

FRAMES: Failure Recovery And Monitoring of Embodied Skills for Humanoid Loco-Manipulation

Ajay Vikram Periasami*, Xinyuan Luo[†], Haoyu Li[†], Xianyi Cheng[†]

*Department of Computer Science, Duke University, Durham, NC, USA

[†]Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC, USA

ajay.vikram@duke.edu, xinyuan.luo@duke.edu, haoyu.harry.li@duke.edu, xianyi.cheng@duke.edu

Abstract—Large language model (LLM) planners can decompose natural-language instructions and select reusable robot skills, but choosing the correct skill does not guarantee successful physical execution. This gap is especially important in humanoid loco-manipulation, where errors during approach, grasping, transport, or placement can invalidate the remainder of a long-horizon plan. We present FRAMES, a failure-aware supervisory framework for the Unitree G1 humanoid that operates above the CEER whole-body controller [1]. A *Planner Agent* selects subtasks through parameterized mid-level skills, while a vision-language-model-based *Monitor Agent* evaluates each skill using temporal multi-view observations and structured robot and contact evidence. Detected failures stop the active skill and provide grounded feedback to a *Recovery Agent*. The framework further includes a *Memory Module* for reusing prior skill experience, and geometric grounding via depth and segmentation. We independently evaluate the monitoring module of the framework in MuJoCo using 100 trials comprising 50 failed and 50 successful executions across five tasks. The monitor detects 48 of 50 failures, correctly accepts 46 of 50 successful executions, and achieves 94.0% overall accuracy. These results provide initial evidence for the monitoring component, while end-to-end evaluation of the complete recovery loop remains ongoing.

Index Terms—humanoid loco-manipulation, failure detection, failure recovery, vision-language models, semantic memory, embodied agents

I. INTRODUCTION

Humanoid robots achieve loco-manipulation by combining locomotion and manipulation to execute long-horizon tasks in human environments. A typical long-horizon task, such as box transport, requires the robot to execute a sequence of subtasks: approaching the object, grasping it, carrying it while walking, and placing it at a new location. The physical coupling of these subtasks means that early errors cascade into downstream failures. An inaccurate approach directly prevents a clean grasp, a premature transition to walking can leave the box unexpectedly supported by the table, and the subsequent motions of walking or turning can ultimately break a secure hold. If these failures go undetected, subsequent actions will be planned using an incorrect estimate of the world state. Achieving reliable humanoid loco-manipulation therefore requires the robot to continuously monitor physical execution, identify failures in real time, and recover autonomously from the observed state.

Recent robotic agents leverage large language models (LLMs) and vision-language models (VLMs) to interpret instructions and select reusable skills. For instance, SayCan

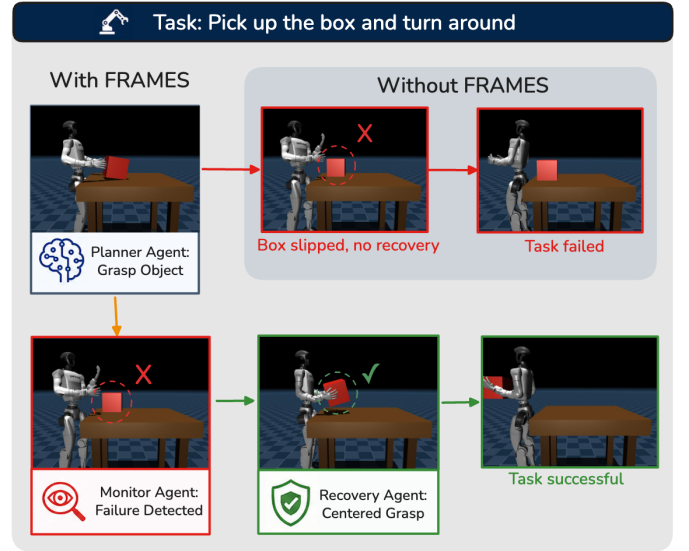


Fig. 1. The planner selects the correct grasp subtask, yet the box can still slip during physical execution. Without failure-aware supervision, the robot proceeds from an invalid state and fails the task. FRAMES detects the failed grasp, invokes a centered grasp, and enables the robot to complete the turn while retaining the box.

