Foundation Models for Enhanced Understanding of Open-World Environments in Failure Recovery

Shufei Chen
School of Interactive Computing
Georgia Institute of Technology
United Statesif
schen964@gatech.edu

Abirath Raju
School of Engineering
Georgia Institute of Technology
United States
araju60@gatech.edu

Abivishaq Balasubramanian
School of Interactive Computing
Georgia Institute of Technology
United States
abivishaq@gatech.edu

Muskan Bawa
School of Engineering
Georgia Institute of Technology
United States
mbawa6@gatech.edu

1 Introduction

As robots undertake increasingly complex and extended tasks, encountering setbacks is inevitable. The essence of robotics lies not just in executing tasks but also in the ability to introspect, evaluate past actions, and communicate missteps in a language as intuitive as natural language. This ability streamlines the debugging process, eliminating the need to navigate through detailed execution logs, and enables autonomous error correction. In addition, failure explanation in the form of a textual summary can be used to automatically generate a corrective plan utilizing an LLM.

A key to achieving this is integrating a comprehensive representation to improve the robot's understanding of its surroundings and interactions, thus enhancing accurate self-assessment and failure explanation. Scene graphs effectively summarize scenes by using nodes to represent objects and their states, and edges to denote spatial relationships between these objects.

Recent studies have focused on using closed-set vocabulary for scene graphs in failure explanation tasks. This approach relies on predefined lists of object states and spatial relations, assigning each recognized object to one of the fixed set of possible states, which significantly limits its adaptability to unseen objects or states. An open-vocabulary scene graph is able to represent information about any objects, object states, and inter-object spatial relationships, regardless of their prior inclusion in the predefined list. Hence, adopting open-vocabulary scene graph representations is vital for enhancing generalization capabilities.

Building on this foundation, our research introduces a novel framework designed to advance failure recovery in robotics, structured around scene description for failure detection. Leveraging state-of-the-art VLM models like the Gemini 1.5 Pro Visual Language Model [1] helps us get the list of objects, their states and the relationships between the objects present in the scene. We evaluate our methodology against baselines, such as the REFLECT model and a Vision-Language Model that directly takes the video input. The RoboFail dataset [2] is used for evaluation. This research represents a significant step towards developing more reliable and autonomous robotic systems

2 Related Work

In the field of Vision-Language Models (VLMs), recent advancements have been instrumental in semantic understanding and spatial reasoning, both essential for improving robotic perception. Models such as CLIP [3] and InstructBLIP [4] have shown exceptional capability in generating language descriptions from images, demonstrating high accuracy in zero-shot learning scenarios. This ability to generalize across various tasks without tailored training is vital for projects aimed at open-vocabulary scene graph generation.

Building on the advancements in Vision-Language Models (VLMs) [1] and the exploration of failure explanations in robotics, the field of scene graph generation is also undergoing significant transformations. Models such as Open-Vocabulary Scene Graph Generation (Ov-SGG) [5] propose an innovative approach. This methodology leverages prompt-based fine-tuning on pre-trained visual-textual pairs, enabling the detection and localization of visual relations between objects, including those unseen during training. By not requiring updates to the pre-trained model's parameters, this approach demonstrates remarkable adaptability, outperforming traditional methods on benchmark datasets like Visual Genome, GQA, and Open Images. This progression towards open-vocabulary scene graphs is pivotal for enhancing the generalization capabilities of robotic systems, promising a more nuanced understanding of their surroundings for effective error explanation and recovery.

Explanations of failures in autonomous systems are increasingly critical. Das et al. [6] and LeMasurier et al. [7] explore robotics failure explanations for human users to help robots recover from failure. Alternatively, failure explanation can be used by a corrective system to autonomously recover from failure without the need for human intervention [2]. Other works incorporate task variety or more complex scenarios [6][7] such as tabletop manipulation, but restrict the scene representation to a closed vocabulary set.

