Scene Graph-Guided Proactive Replanning for Failure-Resilient Embodied Agents

Che Rin Yu[1] Daewon Chae[1] Dabin Seo[1] Yoonha Jang[2] Sangwon Lee[2] Hyeongwoo IM[2] Jinkyu Kim[1]
[1]Korea University [2]KT (Korea Telecom) R&D Center
{eyucherin, cdw098, lemonstar99, jinkyukim}@korea.ac.kr
{yooona.jang, lee.sangwon, im.hyeongwoo}@kt.com

***Abstract*—Replanning is essential for robust robotic behavior in unexpected environments. However, most existing approaches often initiate replanning only after failures occur, which can lead to irreversible errors or inefficient recovery actions. Proactive replanning—detecting and correcting potential failures before execution—offers a promising alternative, yet current methods often rely on manual triggers, dense supervision, or costly annotations. In this work, we propose a novel proactive replanning framework that detects and corrects failures at subtask boundaries by comparing scene graphs against those representing the target precondition states for each subtask, triggering reasoning and corrective planning when graph-level discrepancies arise. This explicit symbolic and spatial grounding enables the system to robustly identify subtle violations of subtask preconditions that are not apparent from object identities alone. Our method requires no dense annotations, minimizes reliance on large language models (LLMs), and supports real-time replanning without resetting the scene. Empirical results with the AI2-THOR simulator demonstrate that our approach effectively identifies semantic and spatial mismatches before execution failures occur, enabling robust and efficient task completion across complex manipulation scenarios.**

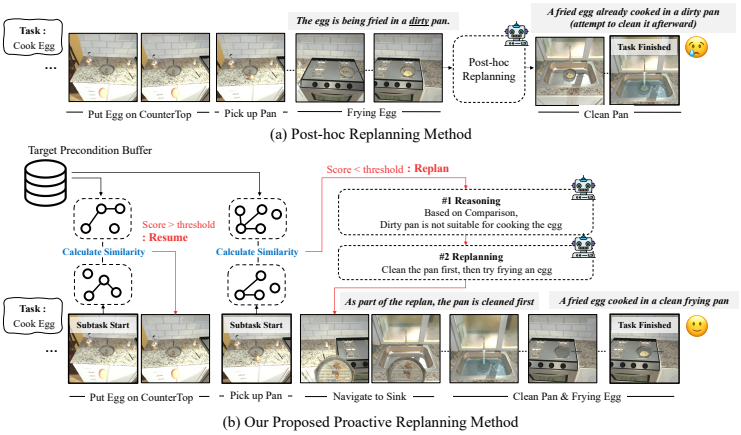


Fig. 1: (a) A conventional post-hoc replanning strategy, where the agent reacts only after a failure occurs—e.g., detecting an egg in a dirty pan and then replanning. In contrast, (b) our proactive strategy monitors preconditions and replans in advance—e.g., identifying the dirty pan beforehand and cleaning it before cooking.

I. Introduction

Autonomous robots are increasingly capable of performing complex tasks, promising transformative applications in unstructured real-world environments [9, 3, 35]. However, to operate reliably, robots must dynamically adapt their behavior in response to unexpected changes, errors, or failures that arise during task execution. This makes replanning a fundamental requirement for robust autonomy, particularly in unpredictable settings where failures may involve irreversible or safety-critical consequences that cannot be mitigated through simple recovery mechanisms.

However, building robust replanning capabilities for autonomous systems remains fundamentally challenging, requiring the following three components: (*i*) determining the optimal time for replanning interventions in an efficient manner, (*ii*) accurately diagnosing the root causes of failures or potential failure conditions, and (*iii*) generating effective and corrective action sequences that can recover progress toward the task goal. Successfully addressing these components is essential for enabling robots to operate safely and reliably across diverse environments.

These challenges are further compounded in long-horizon tasks, where failures often arise from subtle, often overlooked assumptions about spatial configurations [15, 29, 32]. For instance, whether an object is properly placed, partially obstructed, or held can critically affect subtask feasibility. As a result, robust failure reasoning and analysis require not only high-level symbolic abstraction, but also fine-grained spatial grounding that accurately captures both the relational and geometric structure of the environment. This motivates the development of structured spatial representations capable of supporting reliable detection, reasoning, and replanning in response to failures.

While some recent works have made notable progress toward addressing these challenges, many approaches rely mainly on post-hoc mechanisms (i.e., responding only after failures emerge) [20, 11, 27, 31], predefined rule-based triggers [33, 24, 16], or expensive human supervision [36, 28, 19, 5]. Consequently, they struggle to proactively prevent failures in unexpected environments. These limitations highlight the need for proactive replanning frameworks that can detect potential failures before execution and adapt plans accordingly in complex, unforeseen scenarios—much like how humans recognize discrepancies between the current and expected environment and adjust their actions before failures fully

materialize.

In this paper, we propose a proactive replanning strategy in which an agent anticipates potential failures during the execution of long-horizon tasks (e.g., cooking an egg), as illustrated in Figure 1. This approach contrasts with conventional post-hoc replanning strategies that address failures only after they occur (compare Figure 1 (a) and (b)). Specifically, at the beginning of each sub-task (e.g., "picking up a pan"), the agent compares the scene graph of the current environment with an expected scene graph derived from prior successful demonstrations. If the graph-level similarity falls below a predefined threshold, an online replanning process is initiated. This process generates a reasoning chain to identify the likely cause of failure (e.g., "a dirty pan is unsuitable for cooking") and subsequently formulates corrective actions (e.g., "clean the pan before frying the egg"), thereby avoiding the predicted failure and enabling successful task completion. We validate the effectiveness of our method using the AI2-THOR simulator [17], demonstrating the agent's ability to preemptively avoid or recover from potential failures during task execution. Our contributions are as follows:

- We propose a novel proactive replanning approach that preemptively identifies and mitigates potential failures to ensure reliable achievement of target objectives in long-horizon tasks
- We present a lightweight scene graph-based failure anticipation method that, at the onset of each sub-task, assesses whether the current scene is suitable for executing actions necessary to achieve the overall task objectives.
- We demonstrate the effectiveness of our approach, showing that it not only improves failure detection and task success rates compared to baseline methods, but also enhances failure reasoning quality as assessed by human evaluations.

II. Method

A. Problem Formulation

We consider the problem of executing long-horizon tasks, each defined by a desired success condition C_{goal} that describes the final state the robot must achieve (e.g., "a mug filled with coffee is placed on the table"). To achieve C_{goal}, a task $\mathcal{T}$ is decomposed into an ordered sequence of n high-level subtasks, $\mathcal{T} = [a_1, a_2, \ldots, a_n]$, where each subtask a_i corresponds to a semantically meaningful action (e.g., "grab mug," "put mug on coffee machine"). Each subtask is intended to transition the environment closer to satisfying C_{goal}, such that the full execution of $\mathcal{T}$ leads to potential task completion under expected conditions.