[2] combines language-based planning with learned affordance values, while Inner Monologue [3] provides execution feedback to an LLM planner. Similarly, CAPE [4] utilizes failed skill preconditions to generate corrective actions. These approaches primarily use environment feedback to replan between high-level actions, rather than explicitly monitoring the temporal evolution of an ongoing physical skill and interrupting it upon failure.

To handle execution errors more robustly, recent methods have moved toward explicit failure reasoning. REFLECT [5] and AHA [6] utilize multimodal data and VLMs to detect and explain failures, while RACER [7] applies language feedback to learn recovery behaviors. In the domain of execution monitoring, DoReMi [8] tracks dynamic constraints via VLMs, Code-as-Monitor [9] synthesizes executable monitoring programs, and COME-robot [10] integrates visual feedback for mobile-manipulation planning. Despite these advances, most methods focus on fixed-base or wheeled robots, with DoReMi being a notable exception that demonstrates applicability to humanoids. Concurrent to our work, recent humanoid sys-

tems like Cybo-Waiter [11] and POT-VLA [12] have also begun exploring the integration of language planning, geometric grounding, and execution verification. While POT-VLA achieves this by training a unified action model conditioned on shared object tokens, our approach avoids costly policy retraining. Instead, we propose a modular supervisory framework positioned above an existing whole-body controller.

We build our framework on the CEER Controller [1], keeping its low-level policy entirely fixed. The core of our contribution is a modular supervisory layer divided into three core entities: a *Planner Agent* to decompose and execute natural-language tasks via mid-level skills, a *Monitor Agent* to check for failures during skill execution, and a *Recovery Agent* to recover from the detected failures. We also develop a *Memory Module* that supplies reusable experience to the *Planner Agent* and *Recovery Agent*. We also provide depth and segmentation skills for the geometric grounding of LLM-suggested skills. This clean separation of concerns enables advanced planning, monitoring, and recovery behaviors without requiring any modifications or retraining at the whole-body control level.

The contributions of this work are:

- A modular planning and recovery framework that executes humanoid loco-manipulation tasks through parameterized mid-level skills over a fixed whole-body controller, with skill parameterization informed by persistent subtask-level semantic memory and planner-callable SAM3 and RGB-D object grounding.
- A VLM-based *Monitor Agent* that evaluates each skill using multi-view images, structured robot state, and contact information across a temporal window. It distinguishes ongoing progress from success and failure, stops failed skills, and provides failure evidence to the *Recovery Agent*.
- A controlled simulation evaluation of the *Monitor Agent* over 100 trials spanning failed grasps, dropped objects, collisions during approach, wrong-object grasps, and wrong placements. The monitor detects 48 of 50 failures, accepts 46 of 50 successful executions, achieving 94.0% overall accuracy.

These experiments provide initial quantitative evidence for the monitoring component. Evaluation of the complete planning, monitoring, recovery, and memory loop remains ongoing.

II. METHODS

A. Problem Formulation

Given a natural-language instruction g , the objective is to execute a sequence of robot skills that satisfies the instruction while detecting and recovering from execution failures. We use k to index high-level subtasks, n to index monitoring samples collected during subtask k execution, and ℓ to index low-level CEER control steps. For each subtask, $n = 0$ denotes the

observation immediately before execution. The observation at monitoring sample n of subtask k is

$$o_{k,n} = (s_{k,n}, \mathcal{I}_{k,n}), \quad \mathcal{I}_{k,n} = \{I_{k,n}^v\}_{v=1}^{N_{\text{cam}}}, \quad (1)$$

where $s_{k,n}$ is the structured robot and scene state, and $\mathcal{I}_{k,n}$ contains the available camera views. In simulation, $N_{\text{cam}} = 3$, corresponding to robot-perspective, top-down, and side-view RGB images. The structured state is

$$s_{k,n} = (b_{k,n}, \mathcal{O}_{k,n}, \mathcal{U}_{k,n}, \mathcal{H}_{k,n}, \mathcal{K}_{k,n}, \mathcal{F}_{k,n}), \quad (2)$$

where $b_{k,n}$ is the robot base pose, $\mathcal{O}_{k,n}$ contains dynamic-object states, $\mathcal{U}_{k,n}$ contains support-surface states, $\mathcal{H}_{k,n}$ contains hand positions, $\mathcal{K}_{k,n}$ contains contact relations, and $\mathcal{F}_{k,n}$ contains contact forces.