Although current open vocabulary scene graph generation[8][9] usually contains information about the objects in the scene and the relationship between them, they lack information about the state of the object. "Fridge is open" and "Coffee machine is on" are examples of state information for objects. Liu et al. [2] incorporate these states and are able to achieve state-of-the-art results for failure explanation. Therefore an open-vocabulary scene graph generator that also gives the state of the objects is needed. States like "door partially open" are important, therefore the state representation needs to also be from an open set. Existing works on open vocab scene graph generators can be open vocab in terms of the objects it can handle but can be closed vocab in terms of the relationships between the objects. Zhang et al. [9] have a fixed set of spatial relations but can used for unseen objects. For failure explanation being open vocabulary in terms of the spatial relationship is also important. For example, a failure occurs due to a plate being placed at the edge of the table. Usual closed set would encode the information as the plate is on the table, whether the plate is in the middle or edge of the table. This information is insufficient for failure explanation. Therefore there is a need for an open vocabulary scene graph generator which is open set in terms of objects, relations and states of the objects.

This paper's contributions can be summarized as:

- Open-Vocabulary Scene Description: We introduce a novel framework that utilizes open-vocabulary scene graphs that are open-set in terms of the objects, states of objects, and their relationships.
- Evaluation of Vision-Language Models for end-to-end failure explanation: We leverage Gemini 1.5 Pro directly to produce failure explanations from videos of the task being executed.

3 Problem Statement

In this section, we formulate a failure detection problem. An agent is provided with a task objective τ and an action plan A_p that specifies a sequence of action primitives (i.e. subgoals) to achieve the task objective. The action plan itself could be erroneous, meaning that even if executed correctly, it may not lead to achieving the task objective. This is defined as a planning failure. When the action plan is effective but the agent fails to execute the action plan successfully, an execution failure occurs. As the agent executes each action primitive of the action plan, it observes the current state s_t through sensors and obtains an observation O_t at time t. Key features, such as object states and their spatial relations, can be extracted from this raw observation O_t into an abstract form such as a

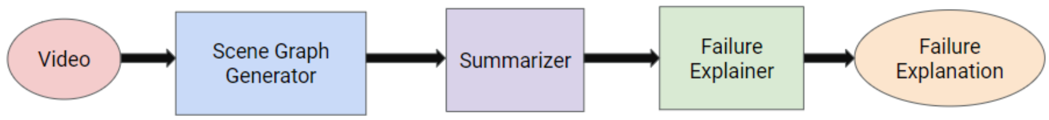

Figure 1: Overview of our method

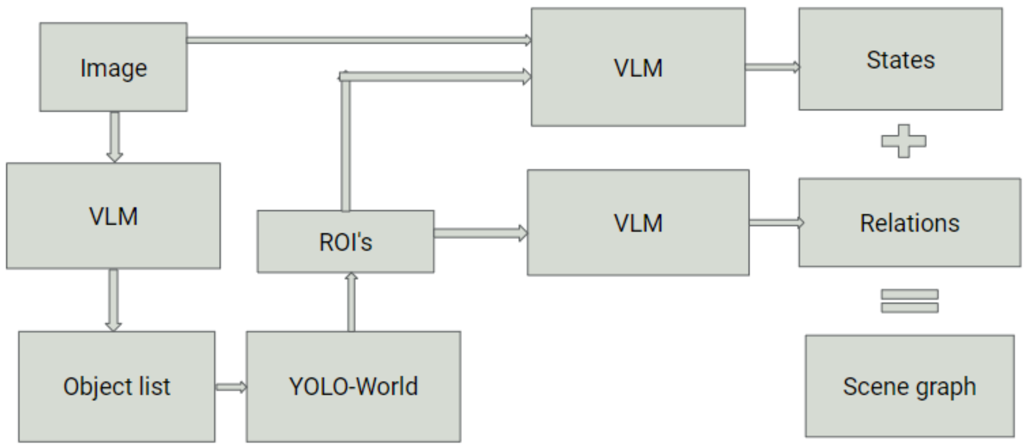

Figure 2: Open vocab scene graph generation

scene graph which we denote as G_t. O_t and G_t represent environment information accessible to the agent, which we collectively denote as E_t. To explain the failure we define a framework F where

$$F : (A_p, E_0, E_1, ..., E_{T1}) \rightarrow e_f$$

Here, e_f denotes the failure explanation, which should contain the following information: failure type c_f, failure step t_f, and failure reason r_f. Failure type c_f identifies whether it is a planning or execution failure. Failure step t_f specifies at which subgoal of the action plan the failure occurs. Failure reason r_f explains the reason of the failure.

Hypothesis: In failure explanation, either the raw observations O or generated scene graphs G can be used as inputs E to the framework. We hypothesize that G is a better representation of the environment information, which results in more accurate identification of the failure type, failure step, and failure reason.