At execution time, the robot receives an RGB-D observation I_i of the environment immediately before executing each subtask a_i, from which it extracts a structured semantic representation in the form of a scene graph G_i that captures the entities and their relations present at that moment. To assess task progress and detect potential failures, the robot compares G_i to the expected scene graph $\hat{G}_i$ inferred from the target precondition state P_i, which may be derived from a successful (i.e., reference) trajectory, manual labels from human annotation, or descriptions generated by large language models (LLMs) based on commonsense knowledge. In our setting, we leverage a dataset of reference trajectories $\mathcal{D}$, where each trajectory $\tau \in \mathcal{D}$ specifies how a high-level task $\mathcal{T}$ should proceed under expected conditions, including the final success condition C_{goal} and the ordered sequence of subtasks $[a_1, a_2, \ldots, a_n]$ required for completion. From these trajectories, we extract the expected precondition state for each subtask, which serves as an intrinsic reference for comparison during execution.

B. 3D Scene Graph Construction

To represent the robot's semantic understanding of the environment prior to executing each subtask, we construct a task-informed scene graph from the robot's RGB-D observation. Each scene graph captures objects, their states, and spatial relationships that are relevant to the current subtask context. Our approach is inspired by the scene graph construction methods proposed in REFLECT [20], which summarize multimodal sensory inputs into symbolic structures for post-hoc failure explanation, and RoboEXP [14], which introduces an action-conditioned 3D scene graph that models interactive and spatial relations between objects and actions.

Instead of generating scene graphs per frame, we construct one at the start of each subtask to check if preconditions are met. From an RGB-D image, we detect objects, segment regions, and infer state attributes via CLIP [23]. Depth is used to project the scene into a 3D point cloud, enabling spatial reasoning. We extract pairwise object relations using spatial heuristics, and use the gripper state to identify robot-object interactions. The resulting graph $G_i = (V_i, E_i)$ contains object nodes, state attributes, spatial edges, and a subtask node for context.

C. Expected Scene Inference

In order to evaluate whether the environment satisfies the conditions for successful subtask execution, we infer expected scene graphs from target precondition states. In this setting, we assume a single reference trajectory generated offline, serving solely as a static reference for expected subtask conditions. During deployment, the robot independently constructs scene graphs from its own perturbed sensory observations and compares them against expected scene graph references for discrepancy detection. If the mismatch exceeds a predefined threshold, the system triggers proactive replanning. This design not only ensures that failure reasoning remain grounded in imperfect inputs, but also provides a consistent and interpretable references for detecting deviations from expected task progress without requiring dense supervision or extensive manual annotations.

D. Graph-Based Discrepancy Analysis for Failure Detection

To assess whether the current environment satisfies the expected conditions for subtask execution, we compare the observed scene graph G_i^{obs} with the expected graph G_i^{exp} using

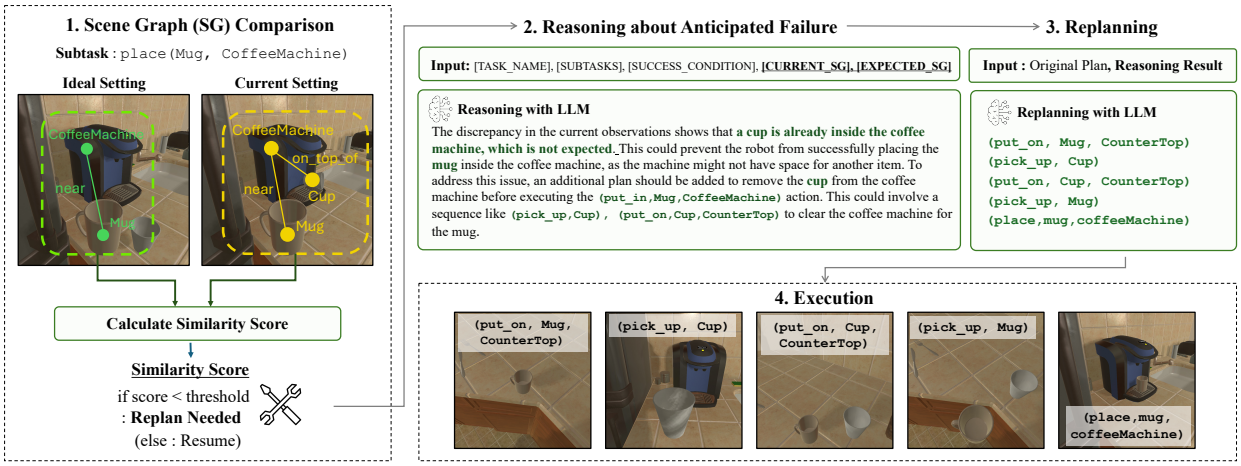


Fig. 2: Our method consists of four steps: (1) compute scene graph similarity to assess subtask feasibility; (2) if below a threshold, use an LLM to infer likely failure causes; (3) generate corrective actions using the original plan and reasoning; (4) execute the revised plan.

a graph based discrepancy analysis algorithm. Each graph consists of a set of object nodes, their associated attributes (e.g., object states), and labeled edges denoting spatial or functional relationships.

Node similarity S_{node} is computed as the average cosine similarity between matched object nodes, using either CLIP embeddings or semantic segmentation features that encode both object class and state. To penalize extra or missing nodes, the sum is normalized by the total number of unique nodes across both graphs. When the similarity score S between the observed and expected scene graphs falls below a threshold (e.g., $S < 0.9$), the system predicts that the current subtask's preconditions are likely violated and initiates proactive replanning. Further explanation on can be found in the appendix.

E. Replanning Strategy

Following the detection of a potential failure, our framework decomposes the replanning process into two modular components: a reasoning module and a replanning module, enabling the robot to proactively adjust its plan before executing a subtask that may otherwise result in failure, as illustrated in Figure 2.The reasoning module is responsible for interpreting the cause of failure. Upon detecting a discrepancy, natural language descriptions of the observed and expected scene graphs, along with the task goal and current subtask context, are passed to a language-based reasoning model (i.e., GPT-4o). The model is prompted to explain why the mismatch may lead to task failure and what changes must be made to the current situation or plan to recover from the failure.

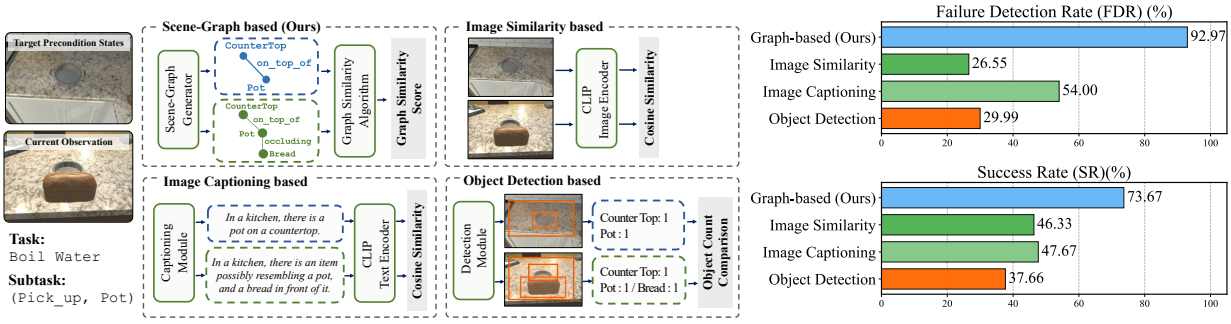
The output of the reasoning module is then passed to the replanning module, which is subsequently invoked to generate recovery plans. This component takes as input the robot's current state, the list of available high-level actions, the observable objects in the scene, and the reasoning-informed constraints or suggestions. The replanning model uses this information to generate a revised action sequence that corrects the issue and restores progress toward the task goal. The new plan is then executed online, enabling the robot to dynamically recover from unexpected deviations or environmental constraints in real time.

