REFLECT: Summarizing Robot Experiences for FaiLure Explanation and CorrecTion

Anonymous Author(s)
Affiliation
Address
email

Abstract: The ability to analyze failed executions and correct its own failures is crucial for an explainable and robust robotic system. Large Language Models (LLMs) have recently demonstrated strong common sense reasoning skills on textual inputs. To leverage the power of LLM for robot failure explanation, we propose a framework REFLECT, which converts multi-sensory data into a hierarchical summary of robot past experiences and queries LLM with a progressive failure explanation algorithm. Finally, conditioned on the explanation, a failure correction planner generates an executable plan for the robot to correct its failure and complete the task. To systematically evaluate the framework, we create the RoboFail dataset and show that our LLM-based framework is able to generate informative failure explanations that assist successful correction planning in both simulation and real world. The dataset and code will be publicly available.

Keywords: Large Language Model, Explainable AI, Task Planning

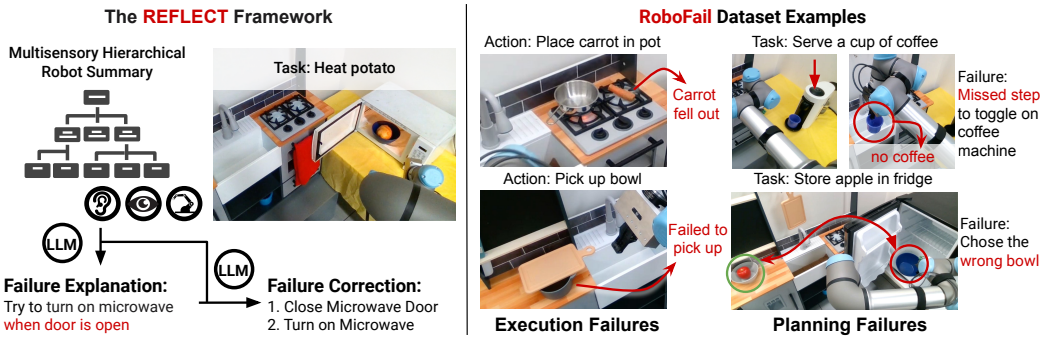


Fig 1: **Summarization for failure explanation and correction.** On the left, we show the REFLECT framework that converts multi-sensory input (RGB-D, audio, proprioception) to a hierarchical summary of robot experiences, which is then used to query Large Language Model (LLM) for failure explanation and correction. The right shows example failure cases in the RoboFail dataset. Results are best visualized on the anonymous website: https://roboreflect.github.io/.

1 Introduction

With the increasing expectations for robots to work on long-horizon tasks in complex environments, failures are inevitable. It is thus essential for a robotic system to identify and analyze failed executions, as such self-reflection ability can increase both **explainability** and **robustness** of the system. This will also help reduce human labor by automatically analyzing the tedious robot data log and allowing the system to potentially self-correct from the failure state.

An effective failure explanation requires several key components: first, the ability to summarize "what happened" by converting the *unstructured, multimodal* raw sensory data into a *structured, unified* format; second, the ability to reason "what was wrong" by inferring from the summary if expected outcomes of the robot plan are achieved; and finally, the ability to plan "what to do" based on the reasoning to correct the failure and complete the task.

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

Recently, Large Language Models (LLMs) [1, 2, 3] have been shown to exhibit strong reasoning [4, 5, 6] and planning [7, 8, 9] capabilities, making it a promising component for explainable and robust robotic systems. However, the remaining challenge lies in how to generate a "textual summary" of raw robot sensory data that can be used in prompts and a systematic way to query LLMs for accurate failure explanation. In other words, how do we transform the robot failure explanation task into a language reasoning task? We observe two important attributes of a good robot summary:

- **Multi-sensory.** The summary should cover all sensory inputs that the robot has access to, such as visual, audio, contact, etc. This is because certain failures can be more easily identified through one type of sensory data than another. For example, it is easier to determine if an object is dropped or if water is running from the faucet using auditory cues rather than visual ones.
- **Hierarchical.** To support effective failure explanations, the robot summary requires multiple levels of abstraction. On one hand, to quickly localize the failure, the highest summary level should focus on verifying whether the subgoals are successful while ignoring execution details. On the other hand, the lower summary levels should maintain enough environmental context for the LLM to generate an informative explanation that can assist in correction planning.

Based on these observations, we introduce REFLECT, a framework that summarizes robot experiences for failure explanation and correction. The framework first generates a **multi-sensory hierarchical summary** with three levels of abstraction. Equipped with this summary, we design a **progressive failure explanation** algorithm to accurately identify failures and generate informative explanations in the context of the environment and robot actions. Conditioned on the failure explanation, the **failure correction planner** generates an executable plan for the robot to correct the failure and complete the task.

