Guardian: Detecting Robotic Planning and Execution Errors with Vision-Language Models

Paul Pacaud, Ricardo Garcia Pinel, Shizhe Chen, Cordelia Schmid
Inria, École normale supérieure, CNRS, PSL Research University
firstname.lastname@inria.fr

Abstract: Robust robotic manipulation requires the ability to detect and recover from failures. While recent Vision-Language Models (VLMs) show promise in detecting manipulation failures, their effectiveness is hindered by limited training data and reliance on single images, restricting their ability to capture fine-grained failure modes and generalize to real-world scenarios. In this work, we introduce a new VLM named Guardian which leverages high-resolution, multi-view visual observations combined with carefully designed language prompts to enhance manipulation failure detection. We propose an automatic failure synthesis pipeline that perturbs successful trajectories in both simulator and real world to generate a diverse set of failure scenarios, covering both task planning and action execution errors. This enables the creation of two new benchmarks: RLBench-Fail and BridgeDataV2-Fail for training and evaluation. Guardian achieves state-of-the-art performance on both benchmarks and demonstrates strong generalization to the manually created real-world RoboFail and UR5-Fail benchmarks. Furthermore, plugging Guardian into a state-of-the-art vision-language manipulation framework improves task success rates in both simulation and real robots.

Keywords: Robotic Manipulation, Failure Detection, Vision-Language Models

1 Introduction

Despite recent advances in task planning and action execution enabled by Large Language Models (LLMs) [1, 2] and Vision-Language Models (VLMs) [3], current robotic manipulation systems remain vulnerable to a range of failures [4, 5, 6]. The LLMs suffer from hallucinations [7], leading to incorrect task plans [8]. Visual models are prone to perception inaccuracies like confusing similar objects [9, 10], and motion planners frequently encounter precision errors or unstable object grasping, such as slippage [11]. These errors accumulate during task execution, severely hindering the reliability of robots in real-world scenarios. In contrast, humans excel in failure detection and recovery by reasoning about visual information and interpreting the semantic context of tasks [12, 13]. Inspired by this, we explore the development of failure detection models that can leverage vision and language to reason about task execution and autonomously identify and rectify errors.

A key challenge in failure detection is how to leverage visual information effectively. REFLECT [14] addresses this by first converting images into textual descriptions and then prompting an LLM for failure detection. However, this multi-stage pipeline is prone to error accumulation. Other recent approaches [15, 16] directly feed images into VLMs, but typically rely on single-view images, which are susceptible to occlusions. To overcome this, AHA [17] concatenates multi-view images across timesteps into a single grid-based image. While this can cover broader visual information, such a compressed format of images can hinder fine-grained spatial-temporal understanding.

Another challenge is the lack of comprehensive robotic failure datasets. Most real-world robot datasets consist of successful demonstrations [18, 19, 20], offering limited insight into failure modes. Manually collecting failure data [21, 22] is time-consuming and often lacks diversity and realism. As a result, existing work [17] mainly uses simulation to construct failure cases, which, however, suffers

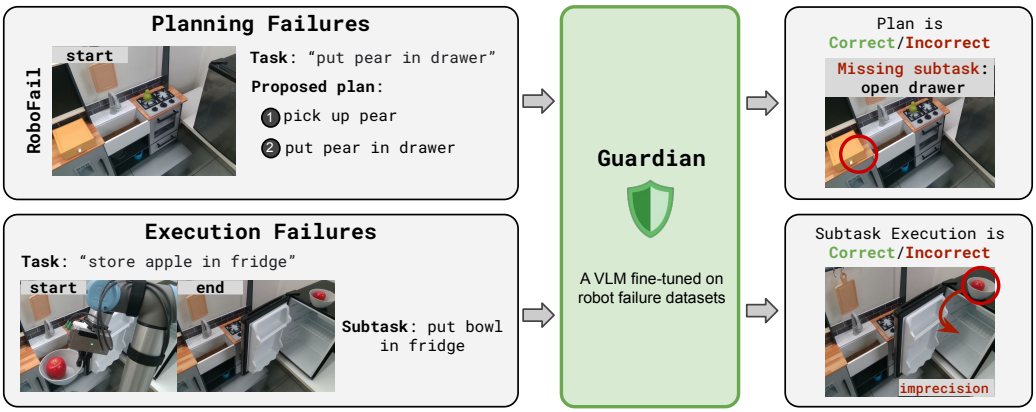

Figure 1: Illustration of our Guardian model - a VLM fine-tuned on our constructed failure datasets. It detects planning failures (top) and execution failures (bottom) in robotic manipulation.

from the well-known sim-to-real gap [23]. This data scarcity significantly hampers the development and evaluation of failure detection methods.

