TEAM-VLA AGENTS: CHAIN-OF-THOUGHT REASONING FOR MULTI-AGENT EMBODIED AUTONOMY WITH CLOSED-LOOP RECOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

For Vision–Language–Action (VLA) based embodied robots, it is essential to establish an autonomous pipeline in open environments, which spans the entire process from goal-driven planning to execution verification and fault recovery. Existing methods often couple high-level planning with reactive execution, lacking explicit reasoning and closed-loop recovery. To address these limitations, we propose the TEAM-VLA Agents (TVA) framework, which adopts a novel paradigm that integrates explicit Chain-of-Thought (CoT) reasoning with multi-agent team collaboration. By leveraging role-based task decomposition and a context-sharing architecture, TVA constructs a CoT-driven autonomous system that unifies task planning, execution monitoring, as well as autonomous failure diagnosis and recovery within a single closed-loop process. This design not only ensures interpretable and traceable task execution but also effectively mitigates error cascades. Empirical evaluations demonstrate that TVA substantially improves success rates on long-horizon tasks and exhibits enhanced robustness and generalizable recovery capabilities across diverse execution failures. Our findings underscore that explicitly incorporating CoT-based reasoning, task verification, and self-recovery mechanisms is a critical step toward achieving reliable, scalable, and truly autonomous embodied intelligence.

1 INTRODUCTION

Vision–Language–Action (VLA) models have become an important paradigm for general-purpose robot control in Embodied Artificial Intelligence (Embodied AI). These models jointly encode visual observations, natural-language instructions, as well as robot states and then directly generate executable actions. In this way, they establish an end-to-end loop from environmental perception and task understanding to physical control, providing a unified foundation for generalization across tasks, objects, and environments (Zitkovich et al., 2023; Kim et al., 2024; Sapkota et al., 2025; Kawaharazuka et al., 2025; Wang et al., 2026).

Recent VLA research has advanced in various domains, such as continuous control, open-world generalization, cross-embodiment adaptation, and efficient deployment. π_0 introduces a flow-matching action expert, while FAST improves autoregressive action tokenization (Black et al., 2024; Pertsch et al., 2025). $\pi_{0.5}$ further enhances unseen-environment generalization through heterogeneous co-training (Intelligence et al., 2025). SmolVLA and OpenVLA-OFT target efficient deployment and fine-tuning (Shukor et al., 2025; Kim et al., 2025), whereas X-VLA and AtomicVLA uses embodiment-specific soft prompts and integrates task planning, atomic skills as well as extensible skill learning respectively (Zheng et al., 2025a; Zhang et al., 2026b). Generalist systems such as GR00T N1, Gemini Robotics, MolmoAct, and Unified VLA further extend VLA models toward humanoid control, spatial reasoning, and general-purpose embodied intelligence (Bjorck et al., 2025; Team et al., 2025; Lee et al., 2025; Wang et al., 2025).

Despite these advances, long-horizon and multi-step tasks in complex real-world environments remain challenging (Zhao et al., 2025; Huang et al., 2026b). Most existing approaches still rely on implicit reasoning within a single policy and tightly couple high-level language planning with low-level reactive control (Huang et al., 2026a; Liu et al., 2026c). Such paradigms generally lack explicit task

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.

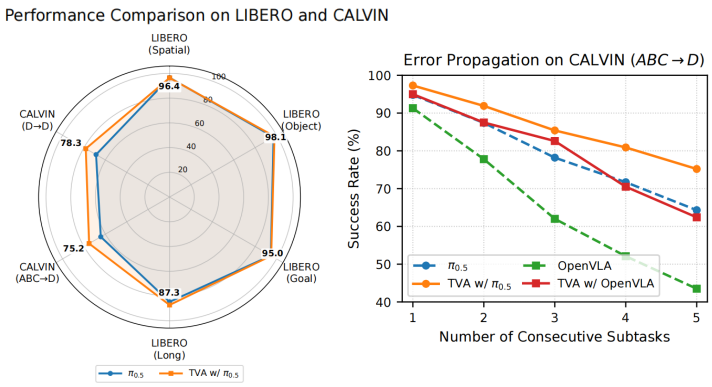


Figure 1: Cross-benchmark performance comparison and long-horizon error propagation analysis. Left: comparison between $\pi_{0.5}$ and TVA w/ $\pi_{0.5}$ across four LIBERO suites and two CALVIN evaluation settings, where the CALVIN results denote the success rates of completing the full five-subtask sequences. Right: success rates for completing one to five consecutive subtasks under the CALVIN ABC→D setting. As the execution horizon increases, TVA exhibits less performance degradation than the baseline models, demonstrating its effectiveness in mitigating error accumulation during long-horizon task execution.

decomposition, progress representation, outcome verification, as well as systematic fault-diagnosis and recovery mechanisms (Yang et al., 2025b; Sun et al., 2025). Consequently, they often fail to identify action deviations in time or revise subsequent decisions according to environmental feedback. When an intermediate step encounters a grasping failure, object slippage, or state deviation, the resulting local error may accumulate during subsequent execution and develop into a cascading failure, as illustrated in Figure 1. This error propagation ultimately increases human intervention and reduces robustness across objects and complex linguistic instructions.

In contrast to end-to-end approaches that rely on implicit reasoning within a single agent, we propose a systematic paradigm based on **explicit Chain-of-Thought (CoT) reasoning and multi-agent collaboration**. As illustrated in Figure 2, the planning, execution, and monitoring agents collaborate through role-based decomposition and a context-sharing mechanism that maintains task goals, execution progress, historical trajectories, and failure information. This design explicitly decouples high-level planning, low-level control, and execution verification, thereby improving the interpretability and robustness of long-horizon task execution. When failures occur, the agents share CoT information to diagnose the root cause, plan recovery strategies, infer recovery trajectories, and dynamically expand the skill library, enabling autonomous closed-loop recovery, as demonstrated in Figure 1. Our main contributions are summarized as follows:

- We propose TEAM-VLA Agents (TVA), a unified closed-loop framework that integrates task planning, VLA-based action execution, execution monitoring, failure diagnosis, and autonomous recovery through role-based agent decomposition and a context-sharing mechanism. By sharing task objectives, reasoning processes, execution states, and historical trajectories, TVA enables interpretable long-horizon task execution with clearly defined agent responsibilities.
- We achieve explicit decoupling of high-level language planning and low-level reactive control, boosting interpretability, robustness and error suppression while reducing human intervention. The introduction of standardized interfaces also enable plug-and-play integration of lightweight and spatially-augmented actuators to enhance reasoning and response speed.
- Experimental results verify our method's SOTA performance in long-horizon complex robotic manipulation tasks with elevated success and recovery rates for reliable embodied autonomy. The ablation studies confirm CoT-based role decomposition and self-recovery as core to system robustness, and the system shows superior generalization on multi-object scenarios and complex linguistic instructions.

2 Background and Related Work

2.1 Explicit Reasoning and Role-Specialized Multi-Agent Collaboration

Recent work introduces explicit reasoning into VLA models for long-horizon planning. CoT-VLA is able to predicts future visual goals (Zhao et al., 2025). ThinkAct and Hume connect high-level reasoning with control through latent planning and value-guided System-2 selection (Huang et al., 2026b; Song et al., 2025). OneTwoVLA, GraphCoT-VLA, and VLA-Reasoner respectively explore adaptive reasoning, structured 3D reasoning, and MCTS-based action search (Lin et al., 2025a; Huang et al., 2026c; Guo et al., 2025b), while some other methods study action-level, verbalizable latent, and spatio-temporal reasoning chains (Zhong et al., 2026; Huang et al., 2026a; Liu et al., 2026c). Agentic Robot, InteractGen, and DynamicVLA further distribute planning, execution, verification, reflection, and temporal reasoning across specialized components (Yang et al., 2025b; Sun et al., 2025; Xie et al., 2026). However, most still lack unified synchronization of reasoning, execution, verification, and recovery throughout the full control loop.

