FAER: Benchmarking VLMs for Failure-Aware Embodied Reasoning

Anonymous ACL submission

Abstract

Failures are inevitable when embodied agents execute complex tasks. Visual-language models (VLMs) serve as the core component of embodied agents in perceiving the environment and making decisions. Assessing the capabilities of VLMs in detecting and reasoning about failures has become increasingly important. Previous work primarily considered low-level manipulation failures (e.g., 3cm grasp offsets), neglecting high-level failures arising during long-horizon task execution (e.g., object-dropping failure in the "clean room" task) by embodied agents. In this paper, we propose **FAER**, a failure-aware benchmark aiming to evaluate the performance of VLMs in terms of *failure detection*, *failure categorization*, *failure description*, and *failure correction* in long-horizon tasks. FAER comprises 3,323 episodes, spanning 3 scenes, 65 tasks, and 83 objects. We assess the performance of 16 widely utilized VLMs and 4 LLMs for FAER tasks. Experimental results show that nearly all VLMs, even GPT-4o, exhibit limited performance in failure detection with a high false negative rate, meaning that they tend to ignore abnormal events, revealing notable gaps in current models' capacity to effectively handle failures. Dataset and code can be found at https://anonymous.4open.science/r/FAER-3C53.

1 Introduction

Developing embodied agents capable of autonomously executing high-level tasks in open, dynamic environments has long been a key goal in artificial intelligence (Lei et al., 2025; Yang et al., 2025; Durante et al., 2024). During execution in actual scenarios, failures inevitably occur owing to external environmental disturbances or intrinsic limited execution capabilities (Guo et al., 2024; Li et al., 2024a; Duan et al., 2025; Liu et al., 2023), leading to the inability to proceed with subsequent tasks. Therefore, detecting, understanding, and correcting failures encountered during execution have emerged as significant research topics (Duan et al., 2025; Liu et al., 2023; Ding et al., 2023) for embodied agents.

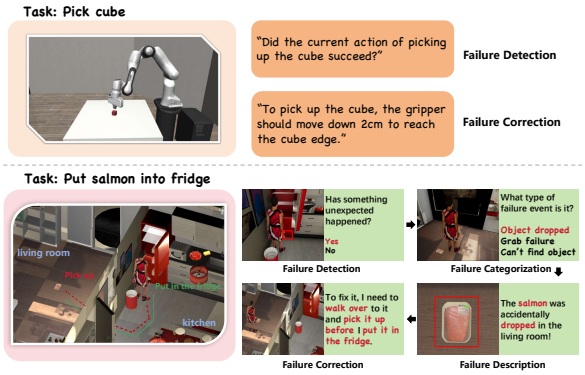

Figure 1: Comparison of our proposed FAER benchmark with previous work. FAER targets failures in long-horizon tasks and covers complex scenes. Besides, FAER provides a series of tasks from failure detection to failure correction, which can comprehensively assess the failure-aware embedded reasoning capability of VLMs.

Recent visual language models (VLMs) (Wang et al., 2024a; Bai et al., 2025; Wang et al., 2024b), trained on vast internet-scale datasets, exhibit exceptional visual perception and reasoning abilities. This allows them to leverage human knowledge extracted from visual data (e.g., images, videos) for embodied task execution. Given these capabilities, VLMs are increasingly integrated as the core component in embodied systems (Guo et al., 2024; Li et al., 2025a; Driess et al., 2023) to process feedback and ground decisions.

Existing work (Duan et al., 2025; Lu et al., 2025; Liu et al., 2023; Ding et al., 2023) has evaluated failure-handling capabilities primarily in constrained, single-scenario settings (e.g., fixed tabletop environments), focusing on low-level manipu-