To address these challenges, we introduce Guardian, built on top of an open-source VLM for failure detection in robotic manipulation. As illustrated in Figure 1, Guardian formulates failure detection as a visual question-answering problem, reasoning over high-level task instructions, subtask descriptions, and the visual scene. It leverages high-resolution, multi-view images to enhance fine-grained spatial-temporal understanding. To support model training, we propose a new failure generation pipeline that automatically perturbs successful robot trajectories in simulator and real world examples. Using this pipeline, we generate two new datasets: RLBench-Fail (19K execution and 8K planning samples) and BridgeDataV2-Fail (11K execution and 6K planning samples). We fine-tune Guardian on our constructed datasets via parameter-efficient fine-tuning. Guardian achieves state-of-the-art performance across RLBench-Fail and BridgeDataV2-Fail datasets, and generalizes well to RoboFail [14] and our constructed UR5-Fail datasets that are manually collected with real robots in a zero-shot manner. We further demonstrate Guardian's plug-and-play capability as a feedback mechanism within a robotic manipulation framework in RLBench simulated and real-world tasks, improving 3DLotus++ [9] via iterative planning and execution monitoring.

In summary, our contributions are threefold:

- We propose a novel failure generation pipeline that automatically perturbs successful trajectories in simulation and the real world, yielding two large datasets RLBench-Fail (simulation) and BridgeDataV2-Fail (real world) for model training and evaluation.
- We introduce Guardian, a fine-tuned VLM for planning and execution failure detection. It integrates multi-view images and reasons over high-level instructions and subtask plans.
- Guardian achieves state-of-the-art performance across four benchmarks, covering both in- and out-of-domain datasets. It further improves task completion of a state-of-the-art vision-language manipulation framework in simulated and real robot tasks.

We will open-source our datasets, code, and models to advance research in robotic failure detection.

2 Related Work

Failure detection in robotic manipulation. Traditional monitoring methods rely on explicit models of tasks, identifying model deviations during planning and execution [24, 25]. Unlike LLM-based approaches, they depend on rigid model-based predictions. Recent works formulate failure detection as a Question Answering (QA) task, by zero-shot prompting LLMs and VLMs [16, 26, 15, 21, 27, 28, 29, 30]. For instance, REFLECT [14] summarizes multi-sensory inputs as texts, including audio, and feeds the text to an LLM, but its reliance on consecutive off-the-shelf models leads to accumulated errors. Consequently, efforts have been made to fine-tune VLMs specifically for robotic manipulation failure detection. SuccessVQA [31] fine-tunes Flamingo using both simulated and

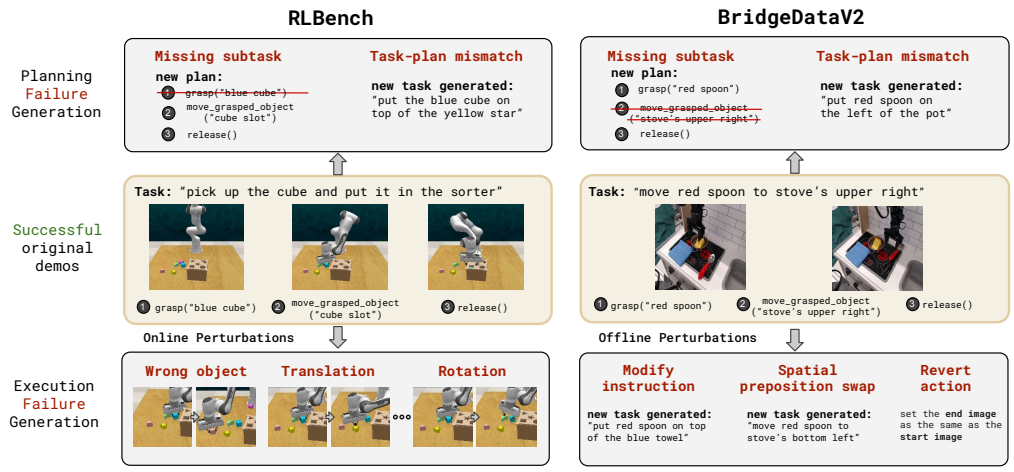

Figure 2: Failure Data Generation Pipeline. We introduce a novel generation pipeline generating failure cases both online in simulation (RLbench), and offline on the real-world dataset (BridgeDataV2). For each positive example, given its correct plan and successful trajectory, we generate a corresponding incorrect plan and unsuccessful trajectory.

real-world data, but is limited to evaluating the final task success, while Guardian is designed for more granular monitoring, checking both the overall plan and the execution of subtasks. Closest to our work, AHA [17] fine-tunes LLaVA-1.5 [32] on a dataset generated in RLBench. AHA concatenates multiple camera viewpoints and keysteps into a single image, leading to compressed visual inputs. Moreover, SuccessVQA and AHA do not release the model or dataset, limiting reproducibility.

Failure detection data generation. Existing datasets for failure detection involve limited settings. RoboFail [14] provides hand-crafted, small-scale simulation and real-world data. Sentinel [15] creates failure cases by rolling out a trained policy on out-of-distribution scenarios through randomization of object scales and poses, but provides data and results to only four tasks. AHA's dataset [17] perturbs trajectories in RLbench, creating over 49K+ image-query pairs. While they provide a scalable data generation pipeline, it is limited to the simulated data and does not generate high-level planning failures. In this work, we propose an automated pipeline to generate extensive datasets with diverse failure cases using simulator and real-robot trajectories.

