specialized agents for task planner, instance-level grounding, and runtime monitoring and diagnosis.

requires VLAs to complete tasks under nominal conditions and to detect, diagnose, and recover from execution failures at inference time.

These challenges have motivated work on runtime recovery in robotic manipulation. RACER combines a VLM supervisor with an actor trained on automatically generated recovery trajectories (Dai et al., 2025). COME-robot monitors execution, identifies failure causes, and performs closed-loop recovery (Zhi et al., 2025). TCoT unifies hierarchical trajectory reasoning, failure detection, and recovery through global–local switching within a VLA framework (Li et al., 2026). These approaches highlight the value of execution monitoring and corrective control. We study a complementary runtime decision problem: distinguishing incorrect target binding from local execution failure and selectively applying the appropriate recovery mechanism. The pretrained VLA remains the default action generator.

As illustrated in Fig. 1, we investigate runtime recovery to improve pretrained VLA robustness without modifying the underlying policy. Our key observation is that perturbations lead to distinct failure types. The first is target failure: incorrect grounding of the language-specified target object or reference relation causes the policy to approach the wrong object instance before grasping. The second is execution failure: the target is correctly identified, but unstable local control during grasping, transport, or placement causes empty grasps, failed lifts, unstable releases, or stalled execution. Distinguishing these failure types enables an external controller to tailor recovery to each case rather than rely on generic retries.

Building on this observation, we propose FIT-MAC, a failure-aware inference-time multi-agent control framework. Specialized agents coordinate task parsing, visual grounding, monitoring, diagnosis, routing, and recovery through a shared execution state. FIT-MAC uses the pretrained VLA as the default action generator and intervenes only when runtime monitoring detects evidence of failure during a grasping or placement attempt. Intervention proceeds through monitoring, diagnosis, routing, and correction. Our method tracks the task stage and execution state, detects stagnation from stage progress, target visibility, and grasping or placement signals, and selects target rebinding or local recovery control accordingly.

This design offers two benefits. First, FIT-MAC distinguishes incorrect target grounding from local execution failure to guide recovery. Second, it intervenes only when observable evidence of failure emerges, limiting disruption to nominal rollouts. FIT-MAC thus preserves the pretrained VLA's general action-generation capability while adding selective, interpretable inference-time recovery tailored to each failure type.

Experiments on LIBERO-Object and LIBERO-Spatial show that FIT-MAC consistently improves the robustness of OpenVLA, π_0, and $\pi_{0.5}$ across diverse perturbations. Gains are largest under position, task, and environment perturbations, where successful execution often requires target rebinding or stage-wise recovery. FIT-MAC also maintains high success rates under object and semantic perturbations, extending its effectiveness across different perturbation types. Our contributions are as follows:

- We propose FIT-MAC, a failure-aware inference-time multi-agent control framework. Specialized agents coordinate closed-loop monitoring, diagnosis, routing, and correction through a shared execution state while retaining the pretrained VLA as the default action generator.
- We introduce attempt-level failure attribution that distinguishes target failures from execution failures using explicit, auditable triggering conditions.
- We design two complementary recovery pathways: target rebinding with closed-loop correction for target failures and bounded local grasping and placement control for execution failures.
- Extensive experiments on LIBERO-Object and LIBERO-Spatial demonstrate consistent robustness gains across OpenVLA, π_0, and $\pi_{0.5}$, particularly under position, task, and environment perturbations.

2 RELATED WORK

Generalist and Long-Horizon Vision-Language-Action Models. Generalist VLA models learn language-conditioned visuomotor policies from heterogeneous robotic data. OpenVLA provides an open-source autoregressive foundation model (Kim et al., 2024). π_0 uses flow matching for continuous action generation, while $\pi_{0.5}$ improves open-world generalization through heterogeneous co-training (Black et al., 2025; Physical Intelligence et al., 2025). NORA improves inference efficiency, X-VLA supports cross-embodiment transfer through soft prompts, and MolmoAct integrates depth-aware perception, spatial reasoning, and action prediction (Hung et al., 2025; Zheng et al., 2025; Lee et al., 2025).

Recent approaches address long-horizon manipulation through hierarchical reasoning, memory, future-state prediction, and spatial representations. AtomVLA decomposes demonstrations into atomic subtasks and scores action chunks using a latent world model (Sun et al., 2026). LaMP incorporates a 3D scene-flow prior, R3DP introduces a real-time 3D-aware policy, and RoboTALES uses task-aligned simulated futures to guide policy learning (Wang et al., 2026b; Zhang et al., 2026b; Gani et al., 2026). These methods modify the policy architecture, representation, or training procedure. FIT-MAC instead preserves the pretrained policy and performs diagnosis and recovery entirely at inference time.

Perturbation Evaluation and Robust Policy Optimization. Benchmarks such as LIBERO-PRO, LIBERO-X, VLATest, adversarial VLA benchmarks, and LIBERO-RECOVER assess VLA robustness to changes in objects, positions, language, observations, and environments (Zhou et al., 2026; Wang et al., 2026a; 2025b;a; Liu et al., 2026a). Existing methods improve robustness through perturbation-based training, post-training, uncertainty-aware action refinement, or latent world-model scoring (Guo et al., 2026; Zhang et al., 2025a; Islam et al., 2026; Sun et al., 2026). VLS instead uses a vision-language model to steer a pretrained policy at inference time (Liu et al., 2026b).

Online Failure Detection and Agentic Runtime Recovery. Runtime recovery methods monitor execution and intervene when failures occur. SAFE, RecoveryChaining, RACER, COME-robot, and TCoT explore failure detection, recovery to resumable states, VLM supervision, execution tracing, and hierarchical recovery (Gu et al., 2025; Vats et al., 2025; Dai et al., 2025; Zhi et al., 2025; Li et al., 2026). GTA-VLA uses interactive visual guidance, including points, boxes, and traces, to steer VLA execution under spatial and visual ambiguity (Ling et al., 2026). Code-based and agentic systems

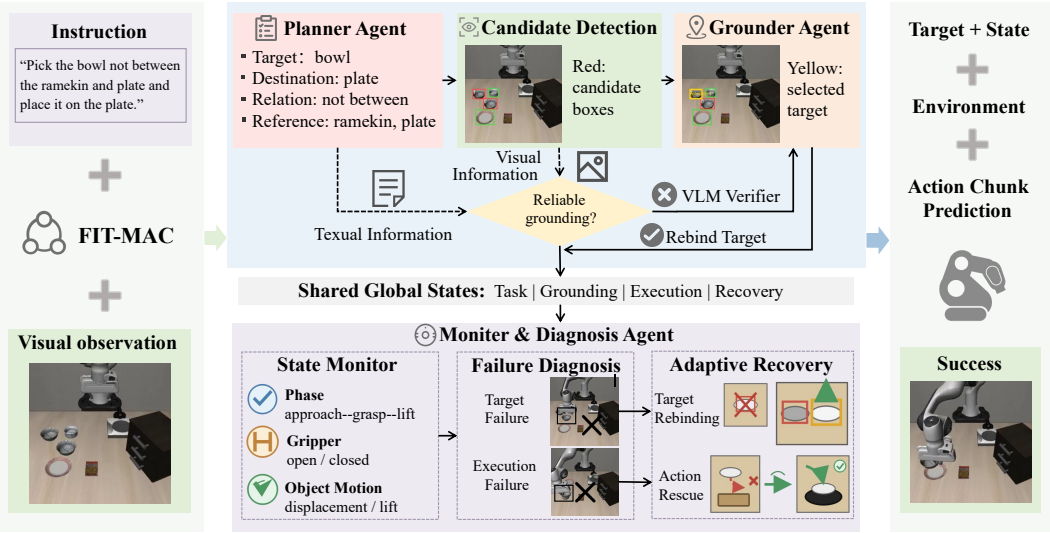

Figure 2: Overview of FIT-MAC. The framework first establishes an instance-level target lock through target rebinding, then monitors stage-specific execution progress and routes detected failures to target correction or bounded local recovery.

extend runtime recovery through execution feedback, visual verification, skill synthesis, and modular coordination (Fu et al., 2026; Zhang et al., 2026a; Garrabé et al., 2025; Singh et al., 2025; Chen et al., 2025; Yang et al., 2025; Yoon et al., 2026).

FIT-MAC explicitly distinguishes target failures from execution failures and routes them to target rebinding with closed-loop correction or bounded local grasping and placement control, respectively. The pretrained VLA remains the default action generator, with intervention triggered only by runtime evidence of failure.

3 METHOD

3.1 OVERVIEW

Given a language instruction x, a visual observation I_t, and a low-dimensional robot state o_t, FIT-MAC augments a frozen VLA policy π_{VLA} with an inference-time control layer. It comprises a Planner Agent, a Grounder Agent, a Monitor-Diagnoser Agent, and an Execution Recovery Module. The Planner Agent parses x into an ordered sequence of manipulation subtasks. Before each subtask, the Grounder Agent maps the instruction's target reference to a visual instance and constructs a grounded prompt $\tilde{x}_{k_t}$. During execution, the Monitor-Diagnoser Agent tracks phase-specific progress and distinguishes target-binding errors from physical execution failures. Let g_t denote the active target-grounding state, k_t the current subtask index, and ϕ_t the current manipulation phase.

When the target lock is reliable and no physical execution failure is detected, the pretrained VLA generates the action:

$$a_t = \begin{cases} \pi_{\text{VLA}}(I_t, \tilde{x}_{k_t}), & d_t = \text{none}, \\ \pi_{\text{pick}}(I_t, o_t, g_t, k_t), & d_t = \text{pick\_failure}, \\ \pi_{\text{place}}(I_t, o_t, g_t, k_t), & d_t = \text{place\_failure}. \end{cases} \tag{1}$$

Target correction is triggered when the target binding is uncertain or inconsistent. The bounded local recovery controllers π_{pick} and π_{place} return control to the pretrained VLA when recovery succeeds, or the recovery budget is exhausted. The VLA parameters remain unchanged.

3.2 TASK DECOMPOSITION AND STAGE MODELING

The Task Planner Agent parses x into a structured representation of the target object, source and destination descriptions, placement relation, and spatial constraints. Each long-horizon rollout is

divided into bounded pick-and-place attempts. The active phase ϕ_t belongs to either the pick-phase set Φ_{pick} or the place-phase set Φ_{place}.

FIT-MAC measures the end-effector–target distance during picking and the object–destination distance during placement:

$$d_t^{\text{tgt}} = \left\| p_t^{\text{ee}} - p_t^{\text{tgt}} \right\|_2, \qquad d_t^{\text{dst}} = \left\| p_t^{\text{obj}} - p_t^{\text{dst}} \right\|_2, \tag{2}$$

where p_t^{ee}, p_t^{tgt}, p_t^{obj}, and p_t^{dst} denote the end-effector, target, object, and destination positions, respectively. The distance reductions over Δ control steps are

$$\Delta d_t^{\text{tgt}} = d_{t-\Delta}^{\text{tgt}} - d_t^{\text{tgt}}, \qquad \Delta d_t^{\text{dst}} = d_{t-\Delta}^{\text{dst}} - d_t^{\text{dst}}. \tag{3}$$

Positive values indicate motion toward the corresponding target. Gripper closure, object–gripper motion, and post-release stability provide additional signals for detecting grasp completion, lifting, and placement.

3.3 Visual Grounding and Target Rebinding

Language instructions describe targets semantically, but robot execution requires identifying a specific visual instance. FIT-MAC therefore performs target rebinding before each subtask. Grounding DINO detects candidate target, destination, and reference objects. The task-level grounder then matches their descriptions and spatial relations to the instruction. When detections are uncertain or ambiguous, a VLM resolver selects the candidate that best matches the instruction. The selected target and destination are stored in the target-grounding state g_t and used to construct the grounded prompt $\tilde{x}_{k_t}$.