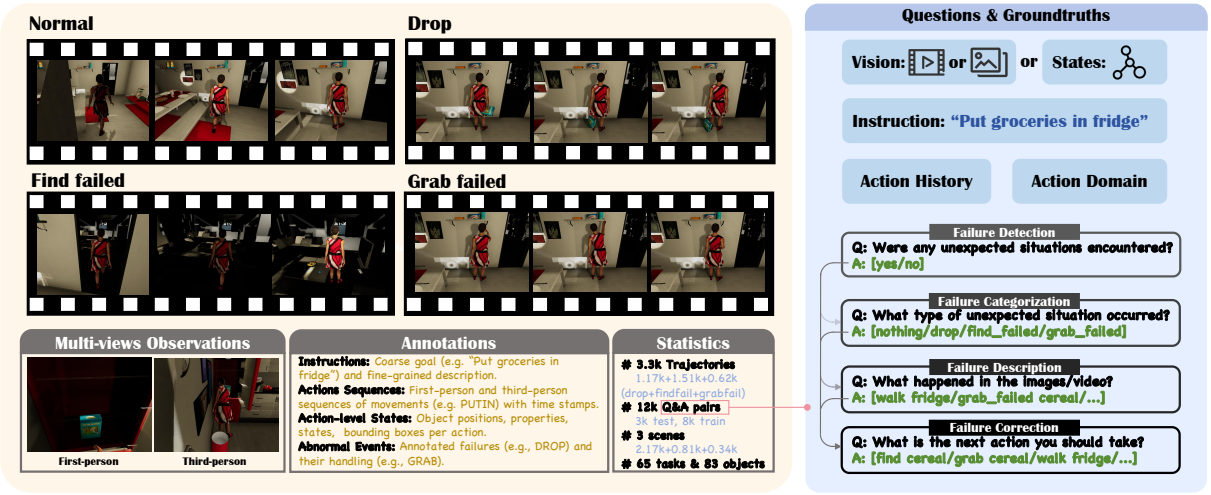

Figure 2: The FAER benchmark encompasses 4 distinct task types addressing 3 common failure events across 3 household scenes. It comprises 4 specific cases: object falling, failed searching, failed grasping, and random normal samples (top left). Tasks follow a cascaded workflow: failure detection, categorization, description, and correction. Critically, each step relies on ground-truth outputs from the preceding stage. Each comprehensively annotated episode includes all 4 tasks and incorporates both first-person and third-person visual perspectives.

lation failures (e.g., 3cm grasp offsets). Their experiments demonstrate that VLMs exhibit limited proficiency in detecting and reasoning about low-level manipulation failures. However, these evaluations neglect complex, high-level failures—such as failing to locate objects—that arise during long-horizon task execution in diverse, realistic scenes (Comparison at Table 5). Consequently, it remains an open question *whether VLMs possess the necessary capabilities for end-to-end perception, understanding, and reasoning about failures directly from visual observations*.

To address this question, this paper constructs a challenging failure-aware benchmark and evaluates widely utilized VLMs on four failure-related tasks. Specifically, we propose FAER (**F**ailure-**A**ware **E**mbodied **R**easoning) benchmark (Figure 1). The FAER benchmark focuses on three typical execution failures: *object drop*, *find failure*, and *grab failure*.

It comprises: (1) 65 task types across 3 scenes, totaling 3,323 episodes with detailed event annotations; (2) Synchronized first-person and third-person visual observations; (3) Four-tiered failure analysis tasks with ground-truth textual labels: *Failure Detection* (identifying failure states), *Failure Categorization* (classifying failure types), *Failure Description* (describing observed scenarios), and *Failure Correction* (generating recovery strategies).

Based on evaluations, we obtain some valuable findings, such as: (1) Failure detection is a bottleneck, as most models show limited performance, with extreme precision-recall imbalance, exhibiting a strong bias to ignore such exception events. (2) Accurate understanding of upstream tasks (detection, categorization, and description) effectively facilitates failure correction. (3) LLMs consistently outperform VLMs by leveraging privileged environmental information in text from simulators.

The main contributions are:

- We propose a failure data generation pipeline `FAER-VirtualHome` which includes failure event injection and realistic rendering of customized failure for constructing scalable failure data.
- We construct a failure-aware evaluation dataset (FAER) for VLMs in embodied long-horizon tasks.
- We assess the performance of failure-aware embodied reasoning of 12 prevailing VLMs/LLMs (e.g., GPT-4o, Gemini-2.5) on FAER tasks.