Learning-based robotic manipulation policies. Recent general robotic policies have achieved remarkable performance on manipulation tasks [33, 9, 19, 34, 35]. However, their real-world applicability is still limited by planning and execution errors leading to task failures [11]. Inspired by Manipulate-Anything [36] and Robot Utility Models [16], which demonstrated significant benefits from integrating VLM verifiers into manipulation pipelines, Guardian is explicitly designed as a plug-and-play verification module trained on specific "failure" data. It can seamlessly integrate into existing policies, enhancing robustness by verifying both planning and execution stages.

3 Robot Failure Datasets Construction

We automatically construct robot failure datasets based on successful robot demonstrations in simulation and real world. Section 3.1 introduces the data sources, and then Section 3.2 describes the failure data generation process. Finally, Section 3.3 presents dataset statistics and quality evaluation.

3.1 Data Sources

Simulated data enables controlled failure generation through procedural perturbations [17], while real robot data reduces the sim-to-real gap but requires significant human supervision [14]. To balance precise control with real-world applicability, we use both simulated and real robot datasets. We propose an automated failure data generation pipeline that derives planning and execution failures from existing successful demonstrations, avoiding manual failure collection. In both domains, tasks are decomposed into subtasks with corresponding video segments, which form the basis for generating failures. Figure 2 (middle) illustrates successful episodes from the simulated and real robot datasets.

Table 1: Robot failure datasets covering execution (exec) and planning (plan) failure categories. The validation and test sets consist of tasks unseen in the training set.

	Source	Auto Failure	Training		Validation		Test	
			Exec	Plan	Exec	Plan	Exec	Plan
RoboFail [14]	Real	×	-	-	-	-	153	30
RLBench-Fail	Sim	✓	12404	5832	1930	924	5282	2080
BridgeDataV2-Fail	Real	✓	8078	4932	1184	544	2362	1166
UR5-Fail	Real	✓	404	224	30	10	100	74

Simulated Data. We use the RLBench [37] simulator, selecting 52 tasks from RLBench-18Task [38] and GemBench [9] benchmarks in our training data. The full list is provided in Section B.4 of the supplementary material. For each task, we generate successful scripted trajectories with varied object placements and segment them into subtasks following 3D-LOTUS++ [9].

Real Robot Data. We use BridgeDataV2 [39] with ECoT annotations [40], which provide fine-grained subtasks and object labels using large vision and language models. We apply automatic cleaning to mitigate annotation noise (details in Section B.3 of the supplementary material). To increase the number of successful trajectories, we augment data by reversing successful executions when applicable, by swapping their start and end images and updating the associated instructions accordingly (e.g., "open drawer" becomes "close drawer", "flip pot upright" becomes "flip pot upside down"). This yields approximately 20% additional successful demonstrations.

3.2 Automated Failure Data Generation

We design failure modes based on established failure taxonomies [14, 17] and analysis of robot policy failures [41]. The failures are categorized into two types: planning and execution. A planning error denotes an incorrect decomposition of a task into subplans, whereas an execution error reflects unsuccessful completion of a subplan.

Planning Failures. As shown in Figure 2 (top), we construct two types of planning failures:

(1) *Missing subtask* - randomly removing a required subtask from the ground-truth subtask sequence.

(2) *Task-plan mismatch*: using an LLM (Mistral-Small-24B) to subtly alter the semantics of the task instruction, creating misalignment with the ground-truth subtasks. Leveraging LLMs enables more diverse planning errors compared to simply omitting subtasks.

Each planning example consists of the overall task instruction, initial images and the subtask sequence.

Execution Failures. In simulation, we perturb actions at the subtask level directly (Figure 2, bottom left), as it is easy to control the robot and generate new outcomes. Given a randomly selected subtask in the trajectory, we modify the ground-truth action through five execution failure modes inspired by [11, 19]: (1) *no close* - gripper fails to close during grasp, (2) *slip* - object drops as the gripper opens mid-movement, (3) *translation* - apply a random 3D positional offset, (4) *rotation* - apply a random angular offset along an axis, (5) *wrong object manipulated* - manipulate a different object, and (6) *wrong object placement* - place the grasped object in an incorrect location.

For real robot data, modifying actions directly is impractical due to current limitations of image editing and generation models. Therefore, we perturb the subtask text instruction paired with the pre-recorded trajectory segment (Figure 2, bottom right) without direct robot control: (1) *modify instruction* - an LLM (prompted with the original instruction and visible objects) generates semantically altered subtask instruction while keeping the original start/end images; (2) *spatial preposition swap* - we heuristically change spatial terms in the instruction (e.g., "move pot to stove's top right" becomes "move pot to stove's top left"), again preserving the images; (3) *revert action* - we retain the original subtask instruction but replace the end image by the start image to simulate a lack of progress.

