FLARE: A Failure-Aware Framework for Autonomous Correction and Recovery in Visual-Language Robotic Manipulation

Supplementary Material

1. Implementation Details

Data Generation and Augmentation. For our primary task data, we use MimicGen [29] with its default parameters. For each task, we begin with a set of 10 human-collected demonstrations that are then expanded to 500 augmented demonstrations. Our perturbation and bridging process, critical for enabling the "retry" capability, is defined by two parameters: a maximum perturbation rotation of $45°$ and a maximum translation of 0.5 meters. Following the MimicGen procedure, we segment each task trajectory into subtasks. Our perturbation & bridging sequences are then injected at the beginning of the full trajectory and at the junctions between these subtasks.

Reset Skill Collection. Our reset skills are designed to be object-centric. The total number of reset skills for a given task is determined by the number of manipulable objects involved. We explicitly scope out unrecoverable situations, such as when an object is too large for the gripper to re-orient (e.g., a toppled coffee machine) or when an object falls outside the robot's reachable workspace (e.g., onto the floor).To mine failure states for reset demonstration, we use Gemini-2.5-Pro to analyze task execution videos, with a sampling temperature set to 0.7. For each identified object-centric reset skill, we collect 20 human demonstrations. This initial dataset is then augmented to 500 demonstrations using the same perturbation & bridging technique applied to the task data, ensuring our reset skills are also robust to varying robot poses.

Training Details. We follow the official training practices of $\pi_{0.5}$ [15]. Specifically, we apply LoRA [14] fine-tuning to the language model and the action expert modules, while all other model parameters remain frozen. We use the Adam optimizer with a constant learning rate of 2.5×10^{-4} and do not employ a cosine learning rate decay schedule.

Evaluation Protocol. For each task, we conducted 50 evaluation trials and reported the average success rate, following established evaluation protocols in prior work [39].

2. Manipulation Tasks

In this section, we briefly describe the 9 manipulation tasks in our evaluation, and the reset task object for demonstration collection in our experiments. Fig. 6 shows the visualization of all these tasks.

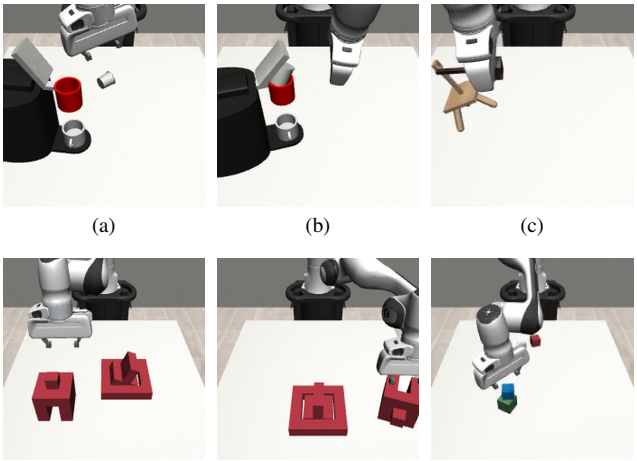

Figure 6. Failure cases for all tasks (stack task excluded as reset is not applicable). (a) Reset coffee pod; (b) Reset coffee machine lid; (c) Reset the needle-hole block; (d) Reset the T-shaped block; (e) Reset the U-shaped block; (f) Reset the blue block.

2.1. Visualization of Task Execution

We visualize a successful execution of the ThreePiece-Assembly task and the corresponding reset instructions in Fig. 7. In this example, the robot first attempts to perform the regular task by picking up the T-shaped block and placing it into the square block. However, the T-shaped block is not placed correctly, triggering a reset. The robot then picks up the T-shaped block and places it aside to reset its pose. After completing the reset, the robot retries the task and successfully completes the assembly.

3. Prompt for Video Analysis and Failure Detection

In this section, we provide the prompts for both video analysis and failure detection during closed-loop inference in Table 6 and Table 7.

4. Broader Impact and Future Work

Broader Impact The core value of the FLARE framework lies in endowing robotic systems with higher autonomy and reliability, which is crucial for advancing VLA models from lab demonstrations to large-scale, dependable deployment in real-world applications.