III. Experiments

Implementation and Evaluation Details. All experiments are conducted in the AI2-THOR simulator [17] using GPT-4o [1] for language reasoning. We construct a precondition buffer by storing RGB-D observations before each subtask in a successful demonstration. At runtime, scene graphs from stored and current observations are compared to detect potential failures. We evaluate on complex tasks from the RoboFail dataset [20], which includes long-horizon manipulation scenarios with diverse failure cases. Additional task details are provided in Section C.

Effects of Proactive Replanning. To evaluate the effectiveness of our proposed proactive replanning method, we construct a benchmark of 10 tasks—extending the RoboFail dataset—with scenarios specifically curated to test the robot's ability to avoid or recover from irreversible failures. We assess performance using task success rate (SR) and Total Execution Time (TET), which measures the average total duration of task execution including time spent on replanning for all tasks. Our method is compared against two baselines that rely on post-hoc replanning with language-based reasoning. The first follows the original REFLECT [20] framework, where replanning is triggered only after the entire task has been completed. The second adapts REFLECT to an online setting, performing verification at the end of each subtask to detect failures. Both baselines are reactive, differing only in whether replanning is triggered at the end of the task or during execution. Further results that illustrate this distinction can be found in Section D.

Analysis of Potential Failure Detection. As shown in Figure 3 (a), we validate our scene graph-based failure detection method by comparing it with the following three alternative approaches: (*i*) image-level comparison using CLIP [23] embedding space, (*ii*) caption-based comparison in which scene descriptions are generated by vision-language models, and



(a) Overview of Potential Failure Detection Approaches (b) Comparison of Performance

Fig. 3: (a) Comparison of baseline methods and our scene graph-based approach for detecting discrepancies between current and expected preconditions. Baselines include: (1) **Image Similarity** via CLIP embeddings, (2) **Image Captioning** via text comparisons, (3) **Object Detection** via object count matching, and (4) our **Scene Graph** method using structured graph analysis. (b) Failure detection and task success rates across all approaches.

TABLE I: Results of proactive replanning compared to alternative methods. (**SR: Success Rate, TET: Total Execution Time in seconds**)

Method	Type	SR (%) ↑	TET (sec) ↓
REFLECT [20]	Offline replanning *after* task	20	151.0
REFLECT-online	Online replanning *after* failure	30	133.9
Ours	Online replanning *before* failure	70	117.0
Baseline	No replanning	0	84.6

(*iii*) object-level similarity scoring based on detected object bounding boxes. Additional implementation details for each method are provided in Section E. Each method receives the same RGB-D observation at the beginning of a subtask and assesses whether the current scene aligns well with the expected conditions derived from a reference trajectory. If the level of discrepancy exceeds a predefined threshold, the method predicts a potential failure. To ensure a fair comparison, we vary the similarity threshold values (90%, 85%, and 80%) and report the average failure detection rates (FDR) and task success rates (SR). Comprehensive results for all thresholds are in Section F.

The evaluation results in Figure 3 (b) demonstrate that our scene graph-based approach consistently achieves higher failure detection rates (FDR) and task success rates (SR) compared to all baseline methods. In contrast, approaches relying solely on visual or semantic cues demonstrate substantially lower performance. While the object detection-based method provides object-level recognition, it lacks the spatial reasoning necessary to evaluate the relational configurations critical for subtask feasibility. These results highlight the importance of incorporating explicit symbolic and spatial reasoning for reliable proactive replanning.

Analysis of Anticipatory Failure Reasoning. Building on the same setup for failure detection, we further evaluate the effectiveness of our framework in reasoning about failure cases. To ensure a fair comparison, we take episodes where all methods correctly identified a failure and evaluate their ability to explain the cause of that failure. For each method, we generate a natural language explanation using a LLM (GPT-4o) [1], conditioned on the method's internal representation of the scene, the current subtask, and the overall task goal. In our approach, the LLM is prompted with structured scene graph differences, whereas for the baselines, it is prompted with either raw image embeddings, generated captions, or detected object-level features.

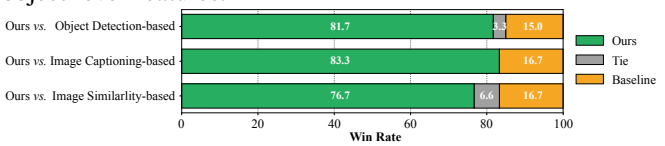


Fig. 4: Human evaluation on reasoning, compared to baseline methods.

To assess explanation quality, we conduct a user study with 15 human evaluators. In each case, evaluators are provided with the ground-truth failure cause and two candidate explanations (one generated by our method and one by a baseline method) and corresponding visual observations. Annotators are instructed to compare the two explanations and select the one better aligned with the ground-truth reasoning. If they find both explanations equally valid or are unsure, they are allowed to select both. More information about this is in Section H. As shown in Figure 4, our method significantly outperforms all baselines in explanation quality, underscoring the critical role of spatial reasoning in failure understanding.

IV. Conclusion

We propose a proactive replanning framework that anticipates and corrects failures during robotic task execution by analyzing scene graph discrepancies. Our method enables early failure detection and real-time recovery without relying on dense supervision or frequent LLM calls. Experiments on the RoboFail benchmark show that our approach significantly improves both failure detection and task success, demonstrating the importance of structured spatial reasoning for reliable and adaptive robot behavior.

V. Acknowledgment

This work was the result of project supported by Korea University - KT (Korea Telecom) R&D Center

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 3, 4, 9

[2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 7

[3] Erik Bauer, Marc Blöchlinger, Pascal Strauch, Arman Raayatsanati, Curdin Cavelti, and Robert K Katzschmann. An open-source soft robotic platform for autonomous aerial manipulation in the wild. *arXiv preprint arXiv:2409.07662*, 2024. 1

[4] Cristina Cornelio and Mohammed Diab. Recover: A neuro-symbolic framework for failure detection and recovery. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12435–12442. IEEE, 2024. 7

[5] Yuchen Cui, Siddharth Karamcheti, Raj Palleti, Nidhya Shivakumar, Percy Liang, and Dorsa Sadigh. No, to the right: Online language corrections for robotic manipulation via shared autonomy. In *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*, pages 93–101, 2023. 1, 7

[6] Yinpei Dai, Jayjun Lee, Nima Fazeli, and Joyce Chai. Racer: Rich language-guided failure recovery policies for imitation learning, 2024. 7

[7] Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. AHA: A vision-language-model for detecting and reasoning over failures in robotic manipulation. In *The Thirteenth International Conference on Learning Representations*, 2025. 7

[8] Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. Every picture tells a story: Generating sentences from images. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*, pages 15–29. Springer, 2010. 7

[9] Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. Mobile ALOHA: Learning bimanual mobile manipulation using low-cost whole-body teleoperation. In *8th Annual Conference on Robot Learning*, 2024. 1

