Towards Failure-Resilient Lifelong Learning Agents through Scene Graph-Guided Proactive Replanning

Che Rin Yu
Korea University
Seoul, South Korea
eyucherin@korea.ac.kr

Daewon Chae
University of Michigan - Ann Arbor
Ann Arbor, USA
daewon@umich.edu

Dabin Seo
Korea University
Seoul, South Korea
lemonstar99@korea.ac.kr

Sangwon Lee
Korea Telecom Research
Seoul, South Korea
lee.sangwon@kt.com

Hyeongwoo IM
Korea Telecom Research
Seoul, South Korea
im.hyeongwoo@kt.com

Jinkyu Kim
Korea University
Seoul, South Korea
jinkyukim@korea.ac.kr

ABSTRACT

When humans perform everyday tasks, we naturally adjust our actions based on the current state of the environment. For example, if we plan to heat a bowl of soup in the microwave and see that there's already a plate inside, we first remove it before proceeding. However, many autonomous robots lack this adaptive awareness. They often follow pre-planned actions that may overlook subtle yet critical changes in the scene, which can result in actions being executed under outdated assumptions and eventual failure. While replanning is critical for robust autonomy, most existing methods respond only after failures occur, when recovery may be inefficient or infeasible. In this work, we present a proactive replanning framework that anticipates and prevents failures before action execution.The key idea is to learn visual preconditions from successful demonstrations. Before each subtask, the system builds a scene graph from RGB-D observations and compares it with an expanding buffer of reference graphs from successful trials. When the similarity falls below a threshold, a lightweight reasoning module diagnoses the discrepancy and generates a corrective sub-plan. As the experience buffer grows, the system scales and generalizes more efficiently, highlighting a progressive path toward more autonomous and effective replanning. Experiments in both the AI2-THOR simulator and real-world platforms show that our approach improves task success and execution performance compared to baselines.

KEYWORDS

Failure Detection and Reasoning; Replanning; Large Language Models; Symbolic Representations

ACM Reference Format:
Che Rin Yu, Daewon Chae, Dabin Seo, Sangwon Lee, Hyeongwoo IM, and Jinkyu Kim. 2026. Towards Failure-Resilient Lifelong Learning Agents through Scene Graph-Guided Proactive Replanning. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/V1X2Y3Z4

This work is licensed under a Creative Commons Attribution International 4.0 License.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/V1X2Y3Z4

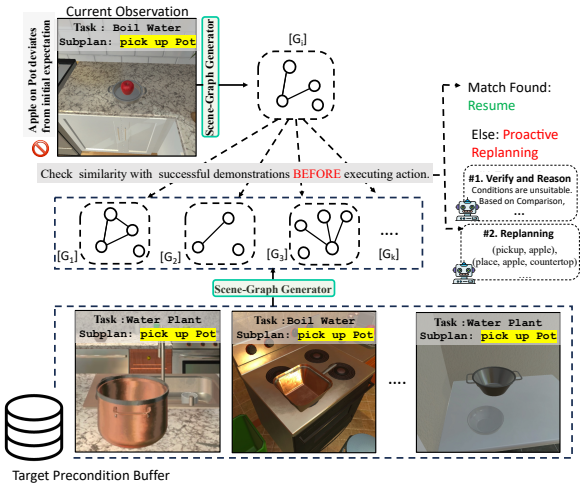


Figure 1: An overview of the proposed framework. Prior to executing each subtask (e.g., "pick up pot"), the model encodes the current visual observation into a scene graph representation and compares it with previously successful demonstrations stored in a target precondition buffer. If a sufficiently similar match is found, the subtask is executed as planned. Otherwise (e.g., when an apple is placed on the pot), the agent performs failure reasoning to infer potential causes and proactively initiates a replanning process.

1 INTRODUCTION

Despite recent advances in embodied intelligence [3, 10, 36], robots still struggle to adapt their plans once deployed in dynamic environments. A plan that was optimal at design time can quickly become invalid as the environmental conditions change (e.g., objects are moved, containers filled, or surfaces obstructed). When such discrepancies remain unnoticed, robots may operate on outdated assumptions, leading to cascading errors or task failure. These challenges highlight the importance of *replanning*, the ability to revise or regenerate new plans based on updated observations, as a fundamental prerequisite for achieving reliable autonomy in unpredictable settings.

However, developing such replanning capabilities for autonomous systems remains a fundamental challenge, as it requires the following four key components: (i) efficiently determining the optimal timing for replanning interventions, (ii) accurately diagnosing the