Each execution example contains the task and subtask descriptions plus pre-/post-action multi-view images. In addition, our pipeline naturally provides fine-grained failure category labels, enabling richer feedback generation beyond binary failure detection in future work.

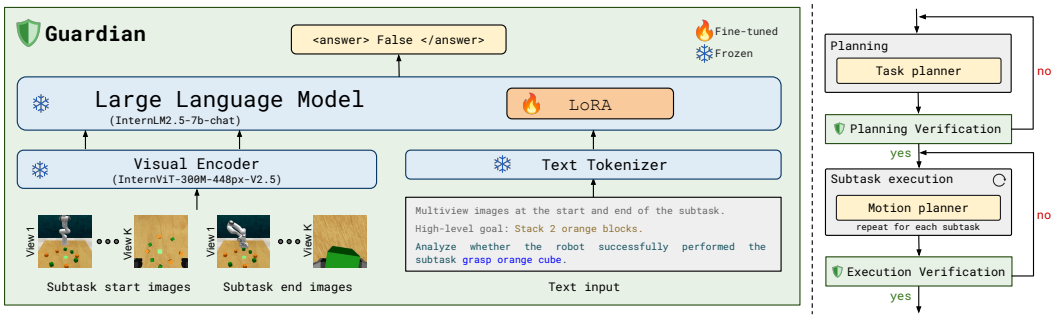

Figure 4: Left: Overview of the Guardian model architecture. Right: Integration of Guardian model into a robot manipulation pipeline for planning and execution verification.

3.3 Dataset Statistics and Evaluation

The resulting datasets, RLBench-Fail and BridgeDataV2-Fail, contain balanced successful and failed examples across both planning and execution. Each dataset is split into training, validation, and test sets, with the validation and test sets featuring unseen tasks/environments to evaluate generalization. Dataset statistics are shown in Table 1.

To measure the quality and diversity of our constructed datasets, i.e., whether the generated failures reflect real policy execution, we run the 3D-LOTUS++ policy [9] on 92 RLBench tasks and manually annotate failure modes for 3 failure episodes per task. As shown in Figure 3, our designed failure modes encompass all failures observed in real executions. Although real failures have more wrong gripper poses in execution and fewer missing subtasks in planning, the distribution of our synthetic and real failures is broadly similar.

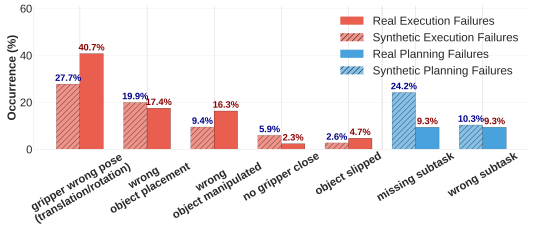

Figure 3: Failure mode distributions in real executions and our constructed data.

4 Method: The Guardian Model

4.1 Problem Formulation

We formulate robot failure detection as a visual question answering problem. For planning verification, given a high-level task instruction T, a proposed plan $P = (P_1, \cdots, P_N)$, and the initial visual context I_{start}, the model VLM_{plan} must access whether P is both consistent with the visual scene and aligned with the intended task:

$$\text{VLM}_{\text{plan}}(I_{\text{start}}, T, P) \rightarrow b_{\text{plan}}, \tag{1}$$

where $b_{\text{plan}} \in \{0, 1\}$ indicates whether the entire plan is valid. The execution verification instead determines whether a subplan P_i is successfully completed given the task goal T, a subtask description P_i, and the visual observations before and after executing the subtask - I_{start} and I_{end}:

$$\text{VLM}_{\text{exec}}(I_{\text{start}}, I_{\text{end}}, T, P_i) \rightarrow b_{\text{exec}}. \tag{2}$$

The planning verifier VLM_{plan} and execution verifier VLM_{exec} can either share the same underlying model or be implemented as separate agents within a multi-agent system.

4.2 Model Architecture and Training

The Guardian model is built upon the state-of-the-art open-source VLM InternVL2.5-8B [42]. As illustrated in Figure 4 (left), it consists of three key components: a text tokenizer that converts natural language prompts into discrete token embeddings, a visual encoder (InternViT-300M) that transforms individual images into visual embeddings, and a transformer-based LLM (internLM2.5-7b-chat) that processes the concatenated multimodal token sequence to predict the answer.

Unlike AHA [17] that concatenates multiple images into a single grid-based image, Guardian processes each image independently through the visual encoder. This design preserves fine-grained

spatial details within each image and allows the model to explicitly reason about spatial and temporal changes for more accurate failure detection.

We fine-tune Guardian using the training splits of our constructed RLBench-Fail and BridgeDataV2-Fail datasets. To enable efficient adaptation, we employed Low-Rank Adaptation (LoRA) to the LLM and only update the LoRA adapter parameters with the other components remain frozen. The model is trained using cross-entropy loss for next token prediction.