In summary, our primary contribution is an LLM-based framework **REFLECT** that leverages a hierarchical summary of multi-sensory robot data for failure explanation and correction. To systematically evaluate REFLECT, we also release the **RoboFail** dataset. The dataset includes 100 failure demonstrations created from the AI2THOR simulation environment and 30 real-world failure demonstrations collected with a UR5e robot arm. We hope the dataset will encourage the development of more explainable and robust embodied AI systems.

2 Related Work

Dense Video Captioning for long-horizon videos has been a challenging task in computer vision. Recent video captioning methods typically train transformer-based models to jointly localize and caption events in videos [10, 11, 12, 13, 14]. Yet it remains a challenge to caption robot videos due to a lack of data. With the emergence of large foundation models, zero-shot video captioning is made possible [15, 16]. These works combine VLMs and LLMs to caption egocentric human activity videos in a zero-shot manner. Extending upon prior works, our approach generates captions that are task-centric and action-centric for temporal robot sensory data in a zero-shot manner, which helps downstream tasks such as robot behavior analysis [17].

Robot Failure Explanation is an important task long studied in HRI to increase human trust in robotic systems. [18, 19] and allow non-expert users to better assist robots in failure scenarios [20, 21]. However, the failure scenarios covered in each work are limited and the methods lack generalizability: Das and Chernova [21] specifically study failure cases in picking, Diehl and Ramirez-Amaro [22] study two pick and place tasks, Song et al. [23] focus on object navigation failures whereas Inceoglu et al. [24] focus on simple object manipulation failures. We propose a LLM-based failure explanation framework that leverages the advanced reasoning ability of LLMs and generalizes well to various failure scenarios including both low-level action execution failures and high-level task planning failures.

Task Planning with Large Language Models Large language models (LLMs) can be leveraged to decompose high-level, abstract task instructions into low-level step-by-step actions executable by agents [7, 9, 25, 26]. Recent works have also demonstrated the self-reflective and self-corrective ability of LLMs based on environment feedback [8, 27, 28, 29]. However, they all assume ground truth environment feedback associated with one or few actions. In this work, we explore the reflective ability of LLMs directly on multi-sensory robot observation data and its temporal reasoning ability on long-horizon robot task executions. Our framework is able to directly operate on real-world robot task execution data with various failure scenarios.

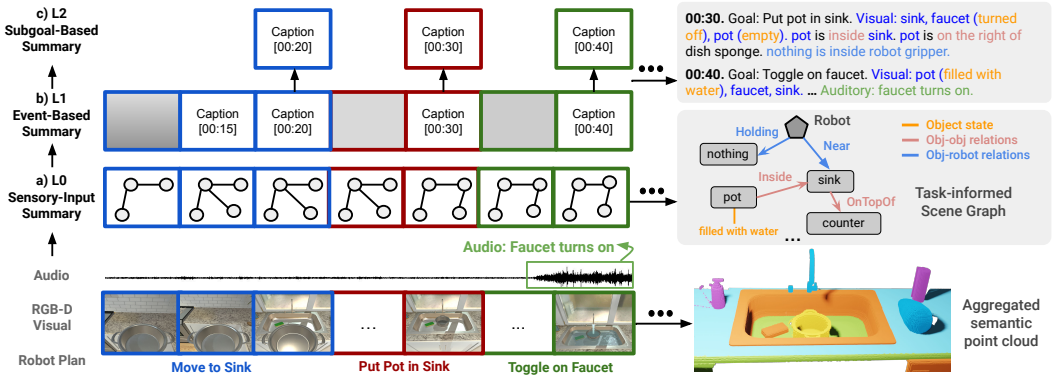


Fig 2: **Hierarchical Robot Summary** is composed of: a) a sensory-input summary that converts multi-sensory robot data (RGB-D, sound, robot states) into task-informed scene graphs and audio summary; b) an event-based summary that generates captions for key event frames; c) a subgoal-based summary that contains the end frame of each subgoal.

3 Method: The REFLECT Framework

We present the REFLECT framework to summarize a robot's past experiences for failure explanation and correction. The framework contains of three components: the hierarchical robot summary module that converts multi-sensory robot data into textual descriptions of robot experiences with three levels of abstraction (§3.1), the progressive failure explanation module that queries LLM to localize and explain the failure (§3.2), and finally the failure correction planner that generates a feasible plan for the robot to correct the failure and complete the task (§3.3).