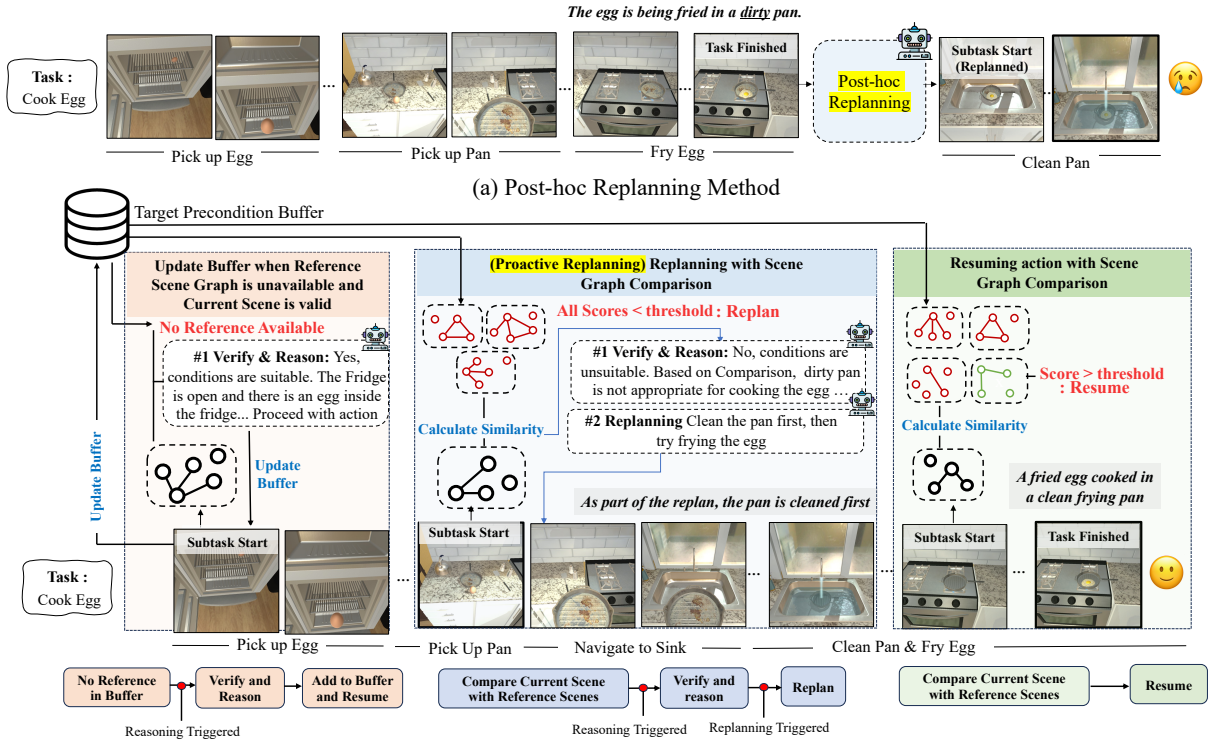


Figure 2: (a) Conventional post-hoc replanning methods [12, 21, 28, 32] react only after a failure (e.g., noticing an egg is fried in a dirty pan). (b) Our proposed proactive replanning method. The agent compares current scene graphs with reference scenes from a *Target Precondition Buffer*. When no reference is available or when similarity scores fall below a threshold, the system invokes an LLM to *verify and reason* about scene validity. If conditions are unsuitable, it replans (e.g., cleaning the pan before frying the egg); otherwise, it proceeds and updates the buffer with validated scenes. This proactive mechanism enables early failure detection, buffer growth, and more reliable task success.

root causes of failures or potential failure conditions, (iii) leveraging past experiences to guide future decision making, and (iv) generating effective and corrective action sequences to recover progress toward the task objectives. Effectively addressing these components is essential for enabling autonomous robots to operate reliably across diverse and dynamically changing environments.

Although recent studies have made notable progress in addressing the challenges of robotic failure recovery, many existing approaches mainly rely on post-hoc mechanisms (i.e., responding only after failures emerge, see Figure 2 (a)) [12, 21, 28, 32], expensive human supervision [5, 20, 29, 37], or predefined rule-based triggers [17, 25, 34], which fail to simultaneously capture the complexity of real-world visual contexts and self-improving behavior of agents.

A key concept underlying proactive replanning is the notion of preconditions, i.e., constraints that must be satisfied before an action can be successfully executed (e.g., a pan must be clean before use, or a container must be empty before placing an object inside). Such preconditions are typically defined manually through hand-engineered rules, which tend to break down under visually diverse or dynamic conditions. To address these limitations, we draw inspiration from human perceptual reasoning. Rather than verifying explicit conditions, humans rely on accumulated perceptual experience, rapidly inferring from visual cues whether an action is feasible, such as recognizing at a glance whether a drawer is open, blocked, or otherwise inaccessible.

While vision language models (VLMs) may seem intuitive for assessing task preconditions, they often overlook spatial inconsistencies [16, 30, 33] and are prone to hallucinations [38, 39]. In contrast, scene graphs provide a structured representation of the environment, enabling the agent to reason about relational and geometric context, such as whether an object is partially visible, obstructed, or held by the agent, which directly affects task feasibility and motivates the development of structured scene representations that capture both relational and geometric context for failure detection and informed replanning.

As shown in Figure 1, we propose a novel proactive replanning method that can detect potential failures prior to action execution and dynamically revise its plans by efficiently checking differences between the current scene and previous successful demonstrations of long-horizon tasks (e.g., boiling water, cooking an egg). This approach contrasts with conventional post-hoc replanning strategies that respond only after failure occur (compare Figure 2 (a) vs. (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 against expected scene graphs derived from prior successful demonstrations, which are stored in an incrementally expanding