2.2 Shared Context Memory for Progress Tracking and Execution Verification

Long-horizon manipulation requires persistent task context to track completed subtasks, environmental changes, and execution outcomes. MemoryVLA and MAP-VLA are able to retain or retrieve task-relevant visual–semantic information (Shi et al., 2025; Li et al., 2025), while Long-VLA models phase-dependent context and HELM combines episodic memory with state verification, rollback, and replanning (Fan et al., 2025; Zeng et al., 2026). Goal2Skill and tool-aligned VLA agents use structured memory and progress feedback for adaptive replanning, whereas EvoVLA incorporates long-term memory into self-evolving policy optimization (Liu et al., 2026b; Lei et al., 2026; Liu et al., 2025). However, these memory mechanisms are typically confined to individual planners, policies, or verifiers, rather than serving as a unified context shared across role-specialized agents.

2.3 Failure Diagnosis, Recovery Trajectory Generation, and Skill Expansion

Recent VLA studies address failures through supervisory correction, recovery-aware data construction, rollback-based replanning, and policy-level safety optimization. For example, both FailSafe, FPC-VLA, and RoboFAC can detect execution failures and generate corrective actions or recovery supervision (Lin et al., 2025b; Yang et al., 2026; Ye et al., 2025), while HELM and Goal2Skill are using execution history and feedback for verification, rollback, and adaptive replanning (Zeng et al., 2026; Liu et al., 2026b). In addition, SafeVLA improves policy safety through constrained learning, RobustVLA evaluates robustness against multimodal perturbations, and VLAC introduces critic-based progress evaluation for real-world policy optimization (Zhang et al., 2026a; Guo et al., 2025a; Zhai et al., 2025). AtomicVLA further supports compositional long-horizon execution through an extensible atomic skill library (Zhang et al., 2026b). Nevertheless, existing methods often separate failure diagnosis, recovery generation, and skill accumulation, which means there is a lack on a closed-loop that unifies planning, execution, monitoring, and recovery through a shared task context.

3 Method

3.1 Framework Overview

Given a high-level instruction $\mathbf{x}$, the observation at time t is $\mathbf{o}_t = (\mathbf{I}_t, \mathbf{s}_t)$, where $\mathbf{I}_t$ and $\mathbf{s}_t$ denote the visual observation and robot state, respectively. As shown in Figure 2, TVA consists of a planner $\mathcal{A}_P$, a VLA executor $\mathcal{A}_E$, and a monitor $\mathcal{A}_M$. The three agents communicate through a shared context memory $\mathcal{M}_t$, which maintains task objectives, execution progress, historical observations, and failure information. This role-specialized design decouples high-level reasoning, low-level action generation, and execution verification, forming a closed-loop of planning, execution, monitoring, and recovery.

The planner decomposes the instruction into a sequence of structured subtasks:

$$\mathcal{P} = \{\mathbf{u}_i\}_{i=1}^{N}, \quad \mathbf{u}_i = \langle \sigma_i, \omega_i, \ell_i \rangle, \tag{1}$$

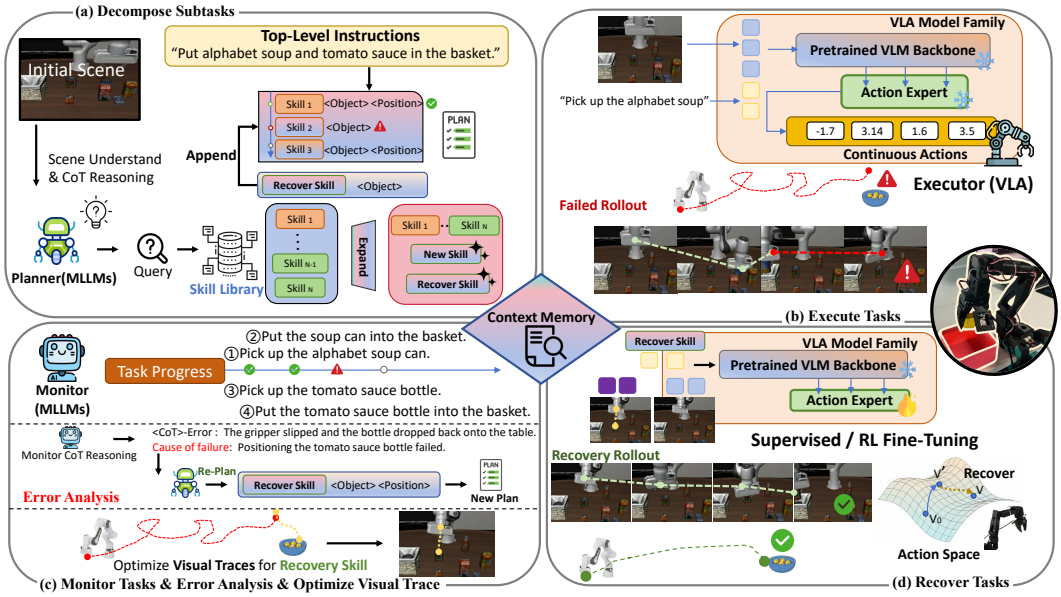


Figure 2: Overview of TEAM-VLA Agents. (a) The planner decomposes high-level instructions into structured subtasks using CoT reasoning and an extensible skill library. (b) The VLA executor generates continuous actions and records failed rollouts in the context memory. (c) The monitor tracks task progress, diagnoses failures, and infers recovery skills and trajectories. (d) Recovery experiences are used to refine the VLA policy and expand the skill library, forming an autonomous planning–execution–monitoring–recovery loop.

where N is the number of subtasks, and σ_i, ω_i, and ℓ_i denote the manipulation skill, target object, and spatial goal of the i-th subtask, respectively. We use i_t to denote the index of the active subtask at time t.

3.2 Planner Agent

The planner employs a Multimodal Large Language Model (MLLM) to directly generate a structured plan conditioned on the instruction, initial observation, skill library, and shared execution context:

$$\mathcal{P} = \mathcal{A}_P\big(\mathbf{x}, \mathbf{o}_0, \mathcal{B}, \mathcal{M}_0\big). \tag{2}$$

Each subtask follows the structured representation $\mathbf{u}_i = \langle \sigma_i, \omega_i, \ell_i \rangle$, specifying the manipulation skill, target object, and spatial goal, respectively. The planner generates the subtask sequence directly, without explicit candidate scoring or ranking. The standardized representation provides a unified task interface for the VLA executor. The planning prompt instructs the MLLM to align the subtasks with the instruction, account for the observed scene and available skills,and order the subtasks according to their dependencies.

3.3 VLA Executor

The VLA executor grounds the active subtask into continuous robot actions. It conditions the original VLA policy on visual observations, robot states, the current subtask, and the retrieved context:

$$\mathbf{z}_t = f_{\text{VLM}}(\mathbf{I}_t, \mathbf{u}_{i_t}, \mathbf{c}_t) \oplus f_s(\mathbf{s}_t), \quad \mathbf{A}_t \sim \pi_{\theta_E}(\cdot \mid \mathbf{z}_t), \tag{3}$$

where f_{VLM} and f_s denote the VLM and robot-state encoders respectively, and $\oplus$ denotes feature concatenation. The context feature $\mathbf{c}_t$ is retrieved from $\mathcal{M}_t$, while $\mathbf{z}_t$ is the resulting condition representation. Moreover, π_{θ_E} denotes the VLA policy parameterized by θ_E, and $\mathbf{A}_t = [\mathbf{a}_t, \ldots, \mathbf{a}_{t+H-1}]$ is an H-step action chunk.

TVA does not alter the architecture or training objective of the underlying VLA model. Autoregressive, diffusion-based, and flow-matching policies can therefore be integrated without modifying

their original inference procedures. The monitor may interrupt an action chunk once subtask success or execution failure is detected.