[10] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 7

[11] Yanjiang Guo, Yen-Jen Wang, Lihan Zha, and Jianyu 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)*, pages 12124–12131. IEEE, 2024. 1, 7

[12] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147. PMLR, 2022. 7

[13] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022. 7

[14] Hanxiao Jiang, Binghao Huang, Ruihai Wu, Zhuoran Li, Shubham Garg, Hooshang Nayyeri, Shenlong Wang, and Yunzhu Li. RoboEXP: Action-conditioned scene graph via interactive exploration for robotic manipulation. In *8th Annual Conference on Robot Learning*, 2024. 2

[15] Amita Kamath, Jack Hessel, and Kai-Wei Chang. What's" up" with vision-language models? investigating their struggle with spatial reasoning. *arXiv preprint arXiv:2310.19785*, 2023. 1, 7

[16] Jinyeon Kim, Cheolhong Min, Byeonghwi Kim, and Jonghyun Choi. Pre-emptive action revision by environmental feedback for embodied instruction following agents. In *8th Annual Conference on Robot Learning*, 2024. 1, 7

[17] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017. 2, 3, 8, 9

[18] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023. 7

[19] Huihan Liu, Alice Chen, Yuke Zhu, Adith Swaminathan, Andrey Kolobov, and Ching-An Cheng. Interactive robot learning from verbal correction. *arXiv preprint arXiv:2310.17555*, 2023. 1

[20] Zeyi Liu, Arpit Bahety, and Shuran Song. REFLECT: Summarizing robot experiences for failure explanation and correction. In *7th Annual Conference on Robot Learning*, 2023. 1, 2, 3, 4, 7, 8, 9

[21] Vicente Ordonez, Girish Kulkarni, and Tamara Berg. Im2text: Describing images using 1 million captioned photographs. *Advances in neural information processing systems*, 24, 2011. 7

[22] Jia-Yu Pan, Hyung-Jeong Yang, Pinar Duygulu, and Christos Faloutsos. Automatic image captioning. In *2004 IEEE International Conference on Multimedia and Expo (ICME)(IEEE Cat. No. 04TH8763)*, volume 3, pages 1987–1990. IEEE, 2004. 7

[23] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 2, 3, 9

[24] Shreyas Sundara Raman, Vanya Cohen, Ifrah Idrees, Eric Rosen, Raymond Mooney, Stefanie Tellex, and David Paulius. Cape: Corrective actions from precondition errors using large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14070–14077. IEEE, 2024. 1, 7

[25] Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning. *arXiv preprint arXiv:2307.06135*, 2023. 7

[26] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 7

[27] Gabriel Herbert Sarch, Yue Wu, Michael J. Tarr, and Katerina Fragkiadaki. Open-ended instructable embodied agents with memory-augmented large language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. 1, 7

[28] Lucy Xiaoyang Shi, Zheyuan Hu, Tony Z. Zhao, Archit Sharma, Karl Pertsch, Jianlan Luo, Sergey Levine, and Chelsea Finn. Yell at your robot: Improving on-the-fly from language corrections. *CoRR*, abs/2403.12910, 2024. 1, 7

[29] Fatemeh Shiri, Xiao-Yu Guo, Mona Golestan Far, Xin Yu, Gholamreza Haffari, and Yuan-Fang Li. An empirical analysis on spatial reasoning capabilities of large multimodal models. *arXiv preprint arXiv:2411.06048*, 2024. 1, 7

[30] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023. 7

[31] Shu Wang, Muzhi Han, Ziyuan Jiao, Zeyu Zhang, Ying Nian Wu, Song-Chun Zhu, and Hangxin Liu. LLM3: Large language model-based task and motion planning with motion failure reasoning. In *Multi-modal Foundation Model meets Embodied AI Workshop @ ICML2024*, 2024. 1, 7

[32] Yutaro Yamada, Yihan Bao, Andrew K Lampinen, Jungo Kasai, and Ilker Yildirim. Evaluating spatial understanding of large language models. *arXiv preprint arXiv:2310.14540*, 2023. 1, 7

[33] Zejun Yang, Li Ning, Haitao Wang, Tianyu Jiang, Shaolin Zhang, Shaowei Cui, Hao Jiang, Chunpeng Li, Shuo Wang, and Zhaoqi Wang. Text2reaction: Enabling reactive task planning using large language models. *IEEE Robotics and Automation Letters*, 2024. 1, 7

[34] Benjamin Z Yao, Xiong Yang, Liang Lin, Mun Wai Lee, and Song-Chun Zhu. I2t: Image parsing to text description. *Proceedings of the IEEE*, 98(8):1485–1508, 2010. 7

[35] Zhecheng Yuan, Tianming Wei, Shuiqi Cheng, Gu Zhang, Yuanpei Chen, and Huazhe Xu. Learning to manipulate anywhere: A visual generalizable framework for reinforcement learning. In *8th Annual Conference on Robot Learning*, 2024. 1

[36] Lihan Zha, Yuchen Cui, Li-Heng Lin, Minae Kwon, Montserrat Gonzalez Arenas, Andy Zeng, Fei Xia, and Dorsa Sadigh. Distilling and retrieving generalizable knowledge for robot manipulation via language corrections. In *2nd Workshop on Language and Robot Learning: Language as Grounding*, 2023. 1, 7

[37] Bohan Zhai, Shijia Yang, Chenfeng Xu, Sheng Shen, Kurt Keutzer, Chunyuan Li, and Manling Li. Halle-control: controlling object hallucination in large multimodal models. *arXiv preprint arXiv:2310.01779*, 2023. 7

[38] Yiyang Zhou, Chenhang Cui, Jaehong Yoon, Linjun Zhang, Zhun Deng, Chelsea Finn, Mohit Bansal, and Huaxiu Yao. Analyzing and mitigating object hallucination in large vision-language models. *arXiv preprint arXiv:2310.00754*, 2023. 7

APPENDIX A
RELATED WORKS

Strategies for Robotic Replanning. There has been a growing interest in replanning strategies in the robotics community where approaches have been introduced based on predefined rule-based triggers [33, 24, 16, 4], human-in-the-loop strategies [36, 28, 5], post-hoc replanning strategies [20, 11, 27, 31], and large vision-language model-based methods [7, 6]. Human-in-the-loop [36, 28, 5] strategies offer operational flexibility by allowing supervisory intervention during task execution, but they introduce scalability challenges, which are impractical for fully autonomous operation, and impose significant labor costs. Post-hoc replanning methods [20, 11, 27, 31], which analyze completed subtasks or trajectories to identify failure points, are shown effective but unsuitable when corrective action must occur before a failure materializes. Moreover, these approaches inherently struggle with irreversible failures where retrospective corrections are infeasible. Lastly, more recent methods leverage vision-language models trained on failed trajectories [7, 6] show promise in recognizing failure scenarios, but require large-scale annotations, incure data collection costs, and often fail to generalize to rare or unseen failure modes.

While some recent works have attempted proactive replanning [33, 24, 16, 4], they rely on pre-defined triggers and react only to anticipated conditions, limiting their ability to handle novel or unforeseen situations. To address this, we present a scene graph-based proactive replanning method, which is flexible and robust in long-horizon robotic tasks as it supports early detection of potential failures and real-time corrective replanning.