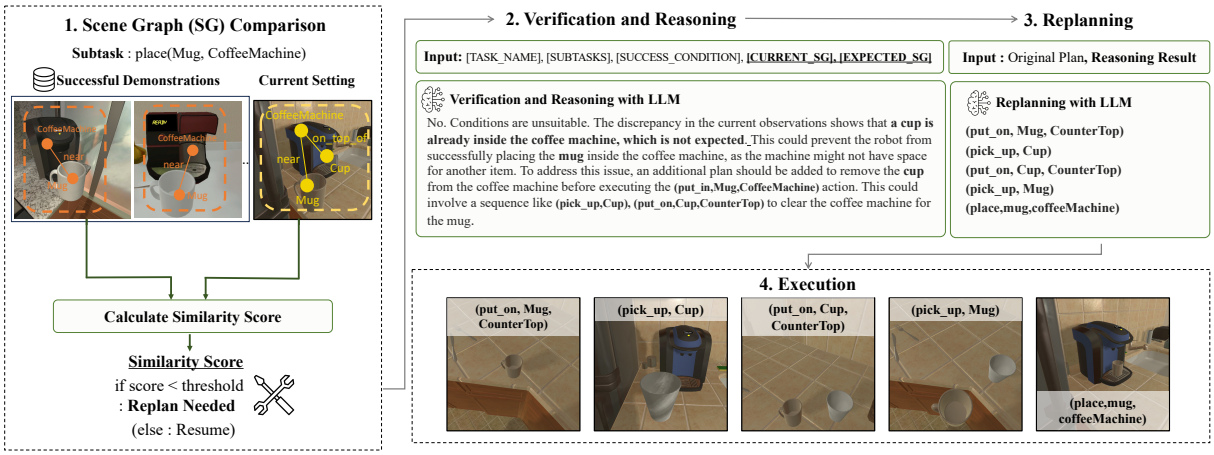


Figure 3: Our approach proceeds in four stages: (1) assess subtask feasibility by comparing the current and reference scene graphs; (2) if similarity is below a threshold, prompt an LLM to verify the scene and identify potential failure causes; (3) derive corrective actions from the reasoning and original plan; and (4) execute the refined plan.

experience buffer. If the similarity between the current and reference scene graphs falls below a threshold, the system proactively verifies the discrepancy and triggers replanning before executing the subtask. This process involves generating a reasoning chain to identify the probable cause of failure (e.g., "a dirty pan is unsuitable for cooking") and formulating corresponding corrective actions (e.g., "clean the pan before frying the egg"), thereby avoiding failure and enabling successful task completion. We validate our approach in the AI2-THOR simulator [18], achieving up to 25–30% higher task success rates than vision–language model based approaches, while completing tasks 20–30% faster on average, demonstrating both improved execution efficiency and the agent's ability to preemptively avoid and recover from potential failures during task execution. We summarize our contributions as follows:

- We present a novel proactive replanning method that leverages scene graphs to structurally evaluate whether subtask preconditions are satisfied, allowing efficient proactive replanning through early detection and correction of potential failures.
- We utilize an incrementally expanding precondition buffer that accumulates validated representations over time, enabling the agent to learn from prior experience, reduce dependence on computationally expensive LLM/VLM-based reasoning, and progressively enhance its lifelong robustness.
- Our experiments conducted on both custom and off-the-shelf datasets, including simulation and real-world platforms, demonstrate that our approach achieves higher task success rates, stronger failure detection accuracy, and more interpretable reasoning compared to existing baselines, as validated through human evaluations.

2 RELATED WORK

Replanning Strategies for Robotics. There has been a growing interest in replanning strategies in the robotics community where various approaches have been introduced based on predefined rule-based triggers [4, 17, 25, 34], human-in-the-loop [5, 7, 29, 37], post-hoc replanning [12, 21, 28, 32], and large vision-language model-based methods [6, 8]. Human-in-the-loop [5, 29, 37] 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 [12, 21, 28, 32], 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 [6, 8] 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.

Large Language Models (LLMs) for Robotics. Multi-modal large language models (LLMs) have been widely used in robotic task planning [2, 13, 13, 14, 26], translating high-level instructions into action sequences or executable code [19, 31]. Similarly, Vision-Language Models (VLMs) interpret visual scenes [9, 22, 23, 35] and detect task-relevant objects [11, 27] 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 [16, 30, 33]. Additionally, LLMs and VLMs may hallucinate [7, 38, 39] and cause errors while assessing environment states, where minor spatial inconsistencies such as occluded objects, misplaced items, or obstructed targets, may silently interfere with task execution. Solely relying on semantics may not detect such failures, highlighting the need for explicit spatial reasoning to detect discrepancies in real time.

3 METHOD

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"). Each subtask

Algorithm 1 Proactive Subtask Validation with Buffer Update

Require: Task $\mathcal{T} = [a_1, \ldots, a_n]$, goal C_{goal}, buffer $\mathcal{B}$, execution threshold τ_{exec}

for each subtask a_i in $\mathcal{T}$ **do**
  Observe environment and build scene graph G_i
  Retrieve reference graphs $\mathcal{R}$ for a_i from buffer
  $s^\star \leftarrow \max_{\hat{G} \in \mathcal{R}} \text{Sim}(G_i, \hat{G})$ ▷ max similarity to references
  if $\mathcal{R} \neq \emptyset$ **and** $s^\star \geq \tau_{\text{exec}}$ **then** ▷ a sufficiently similar reference exists
    Execute a_i
  else ▷ no reference exists or similarity below τ_{exec}
    Use LLM to verify and reason about G_i
    if scene is valid **then**
      add G_i to buffer and execute a_i
    else
      Replan using LLM's explanation and execute new plan
    end if
  end if
end for

is intended to transition the environment closer to satisfying C_{goal}, such that the full execution of $\mathcal{T}$ leads to potential task completion.

At execution time, the robot receives an RGB-D observation I_i of the current environment before executing each subtask a_i. From this observation, the robot constructs a structured semantic representation in the form of a scene graph G_i, which captures the entities present as well as their spatial and relational configurations derived from visual cues. To evaluate the feasibility of executing a_i, the robot compares G_i to reference graphs $\{\hat{G}_i^1, \hat{G}_i^2, \ldots, \hat{G}_i^k\}$, each representing valid configurations observed at the same subtask step across k valid demonstrations.

Buffer Construction from Reference Demonstrations. In order to evaluate whether the environment satisfies the conditions for subtask execution, we infer expected scene graphs from target states derived from the precondtion buffer. If the buffer is empty or if no close match exists, the system calls a language model to verify and reason whether the scene is valid and understand its discrepancies. If confirmed valid, the new configuration is added to the buffer, expanding coverage for future comparisons. These references do not necessarily come from the same overall task, but rather capture valid configurations of the same subtask across different contexts. For example, "pick up mug" may appear in tasks with different goals yet share similar object arrangements.