The instruction is solved sequentially by the *Planner Agent*, one subtask at a time. At high-level decision step k , the system executes the action

$$a_k = (d_k, \sigma_k, \theta_k, C_k), \quad (3)$$

where d_k describes the intended subtask, σ_k is the selected skill API, θ_k contains its execution parameters, and C_k is its monitor contract. For a physical skill, the monitor contract is

$$C_k = (c_k^{\text{pre}}, c_k^{\text{succ}}, c_k^{\text{fail}}), \quad (4)$$

where the fields specify the subtask preconditions, success condition, and failure conditions. Observation and execution-control APIs do not require a monitor contract.

During each physical skill execution, the *Monitor Agent* periodically classifies the skill as `in_progress`, `success`, or `failure`. An `in_progress` verdict allows execution to continue, while a failure interrupts the skill and activates the *Recovery Agent*. After the skill finishes, a final monitor check determines whether control returns to the *Planner Agent*.

B. Planner Agent

At decision step k , the *Planner Agent* receives the original instruction g , current observation $o_k \equiv o_{k,0}$, execution history h_k , and skill library $\mathcal{A}$. It proposes one next action:

$$\begin{aligned} \tilde{a}_k &= (d_k, \sigma_k, \tilde{\theta}_k, C_k) \\ &= \text{PlannerAgent}(g, o_k, h_k, \mathcal{A}), \end{aligned} \quad (5)$$

where $\tilde{\theta}_k$ denotes the initially proposed parameters. The execution history is

$$h_k = \{a_i, \rho_i, \hat{y}_i, z_i\}_{i=0}^{k-1}, \quad (6)$$

where ρ_i is the skill execution result, $\hat{y}_i$ is the *Monitor Agent* verdict, and z_i is its explanation. Monitor fields are empty for skills that do not require monitoring. The skill library is

$$\mathcal{A} = \mathcal{A}_{\text{nav}} \cup \mathcal{A}_{\text{man}} \cup \mathcal{A}_{\text{obs}} \cup \mathcal{A}_{\text{ctrl}}, \quad (7)$$

where the subsets contain navigation, manipulation, observation, and execution-control skills, respectively. Table I lists the available skill APIs and their execution arguments.