Multi-modal Large Language Models (LLMs) in Robotics. Multi-modal large language models (LLMs) have been widely used in robotic task planning [2, 13, 12, 25, 12], translating high-level instructions into action sequences or executable code [30, 18]. Similarly, Vision-Language Models (VLMs) interpret visual scenes [22, 8, 21, 34] and detect task-relevant objects [26, 10] to extract visual information and align semantic cues in the environment. However, VLMs often overlook geometric and precise spatial structures, and struggle with occlusions, spatial constraints, and geometric plausibility in complex scenes [15, 29, 32]. Moreover, LLMs and VLMs may hallucinate [38, 37] and cause errors while assessing environment states, where minor spatial inconsistencies such as occluded objects, misplaced items, or obstructed targets, may silently violate task preconditions in failure detection. Solely relying on high level semantics may not detect such failures, highlighting the need for explicit spatial reasoning to detect discrepancies in real time. To address this, we propose a novel framework combining symbolic scene graph comparisons with selective LLM reasoning for proactive failure detection, while also efficiently leveraging LLMs with minimal inference overhead.

APPENDIX B
GRAPH-BASED DISCREPANCY ANALYSIS FOR FAILURE DETECTION

Node similarity S_{node} is computed as the average cosine similarity between matched object nodes, using either CLIP embeddings or semantic segmentation features that encode both object class and state. To penalize extra or missing nodes, the sum is normalized by the total number of unique nodes across both graphs:

$$S_{\text{node}} = \frac{1}{|\mathcal{V}_{\text{exp}} \cup \mathcal{V}_{\text{obs}}|} \sum_{(v_i^{\text{obs}}, v_i^{\text{exp}})} \cos\left(f(v_i^{\text{obs}}), f(v_i^{\text{exp}})\right) \tag{1}$$

Edge similarity (S_{edge}) and structural similarity (S_{struc}) provide complementary views of structural alignment between graphs. Specifically, S_{edge} measures how well spatial or functional relationships align by computing the ratio of correctly matched edges to the total number of unique edges, while S_{struc} assesses the consistency of node connectivity by measuring differences in node degrees across matched pairs. The two metrics are defined as:

$$S_{\text{edge}} = \frac{|\mathcal{E}_{\text{matched}}|}{|\mathcal{E}_{\text{exp}} \cup \mathcal{E}_{\text{obs}}|} \quad , \quad S_{\text{struc}} = 1 - \frac{1}{N} \sum_{i=1}^{N} \frac{|\deg(v_i^{\text{obs}}) - \deg(v_i^{\text{exp}})|}{D} \tag{2}$$

We compute a graph similarity score S by averaging the three normalized components:

$$S = \text{avg}(S_{\text{node}}, S_{\text{edge}}, S_{\text{struc}}) \tag{3}$$

When the similarity score S between the observed and expected scene graphs falls below a threshold (e.g., $S < 0.9$), the system predicts that the current subtask's preconditions are likely violated and initiates proactive replanning.

APPENDIX C
ROBOFAIL TASK DESCRIPTIONS

The RoboFail dataset, introduced in REFLECT [20], is built in simulation using AI2-THOR [17]. It contains task execution data where various failure cases are manually injected to study failure reasoning. The dataset includes 100 failure scenarios, covering 10 distinct tasks with 10 failure cases per task. About 11% of failures in RoboFail—such as dropped objects—cannot be detected purely from pre-execution observations, we handle these cases by evaluating failure detection immediately before the next subtask, once the effects become visually apparent.

TABLE A1: Polished task descriptions for T1–T10 with corrected numbering and horizontal lines.

Task ID	Task Name	Description
T1	Boil Water	A pot filled with water is placed on a stove burner that is actively heating.
T2	Toast Bread	A slice of bread is positioned inside a toaster that is currently running.
T3	Fry Egg	A cracked egg is cooking in a pan situated on a stove burner that is turned on.
T4	Heat Potato	A potato rests on a plate inside a microwave that is operating.
T5	Make Coffee	A clean mug filled with coffee is placed neatly on the countertop.
T6	Store Egg	A bowl containing an egg is stored securely inside the refrigerator.
T7	Water Plant	The houseplant has been watered and its soil appears moist.
T8	Switch Devices	A closed laptop is placed on the TV stand, and the television is powered on.
T9	Make Salad	A bowl of freshly sliced lettuce, tomato, and potato is kept chilled in the fridge.
T10	Serve Warm Water	A mug of water, heated in the microwave, is served on the dining table.

APPENDIX D
DETAIL RESULTS ON PROACTIVE REPLANNING

Table A2 shows evaluation across a set of robotic tasks under diverse failure injection scenarios, especially in irreversible settings. Each task is injected with a specific failure condition (i.e., incorrect object contents or unintended irreversible actions) to examine the effectiveness of the system's ability to replan. We compare our method against REFLECT [20] and a reactive online variant (REF-online) inspired by the original reflect implementation. For each method, we record whether the task was successfully completed with the new plan and the total execution time in seconds. The success rate is solely determined by the final state of the execution, and whether it satisfied the task provided.

TABLE A2: Experimental results on various failure cases on different tasks. We compare our proposed replanning method with REFLECT [20] and its online version (i.e., REF-online) we implemented based on the original code. Column **Success** shows whether or not the method successfully replanned, avoiding the injected failure, and completed the task. We also record the Total Execution Time (**TET**) in seconds.

Task	Task Description	Failure Injection	Method	Success	TET
Water Plant	Water the houseplant with water	Pot is filled with wine in the beginning of task execution.	REFLECT	False	122
			REF-online	False	78
			Ours	True	43
Water Plant	Water the houseplant with water	Pot is filled with coffee in the beginning of task execution.	REFLECT	False	70
			REF-online	False	43
			Ours	True	43
Cook Egg	Cook an egg on a clean pan	Pan is dirty in the beginning of the task.	REFLECT	True	134
			REF-online	True	144
			Ours	False	159
Cook Egg	Cook an egg on a clean pan	There is a potato on top of a pan in the beginning of task execution.	REFLECT	True	192
			REF-online	True	143
			Ours	True	103
Cook Egg	Cook an egg on a clean pan	The robot cracked the egg inside of the pot instead of the pan.	REFLECT	False	108
			REF-online	False	120
			Ours	True	84
Make Coffee	Serve coffee on a clean mug and place it on the countertop	The mug is dirty in the beginning of task execution.	REFLECT	False	163
			REF-online	True	123
			Ours	False	123
Serve Whole Egg	Serve a whole (uncooked) egg in a bowl	The robot executed the action of cooking the egg.	REFLECT	False	80
			REF-online	False	74
			Ours	True	53
Serve Whole Tomato	Serve an unsliced tomato in a bowl	The robot executed the action of slicing the tomato.	REFLECT	False	120
			REF-online	False	111
			Ours	True	64
Serve Breakfast (simple)	Serve an unsliced apple and sliced lettuce	The robot executed the action of slicing the apple.	REFLECT	False	254
			REF-online	False	250
			Ours	True	257
Serve Breakfast (simple)	Serve an unsliced apple and coffee	The mug is dirty in the beginning of task execution.	REFLECT	False	267
			REF-online	False	253
			Ours	False	241

APPENDIX E
DETAILS ON BASELINE METHODS

To evaluate the effectiveness of our scene graph-based approach for proactive replanning, we compare it against three baseline methods: (i) image similarity using CLIP [23] embeddings, (ii) caption-based comparison using vision-language models, and (iii) object-level similarity scoring based on detected bounding boxes. All methods use the same RGB-D input information.