This strategy enables the robot to adapt to a variety of valid but visually distinct task configurations observed across demonstrations. Through this adaptive process, the robot not only anticipates failures under unseen conditions but also continually enriches its buffer with diverse, validated scenes. As the buffer grows, coverage broadens, reliance on external reasoning decreases, and robustness across heterogeneous environments improves.

3D Scene Graph Construction. We construct a visually grounded, task-informed 3D scene graph from the robot's RGB-D observation, capturing objects, their states, and spatial relationships relevant to the current subtask. Inspired by REFLECT [21] and RoboEXP [15], our method summarizes multimodal inputs into symbolic structures for reasoning about object-action relations.

Instead of generating graphs at every frame, we build one at the start of each subtask to assess whether the environment is suitable for the action. Given an RGB-D image, an object detector identifies bounding boxes and segmentations, and cropped object regions are matched to possible states (e.g., "open," "on," "empty") using CLIP [24] similarity. The depth image projects the segmented view into a 3D semantic point cloud for geometric reasoning, from which pairwise object relations (e.g., on top of, inside, near) are derived. The gripper state determines robot-object interactions (e.g., "held by robot"). The resulting graph $G_i = (V_i, E_i)$ encodes objects, inferred states, and spatial relations.

A subtask node representing a_i is added as contextual metadata, enabling reasoning about whether the current scene satisfies expected preconditions and ensuring that retrieved references are compared within the correct action context, making buffer retrieval and similarity checks both more precise and reliable.

Context-Aware Reference Retrieval. Given the full set of successful demonstrations stored in the buffer, the system first filters reference candidates by comparing subtask nodes. Demonstrations containing the same subtask node as the current subtask a_i are selected as valid references for comparison. This step ensures that only semantically relevant reference subtasks are selected. The corresponding trajectories from which these subtasks were drawn are then used as reference examples for failure detection. To evaluate the feasibility of executing a_i, the robot compares G_i against each of the selected reference graphs $\hat{G}_i^1, \hat{G}_i^2, \ldots, \hat{G}_i^k$ using a structural similarity metric.

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 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 relationships.

Node similarity S_{node} is computed as the average cosine similarity between matched object nodes, using CLIP embeddings and 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:

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

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:

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

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

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

Table 1: Comparison of Success Rate (SR in %) and Task Execution Time (TET in sec).

Method	Type	Buffer	RoboFail Preemptive		RoboFail Post-action		RoboFail		Custom Dataset	
			SR ↑	TET ↓	SR ↑	TET ↓	SR ↑	TET ↓	SR ↑	TET ↓
Baseline (w/o Replanning)	No Replanning	✗	0%	82.75	0%	86.46	0%	83.94	0%	108.53
REFLECT [21]	Offline Replanning *after* entire task execution	✗	66.17%	146.33	56.25%	134.23	63.99%	142.45	58.67%	195.34
REFLECT-online	Online Replanning *after* failure detected	✗	69.11%	130.67	65.14%	**124.46**	67.83%	130.03	66.67%	168.82
VLM (GPT-4o)	Proactive Replanning	✗	48.53%	151.55	43.00%	183.41	46.76%	161.74	50.00%	210.32
+ Image Similarity		✓	1.47%	188.23	3.13%	167.31	2.00%	181.53	3.33%	247.45
+ Image Captioning		✓	54.41%	135.25	46.87%	146.23	51.99%	138.76	52.00%	196.41
+ Object Detection		✓	45.59%	154.55	43.00%	163.67	43.76%	157.47	44.66%	212.43
VLM (Gemini 2.0 Flash)	Proactive Replanning	✗	44.12%	147.55	46.87%	188.41	44.99%	160.63	46.00%	204.56
+ Image Similarity		✓	4.41%	182.45	3.13%	175.32	4.00%	180.17	2.00%	219.34
+ Image Captioning		✓	52.94%	134.21	34.34%	155.23	46.99%	140.94	47.33%	217.32
+ Object Detection		✓	44.12%	151.53	46.88%	182.87	45.00%	161.56	43.33%	213.15
Ours	**Proactive Replanning**	✓	**73.53%**	**104.82**	**71.87%**	128.66	**71.99%**	**111.10**	**78.67%**	**142.85**

When the similarity scores $S_1, S_2, \ldots, S_n$ between the current scene graph G_i and expected scene graphs $\{\hat{G}_i^1, \hat{G}_i^2, \ldots, \hat{G}_i^n\}$ fall below a predefined threshold (i.e., $S_j < 0.9$), the system infers that the current subtask is unlikely to succeed and proactively triggers replanning.

Reasoning Based Replanning. Following the detection of a potential failure, our framework decomposes the replanning process into two modular components: a verification-and-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 3.

The verification-and-reasoning module is triggered if no similar reference exists or the buffer is empty. Natural language descriptions of the observed and expected scene graphs, along with the task goal and current subtask context, are passed to a language model (e.g., GPT-4o). Acting as a verifier, the model determines whether the current scene can still be considered valid for executing a_i. If valid, it also provides reasoning to justify this decision, and the scene is stored in the buffer as a new reference. If invalid, the model explains the cause of the mismatch and specifies what changes are needed to proceed. These outputs are 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.

4 EXPERIMENTS

We evaluate the effectiveness of our graph-based proactive replanning framework through extensive experiments, aiming to investigate the following key questions.