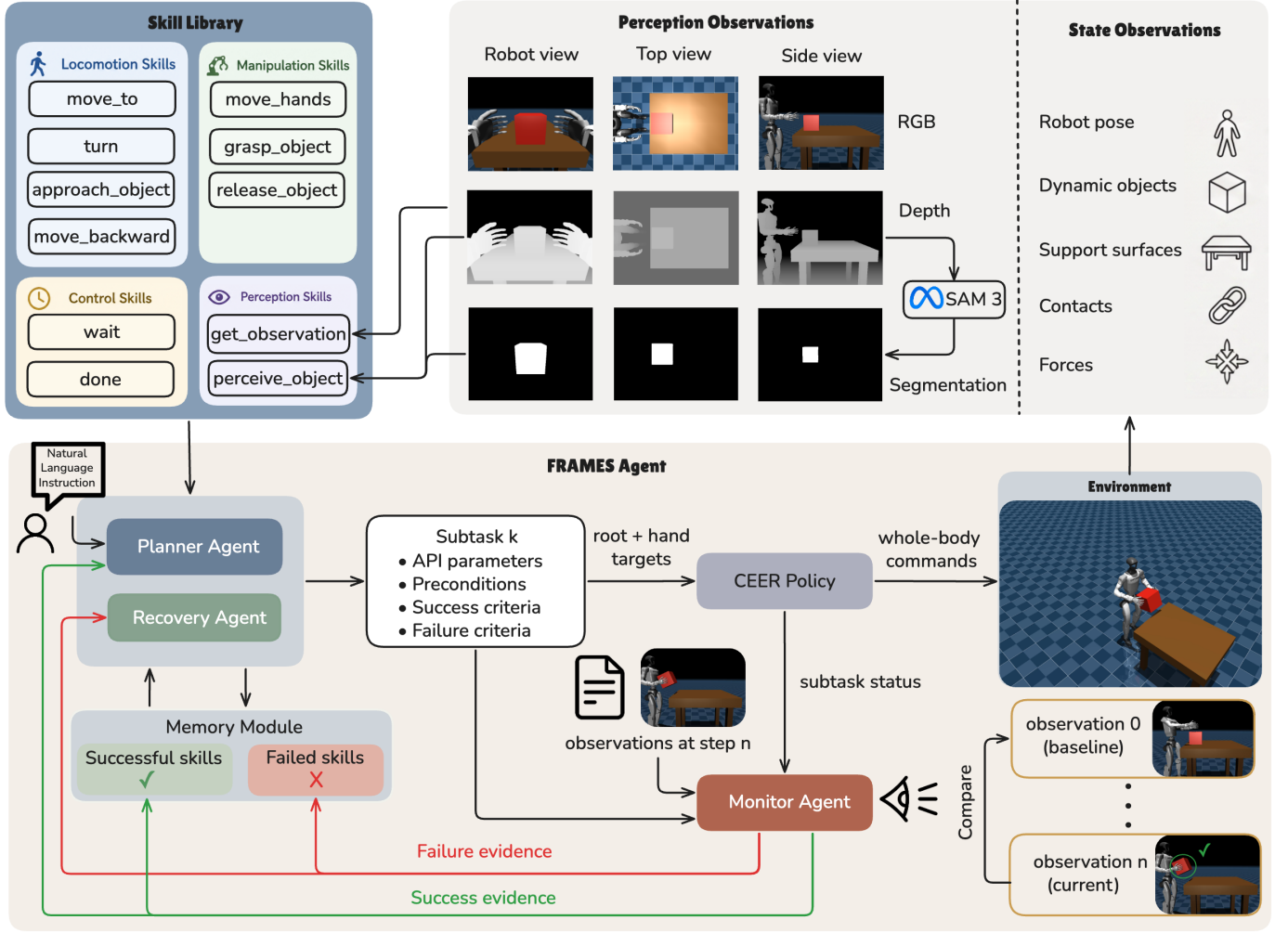


Fig. 2. **Overview of FRAMES.** A natural-language instruction is handled by the *Planner Agent*, which selects a subtask from the skill library together with its API parameters and necessary preconditions and failure and success criteria. The selected subtask is executed through the fixed CEER whole-body controller in the MuJoCo environment. During execution, the *Monitor Agent* compares the current observation against the baseline observation using multi-view perception and structured state information to determine whether the subtask is succeeding or failing. The monitor verdict and supporting evidence are returned immediately to the decision layer, where failures inform the *Recovery Agent*. At the end of each run, completed skill executions are distilled into concise subtask-level records and appended to memory.

After the *Planner Agent* selects skill σ_k with initial parameters $\hat{\theta}_k$, the *Memory Module* retrieves previous records that used the same skill:

$$\mathcal{B}_{\sigma_k} = \{m_i \in \mathcal{B} \mid \sigma_i = \sigma_k\}, \quad (8)$$

where $\mathcal{B}$ is the persistent memory bank and $\mathcal{B}_{\sigma_k}$ contains the retrieved records for the selected skill.

The same *Planner Agent* is then called in a focused parameterization mode to reuse prior experience without allowing memory to alter the intended subtask. It receives the selected skill, initial parameters, current observation, recent execution history, matching memory records, and the skill’s argument schema. It may revise only schema-permitted execution parameters, producing θ_k , while d_k , σ_k , the target object, and C_k remain fixed. If no relevant memory is available or the revised parameters are invalid, the system retains $\theta_k = \hat{\theta}_k$.

The resulting action a_k , defined in Eq. (3), is then passed to the *Skill Executor*.

C. Skill Executor: CEER Controller

We use the pretrained CEER whole-body controller without modification. CEER receives root and end-effector targets and maps them, together with proprioceptive observations, to 29-DoF whole-body commands at 50 Hz. Details of its policy architecture, training procedure, and control abstraction are provided in [1].