Let $\mathcal{C}_t^{\text{sem}}$ denote the instruction-compatible candidates. Runtime geometric verification scores each candidate as

$$S_t(c) = \left\| p_{t,\text{xy}}^c - p_{t,\text{xy}}^{\text{ee}} \right\|_2 + \lambda_z \left| p_{t,z}^c - p_{t,z}^{\text{ee}} \right| \tag{4}$$

yielding the geometrically closest compatible candidate, $\hat{c}_t = \arg\min_{c \in \mathcal{C}_t^{\text{sem}}} S_t(c)$. This score informs target rebinding while preserving semantic grounding.

The selected instance remains locked as the target throughout the current subtask. If its confidence κ_t falls below τ_κ or the locked instance is inconsistent with the current observation, FIT-MAC updates g_t and regenerates the grounded prompt. Target correction thus resolves incorrect or uncertain instance binding before local execution recovery is invoked.

3.4 Runtime Monitoring and Failure Diagnosis

After target rebinding, the pretrained VLA remains the default action generator. The monitor tracks target approach, gripper closure, object–gripper motion, destination approach, and post-release stability. For the current phase ϕ_t, progress is measured using the phase-specific thresholds τ_{tgt}, τ_{dst}, τ_{ee}, and τ_{obj}:

$$\chi_t^{\text{prog}} = \begin{cases} \mathbf{1}\left[\Delta d_t^{\text{tgt}} > \tau_{\text{tgt}} \ \vee \ \|\Delta p_t^{\text{ee}}\|_2 > \tau_{\text{ee}}\right], & \phi_t \in \Phi_{\text{pick}}, \\ \mathbf{1}\left[\Delta d_t^{\text{dst}} > \tau_{\text{dst}} \ \vee \ \|\Delta p_t^{\text{obj}}\|_2 > \tau_{\text{obj}}\right], & \phi_t \in \Phi_{\text{place}}. \end{cases} \tag{5}$$

A phase is considered stalled when no progress is observed during its patience window W_{ϕ_t}:

$$\chi_t^{\text{stall}} = \mathbf{1}\left[\sum_{\tau=t-W_{\phi_t}+1}^{t} \chi_\tau^{\text{prog}} = 0\right]. \tag{6}$$

To detect prolonged execution, FIT-MAC also uses a phase-timeout indicator $\chi_t^{\text{timeout}} = \mathbf{1}[n_t^{\text{phase}} > T_{\phi_t}]$, where n_t^{phase} is the elapsed duration of the current phase and T_{ϕ_t} is its maximum allowed duration.

The diagnosed state is:

$$d_t = \begin{cases} \text{target\_correction}, & \kappa_t < \tau_\kappa \ \vee \ \mathrm{Inconsistent}(g_t, I_t), \\ \text{pick\_failure}, & \phi_t \in \Phi_{\text{pick}} \wedge \left(\chi_t^{\text{stall}} = 1 \vee \chi_t^{\text{timeout}} = 1\right), \\ \text{place\_failure}, & \phi_t \in \Phi_{\text{place}} \wedge \left(\chi_t^{\text{stall}} = 1 \vee \chi_t^{\text{timeout}} = 1\right), \\ \text{none}, & \text{otherwise}. \end{cases} \tag{7}$$

Here, κ_t denotes the confidence of the active target lock, and $\mathrm{Inconsistent}(g_t, I_t)$ indicates a mismatch between the locked instance and the observed scene or task relation. Pick failures occur during approach, grasping, or lifting; place failures occur during transport, placement, release, or post-release stabilization.

3.5 Target Correction and Execution Recovery

FIT-MAC supports two intervention paths. When $d_t =$ target correction, the Grounder Agent reruns object detection, matches candidate instances to the instruction, updates g_t, and regenerates $\tilde{x}_{k_t}$. The pretrained VLA then resumes control with the corrected target binding.

When physical execution fails despite correct target grounding, FIT-MAC retains the target lock and invokes a bounded local recovery policy. The controller selects a phase-specific waypoint p_t^*: a hover point for approach, the grounded target for grasp and lift, the grounded destination for transport and place, and a retreat point for release. The translational recovery command is

$$u_t^{\text{rec}} = \mathrm{clip}\left((p_t^* - p_t^{\text{ee}}) \oslash (0.045, 0.045, 0.035), -1, 1\right), \tag{8}$$

where $\oslash$ denotes element-wise division. The complete recovery action is

$$a_t^{\text{rec}} = \left[\left(u_t^{\text{rec}}\right)^\top, \mathbf{0}_3^\top, a_t^{\text{grip}}\right]^\top, \tag{9}$$

where a_t^{grip} is the gripper command.

Pick recovery succeeds when the object moves consistently with the gripper after closure. Place recovery succeeds when the object is released within the destination region and remains stable. When recovery succeeds, or its budget is exhausted, control returns to the pretrained VLA or execution advances to the next subtask. FIT-MAC thus addresses binding errors through target correction and physical execution failures through local recovery, while retaining the pretrained VLA as the default action generator.

4 Experiments

4.1 Experimental Setup

We evaluate FIT-MAC on LIBERO-Object and LIBERO-Spatial (Liu et al., 2023) using the LIBERO-PRO perturbation protocol (Zhou et al., 2026). We consider five perturbation types: Pos, Task, Env, Obj, and Sem, corresponding to changes in object position, task instruction, environment, object identity, and semantic binding, respectively. We compare OpenVLA with and without FIT-MAC under matched task allocations, episode budgets, and random seeds, and use reported results for other methods.

We use success rate (SR) as the primary metric, complemented by intervention rate (IR) and recovered success rate (RSR). Let N_{all} denote the total number of evaluated episodes, N_{int} the number with at least one corrective action, and $N_{\text{succ,int}}$ the number of intervened episodes that succeed. We define

$$\mathrm{IR} = \frac{N_{\text{int}}}{N_{\text{all}}}, \qquad \mathrm{RSR} = \frac{N_{\text{succ,int}}}{N_{\text{int}}}. \tag{10}$$

IR measures intervention frequency, while RSR measures success among intervened episodes. Together with SR, these metrics characterize FIT-MAC's runtime behavior and task completion.

To assess generality, we also integrate FIT-MAC with π_0 (Black et al., 2025) and $\pi_{0.5}$ (Physical Intelligence et al., 2025), keeping all backbone parameters frozen. We conduct module ablations

Table 1: Comparison with state-of-the-art methods on LIBERO-PRO. The overall average is reported for methods evaluated on all ten perturbation settings.

Method	LIBERO-Object					LIBERO-Spatial					Avg.
	Obj.	Pos.	Sem.	Task	Env.	Obj.	Pos.	Sem.	Task	Env.	
MolmoAct Lee et al. (2025)	92.0	6.0	96.0	0.0	–	90.0	0.0	88.0	0.0	–	–
NORA Hung et al. (2025)	86.0	0.0	92.0	0.0	–	92.0	0.0	91.0	0.0	–	–
X-VLA Zheng et al. (2025)	89.0	2.0	98.0	8.0	–	97.0	0.0	**96.0**	0.0	–	–
AtomVLA Sun et al. (2026)	93.0	10.0	99.0	0.0	–	95.0	16.0	95.0	1.0	–	–
CaP-Agent$_0$ Fu et al. (2026)	–	22.0	–	18.0	–	–	12.0	–	14.0	–	–
RATs Zhang et al. (2026a)	–	61.0	–	63.0	–	–	29.0	–	31.0	–	–
OpenVLA Kim et al. (2024)	98.0	0.0	98.0	0.0	0.0	97.0	0.0	77.0	0.0	89.0	45.9
FIT-MAC (Ours)	**99.0**	**93.4**	**98.8**	**87.2**	**48.8**	**97.2**	**40.2**	86.8	**37.8**	**97.8**	**78.7**

Table 2: Runtime intervention and recovery behavior of FIT-MAC under different perturbations. SR, IR, and RSR denote success rate, intervention rate, and recovered success rate, respectively.

	LIBERO-Object			LIBERO-Spatial		
Perturbation	SR. (%)	IR. (%)	RSR. (%)	SR. (%)	IR. (%)	RSR. (%)
Obj.	99.0	99.8	99.3	97.2	37.8	97.9
Pos.	93.4	99.5	93.8	40.2	87.6	41.3
Sem.	98.8	98.9	99.6	86.8	42.8	98.6
Task	87.2	100.0	87.1	37.8	90.6	35.5
Env.	48.8	50.4	97.5	97.8	21.2	99.1

and sensitivity analyses of intervention criteria, recovery timing, and execution horizon. For the controlled OpenVLA comparison, we average results over five random seeds (7, 11, 13, 17, and 23). Complete per-seed results appear in Appendix A.3.

4.2 Main Results

As shown in Table 1, FIT-MAC improves OpenVLA's average success rate from 45.9% to 78.7% across the ten perturbation settings, a gain of 32.8 %. It outperforms OpenVLA in every setting and achieves the highest success rate in eight of ten settings. The largest gains occur under position and task perturbations, where OpenVLA fails in both suites. FIT-MAC raises success rates to 93.4% and 87.2% on LIBERO-Object and to 40.2% and 37.8% on LIBERO-Spatial, respectively, exceeding all reported baselines in these settings. Under environment perturbations, it improves success rates from 0.0% to 48.8% on LIBERO-Object and from 89.0% to 97.8% on LIBERO-Spatial. These gains accompany consistently strong performance under object perturbations. Although FIT-MAC does not lead under semantic perturbations, it improves on OpenVLA in both suites. Overall, the results support the effectiveness of selective runtime correction, while the lower success rates under spatial position and task perturbations indicate remaining challenges.

In Table 2 we report SR, IR, and RSR for each perturbation to characterize intervention frequency and recovery outcomes. On LIBERO-Object, FIT-MAC intervenes frequently across perturbation types while maintaining high RSR. On LIBERO-Spatial, interventions are most frequent under Pos and Task, the most challenging perturbations. Obj, Sem, and Env require fewer interventions and yield substantially higher RSR.

In Figure 3 we illustrate target correction and execution recovery. In the top row, the instruction specifies the bowl not between the plate and the ramekin, but the policy initially grasps the wrong bowl. FIT-MAC corrects the target binding and redirects manipulation to the intended bowl for placement on the plate. In the bottom row, the policy grasps the correct bowl from the cabinet but places it in the ramekin instead of on the plate. FIT-MAC detects the placement error and invokes local recovery to move the bowl to the instructed destination. These examples illustrate how FIT-MAC addresses target-selection and execution failures through distinct corrective mechanisms.

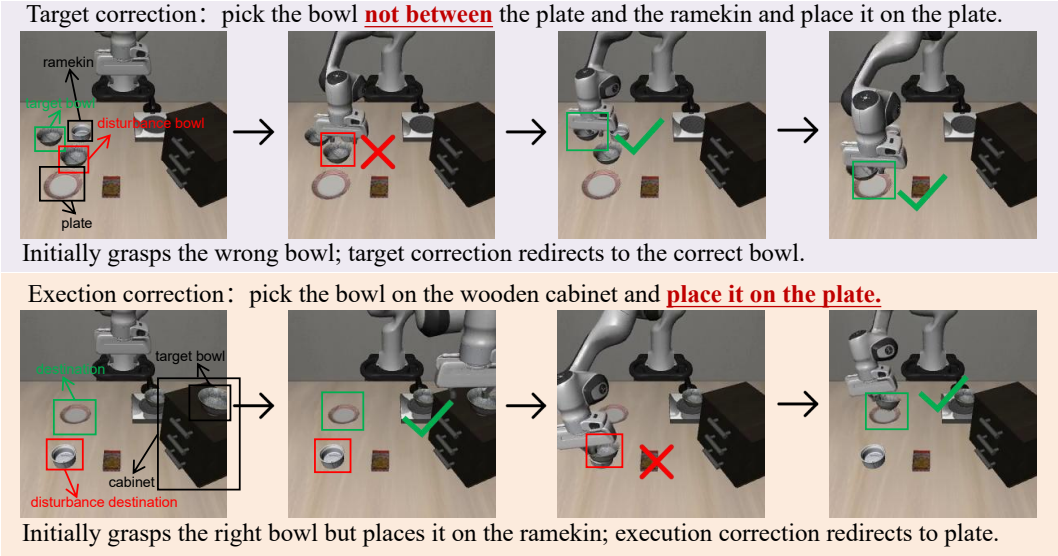

Figure 3: Examples of target correction and execution recovery. The top row shows target rebinding after a grounding failure; the bottom row shows local recovery after a manipulation failure.