(1) How effective is our approach compared to reactive and non-reactive baselines in improving task success rates and reducing total execution time? (Table 1)
(2) Does incrementally scaling the buffer over time affect the system's dependence on expensive LLM queries? (Figure 6)
(3) How well does our method **detect** potential precondition violations and **reason** about possible failures? (Figure 7)
(4) How does the framework perform in real-world scenarios across different robotic platforms? (Table 2, Figure 8)
(5) What is the contribution of subtask conditioning and graph structure according to the ablation analysis? (Table 3)

Effects of Scene Graph Based Proactive Replanning. In this section, we examine the effects of our graph-based proactive replanning approach by evaluating it on four distinct datasets. We first turn to the RoboFail dataset [21], a benchmark of 100 failure scenarios spanning diverse long-horizon tasks. Since RoboFail includes audio data, we exclude this modality and rely solely on visual information, ensuring that all methods reason under the same perceptual conditions. Within RoboFail, we consider two complementary subsets: (i) **RoboFail Preemptive**, which accounts for 68% of the dataset and focuses on failures that can be detected before execution (e.g., attempting to turn on a microwave before closing its door), and (ii) **RoboFail Post-Action**, which covers the remaining 32% and captures failures recognized only after execution (e.g., dropping or misplacing an object). The overall RoboFail results reported in Table 1represent the weighted combination of these two subsets. Additionally, we introduce a custom **Proactive Failure Dataset** consisting of 150 manually curated failure cases across 30 distinct kitchen environments designed to evaluate early-stage failure detection and proactive reasoning.

As baselines, we first consider a no-replanning scenario, where the robot follows the initial task plan without adjustment. We then compare against REFLECT, which uses a scene graph–based hierarchical summary of the robot's actions and replans only after the entire task is completed. In addition, we implement a real-time variant, REFLECT Online, which performs verification after each subtask rather than waiting until the end. Our method extends this further by proactively validating the environment before each subtask begins, enabling it to anticipate and resolve issues in advance, as illustrated in Figure. 4.

To comprehensively assess their capabilities under proactive settings, we evaluate both direct VLM-based detection and three complementary alternatives that also perform pre-execution checks (Figure 5): (i) *image-level comparison* via CLIP embedding similarity between current and reference RGB images, (ii) *caption-based comparison* using VLM-generated scene descriptions, and (iii) *object-level matching* based on detected object counts. In these setups, the VLM serves as the reasoning module, while auxiliary methods

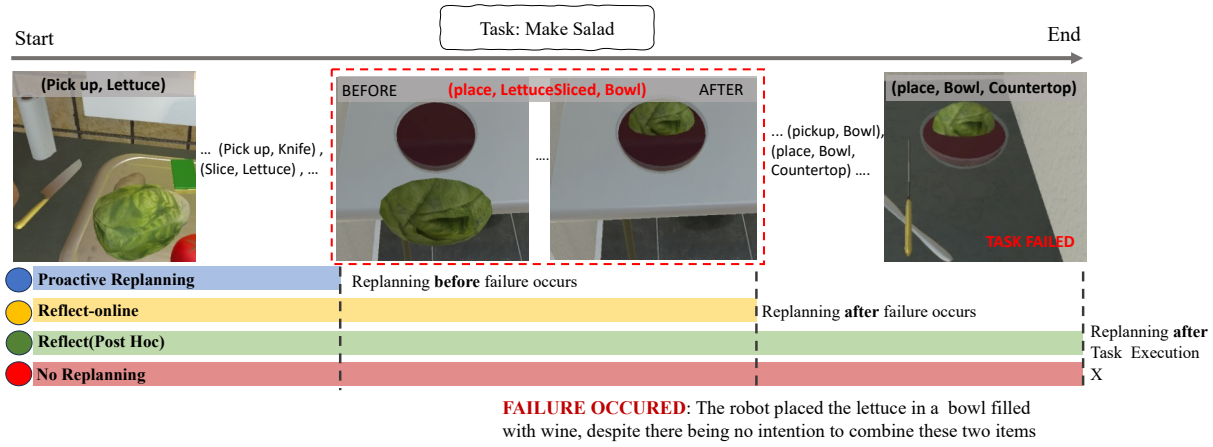


Figure 4: Task execution timelines in a failure-prone "Make Salad" task. The robot must place sliced lettuce into a bowl already filled with wine. Our proactive replanning anticipates and corrects the failure before execution, while REFLECT-online reacts after it occurs, REFLECT (Post-Hoc) replans post-task, and the no-replanning baseline takes no action.

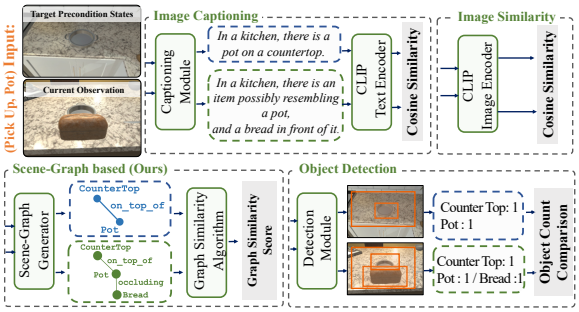


Figure 5: Illustration of four approaches for subtask failure detection: (1) Image Captioning: comparing CLIP text embeddings of generated captions;(2) Image Similarity: CLIP-based similarity between current and expected RGB images; (3) Object Detection : comparing object counts between scenes; (4) Scene Graph (Ours): graph-based discrepancy analysis using structured scene representations.

provide structured validation signals through a growing buffer of verified reference scenes.

We employ GPT-4o as the reasoning model in our framework, except for the Gemini 2.0 Flash baseline, where reasoning is performed by the VLM. For methods incorporating a precondition buffer, we set the similarity threshold to 85%, initialize the buffer as empty, and progressively update it with validated reference scenes during execution.

As shown in Table 1, our graph-based proactive approach consistently outperforms most baselines in both success rate and execution efficiency. Early validation of subtask preconditions allows the agent to bypass unnecessary recovery steps and execute tasks more efficiently. Although our method is designed to detect failures preemptively, high performance on the Post-Action subset shows that it also effectively identifies failures after they occur, often catching them at the beginning of the next subtask. This demonstrates the framework's adaptability and robustness across varied task complexities.