For a physical skill σ_k , the *Skill Executor* converts the selected parameters θ_k and current state into a sequence of root and end-effector targets:

$$u_{k,\ell}^{\text{EE-root}} = G_{\sigma_k}(\theta_k, s_{k,\ell}), \quad (9)$$

TABLE I
SKILL LIBRARY AVAILABLE TO THE PLANNER AND RECOVERY AGENTS

Category	API	Arguments	Function
Locomotion	move_to	(x, y, ψ)	Moves the robot base to an absolute world-frame pose.
	turn	(ψ, T)	Rotates the robot toward an absolute world-frame yaw ψ over duration T .
	move_backward	(x, y)	Moves the base toward an absolute world-frame position while preserving its final orientation.
	approach_object	$(target, d_{\text{stand}}, \Delta\psi)$	Moves the base to a target-facing pose with the specified standoff distance and optional yaw offset.
Manipulation	move_hands	$(p_L, p_R, T_{\text{hold}})$	Moves both hands to robot-base-frame target positions $p_L, p_R \in \mathbb{R}^3$ and maintains them for T_{hold} seconds, without moving the base.
	grasp_object	$(target, p_L, p_R, d_{\text{lift}})$	Grasps the target using specified left- and right-hand contact positions, then lifts it by d_{lift} .
	release_object	None	Opens both hands to release the currently held object.
Observation	get_observation	$(include_images)$	Returns structured state and, when requested, the available camera views.
	perceive_object	$(prompt)$	Locates a text-specified object using SAM3 [13] and depth, and estimates its position, visible size, and hand-to-object distances in the robot-base frame.
Control	wait	(T)	Pauses high-level execution for duration T .
	done	$(r_{\text{claimed}}, z_{\text{task}})$	Requests task completion with a claimed outcome and summary.

where ℓ indexes low-level control steps. CEER executes these targets as

$$q_{k,\ell}^{\text{cmd}} = \pi_{\text{CEER}}(p_{k,\ell}, u_{k,\ell}^{\text{EE-root}}), \quad (10)$$

where $p_{k,\ell}$ is the CEER proprioceptive observation. The Skill Executor updates targets at approximately 10 Hz, while CEER runs at 50 Hz.

After a skill completes or is interrupted, the *Skill Executor* returns

$$\rho_k = (r_k^{\text{status}}, r_k^{\text{message}}, r_k^{\text{data}}), \quad (11)$$

containing whether the skill completed and provides any skill-specific measurements. The *Monitor Agent* separately evaluates whether the observed execution satisfied the subtask contract C_k . Both outputs are added to the execution history and used to select the next planning or recovery action.

D. Monitor Agent

The *Monitor Agent* evaluates whether each physical skill satisfies its subtask contract. It uses skill-local temporal context rather than judging a single observation independently. Before skill k begins, the framework captures and retains a baseline observation

$$o_{k,0} = (s_{k,0}, \mathcal{I}_{k,0}). \quad (12)$$

At monitoring sample n , it captures the current observation $o_{k,n}$. With three camera views, each monitor query therefore contains six images: the robot-perspective, top-down, and side views at skill start, followed by the same three views at the current sample.

The framework also computes a compact structured summary of the change since skill execution began:

$$\Delta s_{k,n} = \phi(s_{k,0}, s_{k,n}). \quad (13)$$

The summary includes object displacement, change in robot-object distance, base-pose changes, and transitions in manipulator, support, and floor contacts. Retaining the same baseline throughout the skill allows the *Monitor Agent* to evaluate cumulative progress rather than only the change between

adjacent samples. The evidence supplied to the *Monitor Agent* is

$$e_{k,n} = (g, d_k, C_k, o_{k,0}, o_{k,n}, \Delta s_{k,n}), \quad (14)$$

where g is the complete task instruction, d_k is the current subtask, and C_k specifies its preconditions, success condition, and failure conditions. The *Monitor Agent* returns

$$(\hat{y}_{k,n}, z_{k,n}) = \text{MonitorAgent}(e_{k,n}), \quad (15)$$

where $\hat{y}_{k,n}$ is `in_progress`, `success`, or `failure`, and $z_{k,n}$ is a concise explanation grounded in the visual and structured evidence. The *Monitor Agent* checks the skill approximately once per second. An `in_progress` result allows execution to continue, while a `failure` result stops the skill and activates the *Recovery Agent*. A `success` result is accepted only after the skill has finished and the *Monitor Agent* confirms the final state.