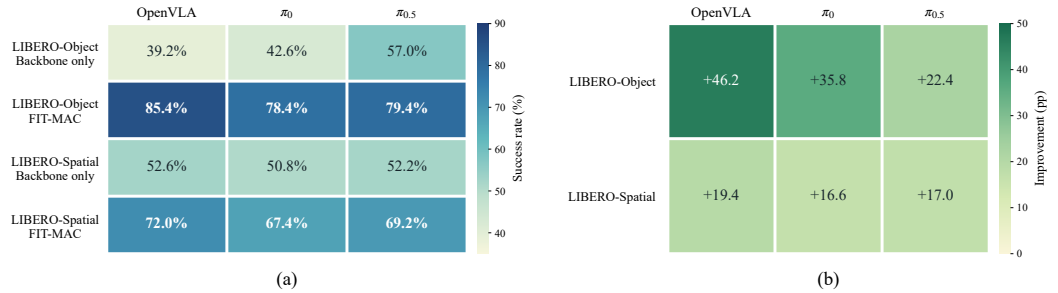

Figure 4: FIT-MAC across VLA backbones on LIBERO-PRO. (a) Success rates of OpenVLA, π_0, and $\pi_{0.5}$ with and without FIT-MAC, averaged over five perturbation types in each suite. (b) Corresponding absolute gains in success rate.

4.3 Generalization across VLA Backbones

We evaluate FIT-MAC with OpenVLA, π_0, and $\pi_{0.5}$ to assess its generality across VLA backbones. As shown in Figure 4(a), FIT-MAC consistently improves average success rates on both LIBERO-Object and LIBERO-Spatial for all three backbones. In Figure 4(b), the absolute gains show that these benefits extend across policies with different architectures and baseline capabilities. These results support the broader applicability of failure-aware runtime correction, which improves robustness while keeping each backbone frozen and retaining it as the default action generator.

4.4 Ablation Study

Module Ablation. In Table 3, We assess each module's contribution through controlled ablations with OpenVLA. Object and Spatial denote success rates averaged over five perturbation types; Avg. is their mean. ΔSR reports the change in percentage points relative to full FIT-MAC. Removing any component reduces performance on both suites. Execution correction contributes most: its removal lowers average success from 78.7% to 54.7%, with drops of 38.4 and 9.6 percentage points on LIBERO-Object and LIBERO-Spatial, respectively.

Hard pick primarily benefits LIBERO-Object, where its removal causes a 35.2-point drop, compared with a 2.0-point drop on LIBERO-Spatial. Hard place contributes substantially to both suites, with corresponding drops of 22.2 and 8.0 points. Target correction provides smaller but consistent gains, with its removal reducing success by 8.2 and 2.2 points. These results support the complementary

Table 3: Module ablation of FIT-MAC with the OpenVLA backbone on LIBERO-PRO.

Configuration	Success Rate (%)			ΔSR (pp)	
	Object	Spatial	Avg.	Object	Spatial
FIT-MAC	**85.4**	**72.0**	**78.7**	0.0	0.0
w/o target correction	77.2	69.8	73.5	-8.2	-2.2
w/o execution correction	47.0	62.4	54.7	-38.4	-9.6
w/o hard pick	50.2	70.0	60.1	-35.2	-2.0
w/o hard place	63.2	64.0	63.6	-22.2	-8.0

Table 4: Hyperparameter sensitivity of FIT-MAC with the OpenVLA backbone on LIBERO-PRO. Default values are shown in bold.

Parameter	Values	Object	Spatial	Avg.
Failure threshold	2 / **3** / 4	85.4 / 85.4 / 85.4	70.8 / 70.6 / 70.6	78.1 / 78.0 / 78.0
Confidence threshold	0.10 / **0.12** / 0.18	85.4 / 85.4 / 85.4	71.0 / 70.6 / 71.0	78.2 / 78.0 / 78.2
Minimum rescue step	8 / **12** / 16	85.4 / 85.4 / 85.4	71.2 / 70.6 / 71.0	78.3 / 78.0 / 78.2
Execution horizon	1 / **2** / 3	68.0 / 85.4 / 87.4	72.0 / 70.0 / 69.8	70.0 / 77.7 / 78.6

roles of target correction and local execution recovery. Complete per-perturbation results appear in Appendix A.4.

Hyperparameter Sensitivity Analysis In Table 4 We assess FIT-MAC's sensitivity to four hyperparameters using OpenVLA. Object and Spatial denote success rates averaged over five perturbation types; Avg. is their mean. Performance remains stable across the tested failure thresholds, confidence thresholds, and minimum rescue steps: LIBERO-Object success stays at 85.4%, and average success varies by at most 0.3 percentage points. We select intermediate values of 3, 0.12, and 12, respectively, to balance failure detection, target reliability, and recovery timing.

The execution horizon has a larger effect. Increasing it from 1 to 2 raises average success from 70.0% to 77.7%, primarily through gains on LIBERO-Object. A horizon of 3 further increases the average to 78.6%, but reduces performance under position and environment perturbations on LIBERO-Spatial. We therefore retain 2 for more balanced performance across perturbation types. Complete per-perturbation results appear in Appendix A.5.

5 Conclusion

We introduced FIT-MAC, a failure-aware multi-agent framework that improves VLA robustness at inference time. FIT-MAC distinguishes target-binding failures from local execution failures and applies targeted corrections without retraining or modifying the pretrained policy. Experiments across VLA backbones and perturbation settings show consistent robustness gains. Intervention and recovery analyses characterize its runtime behavior, while ablations identify execution correction as the largest contributor to these gains. Future work will reduce intervention overhead and improve semantic and spatial grounding for reliable long-horizon execution.

AI use statement

In this work, we used generative AI tools for language refinement and grammar checking. We have not used generative AI tools for designing the proposed method, conducting experiments, generating experimental results, or making scientific claims. All AI-assisted content was reviewed and verified by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

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. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, 2025.

Hongyi Chen, Yunchao Yao, Ruixuan Liu, Changliu Liu, and Jeffrey Ichnowski. Robot failure recovery using vision-language models with optimized prompts. In *2025 American Control Conference*, pp. 1983–1988, 2025. doi: 10.23919/ACC63710.2025.11107751.

Yinpei Dai, Jayjun Lee, Nima Fazeli, and Joyce Chai. RACER: Rich language-guided failure recovery policies for imitation learning. In *2025 IEEE International Conference on Robotics and Automation*, pp. 15657–15664, 2025. doi: 10.1109/ICRA55743.2025.11127799.

Yiguo Fan, Shuanghao Bai, Xinyang Tong, Pengxiang Ding, Yuyang Zhu, Hongchao Lu, Fengqi Dai, Wei Zhao, Yang Liu, Siteng Huang, Zhaoxin Fan, Badong Chen, and Donglin Wang. Long-VLA: Unleashing long-horizon capability of vision language action model for robot manipulation. In *Proceedings of the Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 2018–2037, 2025.

Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Dantong Niu, Fei-Fei Li, Guanya Shi, Jiajun Wu, Shankar Sastry, Yuke Zhu, Ken Goldberg, and Linxi Fan. CaP-X: A framework for benchmarking and improving coding agents for robot manipulation, 2026.

Hanan Gani, Tejal Kulkarni, Madhoolika Chodavarapu, Nicklas Hansen, and Manmohan Chandraker. Robotales: Learning reasoning-guided robot policies via task-aligned simulated futures. In *European Conference on Computer Vision*, 2026. ECCV 2026 accepted paper.

Émiland Garrabé, Pierre Teixeira, Mahdi Khoramshahi, and Stéphane Doncieux. Enhancing robustness in language-driven robotics: A modular approach to failure reduction. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 16717–16724, 2025. doi: 10.1109/IROS60139.2025.11246969.

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 *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1337.

Jianing Guo, Zhenhong Wu, Chang Tu, Yiyao Ma, Xiangqi Kong, Zhiqian Liu, Jiaming Ji, Shuning Zhang, Yuanpei Chen, Kai Chen, Qi Dou, Yaodong Yang, Xianglong Liu, Huijie Zhao, Weifeng Lv, and Simin Li. On robustness of vision-language-action model against multi-modal perturbations. In *International Conference on Learning Representations*, 2026.

Chia-Yu Hung, Qi Sun, Pengfei Hong, Amir Zadeh, Chuan Li, U-Xuan Tan, Navonil Majumder, and Soujanya Poria. NORA: A small open-sourced generalist vision language action model for embodied tasks, 2025.

Md Tanvir Islam, Sai Navaneet Peddapalli, Sangmoon Lee, and Sangtae Ahn. SUREFlow: State-space uncertainty-aware residual flow matching for robust robot manipulation, 2026. Accepted at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, 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, 2024.

Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, Winson Han, Wilbert Pumacay, Angelica Wu, Rose Hendrix, Karen Farley, Eli VanderBilt, Ali Farhadi, Dieter Fox, and Ranjay Krishna. MolmoAct: Action reasoning models that can reason in space, 2025.

Xiang Li, Ya-Li Li, Yuan Wang, Huaqiang Wang, and Shengjin Wang. TCoT: Trajectory chain-of-thoughts for robotic manipulation with failure recovery in vision-language-action model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 6486–6494, 2026. doi: 10.1609/aaai.v40i8.37577.

Yiran Ling, Qing Lian, Jinghang Li, Qing Jiang, Tianming Zhang, Xiaoke Jiang, Chuanxiu Liu, Jie Liu, and Lei Zhang. Guide, think, act: Interactive embodied reasoning in vision-language-action models. In *European Conference on Computer Vision*, 2026. ECCV 2026 accepted paper.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-1939.

Lin Liu, Zhicheng Bao, Lu Zhang, Ziying Song, Wu Yang, Yuzheng Zhuang, Shuai Tao, Wulong Liu, Caiyan Jia, and Huchuan Lu. LIBERO-RECOVER: Beyond task success towards failure recovery in robotic manipulation models. *arXiv preprint arXiv:2609.05178*, 2026a.

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

Physical Intelligence, 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, 2025.

Harsh Singh, Rocktim Jyoti Das, Mingfei Han, Preslav Nakov, and Ivan Laptev. MALMM: Multi-agent large language models for zero-shot robotic manipulation. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 20386–20393, 2025. doi: 10.1109/IROS60139.2025.11247340.

Xiaoquan Sun, Zetian Xu, Chen Cao, Zonghe Liu, Yihan Sun, Jingrui Pang, Ruijian Zhang, Zhen Yang, Kang Pang, Dingxin He, Mingqi Yuan, and Jiayu Chen. AtomVLA: Scalable post-training for robotic manipulation via predictive latent world models, 2026.

Shivam Vats, Devesh K. Jha, Maxim Likhachev, Oliver Kroemer, and Diego Romeres. RecoveryChaining: Learning local recovery policies for robust manipulation. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 9776–9783, 2025. doi: 10.1109/IROS60139.2025.11245856.

Guodong Wang, Chenkai Zhang, Qingjie Liu, Jinjin Zhang, Jiancheng Cai, Junjie Liu, and Xinmin Liu. LIBERO-X: Robustness litmus for vision-language-action models. *arXiv preprint arXiv:2602.06556*, 2026a.

Taowen Wang, Cheng Han, James Chenhao Liang, Wenhao Yang, Dongfang Liu, Luna Xinyu Zhang, Qifan Wang, Jiebo Luo, and Ruixiang Tang. Exploring the adversarial vulnerabilities of vision-language-action models in robotics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6948–6958, 2025a. doi: 10.1109/ICCV51701.2025.00653.

Xinkai Wang, Chenyi Wang, Yifu Xu, Mingzhe Ye, Fucheng Zhang, Jialin Tian, Xinyu Zhan, Lifeng Zhu, Cewu Lu, and Lixin Yang. Lamp: Learning vision-language-action policy with 3d scene flow as latent motion prior. In *European Conference on Computer Vision*, 2026b. ECCV 2026 accepted paper.