We compare two variants of Guardian: (1) a model trained jointly on both planning and execution verification data, and (2) a multi-agent approach that uses two separately trained models for planning and execution verification while sharing the same VLM architecture.

4.3 Integrating Guardian into Robotic Manipulation Framework

Guardian can be seamlessly plugged into existing robotic manipulation pipelines as a verification layer without requiring any architectural modification. Without loss of generality, consider a modular robotic manipulation framework. As shown in Figure 4 (right), Guardian can be inserted at each planning and subtask execution step to detect potential failures. Upon detection, it can trigger replanning strategies or re-execute the corresponding motion policy to facilitate recovery.

5 Experiments

5.1 Experimental Setup

Evaluation datasets. We assess models on four benchmarks covering both simulation and real-robot settings. (1) *RLBench-Fail*: a simulated tabletop environment with a Franka Emika Panda robot arm and four cameras; (2) *BridgeDataV2-Fail*: real-world toy kitchen and tabletop environments with a WidowX 250 6DOF robot arm and a single camera; (3) *RoboFail* [14]: a manually created real-world failure dataset using a UR5 robot arm with a single camera. (4) *UR5-Fail*: we run the 3D-LOTUS++ policy [9] on 32 task variations with a UR5 arm, recording initial and final multi-view images for each subtask. Subtasks are manually labeled as success or failure to obtain execution failure data. For planning failures, we annotate ground-truth plans and generate failure cases using the method described in Section 3.2. More detail is presented in Section A of the supplementary material. Table 1 (bottom) shows the dataset size, where tasks in training, validation and test sets are not overlapped.

Evaluation metric. We report overall accuracy, computed as the average binary classification performance across all samples for failure detection.

Implementation details. We trained our models using LoRA with a rank of 16 and an effective batch size of 16. The training employs a cosine learning rate scheduler with a peak learning rate of $4 * 10^{-5}$, the AdamW optimizer with a weight decay of 0.05, and bf16 precision. We train only on RLBench-Fail and BridgeDataV2-Fail training sets unless otherwise specified. Training is conducted on 2 NVIDIA H100 GPUs and completed in 1.5 hours. During training, we randomly selected either one or four views on RLBench-Fail data to mitigate overfitting to the multiple views. The best checkpoint is selected using the validation sets.

5.2 Comparison with state of the art

Compared methods. We compare our approach against state-of-the-art models, including closed-source models GPT-4.1-mini and GPT-4.1, an open-source VLM InternVL2.5-8B [42], Sentinel [43] and AHA-13B [17] that fine-tunes a VLM on a robotic dataset. We prompt these VLMs with interleaved image and text tokens. Since AHA-13B is not publicly released, we report the numbers from their original paper [17]. Sentinel is dedicated only to execution monitoring, and is thus not evaluated on planning failures. We also include a simple MLP baseline using CLIP text and visual features [44], without relying on LLMs. We compare the Guardian model to a multi-agent Guardian-MA which trains two expert models for detecting planning and execution failures separately.

Results. We present results in Table 2. While GPT-4.1 outperforms the open-source model InternVL 2.5-8B, our Guardian model fine-tuned from InternVL on robotic failure datasets surpasses GPT-4.1 by an average of 14 points on planning and 15 points on execution in in-domain benchmarks. Guardian also exceeds the prior SOTA AHA [17] - a finetuned VLM on RLBench, by over 20% on execution failure detection. While the numbers between ours and AHA are not directly comparable

Table 2: SOTA comparison for failure detection. Our Guardian model and its multi-agent (MA) version are benchmarked against proprietary and open-source VLMs. * denotes the number reported in the AHA paper [17] as neither their trained models nor test set is publicly available.

Model	In-Domain Eval				Out-of-Domain Eval (zero-shot)			
	RLBench-Fail		BridgeDataV2-Fail		Robofail		UR5-Fail	
	Plan	Exec	Plan	Exec	Plan	Exec	Plan	Exec
GPT-4.1-mini-2025-04-14	0.76	0.49	0.87	0.77	0.5	0.78	0.77	0.56
GPT-4.1-2025-04-14	0.79	0.58	0.86	0.78	0.63	0.84	0.89	**0.75**
Sentinel [43]	-	0.60	-	0.69	-	0.79	-	0.66
AHA-13B [17]	-	-	-	-	-	0.64*	-	-
InternVL 2.5-8B	0.68	0.49	0.82	0.73	0.57	0.84	0.84	0.53
CLIP+MLP [44]	0.70	0.53	0.90	0.60	0.60	0.40	0.56	0.53
Guardian (Ours)	**0.94**	0.80	**0.99**	**0.86**	0.63	**0.86**	**0.92**	0.74
Guardian-MA (Ours)	**0.95**	**0.82**	**0.99**	**0.87**	**0.73**	**0.86**	0.87	0.70

due to differences in base VLM and datasets, in Section 5.3 we reproduce AHA on the same base VLM and finetuning data to show our multi-view representation is more effective than the AHA's method. The multi-agent variant, Guardian-MA, provides minor additional gains over Guardian at the cost of increased memory overhead.