E. Recovery Agent

Let n_k^f denote the monitoring sample at which subtask k is declared failed. The framework constructs the failure record

$$f_k = (a_k, \rho_k, z_{k,n_k^f}, \Delta s_{k,n_k^f}, o_{k,n_k^f}). \quad (16)$$

The *Recovery Agent* proposes a corrective action:

$$\tilde{a}_{k+1}^R = \text{RecoveryAgent}(g, o_{k,n_k^f}, h_k, f_k, \mathcal{A}). \quad (17)$$

The *Recovery Agent* uses the same skill library as the *Planner Agent* but is explicitly conditioned on the failed subtask and its observed failure evidence.

After the *Recovery Agent* selects a skill, the *Memory Module* retrieves records for that skill and refines only its permitted execution parameters just like the *Planner Agent*. The resulting recovery action a_{k+1}^R is then executed and monitored using the same procedure as a normal planner action. The *Recovery Agent* may retry the failed skill with revised parameters, select a corrective approach or hand motion, release an unstable contact, or request additional observations.

F. Memory Module

The framework maintains a persistent semantic memory bank $\mathcal{B}$. After each task terminates, including an interrupted run, every completed physical subtask is distilled into a memory record

$$m_i = (d_i, \sigma_i, \theta_i, \rho_i, r_i^{\text{sub}}, z_i, r_i^{\text{task}}, \lambda_i), \quad (18)$$

where r_i^{sub} is the local subtask outcome, r_i^{task} is the final whole-task outcome, and λ_i is a concise lesson describing the observed cause, parameter effect, recovery result, and reuse guidance. Observation and execution-control skills are not stored.

After either the *Planner Agent* or *Recovery Agent* selects a skill σ_k , the Memory Module retrieves records with the same skill:

$$\mathcal{B}_{\sigma_k} = \{m_i \in \mathcal{B} \mid \sigma_i = \sigma_k \wedge r_i^{\text{sub}} = r_i^{\text{task}}\}. \quad (19)$$

Successful subtasks are retrieved only from successful tasks, while failed subtasks are retained as negative examples only from failed tasks. A subtask reported as successful in an ultimately failed task is excluded because its local success may have been transient or incorrect. Within a successful task, only the latest successful execution of each skill is retained.

G. Language-Conditioned Geometric Grounding

To provide the *Planner Agent* with geometric grounding for more informed skill parameterization, it can call `perceive_object` with a text description such as “red box.” The module applies SAM3 [13] to a frame-matched robot-perspective RGB image to obtain a segmentation mask for the requested object. Valid depth measurements inside the mask are used to reconstruct the visible object surface in 3D. Using the camera calibration and robot base pose, the resulting points are transformed into the world and robot-base frames. The module returns the visible-surface center, visible extent, median depth, segmentation confidence, and hand-to-object distances. These measurements support the parameterization of object-relative approach, hand-motion, and grasp skills.

III. PRELIMINARY EXPERIMENTS & RESULTS

A. Experimental Setup

We conduct preliminary experiments in MuJoCo using the 29-DoF Unitree G1 humanoid with the pretrained CEER Controller [1], which was trained in simulation using Isaac Lab with 16,384 parallel environments on four NVIDIA L40s GPUs for 8 hours. OpenAI GPT-5.4-mini is used for the *Planner Agent*, *Recovery Agent*, and *Monitor Agent*. SAM3 provides language-conditioned object segmentation for RGB-D geometric grounding.

B. Monitor Evaluation Protocol

We evaluate the *Monitor Agent* independently on five different tasks. Each task contains 20 trials: 10 failure trials and 10 matched successful executions. Small variations are applied to the robot pose, approach parameters, or manipulation parameters for each of the trials. The required skill sequence

is executed by teleoperating, without involving the *Planner Agent* or *Recovery Agent*. This is done to isolate the *Monitor Agent* performance. The five tasks are defined as follows.

a) Failed grasp.: The robot is instructed to grasp and lift the box from the table. A trial is successful when both hands hold and lift the box, and failed when the box remains on the table after the grasp attempt.

b) Dropped object.: The robot begins each trial holding the box above the table and must maintain its grasp. A trial is successful when the robot continues holding the box and failed when the box slips or falls onto the table or floor.

c) Collision during approach.: The robot is instructed to approach the box without disturbing it. A trial is successful when the robot reaches the box without contact and failed when its hands contact and move the box during the approach.

d) Wrong-object grasp.: A red box and a blue box of the same shape are placed on the table, and the robot is instructed to lift the red box. A trial is successful when the robot lifts the red box and failed when it lifts the blue box.

e) Wrong placement.: A blue target pad and a yellow distractor pad are placed on the table, and the robot is instructed to place the red box on the blue pad. A trial is successful when the box is placed on the blue pad and failed when it is placed on the yellow pad.