Zhijie Wang, Zhehua Zhou, Jiayang Song, Yuheng Huang, Zhan Shu, and Lei Ma. VLATest: Testing and evaluating vision-language-action models for robotic manipulation. *Proceedings of the ACM on Software Engineering*, 2(FSE):1615–1638, 2025b. doi: 10.1145/3729343.

Zhejian Yang, Yongchao Chen, Xueyang Zhou, Jiangyue Yan, Dingjie Song, Yinuo Liu, Yuting Li, Yu Zhang, Pan Zhou, Hechang Chen, and Lichao Sun. Agentic robot: A brain-inspired framework for vision-language-action models in embodied agents. *arXiv preprint arXiv:2505.23450*, 2025.

Sihyung Yoon, Minjong Yoo, Sanghyun Ahn, Seojeong Choi, and Honguk Woo. RoboBRIDGE: A modular framework for bridging policies to robust real-world robotic agents. *arXiv preprint arXiv:2607.27881*, 2026.

Hongyin Zhang, Shuo Zhang, Junxi Jin, Qixin Zeng, Runze Li, and Donglin Wang. RobustVLA: Robustness-aware reinforcement post-training for vision-language-action models. *arXiv preprint arXiv:2511.01331*, 2025a.

Jianke Zhang, Yanjiang Guo, Yucheng Hu, Xiaoyu Chen, Xiang Zhu, and Jianyu Chen. UP-VLA: A unified understanding and prediction model for embodied agent. In *Proceedings of the International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 74911–74922, 2025b.

Junyi Zhang, Jiaxin Ge, Hanjun Yoo, Letian Fu, Zihan Yang, Yaowei Liu, Raj Saravanan, Shaofeng Yin, Justin Yu, Dantong Niu, Zirui Wang, Roei Herzig, Ken Goldberg, Yutong Bai, David M. Chan, Ion Stoica, Angjoo Kanazawa, Jiahui Lei, Haiwen Feng, and Trevor Darrell. Playful agentic robot learning, 2026a.

Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, XinQiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, He Wang, Zhizheng Zhang, Li Yi, Wenjun Zeng, and Xin Jin. DreamVLA: A vision-language-action model dreamed with comprehensive world knowledge. In *Advances in Neural Information Processing Systems*, volume 38, 2025c.

Yuhao Zhang, Wanxi Dong, Yue Shi, Yi Liang, Jingnan Gao, Qiaochu Yang, Yaxing Lyu, Zhixuan Liang, Yibin Liu, Congsheng Xu, Xianda Guo, Wei Sui, Yaohui Jin, Xiaokang Yang, Yanyan Xu, and Yao Mu. R3dp: Real-time 3d-aware policy for embodied manipulation. In *European Conference on Computer Vision*, 2026b. ECCV 2026 accepted paper.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Ya-Qin Zhang, Jiangmiao Pang, Jingjing Liu, Tai Wang, and Xianyuan Zhan. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model, 2025.

Peiyuan Zhi, Zhiyuan Zhang, Yu Zhao, Muzhi Han, Zeyu Zhang, Zhitian Li, Ziyuan Jiao, Baoxiong Jia, and Siyuan Huang. Closed-loop open-vocabulary mobile manipulation with GPT-4V. In *2025 IEEE International Conference on Robotics and Automation*, pp. 4761–4767, 2025. doi: 10.1109/ICRA55743.2025.11127975.

Jiaming Zhou, Ke Ye, Jiayi Liu, Teli Ma, Zifan Wang, Ronghe Qiu, Kun-Yu Lin, Zhilin Zhao, and Junwei Liang. Exploring the limits of vision-language-action manipulation in cross-task generalization. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization, 2026.

A Additional Experimental Results

A.1 Evaluation Protocol

We evaluate OpenVLA, π_0, and $\pi_{0.5}$ on LIBERO-Object and LIBERO-Spatial. Each suite includes five perturbation conditions: Task, Pos, Obj, Sem, and Env. For each method and perturbation setting, we report success rate (SR), intervention rate (IR), and recovered success rate (RSR). SR is the fraction of successful episodes, IR is the fraction with at least one intervention, and RSR is the fraction of intervened episodes that succeed.

Under each evaluation condition, we run 50 episodes per task for the main experiments and 10 episodes per task for the ablation experiments. Multi-seed experiments use five random seeds, $7, 11, 13, 17, 23$, with 10 episodes per task per seed under each evaluation condition. Unless otherwise specified, all methods compared within each experiment use identical task allocations, episode budgets, and random seeds.

A.2 Complete Cross-Backbone Results

In table 5 we report complete results by suite and perturbation for OpenVLA and $\pi_{0.5}$ to assess FIT-MAC's effectiveness across action policies. Backbone-only models do not use FIT-MAC's intervention modules, so their intervention rate (IR) and recovered success rate (RSR) are marked as "–". This breakdown helps distinguish gains from FIT-MAC from differences in baseline robustness across backbones.

As shown in table 6, FIT-MAC consistently improves the average success rate across all three VLA backbones. On LIBERO-Object, FIT-MAC improves OpenVLA, π_0, and $\pi_{0.5}$ from 39.2%, 42.6%, and 57.0% to 85.4%, 78.4%, and 79.4%, respectively. These results correspond to absolute success-rate gains of 46.2%, 35.8%, and 22.4%. On LIBERO-Spatial, FIT-MAC raises the corresponding success rates from 52.6%, 50.8%, and 52.2% to 72.0%, 67.4%, and 69.2%, yielding absolute success-rate gains of 19.4%, 16.6%, and 17.0%. Although the magnitude of improvement varies across backbones and benchmarks, the consistent positive gains show that FIT-MAC remains effective under different underlying action policies.

Table 5: Complete cross-backbone results under five perturbation conditions. Each cell reports SR/IR/RSR, all expressed as percentages (%).

Benchmark	Condition	OpenVLA	OpenVLA + FIT-MAC	π0	π0 + FIT-MAC	π0.5	π0.5 + FIT-MAC
LIBERO-Object	Task	0.0/–/–	87.3/100.0/87.1	0.0/–/–	84.0/98.0/85.7	1.0/–/–	80.0/100.0/80.0
	Pos	0.0/–/–	93.1/99.5/93.8	0.0/–/–	67.0/100.0/67.0	17.0/–/–	67.0/99.0/67.7
	Obj	98.0/–/–	99.0/99.8/99.3	94.0/–/–	89.0/100.0/89.0	98.0/–/–	92.0/100.0/92.0
	Sem	98.0/–/–	98.8/98.9/99.6	90.0/–/–	87.0/97.0/89.7	96.0/–/–	89.0/97.0/91.8
	Env	0.0/–/–	48.8/50.4/97.5	29.0/–/–	65.0/78.0/82.1	73.0/–/–	69.0/79.0/86.1
LIBERO-Spatial	Task	0.0/–/–	37.8/90.6/35.5	0.0/–/–	27.0/100.0/27.0	1.0/–/–	25.0/100.0/25.0
	Pos	0.0/–/–	40.2/87.6/41.3	0.0/–/–	36.0/98.0/34.7	20.0/–/–	37.0/98.0/37.8
	Obj	97.0/–/–	97.2/37.8/97.9	97.0/–/–	97.0/52.0/98.1	97.0/–/–	97.0/52.0/100.0
	Sem	77.0/–/–	86.8/42.8/98.6	97.0/–/–	83.0/60.0/81.7	97.0/–/–	93.0/63.0/88.9
	Env	89.0/–/–	97.8/21.2/99.1	60.0/–/–	94.0/42.0/95.2	46.0/–/–	94.0/41.0/97.6

A.3 Multi-Seed Results

We assess the stability of FIT-MAC across five random seeds, $\{7, 11, 13, 17, 23\}$, using OpenVLA as the backbone on LIBERO-Object and LIBERO-Spatial. For each seed, we run 10 episodes per task under each perturbation condition. As shown in tables 7 and 8, we report success rate (SR), intervention rate (IR), and recovered success rate (RSR) for each perturbation condition on the two benchmarks, respectively.

Table 6: Average success rates (%) of FIT-MAC with different VLA backbones. Results are averaged over the five perturbation types within each benchmark. The Δ rows report absolute gains in percentage points (pp) over the corresponding backbone-only baselines.

Benchmark	Method	OpenVLA	π_0	$\pi_{0.5}$
LIBERO-Object	Backbone only	39.2	42.6	57.0
	FIT-MAC	**85.4**	**78.4**	**79.4**
	Δ (pp)	+46.2	+35.8	+22.4
LIBERO-Spatial	Backbone only	52.6	50.8	52.2
	FIT-MAC	**72.0**	**67.4**	**69.2**
	Δ (pp)	+19.4	+16.6	+17.0

Table 7: Multi-seed results of FIT-MAC on LIBERO-Object with OpenVLA. Each cell reports SR/IR/RSR (%).

Method	**Seed**	**Obj.**	**Pos.**	**Sem.**	**Task**	**Env.**
OpenVLA + FIT-MAC	7	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD
	11	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD
	13	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD
	17	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD
	23	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD

For each suite, perturbation type, and metric, we compute the mean and standard deviation across seeds. Let x_i denote the result for seed i, and $N = 5$. These statistics are defined as

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, \qquad \sigma_x = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}. \tag{11}$$

The seed-level results and their aggregate statistics allow us to distinguish consistent behavior from outcomes that may be dominated by a particular evaluation seed.

Summary. Across the five evaluation seeds, OpenVLA with FIT-MAC achieves an average SR of [TBD] on LIBERO-Object and [TBD] on LIBERO-Spatial, with standard deviations of [TBD] and [TBD], respectively. With $\pi_{0.5}$ as the underlying backbone, FIT-MAC achieves corresponding average SRs of [TBD] and [TBD], with standard deviations of [TBD] and [TBD]. The largest seed-level variation is observed under [TBD] perturbations, whereas performance under [TBD] perturbations remains comparatively stable. Overall, the multi-seed results show that [TBD: state whether FIT-MAC maintains consistent performance across seeds], rather than being driven by a single favorable evaluation seed.

A.4 Complete Module Ablation

We conduct controlled module ablations on OpenVLA and $\pi_{0.5}$ to assess the contribution of each FIT-MAC component. We compare full FIT-MAC, which includes both correctors and both constraints, with four leave-one-module-out variants: w/o target corrector, w/o execution corrector, w/o hard pick, and w/o hard place. Each variant removes one component while keeping the backbone, prompts, evaluation seeds, perturbation settings, and remaining components unchanged.

In table 9 and table 10, we report results on LIBERO-Object and LIBERO-Spatial, respectively. Each cell reports success rate (SR), intervention rate (IR), and recovered success rate (RSR). For backbone-only baselines, IR and RSR are marked as "–". For FIT-MAC variants with no interventions, IR is reported as 0.0% and RSR as "–".

Full FIT-MAC generally outperforms the ablated variants across perturbation settings. Removing the target corrector primarily affects settings involving incorrect target selection, while removing the execution corrector reduces robustness to failures during manipulation. The hard-pick and hard-place ablations assess the contributions of initial target localization and stage-specific action constraints. Although the effects vary across backbones and perturbations, these results support the modules' complementary roles.

Table 8: Multi-seed results of FIT-MAC on LIBERO-Spatial with OpenVLA. Each cell reports SR/IR/RSR (%).

Method	Seed	Obj.	Pos.	Sem.	Task	Env.
OpenVLA + FIT-MAC	7	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD
	11	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD
	13	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD
	17	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD
	23	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD	TBD/TBD/TBD

Table 9: Module ablation results under different perturbation conditions on LIBERO-Object. Each cell reports SR/IR/RSR (%).