For zero-shot generalization, Guardian matches or surpasses GPT-4.1 on planning and execution across two out-of-domain datasets, demonstrating strong transfer to RoboFail and UR5-Fail, which involve new robot embodiments, environments, and tasks.

5.3 Ablation Studies

Multi-view images. Table 3 investigates the image representation choice on failure detection. We fine-tune and evaluate InternVL2.5-8B solely on RLBench-Fail execution data. We vary the number of viewpoints (one or four) and the multi-image format, either separated as in Guardian, or concatenated into a single image as in AHA [17]. For concatenated inputs, we represent the views as rows, and start/end images as columns. In the single-view setting, separating or concatenating two images (start and end) yields similar performance, as the resolution loss from concatenation is minimal. However, in the four-view setting, separating the images significantly boosts the performance by 5 points over concatenation. Concatenating images not only shifts from natural image distributions but also compresses visual information in limited image resolution. Furthermore, using multi-view images consistently outperforms their single-view counterpart.

Table 3: Impact of the image representation (number of views and image format). The AHA method [17] is equivalent to row 3 - concatenating 4 views.

Views	Image Format	RLBench-Fail Execution
1	concat	0.72
	separated	0.72
4	concat	0.82
	separated	**0.87**

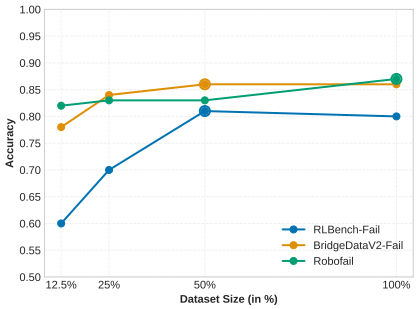

Figure 5: Impact of the training data size on the three execution datasets.

Training data mixture. Table 4 shows the impact of fine-tuning datasets. Row 1 is the baseline InternVL 2.5-8B without fine-tuning. Fine-tuning on simulated RLBench-Fail data (row 2) significantly improves the performance on the same dataset, but the gains do not transfer well to the other two real-world datasets due to the domain gap. A similar trend is observed when raining solely on BridgeDataV2-Fail in row 3. In contrast, combining both datasets during fine-tuning in the last row leads to the best overall performance and improves generalization to the RoboFail benchmark. This indicates the importance of generating diverse planning and execution examples.

Training data size. Figure 5 presents the scaling behaviors of training data size. On RLBench-Fail and BridgeDataV2-Fail test sets, we observe significant gains until 50% and then the performance

Table 4: Impact of different dataset mixture in fine-tuning Guardian model.

Training Data		RLBench-Fail		BridgeDataV2-Fail		Robofail		UR5-Fail	
RLBench-Fail	BridgeDataV2-Fail	Plan	Exec	Plan	Exec	Plan	Exec	Plan	Exec
✗	✗	0.68	0.49	0.82	0.73	0.57	0.84	0.84	0.53
✓	✗	**0.94**	**0.88**	0.88	0.68	0.57	0.70	**0.93**	0.64
✗	✓	0.52	0.48	**0.99**	**0.89**	**0.63**	0.82	0.53	0.53
✓	✓	**0.94**	0.80	**0.99**	0.86	**0.63**	**0.86**	0.92	**0.74**

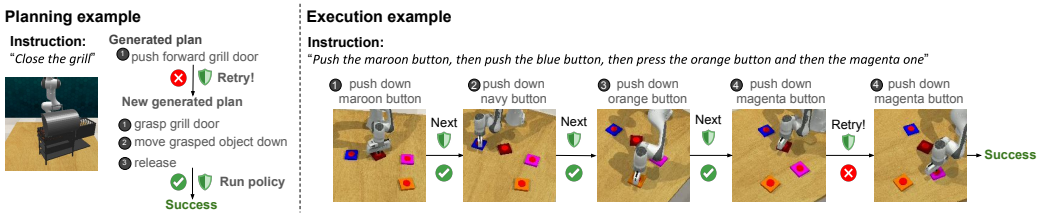

Figure 6: Verification with Guardian during online task execution on RLBench. Left: Successful correction of a generated plan. Right: Successful correction of a subtask execution.

gets saturated. The performance on RoboFail remains stable under low data regimes, but gets better with the full data. This indicates that both data scale and diversity are important.

5.4 Downstream Robotic Tasks

Table 5: Success rate for online task evaluation with and without Guardian.

Guardian	Stack blocks	Intert bulb	Close grill	Stack tower	Push buttons
×	0.30	0.35	0.20	0.10	0.50
✓	0.45	0.50	0.40	0.20	0.60

(a) RLBench results.

Guardian	Put food		Put fruits		Stack cups	
	Norm	Pert	Norm	Pert	Norm	Pert
×	0.60	0.00	**0.60**	0.20	0.80	0.00
✓	**0.80**	**0.80**	**0.60**	**0.40**	**1.00**	**0.40**

(b) Real robot results in normal and perturbed setups.