We report the number of failures correctly detected, the number of successful trials correctly accepted and overall classification accuracy. An `in_progress` verdict returned after skill completion is counted as an incorrect classification.

C. Monitor Evaluation Results

Table II summarizes the results. Across all 100 trials, the *Monitor Agent* correctly detects 48 of 50 failures and accepts 46 of 50 successful executions, corresponding to 94.0% overall accuracy. The monitor correctly classifies all dropped-object and wrong-object-grasp trials. The wrong-object task is particularly important because the monitor identifies the lifted object from camera images without receiving object names or poses. The remaining errors reveal two limitations. Two failed grasps are labelled `in_progress` even though skill execution has ended. Three safe approaches are incorrectly classified as collisions because the hands appear close to the box. One correct placement is also classified as a failure because the box appears to remain held after release. These errors show that the monitor is conservative when physical contact is visually ambiguous. Figure 3 shows qualitative examples of the robot failing and succeeding on the 5 tasks.

IV. CONCLUSION

We presented FRAMES, a failure-aware framework for humanoid loco-manipulation built on top of the CEER whole-body controller. The *Planner Agent* selects parameterized skills and defines how their execution should be checked. The *Monitor Agent* detects failures using temporal multi-view images and structured state information, and stops the current skill when needed. The *Recovery Agent* then selects a corrective action based on the failure evidence. FRAMES

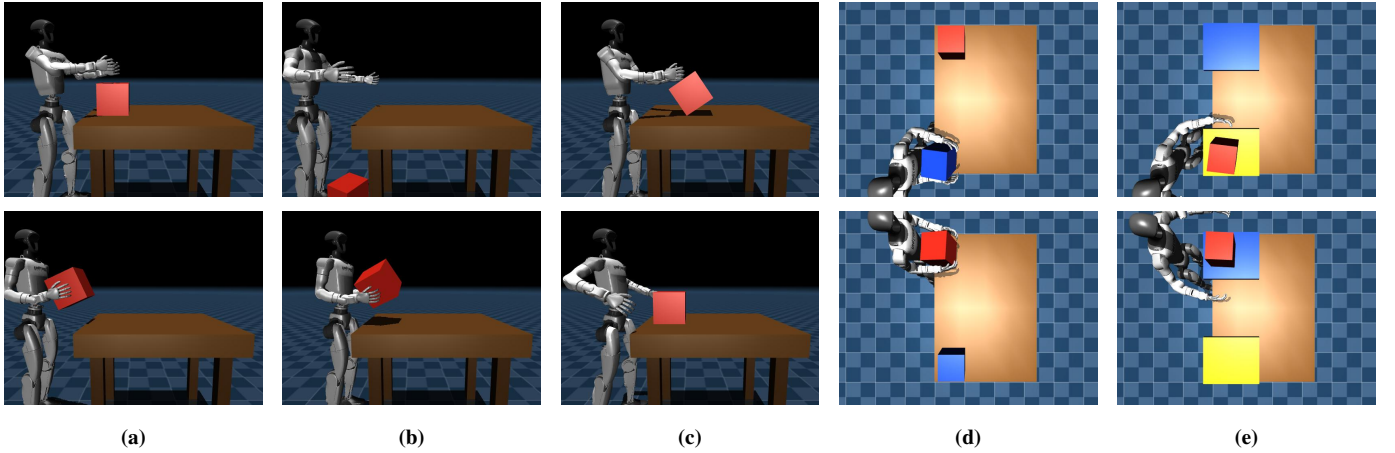


Fig. 3. Representative failure and success outcomes from the five *Monitor Agent* tasks. The top row shows failure conditions, and the bottom row shows matched successful executions: (a) failed versus successful grasp; (b) dropped object versus stable hold; (c) collision versus collision-free object approach; (d) grasping the blue distractor box versus the requested red box; and (e) placing the red box on the yellow distractor pad versus the blue target pad.