Backbone	Configuration	Task	Pos.	Obj.	Sem.	Env.
OpenVLA	Baseline	0.0/–/–	0.0/–/–	98.0/–/–	98.0/–/–	0.0/–/–
	Full FIT-MAC	86.0/100.0/86.0	94.0/99.0/94.9	100.0/100.0/100.0	98.0/98.0/100.0	49.0/49.0/100.0
	w/o target_corrector	69.0/80.0/86.3	70.0/75.0/93.3	100.0/100.0/100.0	98.0/98.0/100.0	49.0/49.0/100.0
	w/o execution_corrector	0.0/0.0/–	4.0/0.0/–	100.0/0.0/–	97.0/0.0/–	34.0/0.0/–
	w/o hard_pick	0.0/0.0/–	4.0/4.0/100.0	100.0/100.0/100.0	98.0/98.0/100.0	49.0/49.0/100.0
	w/o hard_place	25.0/100.0/25.0	60.0/96.0/58.3	100.0/0.0/–	97.0/0.0/–	34.0/0.0/–
π0.5	Baseline	1.0/–/–	17.0/–/–	98.0/–/–	96.0/–/–	73.0/–/–
	Full FIT-MAC	80.0/100.0/80.0	67.0/99.0/67.7	92.0/100.0/92.0	89.0/97.0/91.8	69.0/79.0/86.1
	w/o target_corrector	71.0/83.0/85.5	59.0/94.0/62.8	90.0/100.0/90.0	87.0/100.0/87.0	67.0/79.0/84.8
	w/o execution_corrector	8.0/0.0/–	11.0/0.0/–	92.0/0.0/–	92.0/0.0/–	53.0/0.0/–
	w/o hard_pick	13.0/13.0/100.0	11.0/13.0/84.6	96.0/100.0/96.0	91.0/98.0/92.9	56.0/75.0/74.7
	w/o hard_place	36.0/89.0/31.5	45.0/100.0/45.0	97.0/0.0/–	89.0/0.0/–	55.0/0.0/–

A.5 Complete Hyperparameter Ablation

We assess FIT-MAC's sensitivity to four control parameters governing intervention criteria, recovery timing, and policy query frequency. We vary one parameter at a time while holding the backbone, benchmark, perturbation type, random seed, and all other settings fixed. Each cell reports success rate (SR), intervention rate (IR), and recovered success rate (RSR) for the backbone specified in the corresponding table. Together, these analyses assess sensitivity to intervention criteria, target-confidence requirements, recovery timing, and execution horizon.

A.5.1 Intervention-Gate Sensitivity

The parameter multi-agent-adaptive-intervention-failures sets the minimum number of consecutive actionable failures of the same type required to satisfy the adaptive gate's failure-streak condition. A failure counts only when the current attempt is flagged as a candidate for termination and diagnosed with a target or execution failure. The streak resets when the diagnosed failure type changes.

The gate also requires a sufficient streak of high-confidence target observations. This parameter therefore governs only the failure-streak condition; intervention also depends on other gating conditions and activation signals from strong-evidence and recovery paths. As shown in table 11 and table 12, we evaluate thresholds of $2, 3, 4$ and report results on LIBERO-Object and LIBERO-Spatial , respectively. Lower thresholds allow faster responses to repeated failures, while higher thresholds require more persistent evidence.

A.5.2 Target-Confidence Gate Sensitivity

The parameter multi-agent-adaptive-intervention-confidence sets the minimum target confidence required by the adaptive intervention gate. Confidence can be derived from target probes, shadow grounding, live grounding, and monitor signals. This threshold determines whether target evidence is reliable enough to inform intervention decisions and is distinct from the confidence thresholds of individual grounding or VLM components.

In table 13 and table 14 we evaluate thresholds of $0.10, 0.12, 0.18$ and report results on LIBERO-Object and LIBERO-Spatial, respectively. Lower thresholds admit weaker target evidence, while higher thresholds require greater confidence in the target estimate.

Table 10: Module ablation results under different perturbation conditions on LIBERO-Spatial. Each cell reports SR/IR/RSR (%).

Backbone	Configuration	Task	Pos.	Obj.	Sem.	Env.
OpenVLA	Baseline	0.0/–/–	0.0/–/–	97.0/–/–	77.0/–/–	89.0/–/–
	Full FIT-MAC	38.0/94.0/38.3	41.0/88.0/40.9	97.0/34.0/100.0	85.0/44.0/97.7	99.0/24.0/100.0
	w/o target_corrector	41.0/79.0/36.7	25.0/78.0/25.6	98.0/26.0/100.0	86.0/33.0/100.0	99.0/12.0/100.0
	w/o execution_corrector	28.0/0.0/–	5.0/0.0/–	95.0/0.0/–	86.0/0.0/–	98.0/0.0/–
	w/o hard_pick	28.0/40.0/35.0	41.0/37.0/97.3	97.0/34.0/100.0	85.0/44.0/97.7	99.0/24.0/100.0
	w/o hard_place	36.0/80.0/26.3	5.0/70.0/0.0	95.0/0.0/–	86.0/0.0/–	98.0/0.0/–
π0.5	Baseline	1.0/–/–	20.0/–/–	97.0/–/–	97.0/–/–	46.0/–/–
	Full FIT-MAC	25.0/100.0/25.0	37.0/98.0/37.8	97.0/52.0/100.0	93.0/63.0/88.9	94.0/41.0/97.6
	w/o target_corrector	27.0/100.0/27.0	30.0/97.0/28.9	95.0/52.0/98.1	90.0/64.0/84.4	97.0/42.0/100.0
	w/o execution_corrector	47.0/0.0/–	35.0/0.0/–	96.0/0.0/–	95.0/0.0/–	96.0/0.0/–
	w/o hard_pick	45.0/60.0/56.7	55.0/38.0/86.8	95.0/52.0/96.2	90.0/63.0/84.1	98.0/42.0/100.0
	w/o hard_place	35.0/100.0/35.0	14.0/94.0/10.6	97.0/0.0/–	96.0/0.0/–	96.0/0.0/–

Table 11: Sensitivity to the intervention-gate threshold on LIBERO-Object. Each cell reports SR/IR/RSR (%).

Backbone	Setting	Value	Task	Pos.	Obj.	Sem.	Env.
OpenVLA	FIT-MAC	–	86.0/100.0/86.0	94.0/99.0/94.9	100.0/100.0/100.0	98.0/98.0/100.0	49.0/49.0/100.0
	failures	2	86.0/100.0/86.0	94.0/99.0/94.9	100.0/100.0/100.0	98.0/98.0/100.0	49.0/49.0/100.0
	failures	3	86.0/100.0/86.0	94.0/100.0/94.0	100.0/100.0/100.0	98.0/98.0/100.0	49.0/49.0/100.0
	failures	4	86.0/100.0/86.0	94.0/100.0/94.0	100.0/100.0/100.0	98.0/98.0/100.0	49.0/49.0/100.0
π0.5	FIT-MAC	–	80.0/100.0/80.0	67.0/99.0/67.7	92.0/100.0/92.0	89.0/97.0/91.8	69.0/79.0/86.1
	failures	2	66.0/84.0/78.6	71.0/95.0/74.7	91.0/99.0/91.9	85.0/99.0/85.9	81.0/84.0/96.4
	failures	3	69.0/85.0/81.2	78.0/94.0/83.0	93.0/99.0/93.9	92.0/98.0/93.9	79.0/84.0/92.9
	failures	4	68.0/84.0/81.0	73.0/97.0/75.3	95.0/99.0/96.0	90.0/99.0/90.9	84.0/87.0/95.4

A.5.3 Rescue-Timing Sensitivity

The parameter multi-agent-adaptive-execution-rescue-min-step sets the earliest execution step at which time-based evidence can inform execution recovery. This delay helps avoid mistaking early transient deviations for execution failures. Recovery also requires phase-specific evidence, such as persistent wrong-target behavior, stagnation late in picking or placement, or lack of manipulation progress. Reaching the threshold alone does not trigger recovery.

In table 15 and table 16, we evaluate thresholds of $8, 12, 16$ and report results on LIBERO-Object and LIBERO-Spatial, respectively. Lower thresholds allow earlier consideration of recovery evidence, while higher thresholds give the current action more time to make progress.

A.5.4 Controller-Horizon Sensitivity

The parameter multi-agent-exec-horizon sets the maximum number of predicted actions executed before the next policy query and monitoring update in multi-agent control mode. The effective horizon is also bounded by the open-loop horizon:

$$H_{\text{effective}} = \min\left(H_{\text{open-loop}}, H_{\text{exec}}\right),$$

where H_{exec} denotes the configured execution horizon. This parameter controls feedback frequency without changing the action-chunk length predicted by the pretrained VLA.

In table 17 and table 18, we evaluate horizons of $1, 2, 3, 4$ and report results on LIBERO-Object and LIBERO-Spatial, respectively. Shorter horizons enable more frequent policy queries and monitoring updates, potentially allowing earlier responses to failures. Longer horizons execute more predicted actions before the next update but delay feedback-based correction.

B Additional Qualitative Results

We present additional qualitative examples from LIBERO-PRO to illustrate how FIT-MAC handles target-selection errors and execution failures. Each example traces the robot state before and after intervention, highlighting the initial error, the corrective action, and the subsequent behavior. Red

Table 12: Sensitivity to the intervention-gate threshold on LIBERO-Spatial. Each cell reports SR/IR/RSR (%).

Backbone	Setting	Value	Task	Pos.	Obj.	Sem.	Env.
OpenVLA	FIT-MAC	–	38.0/94.0/38.3	41.0/88.0/40.9	97.0/34.0/100.0	85.0/44.0/97.7	99.0/24.0/100.0
	failures	2	36.0/95.0/35.8	39.0/89.0/41.6	97.0/31.0/100.0	85.0/46.0/97.8	97.0/27.0/100.0
	failures	3	31.0/94.0/30.9	41.0/88.0/40.9	97.0/34.0/100.0	85.0/44.0/97.7	99.0/24.0/100.0
	failures	4	31.0/94.0/30.9	41.0/88.0/40.9	97.0/34.0/100.0	85.0/44.0/97.7	99.0/24.0/100.0
π0.5	FIT-MAC	–	25.0/100.0/25.0	37.0/98.0/37.8	97.0/52.0/100.0	93.0/63.0/88.9	94.0/41.0/97.6
	failures	2	39.0/100.0/39.0	51.0/97.0/49.5	95.0/52.0/94.2	94.0/62.0/91.9	95.0/42.0/95.2
	failures	3	39.0/100.0/39.0	41.0/89.0/33.7	94.0/52.0/94.2	93.0/63.0/93.7	96.0/42.0/100.0
	failures	4	38.0/100.0/38.0	42.0/89.0/34.8	98.0/52.0/100.0	92.0/63.0/90.5	94.0/42.0/97.6

Table 13: Sensitivity to the confidence threshold on LIBERO-Object. Each cell reports SR/IR/RSR (%).

Backbone	Setting	Value	Task	Pos.	Obj.	Sem.	Env.
OpenVLA	FIT-MAC	–	86.0/100.0/86.0	94.0/99.0/94.9	100.0/100.0/100.0	98.0/98.0/100.0	49.0/49.0/100.0
	confidence	0.10	86.0/100.0/86.0	94.0/99.0/94.9	100.0/100.0/100.0	98.0/98.0/100.0	49.0/49.0/100.0
	confidence	0.12	86.0/100.0/86.0	94.0/100.0/94.0	100.0/100.0/100.0	98.0/98.0/100.0	49.0/49.0/100.0
	confidence	0.18	86.0/100.0/86.0	94.0/99.0/94.9	100.0/100.0/100.0	98.0/98.0/100.0	49.0/49.0/100.0
π0.5	FIT-MAC	–	80.0/100.0/80.0	67.0/99.0/67.7	92.0/100.0/92.0	89.0/97.0/91.8	69.0/79.0/86.1
	confidence	0.10	69.0/85.0/81.2	71.0/95.0/74.7	91.0/100.0/91.0	88.0/100.0/88.0	83.0/86.0/95.3
	confidence	0.12	70.0/85.0/82.4	71.0/94.0/75.5	89.0/99.0/89.9	84.0/99.0/84.8	79.0/85.0/92.9
	confidence	0.18	68.0/85.0/80.0	71.0/94.0/75.5	91.0/100.0/91.0	79.0/99.0/79.8	70.0/91.0/76.9

annotations indicate the initially incorrect target or destination, whereas green annotations identify the corrected target or destination. These examples complement the quantitative results by showing how corrections affect the robot's actions during task execution.