Simulation results. To evaluate the practical utility of Guardian, we integrate it into the 3D-LOTUS++ framework [9] and evaluate it on five challenging tasks from the RLBench-Fail test set. For each task, we conducted 20 episodes comparing performance with and without our verification module. If Guardian detects planning or execution failures, we rerun the 3D-LOTUS++ planning module or motion planning policy until Guardian outputs success. Table 5a shows that this integration consistently improves success rates across the five tasks, with gains ranging from 10% to 20%. Figure 6 shows examples of planning and execution failures during online evaluation.

Real robot results. We further integrate Guardian into 3DLotus++ [9] on three unseen manipulation tasks with real robot. We assess the success rate (%) both with and without Guardian across ten episodes per task—five under normal conditions and five subjected to human-induced perturbations. These perturbations are designed to evaluate the robotic policy's ability to replan and recover effectively from erroneous previous actions, rather than blindly continuing its initial plan. Results in Table 5b show that Guardian performs strongly, particularly in perturbed setups for closed-loop planning and execution. More detail is provided in Section A of the supplementary material.

6 Conclusion

This paper presents Guardian, a VLM designed to enhance the robustness of robotic manipulation by accurately detecting planning and execution failures. Addressing the scarcity of failure data, we develop an automated pipeline to generate diverse failure scenarios, resulting in two new benchmarks: RLBench-Fail (simulation) and BridgeDataV2-Fail (real-world). These benchmarks facilitate standardized evaluation and further research in failure detection. Guardian achieves state-of-the-art performance across these datasets and generalizes well to the real-world RoboFail and UR5-Fail benchmarks. Furthermore, we showcase Guardian's plug-and-play utility, improving task success rates when integrated into the 3D-LOTUS++ framework by enabling failure detection and triggering retries in simulated and real-world manipulation tasks.

References

[1] A. Hurst and al. GPT-4o system card. *arXiv:2410.21276*, 2024. URL https://arxiv.org/abs/2410.21276.

[2] A. Grattafiori and al. The Llama 3 herd of models. *arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

[3] Z. Chen, J. Wu, W. Wang, W. Su, G. Chen, S. Xing, M. Zhong, Q. Zhang, X. Zhu, L. Lu, B. Li, P. Luo, T. Lu, Y. Qiao, and J. Dai. InternVL: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *CVPR*, 2024.

[4] R. Sinha, A. Sharma, S. Banerjee, T. Lew, R. Luo, S. M. Richards, Y. Sun, E. Schmerling, and M. Pavone. A system-level view on out-of-distribution data in robotics. *arXiv:2212.14020*, 2023. URL https://arxiv.org/abs/2212.14020.

[5] K. Kawaharazuka, T. Matsushima, A. Gambardella, J. Guo, C. Paxton, and A. Zeng. Real-world robot applications of foundation models: A review. *AR*, 2024.

[6] O. Kroemer, S. Niekum, and G. Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *JMLR*, 2020.

[7] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, and T. Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM TOIS*, 2025.

[8] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147. PMLR, 2022.

[9] R. Garcia, S. Chen, and C. Schmid. Towards generalizable vision-language robotic manipulation: A benchmark and LLM-guided 3D policy. In *ICRA*, 2025.

[10] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox. RVT: Robotic view transformer for 3D object manipulation. In *CoRL*, 2023.

[11] K. Wu, C. Hou, J. Liu, Z. Che, X. Ju, Z. Yang, M. Li, Y. Zhao, Z. Xu, G. Yang, S. Fan, X. Wang, F. Liao, Z. Zhao, G. Li, Z. Jin, L. Wang, J. Mao, N. Liu, P. Ren, Q. Zhang, Y. Lyu, M. Liu, J. He, Y. Luo, Z. Gao, C. Li, C. Gu, Y. Fu, D. Wu, X. Wang, S. Chen, Z. Wang, P. An, S. Qian, S. Zhang, and J. Tang. RoboMIND: Benchmark on multi-embodiment intelligence normative data for robot manipulation. *arXiv:2412.13877*, 2025. URL https://arxiv.org/abs/2412.13877.

[12] H. P. Young. Learning by trial and error. *Games and Economic Behavior*, 2009.

[13] V. Chen, Q. V. Liao, J. W. Vaughan, and G. Bansal. Understanding the role of human intuition on reliance in human-ai decision-making with explanations, 2023.

[14] Z. Liu, A. Bahety, and S. Song. REFLECT: Summarizing robot experiences for failure explanation and correction. In *CoRL*, 2023.

[15] C. Agia, R. Sinha, J. Yang, Z. ang Cao, R. Antonova, M. Pavone, and J. Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. In *CoRL*, 2024.

[16] H. Etukuru, N. Naka, Z. Hu, S. Lee, J. Mehu, A. Edsinger, C. Paxton, S. Chintala, L. Pinto, and N. M. M. Shafiullah. Robot utility models: General policies for zero-shot deployment in new environments. In *ICRA*, 2025.