3.1 Hierarchical Robot Summary

To perform effective failure explanation and correction, we propose a hierarchical summary structure to 1) aggregate and convert robot sensory data over time into a unified structure; 2) summarize the robot experiences for quick localization and accurate explanation of failures. The hierarchical summary structure contains three levels: sensory-input summary, event-based summary, and subgoal-based summary. Each level is discussed in detail below.

3.1.1 Sensory-Input Summary

The sensory-input summary processes unstructured, multimodal robot sensory observations into a unified structure that stores the necessary information for failure explanation.

Visual summary with task-informed scene graphs. To understand the robot's interactions with its surrounding environment, it is important to maintain inter-object relations, robot-object relations (i.e. what object the robot is holding), and object states information. Thus, we propose a scene graph structure where nodes contain information about objects and their states, and edges contain inter-object and robot-object relations. Furthermore, to make the summary succinct and reduce the computation and memory cost, we only include objects in the scene graph that are relevant to the task. Task-relevant objects are objects mentioned in the original robot plan and objects that are spatially related to task-relevant objects.

Given the RGB-D observation at timestep t, we run object detection on the RGB image to obtain a semantic segmentation I_t^S and project it to a 3D semantic point cloud with the observed depth. In addition, for objects that can change states (e.g. microwave can be turned on and off), we detect the corresponding object state by computing the cosine similarity of the image features and the object state text features output by CLIP encoders [30]*. Given the semantic point cloud, we design heuristics for inter-object spatial relations. We consider a total of 8 spatial relations†: inside, on top of, on the left, on the right, above, below, occluding, and near. We also obtain the robot-object relation using gripper state and object detection. To aggregate the 3D point cloud over time frames, we use a similar approach as Li et al. [31] by aligning the newly observed point clouds p_t with the accumulated point cloud from all previous time steps P_{t-1} using 4 heuristic operations: add, update, replace, and delete.†

*The framework assumes access to ground truth object detection and object states in simulation.

†Details of the heuristics can be found in appendix.

Once we have the aggregated point cloud, we construct a scene graph $G_t = \{N + \{n_{robot}\}, E\}$, which describes the object nodes (N) and their spatial relations (E). Each node is defined as $n_i = (c_i, s_i)$, where c_i is the object class, s_i is the object state if any. Each edge contains the spatial relation between the two objects. We add the robot as an additional node n_{robot} and a special relation "inside robot gripper".

Audio summary. In cases where environment changes are hard to identify from visual cues (e.g. something drops from the robot gripper, stove burner turns on but is occluded by an object on top), audio can serve as an additional modality to detect these changes and help identify failures.

Given an input audio stream, we first segment the whole audio clip into several sub-clips by filtering out ranges where the volume is below a certain threshold ϵ. Then for each sub-clip s, we compute its audio-lingual features with a pre-trained audio-language model (e.g. AudioCLIP [32], Wav2CLIP [33]). We also compute the language features for a list of candidate labels L with a frozen language encoder. We calculate the cosine similarity between the audio subclip and the labels in feature space and select the highest-scoring label l^*: $l^* = \text{argmax}_{l \in L}[C(s,l)]$, $C = \frac{f_1(s) \cdot f_2(l)}{||f_1(s)||f_2(l)||}$ where f_1 is the embedding function for audio, f_2 is the embedding function for text.

3.1.2 Event-Based Summary

Given that the sensory-input summary (§3.1.1) computes a scene graph for each frame and thus contains redundant information, the goal of the event-based summary is to select key frames from the sensory-input summary and generate corresponding text captions from scene graphs.

We design a key frame selection mechanism based on visual, audio, and robot events. More specifically, a frame is selected if it satisfies any of the conditions below: 1) The task-informed scene graph of the current frame G_t is different from the previous frame G_{t-1}. 2) The frame is the end of a robot action execution. 3) The frame is the start or end of an audio event.

For each key frame, we convert the corresponding scene graph into text with the following format: [‡] Note that when constructing the visual observation, we only consider objects that are visible in the current frame.

```
[timestep] Action: [robot action]
Visual observation: object1 [state], object2, object3 [state] ... # objects and states
object1 is [spatial relation] object2 ... # object-object relations
object3 is inside robot gripper. # robot-object relations
Auditory observation: [audio summary].
```

3.1.3 Subgoal-Based Summary

The event-based summary (in §3.1.2) stores environment observations throughout the robot task execution. However, it's hard to reason the expected outcomes and identify failures in every one of the low-level actions. It is easier for an LLM to reason about a higher-level intention of the robot (e.g. move to the toaster, put bread slice in the toaster). Thus, we introduce the subgoal-based summary, which consists of observations at the end of each subgoal. The subgoal-based summary enables the failure explanation module to quickly process the robot experience summary and identify failures by checking whether each subgoal is satisfied while ignoring low-level execution details. Once a failure is detected, we can retrieve relevant environment information stored in the event-based summary or sensory-input summary to query LLM for a detailed failure explanation.