Leveraging Buffer Growth to Minimize Reasoning Costs. We evaluate how buffer accumulation enhances scalability and reduces

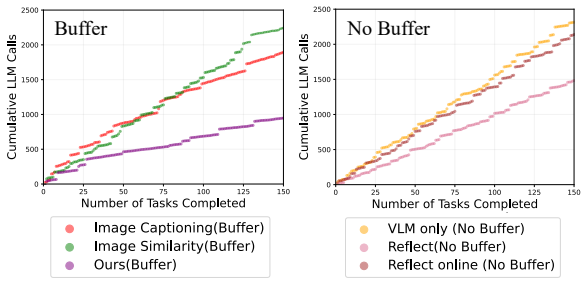


Figure 6: Comparison of LLM call reduction between buffer-based and no buffer methods .The methods start similarly for the first few tasks. Methods with no buffer flatten, while buffer-based methods diverge(even at smaller rates) earlier with fewer calls thereafter.

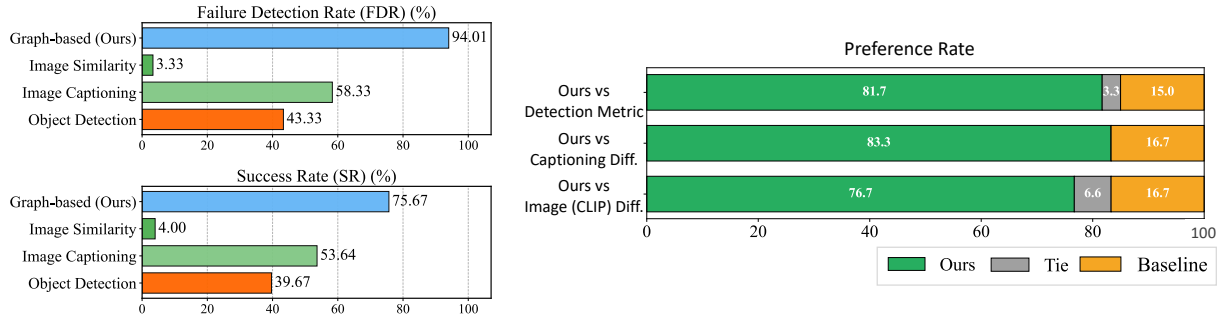
reliance on large language model (LLM) reasoning using our Proactive Failure Dataset, which includes 150 failure scenarios across diverse kitchen setups. As shown in Figure 6, our framework progressively lowers LLM invocations over time, as validated subtask preconditions stored in the buffer no longer trigger redundant reasoning for valid preconditions.

Compared to methods that repeatedly invoke VLMs for both failure detection and reasoning without a buffer, our approach requires fewer reasoning calls as experience grows. While other buffer-based baselines also exhibit gradual improvement, their reduction rate is slower and overall reasoning costs remain higher, primarily because less accurate detection methods trigger unnecessary verification even under satisfied preconditions. By contrast, our graph-based discrepancy analysis enables higher detection, minimizing costly LLM calls and demonstrating strong scalability as the buffer expands.

Analysis of Potential Failure Detection. We empirically compare our scene graph-based method against three alternatives: image-level, caption-based, and object-level comparison to validate its effectiveness. 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 reference trajectories. If no expected configuration yields a similarity

Table 2: Comparison of failure detection and reasoning performance between graph-based and VLM-based methods on the real-world RoboFail dataset.

Candidate	FDR ↑ w/o Obj. Ctx.	FDR ↑ w/ Obj. Ctx.	LCS ↑	ROUGE-L↑	Cosine Similarity ↑	LLM Fuzzy Match Score ↑
GPT-4o (Image Input)	0.57	0.80	11.25	0.314	0.581	0.892
Gemini 2.0 Flash (Image Input)	0.67	0.73	15.00	0.275	0.505	0.752
GPT-4o (Scene Graph Input)	0.77	**0.83**	**20.25**	**0.428**	**0.752**	**0.962**
Gemini 2.0 Flash (Scene Graph Input)	**0.83**	0.80	**20.25**	0.383	0.629	0.955



(a) Comparison of Performance

(b) Human Evaluation on Reasoning, compared to baseline methods

Figure 7: (a) Comparison of failure detection and task success rates across different approaches. (b) Human evaluation study comparing our scene graph-based failure detection explanations against three baselines: object detection, image captioning, and CLIP image similarity.

Table 3: Results of ablation studies.

Failure Detection		Task Completion	
Method	**FDR (%) ↑**	**Method**	**SR (%) ↑**
Ours	**94.01**	**Ours**	**75.67**
w/o Subtask Node	84.33 (9.68% ↓)	w/o Reasoning	37.67 (38.00% ↓)
w/o Structural Matching	82.67 (11.34% ↓)		
w/o Node Matching	74.67 (19.34% ↓)		
w/o Edge Matching	70.33 (23.68% ↓)		

score above the threshold, the method flags the scene as a potential failure case.

We evaluate all methods on the full set of 100 failure scenarios provided by the RoboFail dataset [21]. In this experiment, we vary the similarity threshold values (90%, 85%, and 80%) and report the average failure detection rates (FDR) and task success rates (SR).