By significantly reducing the need for human interven-

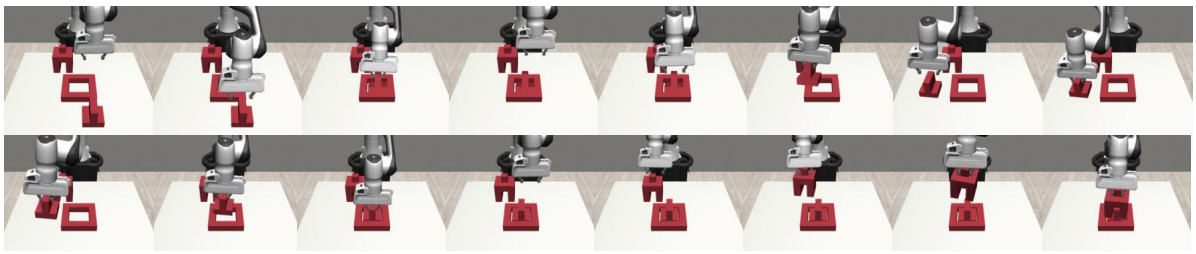

Figure 7. Visualization of a successful task execution for ThreePieceAssembly. Frame 1-4: The robot operates the regular task to assembly the three pieces by first pick-and-place the T-shaped block. Frame 5-8: The T-shaped block is not place appropriately and robot tries to reset it. Frame 9-16: The robot executes the regular task again and succeeds.

tion, FLARE can boost the operational efficiency and safety of industrial automation, service robotics, and professional service domains. This increase in robustness is a critical step in the transition of robots from simple assistive tools to reliable, persistent collaborators, particularly in environments that are repetitive, high-risk, or difficult to access.

As with any highly autonomous system, the powerful self-correction capabilities of FLARE must be accompanied by careful safety and ethical protocols. We therefore emphasize the following:

All recovery and reset actions must strictly adhere to predefined safety constraints to ensure they do not cause unintended harm to the environment or personnel.

We must maintain the system's interpretability and transparency, and ensure that operators retain an emergency override or control switch to guarantee human oversight and accountability remain the final authority over the system.

Future Work While the FLARE framework has markedly advanced VLA models' capabilities in failure recovery, there remains substantial room for exploration in terms of scalability and generalization. Our future work will focus on the following three main directions:

1. Towards Generative and Generalized Recovery Policies: Our current robust "Reset" mechanism relies on a pre-defined library of recovery skills, which limits its ability to handle Out-of-Distribution (OOD) and complex failure states that require non-standard or multi-step, collaborative actions. Future work will explore leveraging the powerful reasoning capabilities of large Vision-Language Models (VLMs) to enable them to autonomously infer and generate custom, ad-hoc recovery action sequences based on the current failure state and the desired goal. Achieving an end-to-end generative recovery policy is a key step in boosting FLARE's generalization.
2. Integrating Online Adaptation and Lifelong Learning: To address the data collection and annotation challenges associated with scaling our current approach, we plan to integrate the FLARE framework into an Active Learning loop. When the model encounters an unseen failure mode with low confidence, the system will be able to automatically trigger data collection and policy fine-tuning in a controlled environment. This mechanism for continuous learning will allow the deployed robot to persistently enhance its own recovery capabilities throughout its operational lifespan.
3. Transfer Validation for Real-World Complexity: Our ultimate goal is to validate FLARE's effectiveness in more challenging, real-world scenarios. This includes testing the system's robustness in environments with dynamic changes, when handling non-rigid or deformable objects, and during long-horizon tasks that require human collaboration. These practical validations will drive the transition of FLARE from a laboratory framework to a real-world utility.

Table 6. The main prompt used in our experiment for video analysis.