4 Methodology

Our research paper adopts the framework from Liu et al. [2] designed to address failure explanation in robotics, which is structured around three primary stages: scene description, summarizing and failure explanation from summary. In Liu et al. [2] scene description stage is only applicable for a close set of objects. For their simulator setup, they leverage ground truth from the simulator. For the real world setup special handling is needed such as pictures for each possible state of the object for each object. For object relationship accurate point clouds are required. Our work focuses on scene description that is open vocabulary targeting failure explanation.

Scene Description Stage Inspired by scene graphs, this stage aims to condense the contents in an image of the scene to textual descriptions that convey the objects present in the scene, the relationship between the objects and the states of each of these objects. To generate this summary a VLM is querying to first generate the list of objects in the scene. This list is then passed to an open vocabulary object detector that gives the bounding box for each of these objects. The bounding box is used to extract regions of the image containing the objects which we refer to as regions of interest, ROIs. The ROIs, objects list and the full image are passed to a VLM querying about the relationship between the objects. A similar query is sent for the state of each object. The response from the query for the relationship and the states is concatenated to form the scene description.

Summarizing Since during each time step the relationship between objects does not change significantly and a majority of the time remains the same there is a lot of redundancy in the information. Therefore the scene description is only obtained at key frames where there is a significant change in the scene. This work does not address the problem of identifying the keyframe. Instead, ground truth keyframes are obtained using Liu et al. [2]. Adopting Liu et al. [2] hierarchical summary, two levels of summary are generated: subgoal level and event level. The subgoal summary is the scene descriptions during keyframes that correspond to the end of a subgoal. The event level summary are scene description at key frames that correspond frame where there is a significant change in the scene. The notion of significant change follows Liu et al. [2]'s definition. An event keyframe is defined as the time step when the relationship between objects changes, the state of an object changes, a new object is visible or a previous visible object is no longer visible.

Failure explanation The failure explanation stage is directly adopted from reflect the only modification was the use of Gemini Pro instead of chatGPT 4. Two types of failure were introduced by Liu et al. [2]: planning failure and execution failure. Planning failures are defined as failures due to an incorrect task plan. This means that even if all the subgoals are successfully executed the main task objective will not be achieved. Execution failure occurs when any subgoal fails to be achieved. For example, dropping a pan while navigating constitutes an execution failure. The algorithm is highlighted in ¡alg ref¿. First, each subgoal with the scene description at that time step is passed to the LLM querying whether the task was successful. If the subgoal is satisfied the next subgoal is checked. If a subgoal fails then an execution failure has occurred. The event level summary till that subgoal is passed to the LLM with the query to get the failure explanation and time step of failure. This response is then passed to again to the LLM querying to parse the timestep.

Algorithm 1 Failure explanation from summarized scene descriptions

Input: Event level summary, $S_E = G_{t0}, G_{t1}, ..$ where $t0, t1, ..$ are event key frames
Input: Subgoal level summary, $S_S = G_{t'0}, G_{t'1}, ...$ where $t'1, t'2, ..$ are subgoal key frame
Input: Task plan, A_p
Initialize: $a_{failed} \leftarrow unassigned$, variable to store the failed subgoal
for G_x in S_S and a in A_p **do**
  $LLM([a, G_x],$ Is subgoal satisfied?)
  if Not satisfied **then**
    $a_{failed} \leftarrow a$
    break
  end if
end for
if a_{failed} is assigned **then**
  $S'_E = S_E$ till key frame of a_{failed}
  $explanation = LLM([S'_E, a_{failed}],$ why did the subgoal fail?)
else
  $explanation = LLM([S_S, A_p],$ what is wrong with this plan?)
end if
return $explanation$

5 Experiments

We use different technologies to make different parts of our framework work. We employ the Gemini 1.5 Pro model to generate scene descriptions, we use the YOLO-world[10] model to obtain the ROI's given an image and finally we use the Gemini 1.0 Pro LLM for failure explanation.

Failure Detection: The RoboFail dataset contains a sensor and ground truth simulator information across all time steps for all the individual events and the subgoals. We then generate the event-level and the sub-goal level summaries using REFLECT's framework. We then extract the keyframes for events and subgoals. We then use these keyframes to locate the corresponding image for each of the frames in the video and feed these images into the Gemini 1.5 Pro VLM to output scene graph summaries. These are the open-vocab event-based and subgoal-based summaries that are fed into the LLM for failure explanation.