For the CLIP-based image similarity baseline, we first compute the similarity score between the expected and current images using CLIP embeddings. If the score exceeds a predefined threshold, we pass both images to GPT-4o and prompt it to describe the differences and identify potential causes of failure. Based on the model's reasoning, we then trigger replanning through an additional LLM call.

For the caption-based comparison baseline, we use GPT-4o [1] to generate captions for both the expected and current RGB images. We then calculate the similarity between the two captions using CLIP embeddings. If the similarity exceeds a threshold, we further prompt GPT-4o with both captions to reason about the differences and replan based on the generated explanation.

For the object-level similarity baseline, we use bounding box detections provided by the simulator [17]. We extract detected objects from both the expected and current images and count the number of objects present in each scene. We compute a similarity score based on the object counts, and if it exceeds a threshold, we pass the lists of detected objects into GPT-4o to

reason about discrepancies and trigger replanning accordingly.

APPENDIX F
FULL RESULTS ON FAILURE DETECTION RATE AND SUCCESS RATE

In our main experiment, we vary the similarity threshold values (90%, 85%, and 80%) and report the mean failure detection rates (FDR) and task success rates (SR). Here, we provide the detailed results for each individual threshold setting to offer a comprehensive view of the performance trends.

TABLE A3: **Success rate of detection based method:** Task results (T1–T10) and total scores across different threshold levels.

Threshold	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	**Total**
90%	10	10	50	50	30	30	30	80	80	40	41.0
85%	10	10	40	40	40	40	30	60	70	40	38.0
80%	10	10	40	30	40	40	30	50	50	40	34.0
Average	10	10	43.33	40	36.66	36.66	30	63.33	66.66	40	37.66

TABLE A4: **Failure detection rate of detection based method:** Task results (T1–T10) and total scores across different threshold levels.

Threshold	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	**Total**
90%	50	40	30	30	30	30	20	20	30	20	30.0
85%	50	30	30	30	30	30	20	20	30	20	29.0
80%	50	30	30	30	30	30	20	20	30	10	28.0
Average	50	33.33	30	30	30	30	20	20	30	26.66	29.99

TABLE A5: **Success rate of image similarity based method:** Task results (T1–T10) and total scores across different Threshold levels.

Threshold	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	**Total**
90%	80	40	50	70	50	40	70	70	30	20	52.0
85%	40	30	50	50	30	70	70	70	20	10	44.0
80%	40	50	50	40	30	50	70	70	20	10	43.0
Average	53.33	40	50	53.33	36.66	53.33	70	70	23.33	13.33	46.33

TABLE A6: **Failure detection rate of image similarity based method:** Task results (T1–T10) and total scores across different Threshold levels.

Threshold	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	**Total**
90%	30	50	30	20	30	20	40	20	40	20	30.0
85%	20	50	30	10	20	20	30	20	40	10	25.0
80%	20	50	30	10	20	20	30	20	40	10	25.0
Average	23.33	50	30	13.33	23.33	20	33.33	20	40	13.33	26.65

TABLE A9: **Success rate of our proposed method:** Task results (T1–T10) and total scores across different Threshold levels.

Threshold	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	**Total**
90%	90	70	70	80	70	70	100	70	70	70	76.0
85%	90	70	70	80	70	70	100	70	60	70	75.0
80%	90	60	70	80	60	70	90	60	50	70	70.0
Average	90	66.66	70	80	66.66	70	96.66	66.66	60	70	73.67

TABLE A10: **Failure detection rate of our proposed method:** Task results (T1–T10) and total scores across different Threshold levels.

Threshold	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	**Total**
90%	100	100	100	100	100	100	100	100	100	100	100.0
85%	90	90	100	90	100	90	90	80	100	90	92.0
80%	90	80	100	90	80	90	90	70	90	90	87.0
Average	93.33	90.0	100	93.33	93.33	93.33	93.33	83.33	96.66	93.33	92.97

TABLE A7: **Success rate of image captioning based method:** Task results (T1–T10) and total scores across different threshold levels.

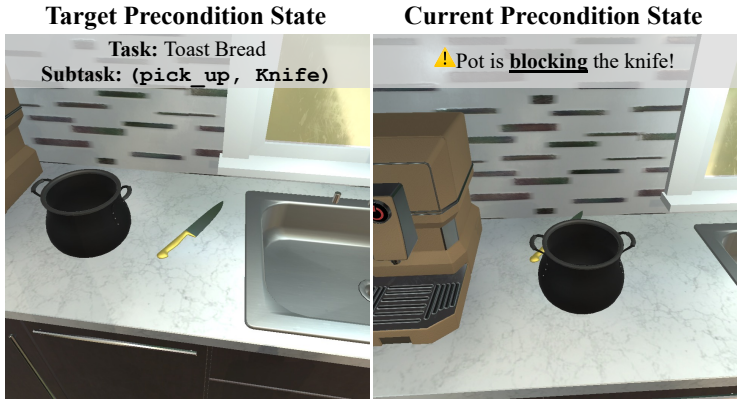
Threshold	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	**Total**
90%	80	40	50	70	50	40	70	70	40	60	57.0
85%	70	40	40	50	30	30	60	70	20	40	45.0
80%	70	40	40	40	30	30	50	70	20	30	42.0
Average	73.33	40	43.33	53.33	33.33	33.33	60	70	26.67	43.33	47.6

TABLE A8: **Failure detection rate of image captioning based method:** Task results (T1–T10) and total scores across different threshold levels.

Threshold	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	**Total**
90%	80	60	70	80	60	60	70	30	70	60	68.0
85%	70	50	60	70	40	40	60	20	50	40	48.0
80%	70	50	60	70	40	30	60	20	50	40	47.0
Average	73.33	53.33	63.33	73.33	46.67	43.33	63.33	23.33	56.67	46.67	54.0

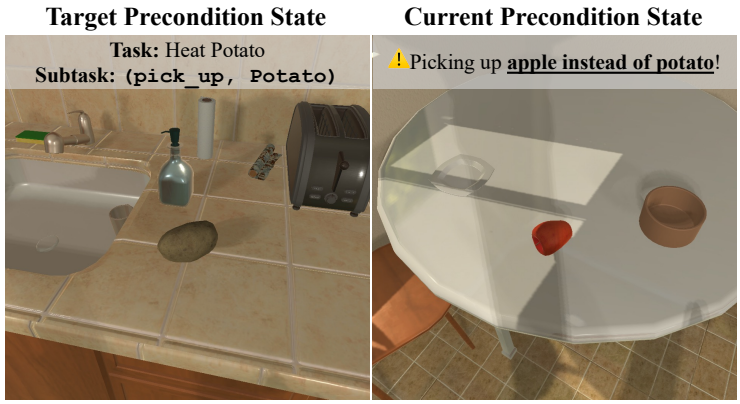
APPENDIX G
ADDITIONAL QUALITATIVE EXAMPLES OF FAILURE REASONING

In this section, we illustrate additional results of failure reasoning on baseline methods and ours. Figure A1 shows reasoning on 'toast bread' task, where failure occurs at subtask '(pick_up, Knife)' because there is a pot blocking the knife. Figure A2 shows reasoning on 'heat potato' task, where failure occurs at subtask '(pick_up, Potato)' because it navigated to an apple and trying to pick up an apple instead of a potato. Our method captures spatial information from scene graph to resolve possible failure at current scene.