3.4 Monitor Agent and Context Sharing

The shared context memory maintains both task-level and execution-level information:

$$\mathcal{M}_t^- = \big\{\mathbf{x}, \mathcal{P}, \mathbf{u}_{i_t}, \mathbf{o}_{1:t}, \mathbf{a}_{1:t-1}, \hat{y}_{1:t-1}, \hat{e}_{1:t-1}\big\}. \tag{4}$$

Here, $y_{1:t}$ and $e_{1:t}$ denote the historical execution states and failure causes respectively. The planner uses this context for task progression and local replanning, the executor retrieves subtask-relevant information, and the monitor compares the current observation with historical execution states.

For the active subtask, the monitor directly evaluates execution using the current and previous observations, the subtask specification, and the shared execution history. It determines whether the subtask is still in progress, has been completed, or has failed, and identifies the failure cause when applicable:

$$(\hat{y}_t, \hat{e}_t) = \mathcal{A}_M\big(\mathbf{o}_{t-1:t}, \mathbf{u}_{i_t}, \mathcal{M}_t^-\big), \tag{5}$$

where $\hat{y}_t \in \{\mathrm{Running}, \mathrm{Success}, \mathrm{Failure}\}$, and $\hat{e}_t$ denotes the predicted failure cause when $\hat{y}_t = \mathrm{Failure}$ and is set to None otherwise. Here, $\mathcal{M}_t^-$ denotes the shared context available before the current monitoring result is recorded.

The monitor assesses whether the observed state satisfies the intended outcome of the active subtask and uses the execution history to inform its judgment. A $\mathrm{Running}$ prediction allows execution to continue, a $\mathrm{Success}$ prediction advances execution to the next subtask or terminates the task if all subtasks are complete, and a $\mathrm{Failure}$ prediction triggers recovery. The predicted execution state and failure cause are then recorded in the shared context memory.

3.5 Recovery Trajectory Inference and Skill Expansion

Once a failure is detected, the planner generates a recovery subtask using the failure cause and shared execution history:

$$\mathbf{u}_t^r = \mathcal{A}_P(\mathbf{x}, \mathbf{u}_{i_t}, \hat{e}_t, \mathcal{M}_t, \mathcal{B}). \tag{6}$$

The recovery subtask $\mathbf{u}_t^r$ specifies the recovery skill, target object, and desired spatial state.

The monitor then selects multiple visual points associated with the robot end-effector, manipulated object, and target region. A multi-point tracking model predicts their future image-space trajectories:

$$\widehat{\mathcal{T}}_t^r = \mathcal{F}_{\mathrm{track}}(\mathbf{I}_{t-L:t}, \mathbf{u}_t^r, \hat{e}_t, \mathcal{M}_t), \tag{7}$$

where L is the historical observation-window length and $\mathcal{F}_{\mathrm{track}}$ denotes the multi-point trajectory predictor. The predicted trajectory set is defined as $\widehat{\mathcal{T}}_t^r = \{\hat{\mathbf{p}}_{t+h}^m\}_{m=1,h=1}^{M,H_r}$, where M is the number of tracked points, H_r is the recovery horizon, and $\hat{\mathbf{p}}_{t+h}^m \in \mathbb{R}^2$ is the predicted image-space coordinate of the m-th point at future step $t+h$.

The predicted tracks are rendered onto the current observation to form a recovery-trajectory image, which provides visual guidance to the VLA executor:

$$\mathbf{I}_t^r = \mathrm{Render}(\mathbf{I}_t, \widehat{\mathcal{T}}_t^r), \quad \mathbf{A}_t^r \sim \pi_{\theta_E}(\cdot \mid \mathbf{I}_t^r, \mathbf{s}_t, \mathbf{u}_t^r, \mathbf{c}_t). \tag{8}$$

Here, $\mathrm{Render}(\cdot)$ overlays the predicted point tracks on the current observation, and $\mathbf{A}_t^r$ denotes the generated recovery action chunk.

After executing $\mathbf{A}_t^r$, the monitor verifies the resulting state. If recovery succeeds, TVA resumes the original plan. Otherwise, the new observation and failure record are written into $\mathcal{M}_t$ for another recovery iteration.

A successful and novel recovery behavior is summarized as a reusable skill σ_r and added to the skill library:

$$\mathcal{B} \leftarrow \mathcal{B} \cup \{\sigma_r\} \quad \text{if} \quad \hat{y}_t^r = \mathrm{Success} \ \wedge \ \nu(\sigma_r) > \delta_n. \tag{9}$$

Here, $\hat{y}_t^r$ denotes the recovery result, $\nu(\sigma_r)$ measures the novelty of the recovered skill relative to the existing library, and δ_n is the novelty threshold. Each new skill stores its language description, applicable failure type, recovery trajectory image, and successful execution record. TVA therefore converts successful recoveries into reusable capabilities while preserving the original VLA training and fine-tuning pipeline. The overall pipeline of TEAM-VLA Agents can be found in Appendix A.

Table 1: LIBERO (Liu et al., 2023) benchmark results comparing the success rates (SR) of representative VLA baselines and TVA across four task suites.

Method	Spatial SR (%) ↑	Object SR (%) ↑	Goal SR (%) ↑	Long SR (%) ↑	Average SR (%) ↑
π_0 (Black et al., 2024)	87.1	63.5	89.7	48.6	72.2
$\pi_{0.5}$ (Intelligence et al., 2025)	**96.6**	97.2	94.6	84.8	93.3
TVA w/ $\pi_{0.5}$ (Ours)	96.4	**98.1**	**95.0**	**87.3**	**94.2**
OpenVLA (Kim et al., 2024)	84.7	88.4	79.2	53.7	76.5
OpenVLA-OFT (Kim et al., 2025)	91.6	95.3	90.6	86.5	91.0
TVA w/ OpenVLA-OFT (Ours)	**93.0**	**95.6**	**91.6**	**90.4**	**92.7**
TraceVLA (Zheng et al., 2025b)	84.6	85.2	75.1	54.1	74.8
SpatialVLA (Qu et al., 2025)	88.2	89.9	78.6	55.5	78.1
SmolVLA (Shukor et al., 2025)	87.1	93.0	88.0	63.3	82.9
TVA w/ SmolVLA (Ours)	**89.1**	**93.9**	**90.4**	**71.5**	**86.2**

Table 2: CALVIN (Mees et al., 2022) benchmark results comparing representative baselines and TVA under the ABC→D and D→D evaluation settings. The table reports the average success rates of completing one to five consecutive instructions over 1,000 evaluation chains, together with the average completed sequence length (Avg. Len.).

Train→Test	**Method**	**No. Instructions in a Row (1,000 Chains)**					**Avg. Len.**
		1	**2**	**3**	**4**	**5**	
ABC→D	OpenVLA (Kim et al., 2024)	91.3%	77.8%	62.0%	52.1%	43.5%	3.27
	Seer (Tian et al., 2025)	96.3%	91.6%	**86.1%**	80.3%	74.0%	4.28
	$\pi_{0.5}$ (Intelligence et al., 2025)	94.8%	87.4%	78.2%	71.7%	64.3%	3.99
	TVA w/ $\pi_{0.5}$ (Ours)	**97.3%**	**91.9%**	85.4%	**80.9%**	**75.2%**	**4.41**
D→D	MDT (Reuss et al., 2024)	93.7%	84.5%	74.1%	64.4%	55.6%	3.72
	RoboUniView (Liu et al., 2024)	96.2%	88.8%	77.6%	66.6%	56.3%	3.85
	FLOWER (Reuss et al., 2025)	97.4%	92.4%	86.9%	81.3%	74.9%	4.35
	$\pi_{0.5}$ (Intelligence et al., 2025)	96.2%	90.1%	85.4%	78.2%	68.7%	4.22
	TVA w/ $\pi_{0.5}$ (Ours)	**98.0%**	**94.7%**	**90.4%**	**86.5%**	**78.3%**	**4.59**

4 EXPERIMENTS

In this section, we evaluate TVA through extensive experiments in both simulated and real-world environments. We assess its effectiveness through comparisons with baseline methods, ablation studies, real-robot deployment, robustness-sensitivity analysis and verification-frequency analysis. Additional details, including implementation details, prompt and context design, experimental settings, and complete quantitative results, are provided in the `Appendix`.