B.1 Target-Correction Cases

In Figure 5, we present representative cases in which the policy initially approaches or grasps an object inconsistent with the instruction. Such errors concern which object the robot acts on, rather than how it executes the manipulation. FIT-MAC detects the mismatch between the selected object and the instruction, corrects the target binding, and redirects subsequent actions toward the intended object. The before-and-after states illustrate how target correction changes the robot's immediate behavior and brings object selection back into alignment with the task instruction.

B.2 Execution-Correction Cases

In Figure 6, we present cases in which the policy selects the correct object but transports or places it at an incorrect destination. For example, the robot may place the object on a cookie box or in another bowl instead of on the instructed plate. Here, the selected object remains consistent with the instruction, but the resulting placement does not satisfy the task requirement. FIT-MAC detects this mismatch and invokes local recovery to redirect the object to the intended destination. The illustrated sequences show how execution correction addresses the placement error while preserving the intended object selection.

B.3 Failure Cases

In Figure 7, we illustrate two failure modes, incorrect target rebinding during grounding and stalled grasping despite correct target identification. In the left example, the robot is instructed to pick the bowl next to the cookie box and place it on the plate. Grounding DINO detects candidate bowls, but the VLM selects the bowl next to the ramekin, producing a target binding inconsistent with the spatial relation specified in the instruction. The controller then follows this incorrect binding and grasps the wrong bowl. This failure originates in target selection rather than low-level control, executing the grasp does not resolve the upstream grounding error.

Table 14: Sensitivity to the confidence threshold on LIBERO-Spatial. Each cell reports SR/IR/RSR (%).

Backbone	Setting	Value	Task	Pos.	Obj.	Sem.	Env.
OpenVLA	FIT-MAC	–	38.0/94.0/38.3	41.0/88.0/40.9	97.0/34.0/100.0	85.0/44.0/97.7	99.0/24.0/100.0
	confidence	0.10	37.0/95.0/36.8	39.0/90.0/41.1	97.0/31.0/100.0	85.0/46.0/97.8	97.0/27.0/100.0
	confidence	0.12	31.0/94.0/30.9	41.0/88.0/40.9	97.0/33.0/100.0	85.0/44.0/97.7	99.0/24.0/100.0
	confidence	0.18	36.0/95.0/35.8	39.0/90.0/41.1	98.0/32.0/100.0	85.0/46.0/97.8	97.0/27.0/100.0
π0.5	FIT-MAC	–	25.0/100.0/25.0	37.0/98.0/37.8	97.0/52.0/100.0	93.0/63.0/88.9	94.0/41.0/97.6
	confidence	0.10	41.0/100.0/41.0	44.0/89.0/37.1	96.0/52.0/98.1	93.0/64.0/90.6	96.0/42.0/97.6
	confidence	0.12	39.0/100.0/39.0	41.0/92.0/35.9	98.0/52.0/98.1	93.0/63.0/88.9	97.0/42.0/97.6
	confidence	0.18	40.0/100.0/40.0	43.0/89.0/36.0	97.0/51.0/98.0	94.0/62.0/93.5	96.0/42.0/97.6

Table 15: Sensitivity to the minimum rescue step on LIBERO-Object. Each cell reports SR/IR/RSR (%).

Backbone	Setting	Value	Task	Pos.	Obj.	Sem.	Env.
OpenVLA	FIT-MAC	–	86.0/100.0/86.0	94.0/99.0/94.9	100.0/100.0/100.0	98.0/98.0/100.0	49.0/49.0/100.0
	rescue_min_step	8	86.0/100.0/86.0	94.0/99.0/94.9	100.0/100.0/100.0	98.0/98.0/100.0	49.0/49.0/100.0
	rescue_min_step	12	86.0/100.0/86.0	94.0/100.0/94.0	100.0/100.0/100.0	98.0/98.0/100.0	49.0/49.0/100.0
	rescue_min_step	16	86.0/100.0/86.0	94.0/99.0/94.9	100.0/100.0/100.0	98.0/98.0/100.0	49.0/49.0/100.0
π0.5	FIT-MAC	–	80.0/100.0/80.0	67.0/99.0/67.7	92.0/100.0/92.0	89.0/97.0/91.8	69.0/79.0/86.1
	rescue_min_step	8	68.0/85.0/80.0	75.0/95.0/78.9	91.0/99.0/91.9	85.0/99.0/85.9	64.0/75.0/84.0
	rescue_min_step	12	63.0/84.0/75.0	72.0/95.0/75.8	93.0/100.0/93.0	87.0/100.0/87.0	63.0/76.0/82.9
	rescue_min_step	16	68.0/84.0/81.0	73.0/94.0/77.7	92.0/100.0/92.0	85.0/100.0/85.0	62.0/78.0/78.2

In the right example, the robot is instructed to pick the bowl next to the ramekin and place it on the plate. The VLM correctly identifies the target among the candidates detected by Grounding DINO, but the gripper repeatedly fails to grasp the bowl after approaching it. These unsuccessful attempts stall execution and prevent the robot from proceeding to placement. This case shows that correct grounding alone does not ensure successful manipulation. Together, the two examples highlight remaining limitations in spatially grounded target selection and grasp execution, either of which can prevent task completion.

C FIT-MAC Execution and Implementation

C.1 Benchmark and Perturbation Setup

LIBERO-Object. This benchmark evaluates object-centric manipulation, requiring the policy to identify the object specified in the instruction and perform the requested interaction. Successful execution therefore depends on both selecting the intended object and carrying out the manipulation. We use this benchmark to examine whether FIT-MAC can address incorrect object selection and execution failures, and how these corrections affect task completion under different perturbation conditions.

LIBERO-Spatial. This benchmark evaluates manipulation guided by spatial relations in the instruction. The policy must identify the intended target in relation to other objects and execute actions consistent with the specified goal. Recognizing the relevant object category alone may be insufficient when the instruction distinguishes the target by its spatial context. We use this benchmark to examine FIT-MAC's ability to correct target bindings and execution errors in tasks that require spatially grounded decisions.

Perturbation Settings. Following LIBERO-PRO, we consider five perturbation types: Task, Pos, Obj, Sem, and Env. Task perturbations modify task instructions, whereas Pos perturbations modify object positions. Obj perturbations change object identities, Sem perturbations modify semantic bindings, and Env perturbations alter environmental conditions. These settings probe different aspects of policy robustness, including instruction interpretation, target grounding, and execution under changes to the scene. We report results separately for each perturbation type and average success rates over the five types when summarizing cross-backbone performance.

Table 16: Sensitivity to the minimum rescue step on LIBERO-Spatial. Each cell reports SR/IR/RSR (%).

Backbone	Setting	Value	Task	Pos.	Obj.	Sem.	Env.
OpenVLA	FIT-MAC	–	38.0/94.0/38.3	41.0/88.0/40.9	97.0/34.0/100.0	85.0/44.0/97.7	99.0/24.0/100.0
	rescue_min_step	8	37.0/95.0/36.8	39.0/90.0/41.1	97.0/31.0/100.0	86.0/47.0/97.9	97.0/27.0/100.0
	rescue_min_step	12	31.0/94.0/30.9	41.0/88.0/40.9	97.0/31.0/100.0	85.0/44.0/97.7	99.0/24.0/100.0
	rescue_min_step	16	36.0/95.0/35.8	39.0/90.0/41.1	97.0/31.0/100.0	86.0/47.0/97.9	97.0/27.0/100.0
π0.5	FIT-MAC	–	25.0/100.0/25.0	37.0/98.0/37.8	97.0/52.0/100.0	93.0/63.0/88.9	94.0/41.0/97.6
	rescue_min_step	8	36.0/100.0/36.0	41.0/90.0/34.4	97.0/52.0/98.1	95.0/63.0/92.1	95.0/42.0/95.2
	rescue_min_step	12	39.0/100.0/39.0	40.0/90.0/33.3	96.0/52.0/96.2	95.0/62.0/91.9	94.0/42.0/95.2
	rescue_min_step	16	41.0/100.0/41.0	44.0/90.0/37.8	97.0/52.0/98.1	94.0/62.0/91.9	97.0/42.0/97.6

Table 17: Sensitivity to the execution horizon on LIBERO-Object. Each cell reports SR/IR/RSR (%).

Backbone	Setting	Value	Task	Pos.	Obj.	Sem.	Env.
OpenVLA	FIT-MAC	–	86.0/100.0/86.0	94.0/99.0/94.9	100.0/100.0/100.0	98.0/98.0/100.0	49.0/49.0/100.0
	exec_horizon	1	36.0/100.0/36.0	58.0/100.0/58.0	100.0/100.0/100.0	97.0/98.0/99.0	49.0/49.0/100.0
	exec_horizon	2	86.0/100.0/86.0	94.0/100.0/94.0	100.0/100.0/100.0	98.0/98.0/100.0	49.0/49.0/100.0
	exec_horizon	3	96.0/100.0/96.0	94.0/99.0/94.9	100.0/100.0/100.0	98.0/98.0/100.0	49.0/49.0/100.0
	exec_horizon	4	77.0/100.0/77.0	79.0/99.0/79.8	99.0/100.0/99.0	98.0/98.0/100.0	49.0/49.0/100.0
π0.5	FIT-MAC	–	80.0/100.0/80.0	67.0/99.0/67.7	92.0/100.0/92.0	89.0/97.0/91.8	69.0/79.0/86.1
	exec_horizon	1	64.0/100.0/64.0	44.0/100.0/44.0	88.0/100.0/88.0	82.0/92.0/89.1	55.0/70.0/78.6
	exec_horizon	2	80.0/100.0/80.0	67.0/99.0/67.7	92.0/100.0/92.0	89.0/97.0/91.8	69.0/79.0/86.1
	exec_horizon	3	68.0/82.0/82.9	68.0/96.0/70.8	90.0/100.0/90.0	93.0/99.0/93.9	67.0/79.0/84.8
	exec_horizon	4	64.0/85.0/75.3	72.0/93.0/77.4	89.0/99.0/89.9	92.0/100.0/92.0	81.0/90.0/90.0

C.2 FIT-MAC Execution Algorithm

FIT-MAC wraps a frozen VLA policy with a runtime controller that performs grounding, failure diagnosis, target correction, and execution correction. Target correction addresses errors in the selected object or destination by revising the grounding result before action generation. Execution correction addresses failures that occur after manipulation has started, including unsuccessful grasping, failed lifting, stalled transport, and incorrect placement. The controller makes routing decisions at action-query boundaries without modifying the parameters of the underlying VLA policy.

Let $\mathcal{P}$ denote the ordered subtask plan, G the persistent grounding record, S the attempt-monitor state, R the persistent execution-correction state, and $\mathcal{H}$ the diagnostic history. The grounding record stores the selected target, target confidence, target-instance lock, image location, and, when available, the destination reference. Once a target instance has been resolved, subsequent monitoring tracks the same instance instead of independently rematching a potentially different visual candidate at every query.

At each action-query boundary, FIT-MAC first identifies the active subtask and initializes or updates its grounding state. The monitor then summarizes the current manipulation attempt, and the diagnoser classifies the observed evidence. Based on the diagnosis, target confidence, target-instance reliability, progress state, and failure persistence, the adaptive gate determines whether target correction or execution correction is authorized. When neither correction branch is activated, FIT-MAC preserves the action generated by the base policy.

Here, $\perp$ indicates that the selected correction controller does not provide a valid action for the current query. In this case, control falls back to the base policy. The effective execution horizon is bounded by both the open-loop action horizon and the configured multi-agent execution horizon:

$$H_{\text{eff}} = \min\left(H_{\text{open-loop}}, H_{\text{exec}}\right), \tag{12}$$

where H_{exec} denotes `multi_agent_exec_horizon`. Thus, the execution horizon controls the maximum number of predicted actions executed before the next policy query and monitoring update in the corresponding multi-agent control mode. It does not replace the original action chunk length of the underlying VLA policy in all execution modes.