2 Related Work

Evaluations for Failure Handling Existing tasks are predominantly limited to low-level manipulation scenarios. AHA (Duan et al., 2025) and RoboFAC (Lu et al., 2025) design tasks around binary classification of robotic manipulation (e.g., "Did grasping succeed?") and simple comprehen-

Figure 3: Data generation workflow: (1) rule-based augmentation of `VirtualHome` scripts with 3 failure events (Drop, Grab-Failure, Find-Failure) and recovery actions; (2) multimodal rendering via `FAER-VirtualHome`, generating synchronized first/third-person videos, action sequences, and state snapshots; (3) creation of 12k Q&A pairs for four failure-reasoning tasks, with a test set balanced across failure/normal cases and unseen scenes.

sion queries, which are limited to desktop environments, focusing on fine-grained actuation failures (e.g., imprecise grasping). RoboFail (from REFLECT (Liu et al., 2023)) evaluates failure detection and explanation in controlled AI2THOR (Kolve et al., 2017) scenes and real-world tabletop settings, with tasks centered on single-object interactions (e.g., object inaccessibility). RACER (Dai et al., 2024) further emphasizes low-level skill assessment, focusing on whether physical interactions (e.g., picking, placing) are executed correctly. These tasks overlook the high-level reasoning capabilities required for complex, long-horizon tasks across various scenarios. Unlike previous work, this paper evaluates failure handling in long-horizon tasks, which not only need to account for changes in the scene but also require more complex failure corrections.

Resources for Failure Handling Evaluation AHA (Duan et al., 2025) introduces 7 manipulation-related failure modes via keyframe modification, object replacement, and reordering through `FailGen`. Its dataset combines multi-view time-series visual observations with queries to form 49k failure-question pairs, focusing on manipulation effects. However, its tasks are limited to binary Q&A and simple explanations for manipulation steps, lacking designs for failure correction to handle open-world failures and remaining confined to fixed operation scenarios. Similarly, RoboFAC (Lu et al., 2025) constructs a large-scale dataset with 78k QA pairs across 16 tasks to address failure analysis and correction, yet it primarily targets specific robotic manipulation failures (e.g., motion planning and execution control errors) rather than high-level semantic deviations in open household activities. The RoboFail dataset (Liu et al., 2023) includes 100 simulated examples and 30 real-world examples, consisting of visual observations, audio, robot states, etc. Its limitations lie in the difficulty of scaling due to manual collection and a similar constraint by the assumption of static environments. COWP (Ding et al., 2023), based on 12 categories of food and beverage-related tasks from ActivityPrograms (Puig et al., 2018), includes potential situations proposed by humans that might hinder task completion (e.g., "Glass is broken", "Faucet has no water"), thereby obtaining 1085 open-world situations. Its limitation is that it does not consider VLMs' detection and understanding of these situations. Unlike prior work, our FAER dataset offers a more fine-grained evaluation of failure-aware reasoning, spanning detection, categorization, description, and correction across diverse scenarios, with both first- and third-person perspectives provided.

3 FAER Bench

3.1 Data Construction

We propose an automated data generation method to construct a multimodal reasoning dataset (Figure 2) enriched with various failure events and their corresponding recovery actions. Specifically, our data generation pipeline consists of two primary steps: (1) automatic insertion of failure events and recovery actions based on a rule-based augmentation framework, and (2) generation of corresponding multimodal data grounded in the augmented task scripts (Figure 3).

3.1.1 Augmentation of Failure Events and Recovery Actions

We enhance the `VirtualHome` dataset by collecting original task scripts and using an LLM to generate additional ones for diversity. We then introduce three failure events programmatically, inserting them at semantically appropriate points with recovery actions: (1) *Drop Event*: Randomly inserts a "drop" action between "grab" and "put" followed by a recovery (regrab or walk + grab). (2) *Find-Failure Event*: Inserts incorrect navigation before object interaction, corrected by a "find" action. (3) *Grab-Failure Event*: Marks a failed "grab" with "grab_false" then retries before the "put". A rule-based engine automates this augmentation, ensuring logical consistency and executable sequences.