[17] 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. In *ICLR*, 2025.

[18] A. Khazatsky and al. DROID: A large-scale in-the-wild robot manipulation dataset. In *RSS*, 2024.

[19] E. Collaboration and al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *ICRA*, 2024.

[20] W. Pumacay, I. Singh, J. Duan, R. Krishna, J. Thomason, and D. Fox. THE COLOSSEUM: A benchmark for evaluating generalization for robotic manipulation. In *RSS*, 2024.

[21] H. Chen, Y. Yao, R. Liu, C. Liu, and J. Ichnowski. Automating robot failure recovery using vision-language models with optimized prompts. *arXiv:2409.03966*, 2024. URL https://arxiv.org/abs/2409.03966.

[22] Q. Bu, J. Cai, L. Chen, X. Cui, Y. Ding, S. Feng, S. Gao, X. He, X. Huang, S. Jiang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

[23] W. Zhao, J. P. Queralta, and T. Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *SSCI*, 2020.

[24] G. De Giacomo, R. Reiter, M. Soutchanski, et al. Execution monitoring of high-level robot programs. In *KR*, 1998.

[25] M. Gianni, P. Papadakis, F. Pirri, M. Liu, F. Pomerleau, F. Colas, K. Zimmermann, T. Svoboda, T. Petricek, G.-J. M. Kruijff, et al. A unified framework for planning and execution-monitoring of mobile robots. *AAAI Workshop on Automated Action Planning for Autonomous Mobile Robots*, 2011.

[26] H. Ha, P. Florence, and S. Song. Scaling up and distilling down: Language-guided robot skill acquisition. In *CoRL*, 2023.

[27] Y. J. Ma, J. Hejna, A. Wahid, C. Fu, D. Shah, J. Liang, Z. Xu, S. Kirmani, P. Xu, D. Driess, T. Xiao, J. Tompson, O. Bastani, D. Jayaraman, W. Yu, T. Zhang, D. Sadigh, and F. Xia. Vision language models are in-context value learners. In *ICLR*, 2025.

[28] K. Shirai, C. C. Beltran-Hernandez, M. Hamaya, A. Hashimoto, S. Tanaka, K. Kawaharazuka, K. Tanaka, Y. Ushiku, and S. Mori. Vision-Language interpreter for robot task planning. In *ICRA*, 2024.

[29] M. Skreta, Z. Zhou, J. L. Yuan, K. Darvish, A. Aspuru-Guzik, and A. Garg. RePLan: Robotic replanning with perception and language models. *arXiv:2401.04157*, 2024. URL https://arxiv.org/abs/2401.04157.

[30] A. Mei, G.-N. Zhu, H. Zhang, and Z. Gan. ReplanVLM: Replanning robotic tasks with visual language models. *IEEE RA-L*, 2024.

[31] Y. Du, K. Konyushkova, M. Denil, A. Raju, J. Landon, F. Hill, N. de Freitas, and S. Cabi. Vision-Language models as success detectors. In *CoLLAs*, 2023.

[32] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. In *NeurIPS*, 2023.

[33] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

[34] A. Goyal, V. Blukis, J. Xu, Y. Guo, Y.-W. Chao, and D. Fox. RVT-2: Learning precise manipulation from few demonstrations. In *RSS*, 2024.

[35] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu. 3D Diffusion Policy: Generalizable visuomotor policy learning via simple 3D representations. In *RSS*, 2024.

[36] J. Duan, W. Yuan, W. Pumacay, Y. R. Wang, K. Ehsani, D. Fox, and R. Krishna. Manipulate-Anything: Automating real-world robots using vision-language models. *arXiv:2406.18915*, 2024. URL https://arxiv.org/abs/2406.18915.

[37] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. RLBench: The robot learning benchmark learning environment. *IEEE RA-L*, 2020.

[38] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-Actor: A multi-task transformer for robotic manipulation. In *CoRL*, 2022.

[39] H. Walke, K. Black, A. Lee, M. J. Kim, M. Du, C. Zheng, T. Zhao, P. Hansen-Estruch, Q. Vuong, A. He, V. Myers, K. Fang, C. Finn, and S. Levine. BridgeData V2: A dataset for robot learning at scale. In *CoRL*, 2023.

[40] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine. Robotic control via embodied chain-of-thought reasoning. In *CoRL*, 2024.

[41] K. Wu, C. Hou, J. Liu, Z. Che, X. Ju, Z. Yang, M. Li, Y. Zhao, Z. Xu, G. Yang, et al. Robomind: Benchmark on multi-embodiment intelligence normative data for robot manipulation. *arXiv preprint arXiv:2412.13877*, 2024.

[42] Z. Chen and al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv:2412.05271*, 2025. URL https://arxiv.org/abs/2412.05271.

[43] C. Agia, R. Sinha, J. Yang, Z.-a. Cao, R. Antonova, M. Pavone, and J. Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. *arXiv preprint arXiv:2410.04640*, 2024.

[44] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.