TABLE II
MONITOR PERFORMANCE ACROSS FIVE BENCHMARKS, EACH
CONTAINING 10 FAILED AND 10 SUCCESSFUL SKILL EXECUTIONS.

Benchmark	Failures Detected	Successful Runs Accepted	Overall Accuracy
Failed grasp	8/10	10/10	90%
Dropped object	10/10	10/10	100%
Collision during approach	10/10	7/10	85%
Wrong-object grasp	10/10	10/10	100%
Wrong placement	10/10	9/10	95%
Overall	48/50	46/50	94%

also uses semantic memory to reuse earlier skill experience, and RGB-D and SAM3 to support object grounding.

Our preliminary evaluation focused on the *Monitor Agent*. Across 100 executions from five tasks, it correctly classified 94 trials, detecting 48 of 50 failures and accepting 46 of 50 successful runs. These results show that the monitoring component can identify several common execution failures, but the complete planning and recovery loop still requires end-to-end evaluation. Future work will improve and evaluate the remaining components and test the full framework across a wider range of long-horizon tasks.

REFERENCES

- [1] X. Luo, X. Chen, X. Yin, H. Wu, B. Xia, Z. Chen, J. Li, B. Chen, and X. Cheng, “CEER: Compliant end-effector and root control as a unified interface for hierarchical humanoid loco-manipulation,” *arXiv preprint arXiv:2605.19981*, 2026.
- [2] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, “Do as i can, not as i say: Grounding language in robotic affordances,” *arXiv preprint arXiv:2204.01691*, 2022.
- [3] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar *et al.*, “Inner monologue: Embodied reasoning through planning with language models,” *arXiv preprint arXiv:2207.05608*, 2022.
- [4] S. S. Raman, V. Cohen, I. Idrees, E. Rosen, R. Mooney, S. Tellex, and D. Paulius, “Cape: Corrective actions from precondition errors using large language models,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 14 070–14 077.
- [5] Z. Liu, A. Bahety, and S. Song, “Reflect: Summarizing robot experiences for failure explanation and correction,” *arXiv preprint arXiv:2306.15724*, 2023.
- [6] J. Duan, W. Pumacay, N. Kumar, Y. R. Wang, S. Tian, W. Yuan, R. Krishna, D. Fox, A. Mandlekar, and Y. Guo, “Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation,” *arXiv preprint arXiv:2410.00371*, 2024.
- [7] Y. Dai, J. Lee, N. Fazeli, and J. Chai, “Racer: Rich language-guided failure recovery policies for imitation learning,” in *2025 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2025, pp. 15 657–15 664.
- [8] Y. Guo, Y.-J. Wang, L. Zha, and J. Chen, “Doremi: Grounding language model by detecting and recovering from plan-execution misalignment,” in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 12 124–12 131.
- [9] E. Zhou, Q. Su, C. Chi, Z. Zhang, Z. Wang, T. Huang, L. Sheng, and H. Wang, “Code-as-monitor: Constraint-aware visual programming for reactive and proactive robotic failure detection,” in *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2025, pp. 6919–6929.
- [10] P. Zhi, Z. Zhang, Y. Zhao, M. Han, Z. Zhang, Z. Li, Z. Jiao, B. Jia, and S. Huang, “Closed-loop open-vocabulary mobile manipulation with gpt-4v,” in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 4761–4767.
- [11] P. Ren, H. Ge, C. Qi, C. Huang, H. Li, J. Zhao, P. Chi, and K. Chen, “Cybo-waiter: A physical agentic framework for humanoid whole-body locomotion-manipulation,” *arXiv preprint arXiv:2603.10675*, 2026.
- [12] P. Ren, H. Ge, J. Zhao, C. Huang, Y. Shi, P. Chi, and K. Chen, “Closing the loop in humanoid vla: Persistent 3d object tokens for verifiable loco-manipulation,” *arXiv preprint arXiv:2607.18016*, 2026.
- [13] N. Carion, L. Gustafson, Y.-T. Hu, S. Debnath, R. Hu, D. Suris Coll-Vinent, C. Ryali, K. V. Alwala, H. Khedr, A. Huang *et al.*, “Sam 3: Segment anything with concepts,” in *International Conference on Learning Representations*, vol. 2026, 2026, pp. 138 846–138 923.