Grounding and Persistent Target State. For each active subtask, the grounding module resolves the target object from the task instruction and the current observation. Candidate detections are

Table 18: Sensitivity to the execution horizon on LIBERO-Spatial. Each cell reports SR/IR/RSR (%).

Backbone	Setting	Value	Task	Pos.	Obj.	Sem.	Env.
OpenVLA	FIT-MAC	–	38.0/94.0/38.3	41.0/88.0/40.9	97.0/34.0/100.0	85.0/44.0/97.7	99.0/24.0/100.0
	exec_horizon	1	38.0/94.0/38.3	40.0/88.0/39.8	97.0/34.0/100.0	86.0/45.0/97.8	99.0/24.0/100.0
	exec_horizon	2	28.0/94.0/27.7	41.0/88.0/40.9	97.0/34.0/100.0	85.0/44.0/97.7	99.0/24.0/100.0
	exec_horizon	3	31.0/95.0/30.5	39.0/90.0/41.1	97.0/31.0/100.0	85.0/46.0/97.8	97.0/27.0/100.0
	exec_horizon	4	29.0/95.0/28.4	39.0/90.0/41.1	97.0/31.0/100.0	84.0/46.0/95.7	97.0/27.0/100.0
π0.5	FIT-MAC	–	25.0/100.0/25.0	37.0/98.0/37.8	97.0/52.0/100.0	93.0/63.0/88.9	94.0/41.0/97.6
	exec_horizon	1	29.0/100.0/29.0	16.0/98.0/14.3	98.0/52.0/100.0	86.0/60.0/80.0	93.0/40.0/97.5
	exec_horizon	2	25.0/100.0/25.0	37.0/98.0/37.8	97.0/52.0/100.0	93.0/63.0/88.9	94.0/41.0/97.6
	exec_horizon	3	35.0/100.0/35.0	45.0/98.0/43.9	95.0/52.0/94.2	91.0/63.0/87.3	98.0/42.0/100.0
	exec_horizon	4	38.0/100.0/38.0	46.0/97.0/44.3	98.0/52.0/100.0	92.0/62.0/88.7	93.0/42.0/92.9

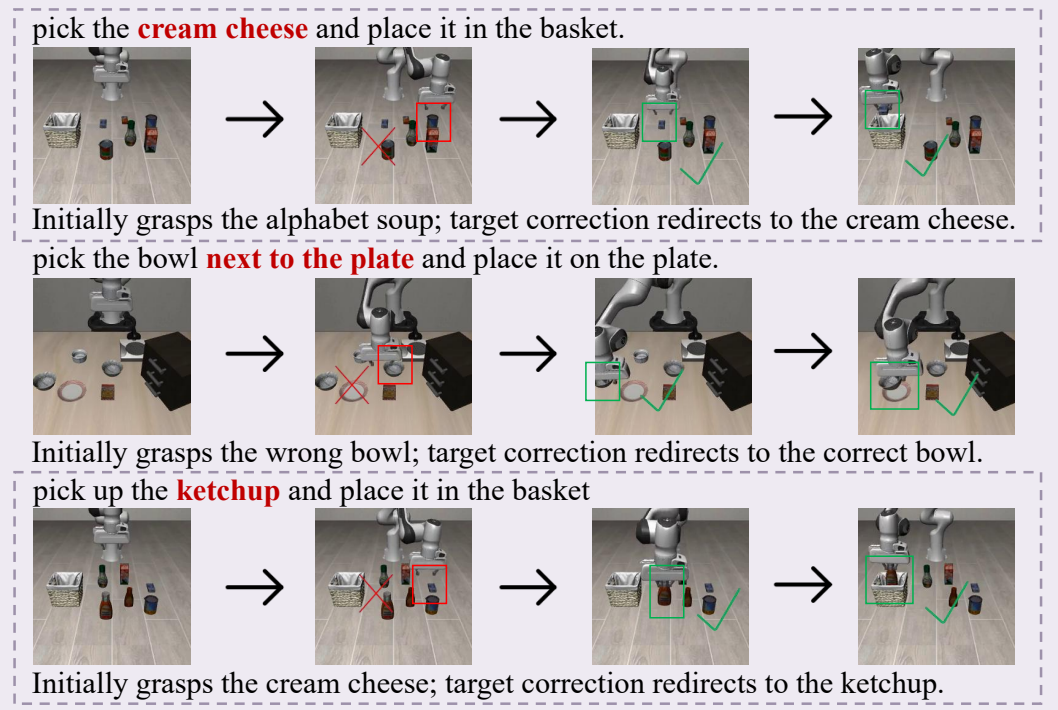

Figure 5: Qualitative target-correction cases. In each row, the policy initially selects or approaches an incorrect object (red). FIT-MAC detects the error, corrects the target binding, and redirects manipulation toward the intended object (green).

filtered using the object category and spatial description. When detections are missing, weak, or ambiguous, an additional language-model query may be used subject to the configured query budget. The resulting grounding record contains the selected target, confidence evidence, target-instance lock, image location, and destination reference when one can be resolved.

The initial target identity is retained across subsequent action queries. Therefore, monitoring and correction operate on a consistent object instance rather than repeatedly rematching an unrelated visual candidate. When the active subtask changes, the grounding state is reinitialized for the new instruction.

Monitoring and Diagnosis. The attempt monitor estimates the current manipulation phase from gripper state, end-effector motion, target lift, target–gripper geometry, and the verifier state. It also tracks the relationship between the robot motion and the resolved target instance. Phase-specific patience rules identify candidate failures, including approaching an incorrect object, unsuccessful grasping or lifting, stalled transport, and failed placement.

The diagnoser combines the current monitor snapshot with the recent diagnostic history. It classifies the current observation as a target-related failure, an execution-related failure, or a non-actionable

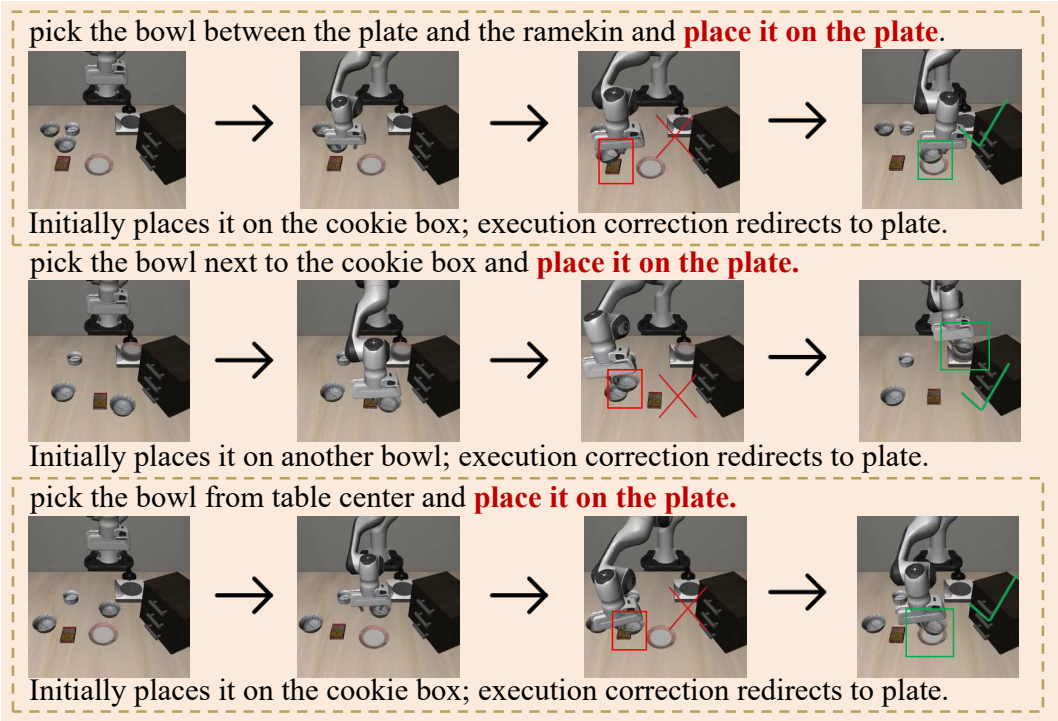

Figure 6: Qualitative execution-correction cases. The robot initially transports or places the selected bowl at an incorrect destination (red). FIT-MAC detects the placement error and redirects the bowl to the instructed plate (green).

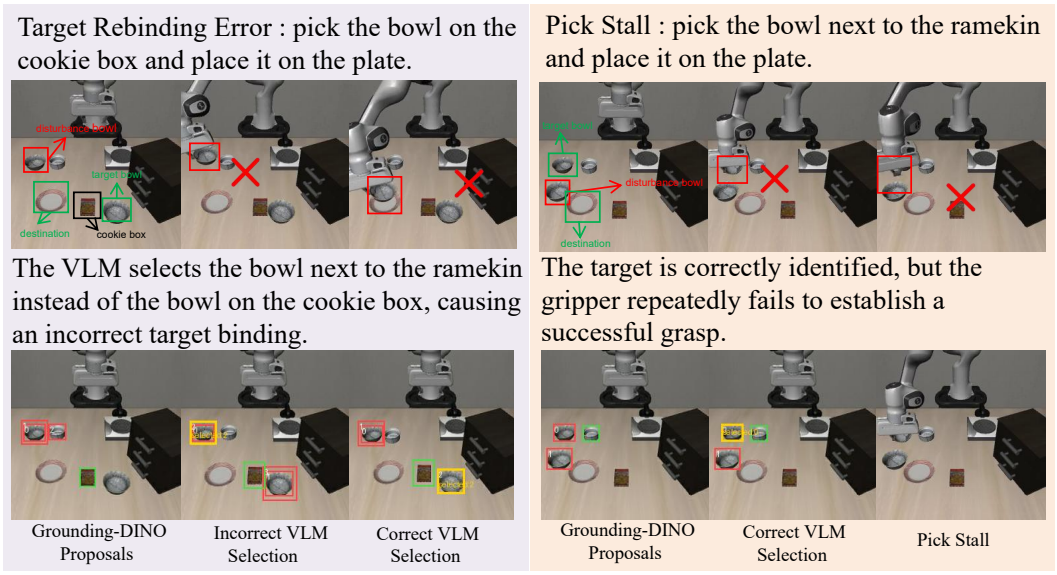

Figure 7: Representative FIT-MAC failures. Left: target rebinding error, The VLM selects the bowl next to the ramekin rather than the instructed bowl, causing the controller to grasp the wrong object. Right: pick stall, The target is correctly identified, but repeated grasp attempts fail.

observation. Target-related failures indicate that the selected object or destination may be incorrect or unreliable. Execution-related failures indicate that manipulation has already started but the expected grasping, lifting, transport, or placement progress has not been observed. An optional language-model assessment can supplement the rule-based diagnosis, while the monitor and diagnosis state remain part of the deterministic runtime trace.