4.1 COMPARATIVE EVALUATION

As shown in Table 1, TVA consistently improves the performance of different VLA backbones on LIBERO (Liu et al., 2023). When integrated with $\pi_{0.5}$ (Intelligence et al., 2025), OpenVLA-OFT (Kim et al., 2025), and SmolVLA (Shukor et al., 2025), TVA increases the average success rate by 0.9, 1.7, and 3.3 percentage points, respectively. The gains are most pronounced on LIBERO-Long, indicating that TVA is particularly effective in improving continuous execution and failure recovery in long-horizon tasks.

Table 2 further shows that TVA consistently outperforms representative baselines on CALVIN (Mees et al., 2022). Under the ABC→D and D→D settings, TVA achieves five-instruction success rates of 75.2% and 78.3%, together with average completed sequence lengths of 4.41 and 4.59, respectively, yielding the best overall results. Its advantage becomes more pronounced as the instruction sequence grows longer, demonstrating that TVA effectively mitigates error propagation across subtasks and improves the stability and robustness of long-horizon execution.

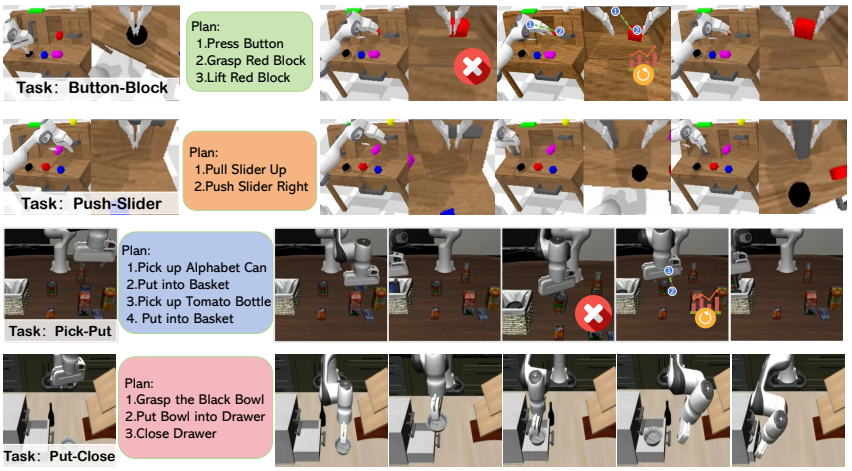


Figure 3: Inference examples of TVA for long-horizon manipulation tasks on the LIBERO and CALVIN benchmarks.

Table 3: Ablation study of TVA w/ $\pi_{0.5}$ on LIBERO and CALVIN. The table reports the average success rate on LIBERO and the success rates of completing five consecutive instructions under the CALVIN ABC→D and D→D settings.

Method	LIBERO Avg. SR (%) ↑	CALVIN ABC→D 5 Instr. SR (%) ↑	CALVIN D→D 5 Instr. SR (%) ↑
w/o TVA	93.3	64.3	68.7
w/o Monitor	93.5	68.1	71.9
w/o CoT Planner	93.9	70.6	74.1
w/o Context	93.9	72.3	75.6
w/o visual trace	93.8	71.5	74.9
w/o skill expansion	94.0	73.1	76.4
Full TVA	**94.2**	**75.2**	**78.3**

Figure 3 shows inference examples of TVA on the LIBERO and CALVIN benchmarks. TVA decomposes long-horizon manipulation instructions into sequential subtasks and coordinates planning, execution, monitoring, and recovery. By continuously tracking progress and correcting intermediate failures, TVA ensures coherent execution across stages and completes complex manipulation tasks.

4.2 Ablation Study

Table 3 shows that removing TVA reduces CALVIN five-instruction success rates by 10.9 and 9.6 percentage points under ABC→D and D→D, respectively, compared with only 0.9 points on LIBERO. Removing the Monitor causes the largest degradation among individual components, followed by the CoT Planner, visual trace, Context, and skill expansion, suggesting that TVA primarily benefits long-horizon execution.

Table 4 further shows progressive gains from enhanced planning and monitoring to Full TVA, with LIBERO-Long success increasing from 76.8% to 87.3% and CALVIN average chain length from 3.82 to 4.59. Full TVA outperforms retry-only monitoring on both metrics, supporting the benefit of its integrated recovery mechanisms.

4.3 Real-Robot Deployment

To examine TVA in a physical manipulation setting, we deploy the framework with $\pi_{0.5}$ on a LeRobot SO101 follower robot arm (6-DoF). The visual sensing system consists of an external third-person RGB camera accessed through OpenCV, providing images at a resolution of 640×480 pixels

Table 4: Progressive Component Ablation Study.

Method	LIBERO-Long Success (%) ↑	CALVIN ABC→D Avg. Chain Length ↑
$\pi_{0.5}$ w/ Naive Planner	76.8	3.82
$\pi_{0.5}$ w/ CoT Planner	80.5	4.08
$\pi_{0.5}$ w/ Monitor (retry-only)	83.6	4.31
Full TVA	**87.3**	**4.59**

and a capture rate of 30 fps. This configuration provides visual observations of the robot, target objects, and surrounding workspace during execution.

Figure 4 presents representative real-world manipulation sequences together with visualizations of the multi-point trajectories. The execution sequences show the robot grasping yellow objects and transferring them to a plate or a red container. The trajectory visualizations overlay colored points and dashed lines on the camera observations to illustrate the image-space relationships and motion guidance associated with the manipulation. Together, these examples qualitatively illustrate physical task execution and the visual trajectory representation used by TVA.

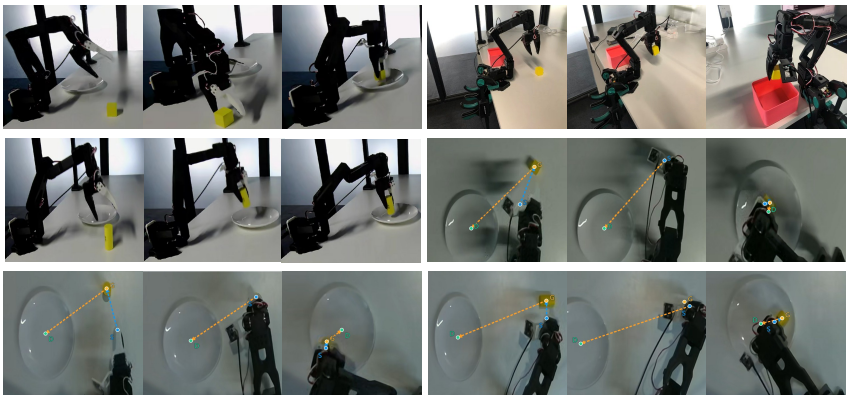


Figure 4: Real-world manipulation and multi-point trajectory visualization on the LeRobot SO101 robot arm. The arm grasps yellow objects and transfers them to a plate or a red container. Colored points and dashed lines represent image-space trajectories for motion guidance.

4.4 Robustness-Sensitivity Analysis

We evaluate TVA in terms of autonomous recovery and failure detection. Recovery Success Rate (RSR) measures the proportion of detected failures resolved without human intervention, while false positive rate (FPR) and false negative rate (FNR) assess monitoring reliability.

As shown in Table 5, TVA w/ $\pi_{0.5}$ achieves the highest RSR: 41.8% on LIBERO-Long, 42.6% on the LIBERO average, and 39.7%/44.5% on CALVIN ABC→D/D→D. The lower RSR under ABC→D is consistent with the added challenge of distribution shifts. SmolVLA also achieves competitive recovery performance, supporting TVA's compatibility with different backbones.

Table 6 reports FPRs of 10.4–11.8% and FNRs of 7.5–8.8%. TVA prioritizes reducing missed failures to limit error propagation in long-horizon tasks, while tolerating moderate verification and recovery overhead from false alarms. Together, monitoring and closed-loop recovery support robust execution across the evaluated benchmarks and backbones.