Video Analysis Query

You are an expert in robotic task failure analysis. Your task is to analyze a video recording of a robot performing a task that ultimately **failed**, and determine the **failure reason**, **recovery strategy**, **failure time point**, and **error group**.
—
Background - The video shows the complete execution process of a robotic task that ended in failure. - You need to carefully inspect the entire sequence and identify the **key moment** and **cause** of failure.
—
Resettable Objects {reset_objects}
—
Concept of *Error Group* An **Error Group** refers to the set of objects that jointly contribute to an erroneous state. Examples:
1. **Single-object error:** The coffee capsule is overturned. → Error Group = [“coffee capsule”]
2. **Multi-object error:** The coffee capsule is jammed in the coffee machine’s insertion slot. → Error Group = [“coffee capsule”, “coffee machine”]
When two or more objects form an undesirable **geometric or dynamic relationship** (e.g., a part stuck in a slot, an object wedged in a mechanism, or a collision with the machine workspace), the error involves multiple objects. To reproduce or reset such an error, **all objects in the group must be transformed together** (i.e., their relative poses must remain consistent). You cannot reset one object independently of the others.
> Note: The reset_object must be included in the error_group list.
—
Reset Selection Rule (Very Important)
When the error involves multiple objects, the selection of the reset_object must respect **operation dependency**:
- If a target object is **blocked, pressed, clamped, or locked** by another, the first step of recovery should address the obstructing object. - Therefore, the reset_object should be the **first object that needs to be manipulated** in order to recover from the failure, not necessarily the final target you want to restore.
Example: Error Group = [“coffee capsule”, “coffee machine”] The coffee machine’s lid is closed (even if not completely), and the capsule is jammed inside the insertion slot. Although the ultimate goal is to reset the capsule, the first step is to **open the lid** — thus, the reset_object is “coffee machine”.
This rule ensures that the chosen reset action is **feasible and logically correct**, avoiding the selection of objects that are blocked or constrained.
—
Analysis Tasks
Please analyze the video and provide the following information:
1. Failure Reason - Describe in detail what caused the task to fail. - Explain why this failure occurred. - If multiple factors contributed, list them in order of importance.
2. Failure Time Point - Specify the exact time (frame index or timestamp) when the failure occurred. - Describe the scene state at that moment. - Explain why this moment represents the critical failure point.
3. Recovery Strategy Based on the failure scene state at that moment, determine the most appropriate recovery strategy: - **retry** – for temporary issues (e.g., unstable grasp, small trajectory deviation) that can be resolved by retrying the same action. - **reset** – for state-altering issues (e.g., toppled cup, displaced tool) that require resetting one or more objects to their correct states.
4. Reset Object (only if the strategy is reset) Based on the failure scene state at that moment, find the object that needs to be reset: - Specify which object should be reset (**choose only one**). - Describe **from what state to what state** it should be reset. - The object must appear in the **Resettable Objects** list. - If the error group has operation dependencies, select the object that should be **manipulated first** during recovery. For instance, if a coffee capsule is trapped under the lid, the first step is to open the lid, so the reset_object is “coffee machine”.
5. Error Group - Identify all objects involved in forming the error state. - List them in an array. - Ensure the reset_object is included in this list. - If only one object is involved, the list should contain just that object.
—
Output Format Please output your analysis in the following **JSON** format:
“‘json
{
“failure_reason”: “Detailed description of the failure cause.”,
“recovery_strategy”: “retry or reset”,
“reset_object”: “Name of the object to reset (null if strategy is retry)”,
“error_group”: [“List of objects involved in the error; must include reset_object if not null”],
“failure_timestep”: 1234,
“failure_description”: “Detailed description of the failure moment.”,
“confidence”: 0.8
} ”’

Table 7. The main prompt used in our experiment for failure case detection during the closed-loop inference for ThreePieceAssembly.

Failure Detection Query

You are an expert in analyzing robot task execution scenes, skilled at identifying abnormal states and suggesting appropriate recovery strategies.
The current task is called **Three Piece Assembly**, and its goal is to "assemble the three pieces."
The correct execution sequence is as follows:
1. Place the **T-shaped block** into the **square block**.
2. Then place the **U-shaped block** on top of the **T-shaped block**.
The following image shows the current state of the task:
Please analyze the current scene. If the **T-shaped block** is visible and not placed flat when inserted into the **square block** — for example, it is tilted or stuck — the answer is **B**. If the **U-shaped block** is not placed flat when put on the **T-shaped block** or is lying on the desk, the answer is **C**. Otherwise, the answer is **A**.
At the end of your answer, clearly state your conclusion in the following format: **The answer is [A] / [B] / [C].**