3.2 Progressive Failure Explanation

The failure explanation algorithm should handle both execution and planning failures, where the former requires action-level observation details and the latter requires task-level information such as task description and robot plan. To do so, the algorithm first identifies the type of failure and then retrieves relevant information from the hierarchical summary to construct the query to LLM for failure explanation. As shown in Fig. 3, the algorithm first goes through each subgoal and verifies the subgoal using the following prompt: [§]

[‡] text color: blue: visual , green: audio , light blue: contact, yellow: summary, orange: final state, failure explanation, brown: timestep, task name, robot subgoal, original robot plan, goal state, blue highlight: LLM output

[§] Complete examples are shown in the appendix.

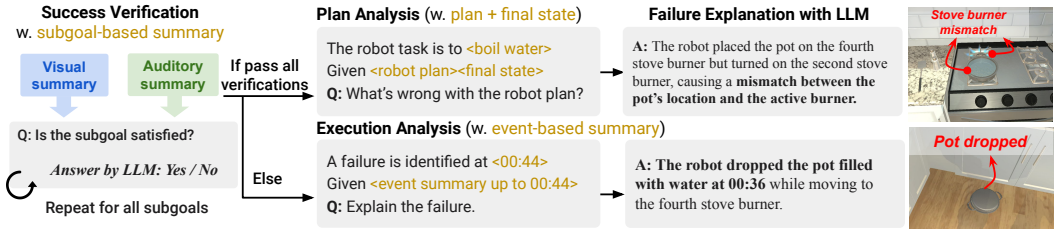


Fig 3: **Progressive Failure Explanation** verifies success for each subgoal. If a subgoal fails, the algorithm enters the *execution analysis* stage for detailed explanation. If all subgoals are satisfied, the algorithm enters *planning analysis* stage to check errors in the robot plan.

> The robot subgoal is [robot subgoal at time t]. Given [subgoal-based summary at time t]
> Q: Is the subgoal satisfied? A: Yes

The LLM is instructed to output 'Yes' or 'No'. If a subgoal is not achieved, then we retrieve history observations stored in event-based summary for failure explanation as follows:

> The robot task is to [task name]. A failure is identified at t. Given [event-based summary up to t]
> Q: Briefly explain what happened at t and what caused the failure?
> A: At 00:44, the robot attempted to put the pot on the fourth stove burner, but the pot was not in its gripper. The failure was caused by the robot dropping the pot filled with water at 00:36 while moving to the fourth stove burner.

In case all subgoals are achieved but the task still failed, then it's likely that the plan itself was incorrect. We use the robot original plan and the final state of the environment to identify errors in the robot plan. The final state is obtained from the scene graph generated from the aggregated semantic point cloud in the last time step without local relations (on the left, on the right, occluding). The prompt looks as follows:

> The robot task is to [task name]. The task is successful if [goal state].
> The robot plan is [original robot plan]. Given [final state]
> Q: What's wrong with the robot plan that caused the robot to fail?
> A: The robot placed the pot on the fourth stove burner but turned on the second stove burner, causing a mismatch between the pot's location and the active burner.
> Q: Which time step is most relevant to the above failure?
> A: 00:49

3.3 Failure Correction Planner

A failure correction planner should generate an executable plan for the robot to correct the failure and complete the task. Prior work [20] has shown that good failure explanations help non-expert users better understand the failure and assist the robot. Analogously, we hypothesize that conditioned on the failure explanation, LLMs can also generate a correction plan for the robot that leads to final task success. The prompt is formatted as below:[§]

> The robot task is to [task name]. The original robot plan is [original robot plan].
> Given [failure explanation] [final state] and [goal state]
> Correction plan: toggle_off (stoveburner-2), toggle_on (stoveburner-4)

To make sure the plan generated by the language model is executable in the environment, we adopt the idea of Huang et al. [26] to map each LLM-generated action to its closest executable action in the task environment using a large pre-trained sentence embedding model.

4 The RoboFail Dataset

The RoboFail dataset consists of failure scenarios in both simulation and real world. For simulation, we generate task execution data in AI2THOR [34] and manually inject failures. The dataset contains a total of 100 failure scenarios, with 10 cases for each of the 10 tasks. We provide the RGB-D observations, sound (20 classes in total), robot state data, as well as ground truth metadata obtained from simulation. The real-world dataset is collected by human teleoperation of a UR5e robot arm in a toy kitchen environment. The dataset contains 11 tasks with a total of 30 failure scenarios. We provide the RGB-D observations

(with Intel RealSense D415), sound (with RØDE VideoMic Pro+), and proprioception data. In addition to the dataset, we also introduce a taxonomy of robot failure scenarios. The distribution of each category in the RoboFail dataset and some examples are visualized in Fig. 4.