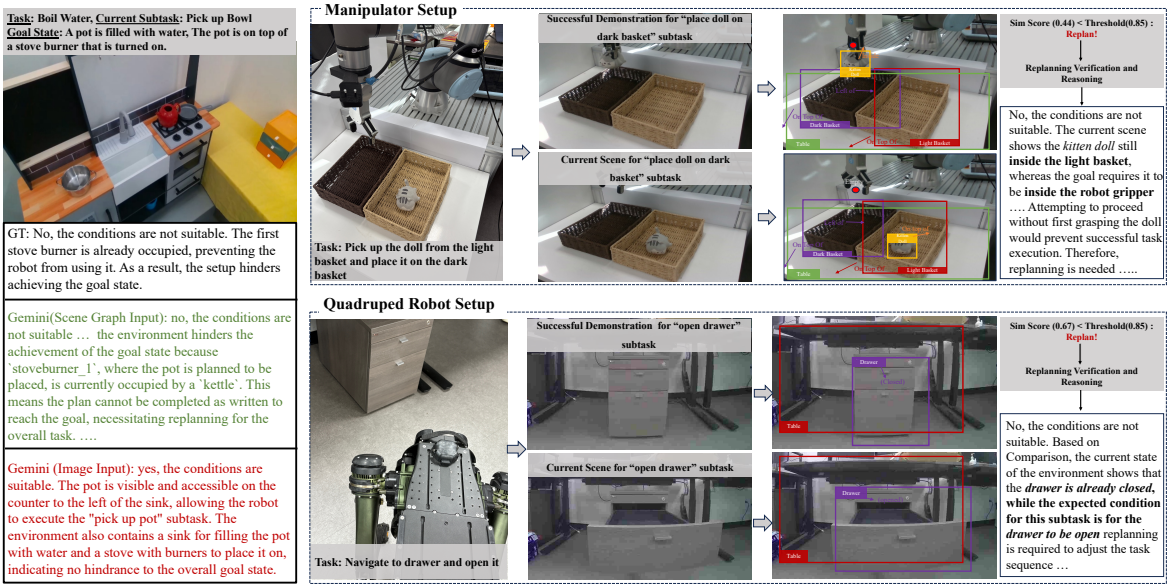
The evaluation results in Figure 7(a) demonstrate that our scene graph-based approach consistently achieves higher failure detection rates (FDR) and task success rates (SR) compared to all baseline methods. In contrast, approaches relying solely on visual or semantic cues, such as CLIP [24] embeddings or captioning, demonstrate substantially lower performance, as they emphasize semantic abstraction without capturing relational structure. 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 leveraging structured visual input to enable explicit symbolic and spatial reasoning for reliable proactive replanning.

Analysis of Anticipatory Failure Reasoning. Building on the same experimental setup from the previous experiment, we further evaluate how effectively each method explains the causes of detected failures. We focus on episodes where all methods correctly identified a failure and compare their ability to generate accurate explanations. For each method, a large language model (GPT-4o) [1] produces a natural language explanation: our approach prompts the model with structured scene graph differences, while baselines use raw image embeddings, generated captions, or detected object-level features.

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) along with corresponding visual observations. Annotators are instructed to compare the two explanations and select the one better aligned with the ground-truth reasoning. If both explanations appear equally valid, they may select both. As shown in Figure 7(b), our method outperforms all baselines in explanation quality, highlighting the importance of spatial reasoning for accurate failure understanding.

Comparison with VLM-Based Methods in Real-World Tasks. While prior experiments were conducted in the AI2-THOR simulator, a potential sim-to-real gap remains. To evaluate real-world robustness, we benchmark our proposed graph-based method against strong vision-language model (VLM) baselines, including GPT-4o and Gemini 2.0 Flash. Each model receives either raw images or scene graph–based textual descriptions and determines whether subtask preconditions are satisfied.

We evaluate on the RoboFail-Real dataset, which includes 30 real-world failure cases collected with a UR5e robot, across two tasks: failure detection (checking subtask feasibility) and failure reasoning (explaining causes). Each task is tested under two prompt settings: (i) a naïve setup without object context and (ii) a contextual setup with object lists. We report reasoning performance for the



(a) Real World RoboFail Dataset Experiments

(b) Real World Manipulator and Quadruped Robot Experiments

Figure 8: (a) Comparison of verification and reasoning processes between VLMs using scene graph and image inputs on the RoboFail-Real dataset. (b) Real-world experiments conducted on a UR5e manipulator and Ghost Robotics Vision 60 robot, illustrating task execution and failure reasoning within their respective environments

contextual setup. Evaluation uses five complementary metrics [8]: FDR for detection accuracy; LCS and ROUGE-L for textual overlap; Cosine Similarity for semantic closeness; and LLM Fuzzy Matching (using Claude-4.5 Sonnet) for teacher–student semantic alignment.

As shown in Table 2, VLMs perform reasonably well in detecting valid preconditions, but the graph-based approach achieves comparable accuracy with far more consistent reasoning. Across LCS, ROUGE-L, Cosine Similarity, and LLM Fuzzy Matching, scene graph inputs consistently yield stronger reasoning performance, demonstrating that explicit structural representations provide a more robust basis for explaining failures in real-world settings.

Real-World Experiments. To further validate real-world applicability, we conduct physical experiments using two distinct robotic embodiments: a UR5e manipulator and a Ghost Robotics Vision 60. Across five real-world tasks, our scene graph–based method consistently detects and explains failures with high accuracy, demonstrating that structural reasoning remains effective beyond simulation. As illustrated in Figure 8(b), the experiments showcase two representative setups—one with their corresponding similarity scores and reasoning outputs.

Ablation Studies. We perform ablation studies to assess the contribution of each component in the scene graph comparison process. As shown in Table 3 (left), removing the subtask node moderately reduces detection accuracy, underscoring the importance of task context. Excluding node or edge matching causes larger drops, confirming that both object identity and relational structure are essential for accurate failure detection. The full model attains the highest detection rate of 94.01%. As shown in Table 3 (right), removing the reasoning module drastically lowers the task success rate from 75.67% to 37.67%, highlighting its critical role in interpreting failures and generating context-aware recovery plans. Without it, the system reverts to uninformed replanning that fails to address underlying causes.