3.1.2 Multimodal Data Generation

To generate multimodal data that reflects the effects of failure events—including videos, action sequences, and environment state information—this paper develops an enhanced simulation environment named `FAER-VirtualHome`. Built upon the original `VirtualHome` platform, `FAER-VirtualHome` extends its functionality to overcome the limitation of not supporting failure event simulation.

Compared with the original `VirtualHome`, `FAER-VirtualHome` offers several key advantages: (1) It supports the execution of three additional failure event types: **drop**, **find_failure**, and **grab_failure**, expanding the original action set. (2) It enables visual rendering of these failure events, thereby providing richer multimodal perceptual data. (3) It explicitly records and displays environment state changes before and after each failure event (Table 1). (4) It allows failure events to be triggered with configurable probabilities, simulating their random occurrence in realistic scenarios, and enabling the evaluation of agents' perceptual robustness and dynamic replanning capabilities.

Using `FAER-VirtualHome`, we generate a series of planning task instances involving both failure events and corresponding recovery actions. Each instance includes a natural language task instruction, a video trajectory with injected failure events, and environment state snapshots recorded before and after key action executions.

3.1.3 Evaluation Data Construction

Using multimodal data with detailed annotations, we construct the evaluation set. The process can be divided into the following steps: (1) data quality control; (2) construction of prompt templates; (3) groundtruth generation; (4) data splitting.

Failure type	State transition
Drop	(object, agent, in) → (object, floor, on)
Find failure	current state → (random_room, agent, in)
Grab failure	unaffected

Table 1: State transitions based on failure events.

Data quality control. To refine the dataset and retain high-quality, commonsense observations, this paper develops an interface tool (Figure 5) for manual quality checks (Table 6). More details can be found in the Appendix 6.

Construction of prompt templates. To standardize input formats for the four reasoning tasks, we design task-specific prompt templates. Each template integrates the overall task instruction, historical action records, and current observational data (images/videos), with output constraints tailored to task requirements (e.g., binary "yes/no" for detection, specific failure types for categorization). Prompt examples are shown in the Appendix.

Groundtruth generation. Ground truth labels were derived from augmented task scripts: failure detection labels ("yes"/"no") indicate the presence of failure events; categorization labels (e.g., "drop" map to predefined types); descriptions follow "verb + object" structures based on scripted actions; corrections use recovery actions inserted in scripts.

Data splitting. To ensure more objective evaluation results, positive (i.e., samples containing failures) and negative samples are balanced by incorporating an equal number of normal samples, which are the prefix frames of the failure frames. Additionally, we expand the dataset by including identical segments with only perspective (e.g., first-person vs. third-person) altered. These two strategies increased the total number of events to 12k, nearly 4 times the original size. Leveraging the expanded dataset, we construct the test set by (1) retaining all samples from the entirely unseen scene to ensure coverage of unseen-scene generalizability, and (2) randomly selecting a subset from other scenes. This stratified partitioning strategy ensures the test set evaluates both novel scenario robustness (Scene 3) and cross-scene adaptability to previously observed contexts, yielding a final test set of 3k samples.

3.2 Failure-Aware Embodied Reasoning Dataset

The FAER benchmark is designed to comprehensively cover failure-aware embodied reasoning in household scenarios, with its overall composition visualized in Figure 2 and detailed statistics summarized in Table 2. Specifically, the dataset spans 3 common household scenes and targets 3 typical failure events, encompassing 4 primary task types (failure detection, categorization, description, and correction) and 4 specific case categories: object dropping, failed finding, failed grasping, and random normal samples (serving as negative controls). For more detailed object categories, please refer to the Appendix. The FAER dataset not only contains 3,323 trajectories with multi-view visual data and detailed annotation files, but also includes about 12,000 Q&A pairs constructed based on these resources, with a subset of about 3,000 randomly selected as the **FAER-test**.