Detection Metric: The discrepancy between the expected and current frames is the absence of one 'Cabinet', one 'Sink', and one 'Faucet' in the current frame, which could cause the robot to fail in achieving the task of picking up the knife if it relies on the sink or faucet for positioning or orientation. To address this, the robot's plan could be refined by updating its environment model to account for the missing objects and adjusting its navigation and manipulation strategies to rely on other available landmarks or objects.
LLM Caption Differences: The robot might fail to pick up the knife because the current caption shows a coffee machine on the left, which could obstruct access to the knife. Additionally, the absence of a toaster in the current caption means the robot cannot place the bread slice inside or turn it on. To fix this, the robot should first ensure the knife is accessible by moving any obstructing objects and locate or place a toaster on the counter.
VLM Image Differences: In the current state, the knife is inside a pot, making it inaccessible for the robot to pick up. Additionally, the toaster is present, but there is no visible bread or sliced bread near it, which is necessary for the task of placing bread inside the toaster and turning it on. To fix this, the robot should first remove the knife from the pot and ensure that the bread is sliced and placed near the toaster before attempting to complete the task.
Ours: The current environment has a discrepancy where the pot is blocking the knife, preventing the robot from successfully executing the action to pick up the knife. This obstruction is not accounted for in the expected scene graph, which assumes the knife is accessible. To address this issue, an additional plan should be added to move the pot out of the way before attempting to pick up the knife. This could involve a subtask like `'(move_obj, Pot, CounterTop-2)'` to ensure the knife is accessible for the robot to pick up.

Fig. A1: **Additional examples of failure reasoning**, comparing ours to other baseline methods.



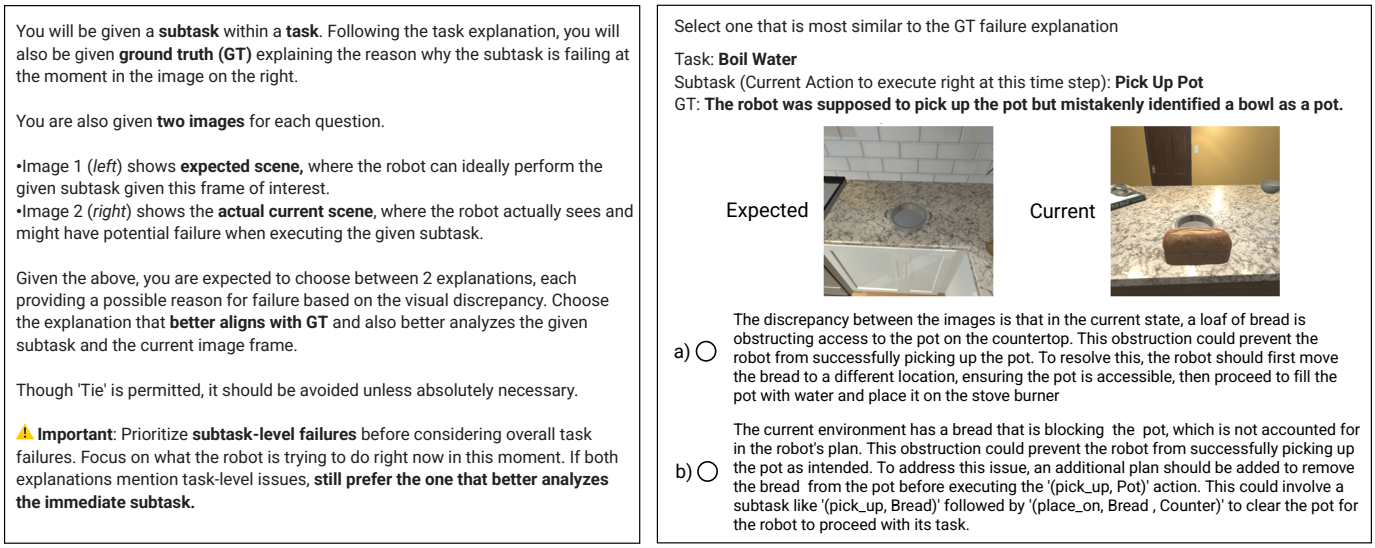
Detection Metric: The robot is likely to fail in achieving the task of picking up the potato and placing it on a plate inside a microwave because the current frame lacks essential objects such as the Potato, Microwave, and Plate, which are present in the expected frame. To address this issue, the robot's plan should be refined to include a verification step that ensures the presence of necessary objects before attempting the task, and if discrepancies are detected, it should trigger a search or navigation routine to locate the missing items or adjust the task to the current environment.
LLM Caption Differences: The robot might fail to achieve the task because the potato is on a table with a plate and bowl, rather than on a countertop near a microwave. To fix this, the robot should first locate the potato, place it on a plate, and then find and use a microwave to complete the task.
VLM Image Differences: The discrepancy between the images is that the potato is on the countertop in the first image, while in the second image, a different object (possibly a red item) is on a table, and there is no visible microwave or plate. This could cause the robot to fail in picking up the potato and placing it on a plate inside a microwave, as the necessary objects are not in the expected locations. To fix this, the robot should first locate the potato, ensure a plate is available, and identify the microwave's location before proceeding with the task.
Ours: The observed discrepancy shows that the robot is detecting an 'apple' in place of the expected 'potato', indicating that the potato may either be missing from the scene or has not been accurately identified by the robot's sensors. This misperception could result in failure to complete the current subtask of picking up the potato. To mitigate this, the robot could incorporate an additional verification step to ensure the potato is present and correctly recognized before attempting to pick it up. This could involve adding a subtask such as '`(scan_for_obj, Potato)`' to confirm the potato's presence and accurate identification within the environment.

Fig. A2: **Additional examples of failure reasoning**, comparing ours to other baseline methods.

APPENDIX H
DETAILS ON HUMAN EVALUATION

As described in our main paper, we conducted a human evaluation of failure reasoning using 15 human annotators. Specifically, we collected 13 scenarios in which both our method and the baseline successfully detected failures. For each scenario, we randomly assigned five annotators to perform the evaluation.

Each annotator was instructed to compare the reasoning results from our method and the baseline, given the ground-truth failure reason and visual observations. As shown in Figure A3, the annotators were provided with labeling instructions along with the following information: (i) the expected scene (i.e., what the robot should observe before executing the subtask), (ii) the actual scene (i.e., what the robot observes in the perturbed environment), and (iii) the ground-truth reason for failure.



(a) (b)

Fig. A3: **Human survey labeling interface**: (a) instructions and (b) an example questionnaire

APPENDIX I
ABLATION STUDIES

We conduct ablation experiments to quantify the contribution of each component in the scene graph comparison process and to assess its impact on failure detection performance. As shown in Table A11 (top), removing the subtask node during scene graph construction leads to a moderate drop in failure detection performance, highlighting the importance of understanding the subtask context. Omitting node matching and edge matching results in even larger decreases, indicating that both object identity alignment and relational structure are critical for accurately identifying potential failures. Removing structural matching also degrades performance, although to a lesser extent. Overall, the full model achieves the highest failure detection rate of 92.97%.