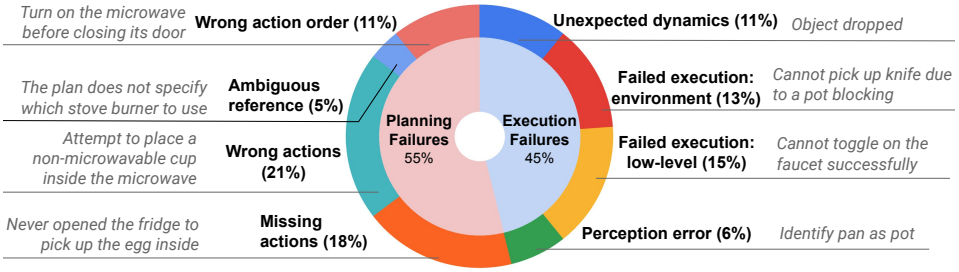


Fig 4: **RoboFail Failure Taxonomy**

5 Evaluation

We systematically evaluate the ability of REFLECT to localize, explain and correct robot failures. In simulation, the framework assumes ground truth object detection and state detection. In real world experiments, we use MDETR [35] for object detection, CLIP [30] for object state detection, and AudioCLIP [32] for sound detection. We use GPT-4 as the LLM. The below metrics are evaluated in our experiments:

- Exp (explanation): percentage of predicted failure explanations that are correct and informative as determined by human evaluators[¶].
- Loc (localization): percentage of predicted failure time that align with actual failure time. A predicted time is considered aligned if it falls within the annotated failure time ranges in the dataset.
- Co-plan (correction planning success rate): percentage of tasks that succeed after correction.

To demonstrate the advantages of our framework, we compare with the following baselines/ablations:

- BLIP2 caption: use a state-of-the-art image caption model, BLIP2 [36], to caption the video frames.
- w/o sound: our approach without sound detection and the audio summary.
- w/o progressive: similar to open-ended Q&A in Socratic Models [15], directly query LLM for failure explanation given the robot summary without the progressive failure explanation algorithm.
- Subgoal only: use only subgoal-based summary for failure explanation.
- LLM summary: convert all sensory-input summary to text and prompt LLM to generate a summary for failure explanation.
- w/o explanation: query LLM for a correction plan without failure explanation.

By evaluating REFLECT on the RoboFail dataset and comparing with the above baselines/ablations, we have the below findings:

	Execution failure			Planning failure		
Method	**Exp**	**Loc**	**Co-plan**	**Exp**	**Loc**	**Co-plan**
w/o progressive	46.5	62.8	60.5	61.4	70.2	64.9
Subgoal only	76.7	74.4	51.2	71.9	73.7	75.4
LLM summary	55.8	67.4	65.1	57.9	54.4	66.7
w/o explanation	-	-	41.9	-	-	56.1
REFLECT	**88.4**	**96.0**	**79.1**	**84.2**	**80.7**	**80.7**

Table 1: Result in Simulation Environments

REFLECT is able to generate informative failure explanations that assist correction planning. Tab. 1 and 2 summarize the evaluation result, where REFLECT achieves the highest performance in explaining, localizing, and correcting failures in both simulation and the real world. The performance slightly decreases in real world due to perception errors. We find that localization is slightly harder for planning failures as the failure is usually not associated with a single timestep. Our framework achieves around 80% correction planning success rates for both failure types.

Audio data is useful for failure explanation. As shown in Tab. 2, the explanation and localization accuracy for [w/o sound] both decrease around 20% on execution failures compared to REFLECT. This is because some unexpected events (e.g. object dropping on floor) or object states hard to identify using visual detectors (e.g. stove burner occluded by a pot on top) are easier to detect through audio.

[¶]Details can be found in appendix.