4.5 Verification-Frequency Analysis

We analyze the trade-off between verification frequency and execution efficiency by validating every 10, 20, or 50 action steps. As shown in Figure 5a, more frequent verification consistently improves

Table 5: Autonomous recovery performance of TVA with different VLA backbones on LIBERO and CALVIN. Recovery Success Rate (RSR) measures the percentage of detected execution failures that are successfully recovered.

Method	LIBERO		CALVIN	
	Long RSR (%) ↑	Avg. RSR (%) ↑	ABC→D RSR (%) ↑	D→D RSR (%) ↑
TVA w/ $\pi_{0.5}$ (Ours)	**41.8**	**42.6**	**39.7**	**44.5**
TVA w/ OpenVLA-OFT (Ours)	35.9	37.1	–	–
TVA w/ SmolVLA (Ours)	38.6	39.8	–	–

Table 6: Failure detection performance of TVA with different VLA backbones on LIBERO and CALVIN. FPR and FNR denote false positive and false negative rates, respectively. Lower is better.

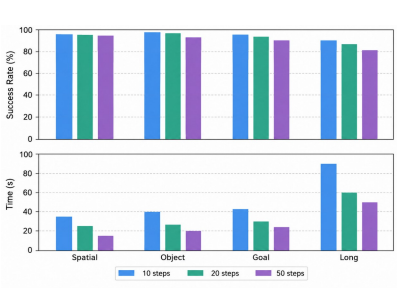
Method	FPR (%) ↓	FNR (%) ↓
TVA w/ $\pi_{0.5}$ (Ours)	**10.4**	8.6
TVA w/ OpenVLA-OFT (Ours)	10.6	**7.5**
TVA w/ SmolVLA (Ours)	11.8	8.8

success rates on LIBERO, with the advantage becoming more pronounced on LIBERO-Long. However, validating every 10 steps introduces substantially higher execution time, whereas a 50-step interval reduces latency at the cost of a clear performance degradation, particularly for long-horizon tasks.

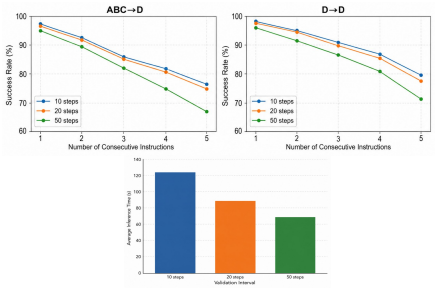
Figure 5b shows that performance gaps between validation intervals widen as instruction chains grow longer. For five consecutive instructions, the 50-step interval performs markedly worse than the 10- and 20-step intervals under both ABC→D and D→D. Increasing the interval from 10 to 20 steps reduces average inference time from approximately 125 s to 89 s while maintaining comparable success rates. We therefore use a 20-step interval by default to balance recovery responsiveness, long-horizon robustness, and computational efficiency. All inference latencies were measured on an NVIDIA GeForce RTX 5090 GPU.

5 Conclusion

This paper presents TVA, a multi-agent framework that unifies task planning, VLA execution, monitoring, diagnosis, and closed-loop recovery through shared context. Upon failure, TVA jointly infers recovery skills and trajectories, verifies the recovered state, and stores successful behaviors in an expandable skill library. This reduces error propagation and improves robustness and interpretability in long-horizon manipulation, supporting autonomous reasoning and recovery capabilities in embodied robots (Feng et al., 2025; Liu et al., 2026a).



(a) LIBERO benchmark.



(b) CALVIN benchmark.

Figure 5: Trade-off between validation frequency and the performance of TVA w/ $\pi_{0.5}$ on the LIBERO and CALVIN benchmarks.

AI USE STATEMENT

We used generative AI tools solely to assist with code development and manuscript polishing. No other uses of generative AI tools were involved in this work.

The authors reviewed the AI-assisted code and verified its correctness through testing. All AI-assisted textual revisions were reviewed to ensure that they accurately preserved the intended meaning and technical content. We take full responsibility for the final content of this work, including all code and text produced with the assistance of generative AI.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, the main text describes the TVA framework, its components, and the evaluation protocols on LIBERO and CALVIN. The appendix provides additional implementation details, model configurations, prompts, and experimental settings. Code and instructions for reproducing the experiments are available at https://anonymous.4open.science/r/teamvla88.

REFERENCES

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Yiguo Fan, Pengxiang Ding, Shuanghao Bai, Xinyang Tong, Yuyang Zhu, Hongchao Lu, Fengqi Dai, Wei Zhao, Yang Liu, Siteng Huang, et al. Long-vla: Unleashing long-horizon capability of vision language action model for robot manipulation. *arXiv preprint arXiv:2508.19958*, 2025.

Yunhai Feng, Jiaming Han, Zhuoran Yang, Xiangyu Yue, Sergey Levine, and Jianlan Luo. Reflective planning: Vision-language models for multi-stage long-horizon robotic manipulation. *arXiv preprint arXiv:2502.16707*, 2025.

Jianing Guo, Zhenhong Wu, Chang Tu, Yiyao Ma, Xiangqi Kong, Zhiqian Liu, Jiaming Ji, Shuning Zhang, Yuanpei Chen, Kai Chen, et al. On robustness of vision-language-action model against multi-modal perturbations. *arXiv preprint arXiv:2510.00037*, 2025a.

Wenkai Guo, Guanxing Lu, Haoyuan Deng, Zhenyu Wu, Yansong Tang, and Ziwei Wang. Vla-reasoner: Empowering vision-language-action models with reasoning via online monte carlo tree search. *arXiv preprint arXiv:2509.22643*, 2025b.

Chi-Pin Huang, Yunze Man, Zhiding Yu, Min-Hung Chen, Jan Kautz, Yu-Chiang Frank Wang, and Fu-En Yang. Fast-thinkact: Efficient vision-language-action reasoning via verbalizable latent planning. *arXiv preprint arXiv:2601.09708*, 2026a.

Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Frank Wang, and Fu-En Yang. Thinkact: Vision-language-action reasoning via reinforced visual latent planning. *Advances in Neural Information Processing Systems*, 38:82782–82802, 2026b.

Helong Huang, Min Cen, Kai Tan, Xingyue Quan, Guowei Huang, and Hong Zhang. Graphcot-vla: A 3d spatial-aware reasoning vision-language-action model for robotic manipulation with ambiguous instructions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18324–18332, 2026c.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Kento Kawaharazuka, Jihoon Oh, Jun Yamada, Ingmar Posner, and Yuke Zhu. Vision-language-action models for robotics: A review towards real-world applications. *IEEE Access*, 2025.

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

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, et al. Molmoact: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025.

Zixing Lei, Changxing Liu, Yichen Xiong, Minhao Xiong, Yuanzhuo Ding, Zhipeng Zhang, Weixin Li, and Siheng Chen. Towards long-horizon embodied agents with tool-aligned vision-language-action models. *arXiv preprint arXiv:2605.13119*, 2026.

Runhao Li, Wenkai Guo, Zhenyu Wu, Changyuan Wang, Haoyuan Deng, Zhenyu Weng, Yap-Peng Tan, and Ziwei Wang. Map-vla: Memory-augmented prompting for vision-language-action model in robotic manipulation. *arXiv preprint arXiv:2511.09516*, 2025.

Fanqi Lin, Ruiqian Nai, Yingdong Hu, Jiacheng You, Junming Zhao, and Yang Gao. Onetwovla: A unified vision-language-action model with adaptive reasoning. *arXiv preprint arXiv:2505.11917*, 2025a.

Zijun Lin, Jiafei Duan, Haoquan Fang, Dieter Fox, Ranjay Krishna, Cheston Tan, and Bihan Wen. Failsafe: Reasoning and recovery from failures in vision-language-action models. *arXiv preprint arXiv:2510.01642*, 2025b.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Fanfan Liu, Feng Yan, Liming Zheng, Chengjian Feng, Yiyang Huang, and Lin Ma. Robouniview: Visual-language model with unified view representation for robotic manipulation. *arXiv preprint arXiv:2406.18977*, 2024.