Dataset Features The FAER dataset is meticulously constructed to enable a fine-grained evaluation of embodied reasoning under failure conditions. A key feature is the inclusion of synchronized first-person and third-person visual perspectives for each event, allowing for a comprehensive assessment of a model's perceptual abilities from different viewpoints. Each episode within the dataset is annotated to support the four-tiered task hierarchy (Detection, Categorization, Description, and Correction). Ground-truth labels are derived directly from the augmented task scripts, providing clear and objective targets—from binary labels (yes/no) for detection to structured verb + object phrases for description and correction. To ensure a robust and unbiased evaluation, the dataset is carefully balanced with an equal number of normal (non-failure) and abnormal (failure) samples. Furthermore, the test split is specifically designed to measure generalization by including samples from a completely unseen scene (Scene 3) alongside data from previously seen environments. Table 2 details the dataset statistics, including a total of 3,323 trajectories, covering 65 tasks and involving 83 objects, along with counts of different failure types.

Failure	Num	Scene1	Scene2	Scene3	Task	Object
Drop	1,176	732	255	189	57	77
Find failure	1,518	1,028	385	105	52	78
Grab failure	629	411	172	46	41	73
Total	3,323	2,171	812	340	65	83

Table 2: Dataset statistics.

3.3 Failure-Aware Embodied Reasoning Tasks

Formulation We formalize a framework for an agent's interaction with the environment. The agent receives a sequence of multi-modal observations $O = (o_{t_1}, o_{t_2}, \ldots, o_{t_n})$, where o_{t_i} denotes observation data (e.g., images, video, text) at timestep t_i. Specifically, in the image modality, we uniformly sample 4 images of the current event as observations; in the video modality, observations consist of a video segment capturing the current state; in the text modality, observations are replaced by text-described states. Additionally, the input includes the task goal G, action history H and optional action domain A. Together with an auxiliary text query Q, our tasks are formulated as a four-tuple (O, G, H, Q, A). The output consists of short phrase-based tokens (e.g., "yes", "grab cup").

Tasks FAER defines four complementary tasks: Failure Detection, Failure Categorization, Failure Description, and Failure Correction. Collectively, these tasks are designed to comprehensively evaluate agents' perceptual and reasoning capabilities (Cheng et al., 2024; Jia et al., 2022).

Failure Detection evaluates state estimation and failure recognition by mapping observations to a binary decision, $O_1 = \{yes, no\}$, testing whether the agent can identify abnormal execution states in dynamic environments ($O_t \rightarrow S_t$).

Failure Categorization requires predicting the failure type by abstracting perceptual evidence into a discrete category, framed as inverse action modeling over state transitions, $\{O_{t-1}, O_t\} \rightarrow A_{t-1}$.

Failure Description further probes inverse action modeling by requiring a structured, grounded description of the failure event in the format "event verb + object" (e.g., `grab_fail apple`, `walk fridge`, `find TV`). This tests whether models can compress visual and contextual cues into task-relevant, semantically grounded event representations, reflecting context awareness and instruction following (Zhang et al., 2025).

Failure Correction evaluates policy prediction under failures by generating the next action conditioned on recent interaction history, $\{O_{t-k}, A_{t-k} \ldots, O_{t-1}, A_{t-1}, O_t\} \rightarrow A_t$, with outputs constrained to "verb + object" or verb-only forms (e.g., "grab apple", "find TV", "standup").