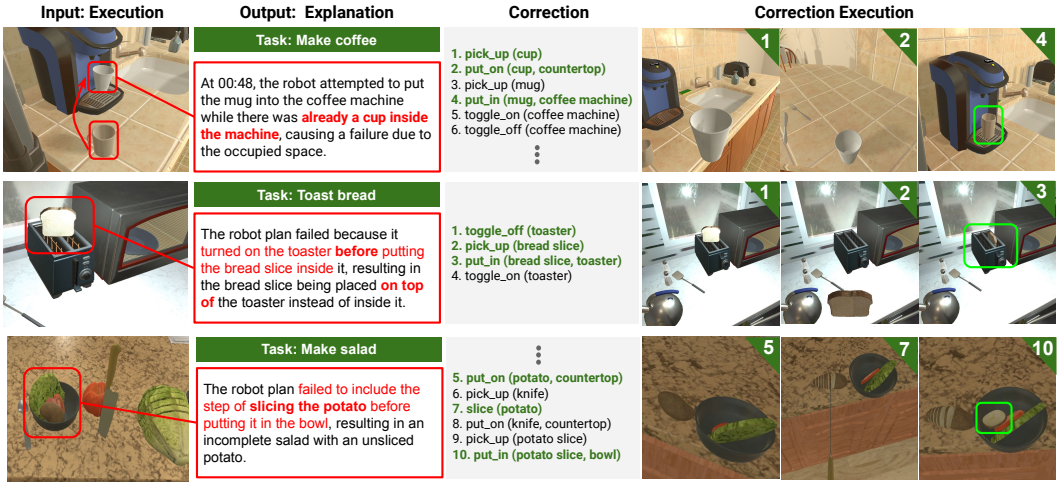


Fig 5: **Qualitative Results in Simulation.** Given a failed robot task execution, REFLECT is able to generate succinct and informative failure explanations for both execution and planning failures. Conditioned on the explanation, a plan can also be generated for the robot to correct the failure and proceed with the task.

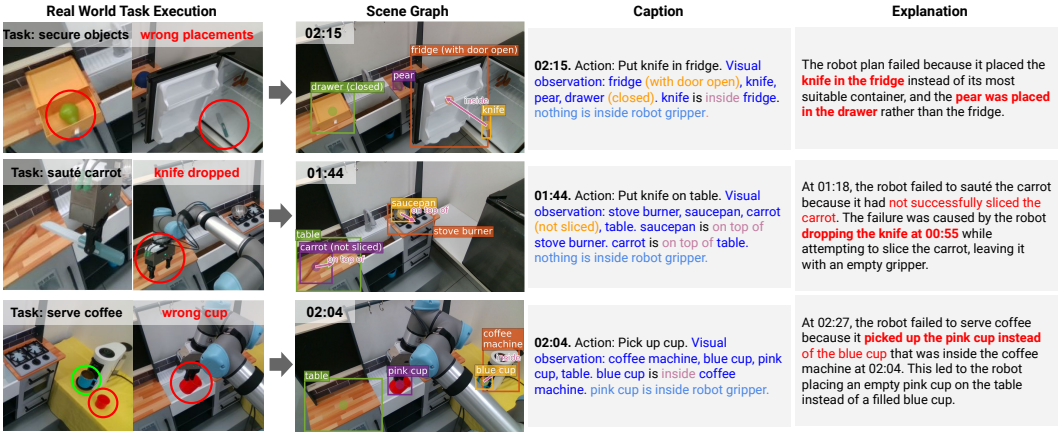
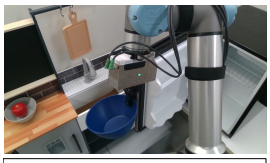


Fig 6: **Qualitative Results in Real World.** REFLECT is able to summarize and generate informative failure explanations for real world robot executions. The above figure shows three failed task executions on the left, the corresponding scene graph and caption for one key frame in the middle, and the LLM-generated failure explanation on the right. More results can be found on the project website: https://roboreflect.github.io/.

Task-relevant object spatial and state information is crucial. As shown in Tab. 2, [BLIP2 caption] achieves the worst performance in all scenarios because the captions generated by BLIP2 lack the necessary information for failure explanation. In contrast, our zero-shot caption method is designed to capture environment information such as object states and spatial relations, which are task-relevant and crucial for failure explanation. The figure on the right shows that REFLECT is able to summarize object states such as "fridge (with door open)" and spatial relations such as "apple inside white bowl", which are not present in the BLIP2 caption.



BLIP2 caption: a robot is holding a bowl in a kitchen.

REFLECT: fridge (with door open), apple, white bowl, dark blue bowl. apple is inside white bowl. dark blue bowl is inside robot gripper.

Progressive failure explanation is important. Our progressive failure explanation algorithm leverages the hierarchical summary to first identify the failure with the subgoal-based summary, and then query LLM for failure explanation accordingly. Comparing with [w/o progressive] in Tab. 1 and 2, the progressive algorithm helps with more accurate localization and more informative failure explanation. To better understand the difference, a qualitative example is shown in Fig. 7: although [w/o progressive] mentions that the task failed because the robot did not have an egg in its gripper to put in the pan, it does not reason why the egg was not present. In contrast, REFLECT identifies failure in the "pick up egg" action and then queries the event-based summary to infer the exact reason – "fridge was closed".