Algorithm 1 FIT-MAC runtime execution

Require: Instruction x, base policy π, environment $\mathcal{E}$, configuration θ, execution budget T
Ensure: Episode trace $\mathcal{L}$ and verified outcome y

$\mathcal{P} \leftarrow$ PARSEANDDECOMPOSE(x)
Initialize grounding record G, monitor state S, correction state R
Initialize verifier V, diagnostic history $\mathcal{H}$, action queue Q, and trace $\mathcal{L}$
$y \leftarrow$ UNKNOWNOUTCOME
for $t = 0, \ldots, T-1$ **do**
  if $Q = \emptyset$ **then**
    $o_t \leftarrow$ OBSERVE$(\mathcal{E})$
    $j \leftarrow$ SELECTACTIVESUBTASK$(\mathcal{P}, V, G)$
    $x_j \leftarrow \mathcal{P}[j]$
    if x_j is new or G is not initialized **then**
      $G \leftarrow$ GROUNDTARGETANDDESTINATION(x_j, o_t, θ)
    end if
    $S \leftarrow$ MONITORATTEMPT$(S, R, G, o_t, V, \mathcal{E})$
    $(D, \mathcal{H}) \leftarrow$ DIAGNOSE$(S, \mathcal{H}, G, \theta)$
    $\Gamma \leftarrow$ ADAPTIVEGATE$(D, S, G, \mathcal{H}, \theta)$
    if HASPENDINGEXECUTIONCORRECTION(R) **then**
      $M \leftarrow$ pending execution correction
    else if Γ.allow_target_correction **then**
      $M \leftarrow$ target correction
    else if Γ.allow_execution_correction **then**
      $M \leftarrow$ execution correction
    else
      $M \leftarrow$ base policy
    end if
    if $M =$ target correction **then**
      $G \leftarrow$ CORRECTTARGETGROUNDING(G, S, o_t, θ)
      $\ell \leftarrow$ BUILDPOLICYINSTRUCTION(x_j, G, θ)
      $A \leftarrow \pi(\ell, o_t)$
    else if $M \in \{$pending execution correction, execution correction$\}$ **then**
      $(A, R) \leftarrow$ RUNEXECUTIONCORRECTION$(R, G, o_t, S, V, \mathcal{E}, \theta)$
      if $A = \perp$ **then**
        $\ell \leftarrow$ BUILDPOLICYINSTRUCTION(x_j, G, θ)
        $A \leftarrow \pi(\ell, o_t)$
      end if
    else
      $\ell \leftarrow$ BUILDPOLICYINSTRUCTION(x_j, G, θ)
      $A \leftarrow \pi(\ell, o_t)$
    end if
    $H_{\text{eff}} \leftarrow$ EFFECTIVEHORIZON(θ, Γ)
    $Q \leftarrow$ PREFIX$(A, \min(H_{\text{eff}}, |A|))$
    Append the routing decision and controller state to $\mathcal{L}$
  end if
  $a_t \leftarrow$ POPFRONT(Q)
  $(o_{t+1}, \text{terminal}) \leftarrow \mathcal{E}$.STEP$(()a_t)$
  Append (a_t, o_{t+1}) to $\mathcal{L}$
  if terminal **then**
    break
  end if
end for
$y \leftarrow$ VERIFYEPISODEOUTCOME$(\mathcal{E}, V, \mathcal{L})$
return $(y, \mathcal{L})$

Adaptive Intervention Gate. A diagnosis does not automatically trigger a correction action. The adaptive gate evaluates whether the observed evidence is sufficiently reliable and persistent for intervention. Its decision considers the diagnosed failure type, the target confidence, the reliability of the target-instance lock, persistent lack of progress, phase-specific execution evidence, and motion away from the selected target. The gate also uses failure-streak and high-confidence-streak conditions when repeated evidence is required.

For the object-, semantic-, and environment-related perturbation routes, the gate applies an additional progress guard. When the base policy is making recognized progress, the controller avoids inter-

rupting the current execution unless stronger wrong-target or execution-failure evidence is available. Consequently, failure diagnosis and correction activation are separate decisions, FIT-MAC can detect and record a failure while continuing to observe the base policy when the evidence is insufficient for safe correction.

Target Correction. Target correction operates on the target-level state used to construct the policy instruction. When the monitor and diagnoser indicate a wrong-target approach, an unreliable target lock, an ambiguous target instance, or an incorrect destination reference, the controller revisits the available grounding evidence. It can update the selected target, target-instance lock, or destination representation, and then rebuild the instruction supplied to the base policy.

Target correction changes which object or destination the policy is instructed to manipulate. It does not directly replace the low-level action controller. After the corrected grounding record is committed, the base policy generates the next action chunk using the revised target information.

Execution Correction. Execution correction operates after manipulation has started and maintains persistent phase-dependent controller state across action queries. The execution-correction controller dispatches the appropriate routine according to the current manipulation phase and the available failure evidence.

The pick-correction routine approaches the selected target, descends, closes the gripper, and verifies whether the target object has been lifted. If the lift test is unsuccessful, the controller can reopen the gripper and initiate another controlled descent. Once carrying evidence is confirmed, control is transferred to the placement routine.

The place-correction routine verifies that the object is being carried, resolves a valid destination anchor, lifts the carried object, transports it above the destination, corrects the object–destination alignment, descends, releases the object, and retreats. The geometric placement controller is applied only when the destination can be represented by a direct anchor. Instructions that require a more complex destination relation remain under the base-policy execution path.

After release, a post-release settling routine may keep the gripper open and apply a small upward motion for a configured number of queries. This step reduces the likelihood that the object remains in contact with the gripper or is immediately disturbed after release. Execution correction is cancelled when its validity conditions fail, when the current attempt transitions to an incompatible phase, or when the correction query budget is exhausted. In all such cases, the controller returns $\perp$, and control falls back to the base policy.

C.3 Representative Runtime Traces

In table 19 we present an event-level trace of a successful LIBERO-Spatial episode under Task perturbation. The instruction is *"Pick the akita black bowl not between the plate and the ramekin and place it on the plate."* The initial observation contains two detected black bowls, requiring the grounder to resolve the specified spatial relation before execution.

Target correction selects the intended object from detected candidates using the instruction and, when needed, VLM-based candidate selection. It then establishes a persistent target-instance lock. Execution correction applies phase-specific control to recover from failures during approach, grasping, lifting, transport, alignment, or placement.

C.4 Agent Prompts

We document the prompts and output interfaces for FIT-MAC's three agents, covering task planning, target grounding, and execution monitoring. The Task Planner Agent converts the instruction into a structured task description that specifies the intended manipulation. The Grounder Agent identifies the target instance and destination, linking the task description to the objects in the scene. The Monitor & Diagnoser Agent monitors execution and detects actionable failures that require corrective intervention. For each agent, we provide the prompt and expected output format to clarify its role and the information it passes to subsequent components. Together, these interfaces describe how FIT-MAC connects instruction interpretation, scene grounding, and failure diagnosis during task execution.

Table 19: Representative runtime trace of a successful LIBERO-Spatial episode under Task perturbation. Target correction resolves ambiguity between candidate bowls during grounding. Execution correction activates after a wrong-target approach and remains active through placement correction.

Query	Step	Phase	Event / evidence	Diagnosis and route	Controller state
1	10	Pick approach	Two bowl candidates are detected. Grounding resolves the spatial relation and selects the black bowl[1] with VLM confidence 0.95.	Target grounding; target correction completed.	Target-instance lock established; base policy starts execution.
5	42	Pick approach	The robot continues approaching the resolved target.	None; baseline execution.	Base policy continues with an eight-action chunk.
6	50	Pick approach	The monitor reports a wrong-target approach.	Execution failure; execution correction activated.	Hard-pick correction starts in approach-hover phase.
7	51	Pick approach	The correction state remains pending.	Execution failure; correction remains active.	Hard-pick controller continues with one corrective action.
30	74	Place transport	The pick attempt transitions successfully to placement. The object is reported as lifted and carried.	Successful pick-to-place transition.	Destination-locked place correction starts in lift-carry phase.
31	75	Place transport	The monitor reports insufficient placement progress and a placement stall.	Execution failure; correction remains active.	Destination is locked to the plate_1.
33	77	Place transport	The object has not reached the destination.	Execution failure; correction remains active.	Move-above-destination phase performs alignment.
55	99	Place transport	The controller progresses to the descent stage.	Execution failure; correction remains active.	Descent-to-destination phase; release height not yet reached.
59	103	Place descend	The object reaches the release condition.	Execution correction; release permitted.	Release phase enters the release-ready state.
60	104	Place release	The controller holds the release state while waiting for the gripper to open.	No new failure; correction remains pending.	Release-hold phase.
61	105	Place completion	The environment returns reward 1 and terminates successfully.	Successful completion.	Episode ends after the final release-hold action.

C.4.1 Task Planner Agent

The task planner converts the natural-language LIBERO instruction into a canonical instruction and a coarse-grained execution plan. The planner keeps the same granularity as the original instruction and does not unnecessarily split a simple pick-and-place instruction into separate locate, grasp, move, and release subtasks. A rule-based planner is used as the fallback, while a local language model can optionally produce the same structured interface.

The system prompt is:

Task Planner System Prompt

You are a LIBERO task decomposition agent.

Given a natural−language manipulation instruction, produce a structured task description. Output JSON only with the following keys:

canonical_instruction
plan
action
target_object
source_phrase
destination
destination_prep
destination_relation
confidence

Keep the plan at the same granularity as the instruction. Do not invent locate, grasp, move, or release steps when they are not explicitly required. Preserve object identities, spatial relations, and destination references. Return only valid JSON.

C.4.2 GROUNDER AGENT

The grounder first obtains object candidates from Grounding-DINO, parses the task-level target and spatial relation, and invokes a VLM candidate selector when the candidate set is ambiguous or the detector confidence is insufficient. The resulting target record is used to establish a target-instance lock that is maintained during subsequent manipulation.

For candidate-based grounding, the system prompt is:

Grounder System Prompt

You are a LIBERO grounding agent.

Follow the instruction to select the correct target object.

The green−labeled box is a reference object when present.
The red−labeled boxes are the only candidate target objects that may be selected.
Blue−labeled boxes are extra detections shown for context only and are not selectable.

Do not select the reference object, the robot arm, or any other non−target object.

Output JSON only with the following keys:
selected_index, confidence, reason

Use zero−based indexing. Choose the target by the source phrase and its spatial relation to the reference object. Do not choose by candidate order or detector score alone.

The corresponding user prompt is:

Grounder User Prompt

Instruction: <instruction>
Canonical instruction: <canonical_instruction>
Source phrase: <source_phrase>
Target object: <target_object>
Reference objects: <reference_objects>

Candidates:
<candidate_list>

Other detected instances:
<other_instances>

Selection rules:
– Select only a red candidate box.
– Use the source phrase and its relation to the reference object.
– Ignore candidate order unless multiple candidates are equally plausible.
– Do not select a reference object or an extra detection.
– Return only valid JSON.

C.4.3 MONITOR & DIAGNOSIS AGENT

The monitor tracks manipulation progress using deterministic state and motion signals. It does not directly replace the controller's action policy. The diagnoser consumes the monitor snapshot and assigns one of the failure labels from none, target failure,execution failure, or success. An optional language-model critic can provide an additional diagnosis, but its output is normalized, confidence-clamped, and merged with the rule-based evidence before it is used by the recovery router.

The diagnoser first checks task completion, then determines whether there is an actionable failure. If no attempt-end condition is present, it returns none. If a correction is already pending, it returns an intermediate diagnosis indicating that recovery is still in progress. Evidence for target failure includes target ambiguity, target-probe mismatch, wrong-target evidence, and an instruction–target mismatch. Evidence for execution failure includes phase stalls, repeated stalls, wrong-target approach after execution has started, failed grasp or lift, failed placement, and lack of motion progress.

When enabled, the optional critic receives the following system prompt:

Diagnosis System Prompt

You are a LIBERO critic agent.

Inspect the task instruction and the execution snapshot. Output JSON only with the following keys:

failure_type
reason
confidence
evidence_tags

failure_type must be one of:
none, target_failure, execution_failure, success.

Use target_failure when the target object is likely wrong, ambiguous, or mismatched with the instruction. Use execution_failure when the target seems correct but the robot stalled, failed to grasp, failed to place, or made no progress. Use none when there is no actionable failure. Keep the reason short and factual.

The critic user prompt is:

Diagnosis User Prompt

Task label: <active_instruction>
Snapshot:
{
"route": "<route>",
"phase": "<phase>",
"attempt_kind": "<attempt_kind>",
"attempt_id": <attempt_id>,
"phase_steps": <phase_steps>,
"stall_reason": "<stall_reason>",
"attempt_end_candidate": <true_or_false>,
"gripper_closed": <true_or_false>,
"target_lifted": <true_or_false>,
"target_selected": "<target_selected>",
"target_confidence": <target_confidence>,
"progress_flags": <progress_flags>,
"verifier_status": "<verifier_status>",
"target_probe": <target_probe>,
"rule_hint": "<rule_hint>",
"recent_contexts": <recent_contexts>,
"recent_diagnoses": <recent_diagnoses>
}

Return only valid JSON.

D VIDEO DEMONSTRATIONS