Model	Failure Detection				Failure Categorization				Failure Description				Failure Correction				Avg. Acc.
	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1	
Image																	
InternVL3-2B-Instruct	50.01	0.00	0.00	0.00	53.90	**100.00**	7.78	14.44	24.41	30.00	38.40	33.68	37.70	36.27	32.52	34.29	41.51
InternVL3-8B-Instruct	50.24	**100.00**	0.46	0.92	54.10	55.48	41.40	47.42	35.05	**58.32**	59.57	58.94	55.03	55.31	52.19	53.70	48.61
InternVL3-38B-Instruct	51.51	77.08	4.26	8.07	57.30	97.02	15.05	26.06	40.18	29.48	14.12	19.09	56.58	92.53	14.30	24.77	51.39
Qwen3-VL-8B-Instruct	57.73	77.01	22.02	34.25	56.09	92.03	13.32	23.27	26.20	18.66	14.18	16.11	50.59	54.54	6.92	12.28	47.65
Qwen2.5-VL-7B-Instruct	50.12	**100.00**	0.23	0.46	50.10	**100.00**	0.17	0.34	10.66	0.57	0.46	0.51	50.41	51.03	19.89	28.62	40.32
Qwen2.5-VL-32B-Instruct	55.63	95.77	11.76	20.95	57.62	96.15	15.85	27.22	29.14	20.47	24.05	13.88	56.41	**75.45**	18.97	30.32	51.26
Qwen2.5-VL-72B-Instruct	62.29	72.56	39.50	51.15	66.70	80.69	43.88	56.85	36.20	37.44	41.17	39.21	62.81	66.54	51.49	58.06	57.00
Llava-v1.6-mistral-7B-hf	55.69	53.64	83.50	65.32	50.44	68.29	1.61	3.15	1.87	1.34	1.32	1.33	30.52	7.64	3.51	4.81	34.63
Llama-3.2-11B-Vision-Instruct	51.28	50.68	**93.31**	65.68	34.30	23.56	14.01	17.57	3.48	2.97	2.94	2.95	35.57	32.24	26.23	28.93	31.16
Gemini-2.5-flash	58.23	77.77	23.01	35.51	68.49	67.35	71.74	69.48	28.25	27.38	26.35	26.86	71.92	68.25	81.94	74.47	56.72
Gemini-2.5-flash-thinking	60.04	63.84	46.37	53.72	71.75	74.84	65.82	70.04	40.51	40.22	36.12	38.06	72.08	73.55	69.01	71.21	61.10
Gemini-2.5-pro	59.38	71.95	46.00	56.12	74.66	74.49	74.24	74.36	31.69	33.08	35.13	34.07	74.05	70.28	83.33	76.25	59.95
Gemini-3-flash-preview	**70.94**	72.57	67.30	69.84	**81.38**	75.96	**91.80**	**83.13**	43.99	37.86	27.43	31.81	69.30	59.67	82.95	69.41	66.40
GPT-4o	70.62	70.08	71.90	**70.98**	74.71	74.49	74.97	74.73	**60.46**	**58.32**	**72.93**	**64.81**	**77.47**	74.24	**84.14**	**78.88**	**70.81**
Video																	
MiniCPM-V-4	49.89	0.00	0.00	0.00	**52.40**	60.04	14.3	**23.10**	4.46	2.46	2.36	2.41	41.59	7.8	1.55	2.59	37.09
LLaVA-NeXT-Video-7B-hf	37.67	20.92	**8.88**	**12.47**	43.15	30.42	**10.66**	15.79	1.75	3.18	3.28	3.23	19.77	14.96	12.91	13.86	25.59
InternVL3-2B-Instruct	50.01	0.00	0.00	0.00	51.80	**98.43**	3.63	7.00	15.53	19.07	21.28	20.11	40.81	34.99	21.45	26.60	39.54
InternVL3-8B-Instruct	**50.61**	**64.38**	2.71	5.20	49.63	46.28	4.67	8.48	**21.79**	**25.06**	**28.37**	**26.61**	**42.75**	**38.09**	**23.24**	**28.87**	**41.20**
Text																	
Meta-Llama-3-8B-Instruct	60.79	56.46	**94.23**	70.61	44.59	45.45	54.26	49.47	22.91	29.44	38.81	33.48	44.42	46.12	66.55	54.48	43.18
Meta-Llama-3-70B-Instruct	**73.91**	**74.71**	72.26	**73.46**	71.69	73.44	**67.93**	70.58	30.84	37.28	56.22	44.83	60.82	59.51	67.64	63.31	59.32
Qwen2.5-7B-Instruct	55.46	73.80	16.89	27.49	63.36	76.94	38.11	50.97	26.75	31.18	38.58	34.49	48.02	48.21	53.69	50.80	48.40
Qwen2.5-72B-Instruct	73.62	74.13	72.54	73.33	**76.90**	**83.01**	67.64	**74.54**	**38.57**	**42.60**	**65.91**	**51.75**	**77.39**	**80.56**	**72.20**	**76.15**	**66.62**
Avg.	56.12	61.24	35.32	36.16	59.32	73.59	36.95	40.36	27.94	26.65	29.50	27.19	53.46	52.17	38.48	43.76	49.07