Hierarchical structure is important. The performance decrease of [Subgoal only] shows the importance of event-based summary in the hierarchical summary as it stores intermediate observations that can be retrieved for better failure localization and explanation. Consider the scenario when the robot accidentally dropped the pot it was holding when moving to stove burner. [Subgoal only] only infers that the object was not in the robot gripper at the end of the subgoal execution. In contrast, the event-based summary stores auditory observation of "something drops on ground" and visual observation of "nothing is inside robot gripper", which helps identify that the pot was dropped and the exact time step the failure occurred.

	Execution failure		Planning failure	
Method	**Exp**	**Loc**	**Exp**	**Loc**
BLIP2 caption	6.25	25.0	35.7	57.1
w/o sound	50.0	68.8	78.6	78.6
w/o progressive	43.8	81.3	71.4	78.6
Subgoal only	56.3	62.5	71.4	78.6
LLM summary	37.5	75.0	64.3	71.4
REFLECT	**68.8**	**93.8**	**78.6**	**78.6**

Table 2: Result in Real World Environments

In addition, the hierarchical structure is a better way to condense the sensory-input summary for failure explanation. We implement an alternative [LLM summary], which prompts LLM to summarize the sensory-input summary. The performance decreases significantly in both simulation and real-world as the LLM-generated summary loses information that is relevant for failure explanation. As shown in Fig. 7, the summary and explanation of [LLM summary] only mention that the robot tried to pick up the egg and the gripper remained empty, but did not mention that the fridge was closed, which is the actual failure cause.

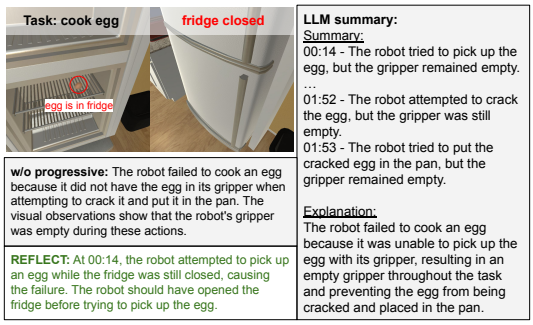


Fig 7: [w/o progressive] vs. [LLM summary] vs. Ours

Failure explanation helps correction. From Tab. 1, we observe that the correction planning success rate significantly decreases for [w/o explanation]. This is because the failure explanation can guide LLM to generate a correction plan based on the failure cause. As shown in Fig. 8, given the failure reason that the mug cannot be put inside the coffee machine due to the presence of a cup, REFLECT generates a plan to move the cup away and then proceed with the task. Whereas [w/o explanation] simply repeats the original plan without taking any actions to address the cause of failure.

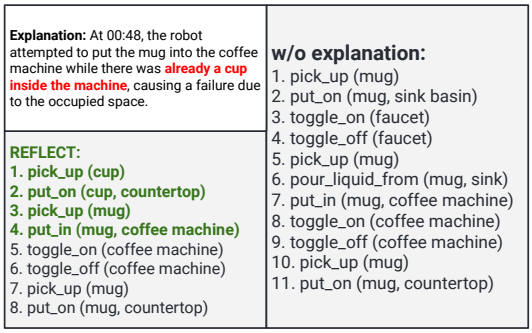


Fig 8: Failure explanation helps correction planning.

Limitation & Future Work. The performance of our results can benefit from better perception models. Future work can consider developing better perception methods that are more accurate and capture more information. In addition, we study one use case of the hierarchical summary for robot failures, but we believe the summary is also useful for other tasks such as analyzing the efficiency and safety of robot behaviors or collaborating with humans. Finally, we release the RoboFail dataset with 8 categories and 130 failure scenarios in simulation and real world. Yet this only covers a limited percentage of all robot failure scenarios. We encourage contribution to the RoboFail dataset by providing the interface to generate failure data in simulation. Real-world data in any setup is welcome too.

6 Conclusion

We study the ability of Large Language Models to explain and correct robot failures. We propose a framework REFLECT which converts temporal, multi-sensory robot data into a hierarchical summary of robot experiences, then queries LLM to localize and explain the robot failures using a progressive failure explanation algorithm, and finally generates a plan for the robot to correct the failure. To evaluate the framework, we release a dataset of robot failed executions in both simulation and real-world and show that REFLECT achieves better performance as compared to several baselines and ablations. We encourage future work to improve the framework and explore more use cases of the robot summary.

References

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[2] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[3] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[4] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[5] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.

[6] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[7] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.

[8] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, N. Brown, T. Jackson, L. Luu, S. Levine, K. Hausman, and B. Ichter. Inner monologue: Embodied reasoning through planning with language models. In *arXiv preprint arXiv:2207.05608*, 2022.