5 CONCLUSION

We propose a novel proactive replanning framework that detects and mitigates potential failures before they occur through structured visual understanding. By leveraging scene graph comparisons between the current observation and reference trajectories extracted from successful experiences, our method enables pre-execution detection with online replanning. Additionally, by leveraging an incrementally expanding buffer, the system effectively reduces dependence on costly reasoning calls while maintaining robust performance across diverse tasks. Experimental results on simulated and real world benchmark demonstrate that our approach outperforms baselines. Overall, we suggest that proactive, structured replanning offers a promising path forward for enabling safe and adaptive robot behavior in complex environments.

ACKNOWLEDGMENTS

This work was the result of a project supported by the KT (Korea Telecom)–Korea University R&D Center. Also, this work was partly supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) under the Leading Generative AI Human Resources Development program (IITP-2025-RS-2024-00397085, 20%), the Artificial Intelligence Star Fellowship Support Program to Nurture the Best Talents (IITP-2025-RS-2025-02304828, 20%), the Korea Institute of Science and Technology (KIST) Institutional Program (Project No. 2E33611, 10%), and grant RS-2022-II220043 ("Adaptive Personality for Intelligent Agents," 10%), funded by the Korea Government (MSIT). Daewon Chae is supported by the Hyundai Motor Chung Mong-Koo Foundation.

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. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

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

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

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

[5] Yuchen Cui, Siddharth Karamcheti, Raj Palleti, Nidhya Shivakumar, Percy Liang, and Dorsa Sadigh. 2023. 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*. 93–101.

[6] Yinpei Dai, Jayjun Lee, Nima Fazeli, and Joyce Chai. 2024. RACER: Rich Language-Guided Failure Recovery Policies for Imitation Learning. arXiv:2409.14674 [cs.RO]

[7] Venkata Naren Devarakonda, Ali Umut Kaypak, Shuaihang Yuan, Prashanth Krishnamurthy, Yi Fang, and Farshad Khorrami. 2025. Multitalk: Introspective and extrospective dialogue for human-environment-llm alignment. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 10737–10743.

[8] Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. 2025. AHA: A Vision-Language-Model for Detecting and Reasoning Over Failures in Robotic Manipulation. In *The Thirteenth International Conference on Learning Representations*.

[9] Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. 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*. Springer, 15–29.

[10] Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. 2024. Mobile ALOHA: Learning Bimanual Mobile Manipulation using Low-Cost Whole-Body Teleoperation. In *8th Annual Conference on Robot Learning*.

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

[12] Yanjiang Guo, Yen-Jen Wang, Lihan Zha, and Jianyu Chen. 2024. Doremi: Grounding language model by detecting and recovering from plan-execution misalignment. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 12124–12131.

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

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

[15] Hanxiao Jiang, Binghao Huang, Ruihai Wu, Zhuoran Li, Shubham Garg, Hooshang Nayyeri, Shenlong Wang, and Yunzhu Li. 2024. RoboEXP: Action-Conditioned Scene Graph via Interactive Exploration for Robotic Manipulation. In *8th Annual Conference on Robot Learning*.

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

[17] Jinyeon Kim, Cheolhong Min, Byeonghwi Kim, and Jonghyun Choi. 2024. Preemptive Action Revision by Environmental Feedback for Embodied Instruction Following Agents. In *8th Annual Conference on Robot Learning*.

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

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

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

[21] Zeyi Liu, Arpit Bahety, and Shuran Song. 2023. REFLECT: Summarizing Robot Experiences for Failure Explanation and Correction. In *7th Annual Conference on Robot Learning*.

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

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

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

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

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

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

[28] Gabriel Herbert Sarch, Yue Wu, Michael J. Tarr, and Katerina Fragkiadaki. 2023. Open-Ended Instructable Embodied Agents with Memory-Augmented Large Language Models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

[29] Lucy Xiaoyang Shi, Zheyuan Hu, Tony Z. Zhao, Archit Sharma, Karl Pertsch, Jianlan Luo, Sergey Levine, and Chelsea Finn. 2024. Yell At Your Robot: Improving On-the-Fly from Language Corrections. *CoRR* abs/2403.12910 (2024).

[30] Fatemeh Shiri, Xiao-Yu Guo, Mona Golestan Far, Xin Yu, Gholamreza Haffari, and Yuan-Fang Li. 2024. An Empirical Analysis on Spatial Reasoning Capabilities of Large Multimodal Models. *arXiv preprint arXiv:2411.06048* (2024).

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

[32] Shu Wang, Muzhi Han, Ziyuan Jiao, Zeyu Zhang, Ying Nian Wu, Song-Chun Zhu, and Hangxin Liu. 2024. LLM3: Large Language Model-based Task and Motion Planning with Motion Failure Reasoning. In *Multi-modal Foundation Model meets Embodied AI Workshop @ ICML2024*.

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

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

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

[36] Zhecheng Yuan, Tianming Wei, Shuiqi Cheng, Gu Zhang, Yuanpei Chen, and Huazhe Xu. 2024. Learning to Manipulate Anywhere: A Visual Generalizable Framework For Reinforcement Learning. In *8th Annual Conference on Robot Learning*.

[37] Lihan Zha, Yuchen Cui, Li-Heng Lin, Minae Kwon, Montserrat Gonzalez Arenas, Andy Zeng, Fei Xia, and Dorsa Sadigh. 2023. Distilling and Retrieving Generalizable Knowledge for Robot Manipulation via Language Corrections. In *2nd Workshop on Language and Robot Learning: Language as Grounding*.

[38] Bohan Zhai, Shijia Yang, Chenfeng Xu, Sheng Shen, Kurt Keutzer, Chunyuan Li, and Manling Li. 2023. HallE-Control: controlling object hallucination in large multimodal models. *arXiv preprint arXiv:2310.01779* (2023).

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