Table 3: Quantitative evaluation of 14 models across image, video, and text modalities. The four tasks correspond to Failure Detection, Failure Categorization (Type Judge), Failure Description (What Happened), and Failure Correction (Next Action). Metrics include Accuracy (Acc.), Precision (Pre.), Recall (Rec.), and F1. Best results are in **bold**, second best are underlined.

This task assesses closed-loop recovery by requiring targeted corrective actions derived from contextual semantics and state transitions.

4 Experimental Setup

We use **FAER-test** as test set, which includes four designed tasks in egocentric and exocentric views with negative and positive samples. We evaluate models on the four tasks mentioned above in *self-cascade* mode, where the previous responses from the models are included in the input for the current task.

4.1 Baselines

We evaluate a comprehensive set of models spanning image, video, and text modalities to systematically assess failure reasoning capabilities. For the **image modality**, we include open-source models such as the Qwen2.5-VL series (7B, 32B, 72B) (Bai et al., 2025), Qwen3-VL-8B-Instruct, the InternVL3 series (2B, 8B, 38B) (Zhu et al., 2025), Llava-v1.6-mistral-7B-hf (Liu et al., 2024), and Llama-3.2-11B-Vision-Instruct. We also evaluate proprietary models including the Gemini-2.5 family (Flash, Pro, Flash-Thinking), Gemini-3-flash-preview, and GPT-4o. For the **video modality**, we test MiniCPM-V-4, LLaVA-NeXT-Video-7B-hf, and the InternVL3 series (2B, 8B). Finally, for the **text modality** (using ground-truth state descriptions), we employ Qwen2.5-Instruct (7B, 72B) (Team, 2024) and Meta-Llama-3-Instruct (8B, 70B) (AI@Meta, 2024). This diverse selection allows us to analyze the impacts of parameter scale, accessibility (open vs. closed), functional design (reasoning vs. non-reasoning), and input modality on performance.

4.2 Evaluation Metrics

We use accuracy—calculated via token-level comparison between model-generated phrases and groundtruth—as the primary metric, with precision, recall, and F1-score also reported.

4.3 Prompts

To systematically evaluate the embodied reasoning capabilities of VLMs in handling failure events during long-horizon task execution, we design a series of prompts corresponding to four reasoning tasks: failure detection (Figure 10), failure categorization (Figure 11), failure description (Figure 12), and failure correction (Figure 13). The images used for reasoning consist of **4** frames uniformly sampled from the latest short **sub-segment** of the video, while the video input corresponds to the current sub-segment of the full video, processed with a maximum of **32** frames. More details are shown in the Appendix 6.

5 Results and Analysis

5.1 Main Results and Findings

The main experimental results are shown in Table 3. We systematically evaluate four task categories: failure detection, failure categorization, failure description, and failure correction. We report statistical performance metrics: accuracy, precision, recall and F1. Overall, GPT-4o outperforms all models across modalities, achieving the highest accuracies in all four failure tasks, with Gemini-2.5-flash-thinking ranking second. We have the following key findings:

Most models exhibit a bias between precision and recall. Most models show high precision but very low recall, with the Qwen and Intern series being particularly notable. For example, in the failure detection task, Qwen2.5-VL-7B-Instruct achieves 100% precision (note that if the model only answers a few "yes" responses and hits the target, its precision rate is 100%) but only 0.23% recall, which demonstrates an **optimism bias**. The same phenomenon is observed in the Intern series and Gemini-2.5-flash, indicating that these models cannot effectively identify positive samples. In contrast, Llama series models display sensitivity with high recall but low precision, suggesting that they tend to return positive predictions. In the failure categorization task, nearly all models except the GPT series also exhibit this pattern, indicating their inability to properly classify positive samples. This bias is somewhat mitigated in the failure description and failure correction tasks. This is because, in the experimental setup, later tasks are provided with more reasoning conditions, which improves their performance.