[9] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*, 2022.

[10] T. Wang, R. Zhang, Z. Lu, F. Zheng, R. Cheng, and P. Luo. End-to-end dense video captioning with parallel decoding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6847–6857, 2021.

[11] C. Deng, S. Chen, D. Chen, Y. He, and Q. Wu. Sketch, ground, and refine: Top-down dense video captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 234–243, June 2021.

[12] Q. Zhang, Y. Song, and Q. Jin. Unifying event detection and captioning as sequence generation via pre-training. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVI*, pages 363–379. Springer, 2022.

[13] W. Zhu, B. Pang, A. V. Thapliyal, W. Y. Wang, and R. Soricut. End-to-end dense video captioning as sequence generation. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5651–5665, 2022.

[14] A. Yang, A. Nagrani, P. H. Seo, A. Miech, J. Pont-Tuset, I. Laptev, J. Sivic, and C. Schmid. Vid2seq: Large-scale pretraining of a visual language model for dense video captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10714–10726, 2023.

[15] A. Zeng, M. Attarian, B. Ichter, K. Choromanski, A. Wong, S. Welker, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke, and P. Florence. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv*, 2022.

[16] Z. Wang, M. Li, R. Xu, L. Zhou, J. Lei, X. Lin, S. Wang, Z. Yang, C. Zhu, D. Hoiem, et al. Language models with image descriptors are strong few-shot video-language learners. *arXiv preprint arXiv:2205.10747*, 2022.

[17] C. DeChant and D. Bauer. Summarizing a virtual robot's past actions in natural language. *arXiv preprint arXiv:2203.06671*, 2022.

[18] P. Khanna, E. Yadollahi, M. Björkman, I. Leite, and C. Smith. User study exploring the role of explanation of failures by robots in human robot collaboration tasks. *arXiv preprint arXiv:2303.16010*, 2023.

[19] S. Ye, G. Neville, M. Schrum, M. Gombolay, S. Chernova, and A. Howard. Human trust after robot mistakes: Study of the effects of different forms of robot communication. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1–7, 2019. doi:10.1109/RO-MAN46459.2019.8956424.

[20] D. Das, S. Banerjee, and S. Chernova. Explainable ai for robot failures: Generating explanations that improve user assistance in fault recovery. In *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, pages 351–360, 2021.

[21] D. Das and S. Chernova. Semantic-based explainable ai: Leveraging semantic scene graphs and pairwise ranking to explain robot failures. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3034–3041. IEEE, 2021.

[22] M. Diehl and K. Ramirez-Amaro. Why did i fail? a causal-based method to find explanations for robot failures. *IEEE Robotics and Automation Letters*, 7(4):8925–8932, 2022.

[23] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. *arXiv preprint arXiv:2212.04088*, 2022.

[24] A. Inceoglu, E. E. Aksoy, A. C. Ak, and S. Sariel. Fino-net: A deep multimodal sensor fusion framework for manipulation failure detection. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6841–6847. IEEE, 2021.

[25] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022.

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

[27] S. S. Raman, V. Cohen, E. Rosen, I. Idrees, D. Paulius, and S. Tellex. Planning with large language models via corrective re-prompting. *arXiv preprint arXiv:2211.09935*, 2022.

[28] N. Shinn, B. Labash, and A. Gopinath. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*, 2023.

[29] M. Skreta, N. Yoshikawa, S. Arellano-Rubach, Z. Ji, L. B. Kristensen, K. Darvish, A. Aspuru-Guzik, F. Shkurti, and A. Garg. Errors are useful prompts: Instruction guided task programming with verifier-assisted iterative prompting. *arXiv preprint arXiv:2303.14100*, 2023.

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

[31] X. Li, D. Guo, H. Liu, and F. Sun. Embodied semantic scene graph generation. In A. Faust, D. Hsu, and G. Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1585–1594. PMLR, 08–11 Nov 2022.

[32] A. Guzhov, F. Raue, J. Hees, and A. Dengel. Audioclip: Extending clip to image, text and audio. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 976–980. IEEE, 2022.

[33] H.-H. Wu, P. Seetharaman, K. Kumar, and J. P. Bello. Wav2clip: Learning robust audio representations from clip. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4563–4567. IEEE, 2022.

[34] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, M. Deitke, K. Ehsani, D. Gordon, Y. Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[35] A. Kamath, M. Singh, Y. LeCun, G. Synnaeve, I. Misra, and N. Carion. Mdetr-modulated detection for end-to-end multi-modal understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1780–1790, 2021.

[36] J. Li, D. Li, S. Savarese, and S. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.