Isabella Liu, An-Chieh Cheng, Rui Yan, Geng Chen, Ri-Zhao Qiu, Xueyan Zou, Sha Yi, Hongxu Yin, Xiaolong Wang, and Sifei Liu. Long-horizon manipulation via trace-conditioned vla planning. *arXiv preprint arXiv:2604.21924*, 2026a.

Zeting Liu, Zida Yang, Zeyu Zhang, and Hao Tang. Evovla: Self-evolving vision-language-action model. *arXiv preprint arXiv:2511.16166*, 2025.

Zhen Liu, Xinyu Ning, Zhe Hu, Xinxin Xie, Weize Li, Zhipeng Tang, Chongyu Wang, Zejun Yang, Hanlin Wang, Yitong Liu, et al. Goal2skill: Long-horizon manipulation with adaptive planning and reflection. *arXiv preprint arXiv:2604.13942*, 2026b.

Zhuoyang Liu, Jiaming Liu, Hao Chen, Jiale Yu, Ziyu Guo, Chengkai Hou, Chenyang Gu, Xiangju Mi, Renrui Zhang, Kun Wu, et al. Last $_{\{0\}}$: Latent spatio-temporal chain-of-thought for robotic vision-language-action model. *arXiv preprint arXiv:2601.05248*, 2026c.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

Moritz Reuss, Ömer Erdinç Yağmurlu, Fabian Wenzel, and Rudolf Lioutikov. Multimodal diffusion transformer: Learning versatile behavior from multimodal goals. *arXiv preprint arXiv:2407.05996*, 2024.

Moritz Reuss, Hongyi Zhou, Marcel Rühle, Ömer Erdinç Yağmurlu, Fabian Otto, and Rudolf Lioutikov. Flower: Democratizing generalist robot policies with efficient vision-language-action flow policies. *arXiv preprint arXiv:2509.04996*, 2025.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 2024.

Ranjan Sapkota, Yang Cao, Konstantinos I Roumeliotis, and Manoj Karkee. Vision-language-action models: Concepts, progress, applications and challenges. *arXiv preprint arXiv:2505.04769*, 2025.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Haoming Song, Delin Qu, Yuanqi Yao, Qizhi Chen, Qi Lv, Yiwen Tang, Modi Shi, Guanghui Ren, Maoqing Yao, Bin Zhao, et al. Hume: Introducing system-2 thinking in visual-language-action model. *arXiv preprint arXiv:2505.21432*, 2025.

Nan Sun, Bo Mao, Yongchang Li, Chenxu Wang, Di Guo, and Huaping Liu. Transforming monolithic foundation models into embodied multi-agent architectures for human-robot collaboration. *arXiv preprint arXiv:2512.00797*, 2025.

Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 92033–92052, 2025.

Qiuyue Wang, Mingsheng Li, Jian Guan, Jinhui Ye, Sicheng Xie, Yitao Liu, Junhao Chen, Zhixuan Liang, Jie Zhang, Xintong Hu, et al. Qwen-vla: Unifying vision-language-action modeling across tasks, environments, and robot embodiments. *arXiv preprint arXiv:2605.30280*, 2026.

Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified vision-language-action model. *arXiv preprint arXiv:2506.19850*, 2025.

Haozhe Xie, Beichen Wen, Jiarui Zheng, Zhaoxi Chen, Fangzhou Hong, Haiwen Diao, and Ziwei Liu. Dynamicvla: A vision-language-action model for dynamic object manipulation. *arXiv preprint arXiv:2601.22153*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Yifan Yang, Zhixiang Duan, Tianshi Xie, Fuyu Cao, Pinxi Shen, Peili Song, Chenyang Zhao, Piaopiao Jin, Guokang Sun, Shaoqing Xu, et al. Fpc-vla: A vision-language-action framework with a supervisor for failure prediction and correction. *Expert Systems with Applications*, pp. 131742, 2026.

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

Zewei Ye, Weifeng Lu, Minghao Ye, Tao Lin, Shuo Yang, Junchi Yan, and Bo Zhao. Robofac: A comprehensive framework for robotic failure analysis and correction. *arXiv preprint arXiv:2505.12224*, 2025.

Zijian Zeng, Fei Ding, Huiming Yang, and Xianwei Li. Helm: Harness-enhanced long-horizon memory for vision-language-action manipulation. *arXiv preprint arXiv:2604.18791*, 2026.

Shaopeng Zhai, Qi Zhang, Tianyi Zhang, Fuxian Huang, Haoran Zhang, Ming Zhou, Shengzhe Zhang, Litao Liu, Sixu Lin, and Jiangmiao Pang. A vision-language-action-critic model for robotic real-world reinforcement learning. *arXiv preprint arXiv:2509.15937*, 2025.

Borong Zhang, Yuhao Zhang, Jiaming Ji, Yingshan Lei, Juntao Dai, Yuanpei Chen, and Yaodong Yang. Safevla: Towards safety alignment of vision-language-action model via constrained learning. *Advances in Neural Information Processing Systems*, 38:153335–153373, 2026a.

Likui Zhang, Tao Tang, Zhihao Zhan, Xiuwei Chen, Zisheng Chen, Jianhua Han, Jiangtong Zhu, Pei Xu, Hang Xu, Hefeng Wu, et al. Atomicvla: Unlocking the potential of atomic skill learning in robots. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20743–20754, 2026b.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 1702–1713, 2025.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025a.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. In *International Conference on Learning Representations*, volume 2025, pp. 54277–54296, 2025b.

Linqing Zhong, Yi Liu, Yifei Wei, Ziyu Xiong, Si Liu, and Guanghui Ren. Acot-vla: Action chain-of-thought for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8152–8162, 2026.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

A Detailed Algorithm of TEAM-VLA Agents

Algorithm 1 Closed-Loop Execution of TVA

Require: Instruction $\mathbf{x}$, observation $\mathbf{o}_0$, skill library $\mathcal{B}$
$\mathcal{P}, \mathcal{M}_0 \leftarrow \textsc{Plan}(\mathbf{x}, \mathbf{o}_0, \mathcal{B})$
for $\mathbf{u}_i \in \mathcal{P}$ **do**
  $\mathbf{A}_t \leftarrow \mathrm{VLA}(\mathbf{o}_t, \mathbf{u}_i, \mathcal{M}_t)$; execute $\mathbf{A}_t$
  $\hat{y}_t, \hat{e}_t \leftarrow \textsc{Monitor}(\mathbf{o}_{t:t+1}, \mathbf{u}_i, \mathcal{M}_t)$; update $\mathcal{M}_t$
  if $\hat{y}_t = \mathrm{Failure}$ **then**
    $\mathbf{u}_t^r, \widehat{\mathcal{T}}_t^r \leftarrow \textsc{Recover}(\hat{e}_t, \mathbf{u}_i, \mathcal{M}_t, \mathcal{B})$
    $\mathbf{A}_t^r \leftarrow \mathrm{VLA}(\textsc{Render}(\mathbf{I}_t, \widehat{\mathcal{T}}_t^r), \mathbf{u}_t^r, \mathcal{M}_t)$
    Verify recovery; update $\mathcal{M}_t$ and expand $\mathcal{B}$ if novel
  end if
end for
return Task result $\hat{y}$

B Experimental Setup

B.1 Benchmark

LIBERO (Liu et al., 2023) is a simulation benchmark for lifelong robot learning comprising four task suites: LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long. We place particular emphasis on LIBERO-Long to evaluate complex, multi-step manipulation.

- **LIBERO-Spatial**: Tests spatial understanding by requiring the robot to manipulate objects according to their relative positions.
- **LIBERO-Object**: Evaluates object understanding and generalization across target objects with diverse appearances and shapes.
- **LIBERO-Goal**: Tests instruction following through different task objectives within the same scene layout.
- **LIBERO-Long**: Evaluates long-horizon execution through compositional manipulation tasks requiring multiple sequential steps.