Furthermore, as shown in Table A11 (bottom), task success rates drop from 73.67% to 37.67% when the reasoning module is removed, highlighting its critical role in effective replanning. Without reasoning, the system struggles to interpret the causes of failure and devise corrective actions that are properly aligned with the environment's constraints and the task's goals. Instead, it resorts to less informed or naive replanning strategies that often fail to resolve underlying issues, resulting in unsuccessful task completions.

TABLE A11: Failure detection and task completion results.

Failure Detection		**Task Completion**	
Method	**FDR (%) ↑**	**Method**	**SR (%) ↑**
Ours	**92.97**	Ours	**73.67**
w/o Subtask Node	84.33 (8.64% ↓)	w/o Reasoning	37.67 (36.0% ↓)
w/o Structural Matching	82.67 (10.3% ↓)		
w/o Node Matching	74.67 (18.3% ↓)		
w/o Edge Matching	70.33 (22.6% ↓)		

APPENDIX J
LLM PROMPTS USED IN OUR EXPERIMENTS

In this section, we provide the LLM prompts used in our experiments. Table A12, A13, A14 and A15 show the prompts used during the reasoning stage for the graph-based discrepancy algorithm, image similarity-based method, captioning-based method, and object detection-based method, respectively. Table A16 presents the prompt used for replanning after the reasoning stage.

TABLE A12: The prompt used to perform reasoning in our method.

Method: Graph Based Discrepancy Algorithm (Ours)

System Prompt: You are given the current robot state, the goal condition, the current scene graph, the expected scene graph, and the robot plan. Briefly explain what is wrong with the current environment that could cause the robot to fail in 1-2 sentence. Also in 1-2 sentences, describe any additional plans to be added to the current plan that could help the robot successfully execute the task

User Prompt:
The robot task is to [TASK_NAME]. The sequence of subtasks to achieve this goal is [SUBTASKS]
The task is considered successful if [SUCCESS_CONDITION]
Here's the robot observation before executing the current subtask [SUBTASK]: [CURRENT_OBSERVATION]. Here are the expected observations [EXP_OBSERVATIONS]
Q: The node represents the objects in the scene and their states whereas the edges represent the relationship between the objects. Carefully consider the current scene graph and the expected scene graph and determine what could be wrong with the environment that could cause the robot to fail in executing the current subtask or the entire plan.
A:

TABLE A13: The prompt used to perform reasoning in image similarity based method.

Method: Image Similarity

System Prompt: You are given the current robot state, the goal condition, the difference between expected and current environmental states, and the robot plan. Briefly explain what is wrong with the current environment that could cause the robot to fail in 1-2 sentence. Also in 1-2 sentences, describe any additional plans to be added to the current plan that could help the robot successfully execute the task

User Prompt:
The robot task is to [TASK_NAME]. The sequence of subtasks to achieve this goal is [SUBTASKS]
The task is considered successful if [SUCCESS_CONDITION]
Here's the robot difference in observation between expected and current states before executing the current subtask [SUBTASK]:[IMAGE_DIFFERENCE]. Carefully consider the difference and determine what could be wrong with the environment that could cause the robot to fail in executing the current subtask or the entire plan.
A:

TABLE A14: The prompt used to perform reasoning in image similarity based method.

Method: Image Similarity

System Prompt: You are given the current robot state, the goal condition, the difference between expected and current environmental states, and the robot plan. Briefly explain what is wrong with the current environment that could cause the robot to fail in 1-2 sentence. Also in 1-2 sentences, describe any additional plans to be added to the current plan that could help the robot successfully execute the task

User Prompt:
The robot task is to [TASK_NAME]. The sequence of subtasks to achieve this goal is [SUBTASKS]
The task is considered successful if [SUCCESS_CONDITION]
Here's the robot difference in observation between expected and current states before executing the current subtask [SUBTASK]:[IMAGE_DIFFERENCE]. Carefully consider the difference and determine what could be wrong with the environment that could cause the robot to fail in executing the current subtask or the entire plan.
A:

TABLE A15: The prompt used to perform reasoning in detection based method.

Method: Object Detection

System Prompt: You are given the current robot state, the goal condition, the current detected objects in the scene, the expected objects detected in the scene ,and the robot plan. Briefly explain what is wrong with the current environment that could cause the robot to fail in 1-2 sentence. Also in 1-2 sentences, describe any additional plans to be added to the current plan that could help the robot successfully execute the task

User Prompt:
The robot task is to [TASK_NAME]. The sequence of subtasks to achieve this goal is [SUBTASKS]
The task is considered successful if [SUCCESS_CONDITION]
Here's the robot observation before executing the current subtask [SUBTASK]: [CURRENT_DETECTION]. Here are the expected observations [EXPECTED_DETECTION]
Q: Carefully consider the differences in detected items and determine what could be wrong with the environment that could cause the robot to fail in executing the current subtask or the entire plan.
A:

TABLE A16: The prompt used to perform replanning in our experiment.

System Prompt:
[Available_actions]. Provide a recovery plan using ONLY the available actions to achieve the SUCCESS STATE after a failure. Follow these rules STRICTLY:

1. Available Actions: pick_up, put_in, put_on, open, close, toggle_on, toggle_off, slice, crack, pour

2. Constraints:
- Robot has ONE gripper (can hold only ONE object at a time).
- NEVER skip required steps (e.g., 'pour' for cleaning, 'toggle_on' for cooking).

3. Replan Logic:
- Start from the CURRENT PLAN state (DO NOT restart from initial plan unless failure occurred at step 1).
- The SUCCESS STATE [SUCCESS_CONDITION] MUST be achieved—verify ALL steps against it.

4. Output Format:
- ONLY return a list of actions (NO descriptions, Python, symbols, or extra text).
- Follow the EXACT format of the initial plan (e.g., '(action, object, target)').

![IMPORTANT] PRIORITIZE THE SUCCESS STATE: Every action must DIRECTLY contribute to achieving [SUCCESS_CONDITION].

User Prompt:

Task: [TASK_NAME]

Initial plan: [PLAN]

Failure reason: [FAILURE_REASON]

SUCCESS Condition: [SUCCESS_CONDITION]

Current plan the robot is about to execute: [CURRENT_PLAN]

Current observation:
- Objects: [OBJECTS_IN_SCENE]

Constraints:
- Single-arm robot: Gripper must be EMPTY to pick up objects.

Correction plan (ONLY ACTIONS, NO COMMENTS):

APPENDIX K
LIMITATIONS

Though we demonstrate the effectiveness of preemptively identifying failures through scene graph discrepancies, several limitations remain. Our evaluation was limited to failure cases provided by the RoboFail dataset, which may not capture the full range of possible failure modes. It is therefore worth applying our proposed method to more diverse scenarios to assess its generalizability. Moreover, while our current experiments are conducted in simulation, evaluating the method on real robotic platforms is a critical next step. Although real-world deployment poses practical challenges, we are excited about future implementations on physical robots to validate the system's effectiveness beyond simulated environments.