The necessity for open-vocabulary action primitives emerges, necessitating their association with unseen objects. Since our study primarily focuses on generating open-vocabulary scene descriptions rather than an expansive action space, we maintain a closed-set action space for simulation data by augmenting action primitives for each object of the predefined list. For the data collected in the real world on the Stretch robot, we teleoperate the robot, thus emulating open vocab action primitives.

Baselines

Our method is evaluated against two baselines: REFLECT [2] and a Vision-Language Model (VLM) baseline, representing closed and open vocabulary approaches, respectively. The closed vocabulary REFLECT baseline is utilized without modifications to validate its performance on an expanded dataset.

$$F_{REFLECT} : (A_p, G_0, G_1, ..., G_T) \rightarrow e_f$$

The VLM baseline involves prompting Gemini 1.5 Pro [1]. This model allows a large prompt with millions of tokens corresponding to up to 3 hours of video data. Leveraging this, the entire video of the task execution, $(O_0, O_1, ..., O_T)$ is passed along with the task plan(τ) and Gemini is expected to output the failure explanation.

$$F_{GEMINI} : (A_p, O_0, O_1, ..., O_T) \rightarrow e_f$$

Framework Evaluation

To assess the efficacy of our framework, we will compare its performance against established baselines using two metrics from REFLECT [2]:

- Explanation (Exp): This metric evaluates the accuracy and informativeness of predicted failure explanations, as judged by human evaluators.
- Localization (Loc): It measures the alignment of predicted failure times with actual failure instances within the dataset, considering predictions accurate if they fall within the annotated time ranges.

Dataset

To evaluate our framework and the baseline, we will use the RoboFail dataset, as mentioned in the REFLECT paper [2], which consists of 10 tasks, each with 10 failure cases. To highlight the benefits of using an open-vocabulary approach for scene descriptions, we select two out of the 10 tasks to evaluate our framework for all 10 failure cases. We also evaluate real-life teleoperation data collected on the Stretch robot but this result is not a part of our metrics. Please refer to the Appendix section for detailed further analysis.

6 Results

In the evaluation of Gemini 1.5 Pro conducted across 10 varied kitchen scenarios, findings highlight notable deficiencies in the system's performance. Specifically, Gemini's ability to recognize planning errors was subpar, as evidenced by its reduced accuracy in both failure explanation and localization. Even in instances where Gemini successfully pinpointed the failure step and identified the error, it consistently failed to attribute these failures to planning errors, suggesting a fundamental challenge in distinguishing between different types of errors.

Further examination revealed that, while Gemini could accurately localize the step of failure, it was less adept at providing accurate reasons for these failures. A frequent error involved incorrectly describing objects as "dropped to the floor" for explaining failures, regardless of whether the object was actually mishandled or not even picked up due to obstructions by other objects. This pattern indicates a possible over-reliance on the visual cue of an empty robot gripper to infer failure reasons, leading to a mischaracterization of the failure scenarios.

Additionally, Gemini demonstrated a marked preference for processing instructions in natural language (e.g., "put the pot on the stove") over structured tuple formats (e.g., "(put_on, Pot, Stove)"). This finding underscores the model's inclination towards natural language comprehension, likely a reflection of its training predominantly on text-based data, and highlights a limitation when interacting with inputs presented in less familiar formats.

The analysis of our framework's efficacy, as depicted in the provided table, elucidates several critical differences in performance between our methods (few-shot and zero-shot prompting) and established baselines (Gemini and REFLECT).

Comparison of Zero-Shot Prompting with Gemini

Zero-shot prompting, which abstracts raw observations into scene graphs, shows a distinct approach compared to Gemini. Gemini's performance, with explanation accuracy at 57.5% for execution failures and 45.0% for planning failures, suggests that it retains a moderate level of detail necessary for generating useful failure explanations. In contrast, zero-shot prompting, with lower scores (22.0% for explanation and 44.4% for localization in execution failures), might suffer from the abstraction process. The generation of scene graphs potentially strips away nuanced information that is critical for accurate failure diagnostics, reflecting a loss of detail that can be pivotal in understanding complex interactions in robotic tasks.