CALVIN (Mees et al., 2022) is a simulation benchmark for long-horizon, language-conditioned robotic manipulation. It comprises four environments (A–D) and 34 tasks involving movable objects and articulated scene elements. Evaluation requires completing five consecutive language-specified subtasks without resetting the environment, assessing reliable execution under evolving scene states.

- **Language-Conditioned Manipulation**: Requires grounding natural language instructions in visual observations and translating specified goals into manipulation actions.
- **Diverse Manipulation Skills**: Covers lifting, pushing, stacking, and interactions with articulated components and switches.
- **Cross-Environment Generalization**: Environments A–D vary in appearance and spatial configuration. The ABC→D setting evaluates transfer from training environments A–C to the unseen environment D.
- **Long-Horizon Evaluation**: Uses 1,000 sequences of five subtasks, reporting success rates for completing at least k consecutive subtasks ($k \in \{1, \ldots, 5\}$) and average chain length.

B.2 Implementation Details

The configurations of the agents in our multi-agent collaborative architecture are summarized in Table 7. Both the Planner and Monitor use large language models with chain-of-thought (CoT) reasoning capabilities for reasoning and analysis. The downstream vision-language-action (VLA) execution models span multiple architectural paradigms.

Table 7: Model configurations for individual agents.

Agent	Configuration
Planner	Qwen3-VL-8B-Thinking (Yang et al., 2025a)
Executor	OpenVLA (Kim et al., 2025) $\pi_{0.5}$ (Intelligence et al., 2025) SmolVLA-0.24B (Shukor et al., 2025)
Monitor	Qwen3-VL-8B-Thinking (Yang et al., 2025a)

C Details on Skill Library and Subtask Decomposition

C.1 Extensible Skill Library ($\mathcal{B}$)

The skill library $\mathcal{B}$ provides the planner $\mathcal{A}_P$ with a structured vocabulary for decomposing high-level instructions into executable manipulation subtasks. Its entries describe operations supported by the underlying VLA executor $\mathcal{A}_E$. Consistent with Eq. 1, an instantiated subtask is represented as

$$\mathbf{u}_i = \langle \sigma_i, \omega_i, \ell_i \rangle, \tag{10}$$

where σ_i identifies the manipulation skill, ω_i specifies the target object, and ℓ_i describes the desired spatial goal. For a grasping skill, the goal is to establish a grasp on the specified object. For a placement skill, it identifies the destination surface or container. We use $\varnothing$ when the skill and target object already specify the intended outcome without a separate spatial argument.

Each skill entry associates a skill identifier with an argument schema and a language description. The argument schema follows the shared object–position convention, while the description specifies the intended operation and outcome. This interface separates subtask specification from continuous action generation: the planner selects and instantiates a skill, and the VLA executor grounds the resulting subtask into robot actions using the current observation, robot state, and retrieved context, as described in Eq. 3.

Table 8 lists representative entries. These entries illustrate the interface rather than define a fixed or exhaustive skill set. Their descriptions support instruction–skill matching, while their argument schemas help identify the objects and spatial conditions required for execution. Together with the initial observation and shared context memory $\mathcal{M}_0$, this information conditions direct plan generation in Eq. 2. The planning prompt asks the planner to align the subtasks with the instruction, account for the observed scene and available skills, and respect subtask dependencies. These are instructions for plan generation, not numerical criteria for scoring or ranking candidate plans.

Table 8: Representative entries in the extensible skill library $\mathcal{B}$. Arguments correspond to the target object ω_i and spatial goal ℓ_i of an instantiated subtask. $\varnothing$ indicates that no separate spatial argument is required.

Skill	Arguments (ω_i, ℓ_i)	Language description
`pick_up`	(target object, $\varnothing$)	Grasp and lift the target object so that it is held by the robot.
`put_on`	(held object, target surface)	Place the held object on the specified support surface.
`put_in`	(held object, container interior)	Place the held object inside the specified container.
`open_drawer`	(drawer handle, $\varnothing$)	Pull the target drawer open using its handle.
`close_drawer`	(drawer, $\varnothing$)	Push the target drawer closed.
`turn_on`	(device control, $\varnothing$)	Manipulate the target control to activate the device.

An existing skill can be instantiated with different objects and spatial goals without creating a new entry. For example, `put_on` can specify placement on different plates by changing its arguments. The object's current location is grounded from the observation and shared context, whereas the spatial argument specifies the desired destination. Skill expansion additionally allows successful recovery behaviors to become reusable entries, as detailed below.

C.2 Instantiation of the Subtask Sequence $\mathcal{P}$

Consider the high-level instruction $\mathbf{x}$: *"Put the white mug on the left plate and put the yellow-and-white mug on the right plate."* Given the initial observation, the skill library $\mathcal{B}$, and the shared execution context, the planner grounds the referenced objects and destinations and directly generates an ordered sequence according to Eq. 2. For this example, $\mathcal{P} = \{\mathbf{u}_i\}_{i=1}^{4}$. Assuming that both mugs are initially on the tabletop and the gripper is empty, one valid decomposition is

$$\mathbf{u}_1 = \langle \texttt{pick_up}, \text{white mug}, \varnothing \rangle, \tag{11}$$

$$\mathbf{u}_2 = \langle \texttt{put_on}, \text{white mug}, \text{left plate} \rangle, \tag{12}$$

$$\mathbf{u}_3 = \langle \texttt{pick_up}, \text{yellow-and-white mug}, \varnothing \rangle, \tag{13}$$

$$\mathbf{u}_4 = \langle \texttt{put_on}, \text{yellow-and-white mug}, \text{right plate} \rangle. \tag{14}$$

Each subtask specifies an intended outcome and has execution requirements determined by the current scene and robot state. For $\mathbf{u}_1$, the white mug must be identifiable and accessible, and the gripper must be available. The intended outcome is that the robot holds the white mug. Subtask $\mathbf{u}_2$ requires this grasp to be maintained and an accessible placement region on the left plate. Its intended outcome is that the white mug rests on the left plate and the gripper is empty. Subtask $\mathbf{u}_3$ then establishes a grasp on the yellow-and-white mug, and $\mathbf{u}_4$ places it on the right plate while preserving the first placement result.

This decomposition illustrates the considerations specified in the planning prompt. Each mug is associated with its instructed destination, and the decomposition assumes that the objects, placement regions, and required skills are available. Each placement follows its corresponding grasp; under the assumed single-gripper setup, the first mug is released before the second is grasped. The planner expresses these relationships through the generated subtask sequence rather than through explicit candidate scores or a weighted selection objective. The two mug-transfer operations may be reversed if scene conditions permit, since the instruction does not impose a unique ordering between them.

During execution, i_t identifies the active subtask $\mathbf{u}_{i_t}$. The VLA executor generates action chunks conditioned on this subtask, the current visual observation and robot state, and the context retrieved from $\mathcal{M}_t$. The monitor $\mathcal{A}_M$ directly predicts the execution state from the current and previous observations $\mathbf{o}_{t-1:t}$, the active subtask $\mathbf{u}_{i_t}$, and the context $\mathcal{M}_t^-$, as specified in Eq. 5. Here, $\mathcal{M}_t^-$ contains the context available before the current monitoring result is recorded. The monitor assesses the intended subtask outcome using these inputs; no explicit semantic similarity score or progress increment is computed. For example, the expected outcome of $\mathbf{u}_1$ is a successful grasp and lift, whereas that of $\mathbf{u}_2$ is placement of the white mug on the left plate.

The monitor returns an execution state $\hat{y}_t \in \{\mathrm{Running}, \mathrm{Success}, \mathrm{Failure}\}$ and a failure cause $\hat{e}_t$, which is set to None when the predicted state is not $\mathrm{Failure}$. A $\mathrm{Running}$ prediction keeps the current subtask active. A $\mathrm{Success}$ prediction advances execution to the next subtask, or terminates the task when the final subtask is complete. A $\mathrm{Failure}$ prediction triggers recovery. The monitoring result is then written to the shared memory, denoted $\mathcal{M}_t$ after this update, for subsequent planning and execution. Thus, progression through $\mathcal{P}$ depends on monitored outcomes rather than solely on the completion of an action chunk.