Comparison of Zero-Shot Prompting with REFLECT

The comparison with REFLECT, which achieves significantly higher scores (88.4% in explanation and 96.0% in localization for execution failures), highlights the challenges of our open-vocabulary, zero-shot approach. REFLECT's use of closed-set scene graphs provides a structured and less ambiguous method for failure analysis. For instance, in the scenarios involving a salt and pepper shaker and a faucet, the specificity in REFLECT's closed vocabularies helps maintain clear distinctions between similar-looking objects with different functionalities. Our zero-shot method, by relying on open vocabularies, can introduce ambiguities that degrade the system's ability to accurately pinpoint and explain failures.

Comparison between Zero-Shot and Few-Shot Prompting

When comparing zero-shot and few-shot prompting methods, it becomes apparent that few-shot prompting (15.0% in explanation and 30.7% in localization for execution failures) even underperforms relative to zero-shot. This might be due to few-shot prompting's reliance on examples that do not necessarily align closely with the specific task or failure type at hand. The method tends to incorporate irrelevant or only partially relevant information from the example responses, which does not effectively mimic the needed response style or content accuracy. This inadequacy results in responses that fail to adequately address the subtleties and specifics of the queried failures, further emphasizing the importance of tailored and context-specific training data.

Specific Failure Examples:

- Salt-and-Pepper Shaker/Faucet Problem: Gemini faced difficulty in differentiating between objects with similar appearances but different functionalities, leading to incorrect action recommendations.
- Grounding Problem: The absence of a physical embodiment for the robot's gripper in Gemini's training simulations prevented it from developing a realistic understanding of spatial interactions, which affected its ability to ground actions properly in the physical world.
- Dirty Mug Problem: This scenario emphasized the critical need for integrating a holistic task plan within the system's operational framework. Although Gemini identified the mug's dirty condition early in the task sequence, it failed to recognize the subsequent need

for cleaning the mug as part of the preparation for making coffee. This oversight highlights a significant limitation in Gemini's ability to anticipate and sequence multiple subgoals based on the overall task context. Instead of incorporating future steps into its processing, the system prematurely flagged the dirty mug as a stand-alone issue, neglecting the interdependent nature of tasks in realistic settings.

These findings underscore the need for improvements in the model's ability to handle diverse and dynamic real-world scenarios. The integration of more robust scene understanding capabilities, especially in distinguishing between visually similar but functionally different objects, and better grounding in physical interactions, would be crucial steps forward for the development of more reliable autonomous systems.

Accuracy and Informativeness of Failure Explanation				
	Execution failure		Planning failure	
	Explanation	Localization	Explanation	Localization
Gemini 1.5 Pro	57.5%	66.7%	45.0%	50.0%
REFLECT	88.4%	96.0%	84.2%	80.7%
Zero-Shot Prompting	22.0%	44.4%	-	-
Few-Shot Prompting	15.0%	30.7%	-	-

Table 1: Performances on RoboFail Simulated Dataset

7 Conclusion

In this study, we have developed a framework for enhancing failure recovery in robotics through the integration of open-vocabulary scene graph representations. Our framework demonstrates the potential for significantly improving the robot's understanding of its environment, thereby refining its ability to identify, localize, and explain failures effectively.

Our comparative analysis with established baselines like Gemini and REFLECT highlights the challenges and advantages of adopting an open-vocabulary approach. While Gemini provides a moderate level of detail aiding in failure explanation, its lack of processing of the raw inputs (i.e. scene graph representations) results in the lower accuracy of failure localization and explanation, as compared to REFLECT. In contrast, REFLECT's closed-set scene graphs, though more restrictive, offer higher accuracy due to their structured nature, reducing ambiguity in failure analysis.

The zero-shot and few-shot prompting methods explored in our study underscore the difficulties of training with limited context-specific data. The few-shot method, in particular, illustrated a tendency to incorporate irrelevant information, emphasizing the need for a more tailored and precise prompting approach to handle the complex nature of robotic tasks effectively.

Furthermore, specific failure scenarios such as the Salt and Pepper Shaker/Faucet Problem and the Dirty Mug Problem reveal critical areas for improvement in our model's scene understanding capabilities and its ability to anticipate and plan multiple subgoals within a task sequence.

Moving forward, the integration of more robust scene understanding capabilities, particularly in distinguishing between visually similar but functionally different objects, and better grounding in physical interactions will be crucial. These enhancements are expected to develop more reliable and autonomous robotic systems capable of adapting to a broader array of tasks and real-world scenarios. Our continued efforts in refining the open-vocabulary approach and exploring its implications for robotics failure diagnosis and recovery hold promising potential for future advancements in the field.