C.3 Recovery-Driven Skill Expansion

When the monitor detects a failure, the planner uses the original instruction $\mathbf{x}$, the active subtask $\mathbf{u}_{i_t}$, the predicted failure cause $\hat{e}_t$, the updated shared context memory $\mathcal{M}_t$, and the skill library $\mathcal{B}$ to generate a recovery subtask, as specified in Eq. 6. The recovery subtask follows the same structured interface:

$$\mathbf{u}_t^r = \langle \sigma_t^r, \omega_t^r, \ell_t^r \rangle, \tag{15}$$

where the three components identify the recovery skill, target object, and desired spatial state, respectively.

For example, if the white mug slips from the gripper during its transfer to the left plate, the original placement subtask may no longer be directly executable. If the mug remains accessible, a recovery subtask can specify re-establishing a grasp on it. The planner grounds this recovery in the updated

observation and execution history, since the mug's location may have changed. This example illustrates how failure information changes the immediate execution objective while retaining the original instruction as the task-level goal.

Following Eq. 7, the monitor selects visual points associated with the end-effector, manipulated object, and target region. The multi-point trajectory predictor uses the historical images, recovery subtask, failure cause, and shared context to predict their future image-space trajectories. These tracks are overlaid on the current observation to form $\mathbf{I}_t^r$. The VLA executor then generates a recovery action chunk conditioned on this image, the robot state, the recovery subtask, and the retrieved context, as described in Eq. 8. The predicted tracks provide visual guidance; the VLA policy remains responsible for generating continuous robot actions.

After recovery execution, the monitor directly evaluates the resulting state against the intended outcome of the recovery subtask, using the observations and shared execution history. A successful recovery allows TVA to resume the original plan; recovery success does not by itself imply completion of the interrupted subtask. Otherwise, the updated observation and failure record are incorporated into $\mathcal{M}_t$ to support another recovery iteration. In the mug example, successful regrasping restores the condition needed to continue the interrupted placement.

Successful recovery does not automatically create a new library entry. As defined in Eq. 9, the recovered behavior is added only when the monitor reports success and its novelty exceeds the threshold δ_n:

$$\mathcal{B} \leftarrow \mathcal{B} \cup \{\sigma_r\} \quad \text{if} \quad \hat{y}_t^r = \text{Success} \ \wedge \ \nu(\sigma_r) > \delta_n. \tag{16}$$

Here, σ_r denotes the reusable skill summarized from the successful recovery. In addition to its identifier and compatible argument schema, the new entry stores its language description, applicable failure type, recovery-trajectory image, and successful execution record. These records associate the skill with the circumstances in which it succeeded and the visual guidance used during execution.

Reusing a skill with different arguments and expanding the skill library serve distinct purposes. Applying an existing skill to a different mug or plate changes its arguments without necessarily introducing a novel behavior. By contrast, a successful recovery that satisfies the novelty criterion contributes a reusable entry to $\mathcal{B}$. This expansion enriches the planner's available skill repertoire while preserving the structured subtask interface and the original architecture, training objective, and fine-tuning pipeline of the underlying VLA model.

D Prompt Details

This section details the structured prompt designs (Sahoo et al., 2024) for the Planner and Monitor agents. Through a closed-loop coordination mechanism, this design translates high-level instructions into executable robotic arm actions , while simultaneously enabling the analysis of error causes to perform corrective re-planning.

D.1 Planner Prompt

As shown in Figure 6, the Planner agent serves as the decision-making core, responsible for decomposing complex tasks into subtask sequences. Its prompt is structured with the following key components:

- **Atomic Skill Library:** A predefined set of robotic manipulation primitives (e.g., *pick up*). Each skill is defined by its manipulation predicate, parameter space (e.g., target objects), and a natural language semantic descriptor to ensure the agent understands the physical implications of each action.
- **Reasoning Protocol:** To ensure logical consistency, the Planner follows a three-step chain-of-thought process:
  1. *Environment Detection*: Assessing whether target objects are within the field of view and if the robot's current state permits the action.
  2. *Logical Derivation*: Identifying the most immediate physical action required based on the task progress.

3. *Causal Entailment*: Predicting environment changes post-execution to ensure alignment with the global goal.

- **Error-Driven Re-Planning:** When a failure is reported, the Planner discards affected plans and generates a compensatory sequence. A strict constraint is applied to ensure the first subtask of the new plan directly remediates the identified error.

D.2 MONITOR PROMPT

As shown in Figure 6, the Monitor agent acts as a real-time supervisor to verify execution outcomes via a *Verification Protocol*:

- **Visual Evidence Extraction:** The agent observes image sequences to determine if the final pose of the target object aligns with subtask expectations.
- **Anomaly Detection:** It identifies unintended collisions or grasp failures during the execution trajectory.
- **Causal Analysis:** If a subtask is deemed unsuccessful (Status: `NO`), the Monitor generates a detailed causal analysis of the failure, which is fed back to the Planner for re-planning.

Planner PROMPT

As a planner specializing in robotic arm manipulation tasks, your objective is to decompose high-level instructions into a series of subtasks based on the provided atomic skill library. These subtasks will then be delivered to a VLA model serving as the executor to carry out low-level physical actions. During the execution process, your team member—the Monitor—is responsible for detecting the progress of execution. If an error is identified, the Monitor will provide a causal analysis; subsequently, you must re-plan the sequence of subtasks based on the cause of the error and the environmental state to achieve the robotic arm's execution goal.
Atomic Skill Library
{"manipulation": pick up
"parameter space": [object]
"natural language semantic descriptor": Firmly grasp the target object on the tabletop.
}.....
Analyze the scene state to assist in planning.
- Scene context: [Information and positional relationships of objects on the operation platform]
- Robotic arm state: [Position and status of the gripper]
- Monitor Feedback: Error Analysis
{IF_ERROR_DESCRIPTION_ELSE_NONE}
Reasoning Protocol
1. Environment Detection: Is the target object within the field of view? Does the current robot state permit this action?
2. Logical Derivation: To achieve the high-level instruction, what is the most immediate physical action required following the completion of previous tasks?
3. Causal Entailment: If this action is executed, how will the environment change? Does it align with the final goal?
Error-Driven Re-Planning Module
When the scene state contains error information, you must prioritize handling the anomaly over continuing the original plan.
Recovery(Re-Plan) Logic:
1. Analyze Error Cause: Interpret the physical failure reported by the Monitor. For example, if the Error Cause is "Object Dropped," you must insert a new "pick up" subtask.
2. Update Subtasks: Discard the affected subsequent plans and generate a new sequence based on the current compromised environmental state.
3. Constraint: Ensure the first subtask of the new plan directly remediates the reported issue.
Inputs
- Task: {Task_Description}
- Visual Observation: {IMAGE_FRAMES}
- Error Analysis {IF_ERROR_DESCRIPTION_ELSE_NONE}
Output Format
- Subtask List: {Subtask_IDX}

Monitor PROMPT

You are an expert in robotic task monitoring. Your mission is to verify whether the execution trajectory of subtasks has achieved the expected results based on the execution image sequence and to generate a record of successfully completed subtasks. If an error occurs during the execution process, you are required to analyze the cause of the failure.
Verification Protocol
Please perform the analysis through the following Chain-of-Thought (CoT):
1. Visual Evidence: Observe the images. Does the final pose of the target object align with the subtask's expectations? Has the robotic arm successfully disengaged contact?
2. Anomaly Detection: Did any unintended collisions or grasp failures occur during the execution process?
3. Status Determination:
If the task is successful: status = YES.
If the task fails: status = NO. Provide a detailed description of the error cause.
Inputs
- Subtask List: {SUBTASK_DESCRIPTION}
- Visual Observation: {IMAGE_FRAMES}
Output Format
- Subtask status: {SUBTASK_DESCRIPTION, YES/NO}
- Error Analysis: {IF_ERROR_DESCRIPTION_ELSE_NONE}

Figure 6: Prompt details