References

[1] M. Reid, N. Savinov, D. Teplyashin, D. Lepikhin, T. Lillicrap, J.-b. Alayrac, R. Soricut, A. Lazaridou, O. Firat, J. Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[2] Z. Liu, A. Bahety, and S. Song. Reflect: Summarizing robot experiences for failure explanation and correction, 2023.

[3] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision, 2021.

[4] W. Dai, J. Li, D. Li, A. M. H. Tiong, J. Zhao, W. Wang, B. Li, P. Fung, and S. Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning, 2023.

[5] T. He, L. Gao, J. Song, and Y.-F. Li. Towards open-vocabulary scene graph generation with prompt-based finetuning, 2022.

[6] 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*, HRI '21. ACM, Mar. 2021. doi: 10.1145/3434073.3444657. URL `http://dx.doi.org/10.1145/3434073.3444657`.

[7] G. LeMasurier, A. Gautam, Z. Han, J. W. Crandall, and H. A. Yanco. Reactive or proactive? how robots should explain failures. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction (HRI '24)*, page 26, New York, NY, USA, 2024. ACM.

[8] K. Tang, Y. Niu, J. Huang, J. Shi, and H. Zhang. Unbiased scene graph generation from biased training. In *Conference on Computer Vision and Pattern Recognition*, 2020.

[9] Y. Zhang, Y. Pan, T. Yao, R. Huang, T. Mei, and C.-W. Chen. Learning to generate language-supervised and open-vocabulary scene graph using pre-trained visual-semantic space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2915–2924, 2023.

[10] T. Cheng, L. Song, Y. Ge, W. Liu, X. Wang, and Y. Shan. Yolo-world: Real-time open-vocabulary object detection. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2024.

Appendices

A Baseline Results

This section presents example prompts and results for evaluating the performance of Gemini Pro 1.5 on two distinct tasks.

A.1 Task 1: Boil Water

Prompt: The video shows the following task plan being executed for the main goal of "boil water in a pot".

- Pick up pot
- Put pot in sink
- Toggle on faucet
- Ignore

- Toggle off faucet
- Pick up bowl
- Put bowl on stove burner
- Toggle on stove burner
- Terminate

Question: Is the action plan appropriate for reaching the goal? If so, has the above task been executed successfully? At which step did the error occur and how?

Answer: The task plan was not executed successfully. The action plan is appropriate for the main goal of ”boil water in a pot”. The error occurred at step 6, ”Put bowl on stove burner”. Instead of placing the pot on the stove burner, the robot placed the bowl on the stove burner.

Ground Truth Failure Reason: ”A bowl is put on top of the stove burner instead of a pot.”

A.2 Task 2: Put Fruits in Bowl

Prompt: The video shows the following task plan being executed for the main goal of ”put all visible fruits in a bowl”.

- Pick up strawberry
- Put strawberry in bowl
- Pick up pear
- Put pear in bowl
- Pick up carrot
- Put carrot in bowl

Question: Is the action plan appropriate for reaching the goal? If so, has the above task been executed successfully? At which step did the error occur and how?

Answer: The action plan is appropriate for reaching the goal of ”put all visible fruits in bowl”. However, the task was not executed successfully. The error occurred at step 5: ”Pick up carrot”. A carrot is a vegetable, not a fruit, so it should not have been placed in the bowl.

Ground Truth Failure Reason: ”The robot should not put carrot in the bowl since it is not a fruit.”

A.3 Task 3: Secure Objects

Prompt: The video shows the following task plan being executed for the main goal of ”secure objects”.

- Pick up knife
- Put knife in fridge
- Close fridge
- Pick up pear
- Put pear near drawer
- Open drawer
- Pick up pear
- Put pear in drawer
- Close drawer

Question: Is the action plan appropriate for reaching the goal? If so, has the above task been executed successfully? At which step did the error occur and how?

Answer: The action plan is appropriate for reaching the goal of ”secure objects”. However, the task was not executed successfully. The error occurred at step 5, ”Put pear near drawer”. Instead of placing the pear near the drawer, the robot dropped it on the floor.

Ground Truth Failure Reason: ”The knife should be placed inside the drawer and the pear in the fridge. The current placements are flipped.”