Failure detection and failure description are tougher. Almost all models, even GPT-4o, exhibit limited performance in both the failure detection task and the failure description task. In terms of the average performance across all models, the scores for failure detection and failure description are 56.12% and 27.94% respectively. In the failure detection task, models are required to detect whether a failure has occurred. This places higher demands on the models' state estimation capabilities, which involve understanding the current task target and comparing it with the actual state. From the experiments, most models lack such perceptual ability, achieving an accuracy of approximately 50%. This indicates that the models' responses have polarity characteristics (it should be noted that if all answers are consistent, the accuracy will also reach 50%, because the number of negative samples and positive samples in the test is almost equal). In failure description task, models are ask to output "verb (+ object)", which imposes higher demands on understanding visual detail and main goal.

Accurate upstream reasoning facilitates downstream tasks. To investigate whether correct reasoning in upstream tasks genuinely benefits downstream performance, we introduced ground-truth labels as clean prior information for downstream tasks, replacing the model's self-generated responses. The results, presented in Table 8 of the Appendix, demonstrate a consistent improvement (approximately 15%) across downstream tasks. This suggests that a model's intrinsic understanding of the failure directly influences the quality of its final decision-making.

Textual models with textual states perform more robustly than visual models. We observe that textual models from the same series (e.g., Qwen2.5) achieve better average performance (+8.85%) than visual models in almost all tasks. For textual model experiments, we replace visual images in prompts with state descriptions derived from privileged information, testing Qwen2.5-7B-Instruct, Meta-Llama-3-8B-Instruct and their larger-parameter variants (Table 3). We find that language models utilizing text-described states outperform those using raw visual information: for instance, Qwen2.5-72B-Instruct has a higher average accuracy (66.62%) than Qwen2.5-VL-72B-Instruct (57.00%), with improvements across the first three tasks. This may indicate that visual models suffer from information distortion.

Additional findings are presented in the Ap-

Model	Failure Detection				Failure Categorization				Failure Description				Failure Correction				Avg. Acc.
	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1	
GPT-4o (4 images)	70.62	70.08	71.90	70.98	**88.47**	88.39	88.54	**88.47**	**82.92**	75.40	97.75	**85.13**	99.71	99.83	99.60	99.71	**85.43**
GPT-4o (2 images)	67.82	68.48	66.03	67.23	88.24	89.97	86.07	87.98	81.39	73.66	97.75	84.01	99.94	99.94	99.94	99.94	84.34
GPT-4o (w/o images)	54.16	57.34	32.41	41.41	83.71	98.99	68.07	80.67	64.91	60.69	84.60	70.68	**99.97**	99.94	100.00	**99.97**	75.69
GPT-4o (w/ text states)	**72.18**	66.59	88.98	**76.17**	87.78	92.52	82.20	87.06	71.72	65.81	90.36	76.16	99.88	99.76	100.00	99.88	82.89
GPT-4o (independent)	69.45	70.04	68.80	69.41	70.24	65.94	84.59	74.11	59.86	56.15	89.91	69.13	75.99	79.06	70.78	74.69	68.89

Table 4: Image and prompt mode ablation studies on GPT-4o. Image ablation includes image count (4, 2), absence of images (w/o images), and inclusion of text states (w/ text states). Prompt ablation includes the independent prompt mode (independent). All experiments (except independent) use the *groundtruth-cascade* setting, where ground truth from upstream tasks is provided as input to mitigate error propagation. We report Accuracy (Acc.), Precision (Pre.), Recall (Rec.), and F1 score.

pendix 6.

5.2 Further Analysis of GPT-4o

Since GPT-4o performs best across all tasks, we use it as the upper bound for embodied agents' failure handling. To mitigate the impact of cumulative errors, we replaced the model-generated answers from preceding tasks with the ground truth during the inference of the current task. This setting, which we term *groundtruth-cascade*, is applied in the experiments shown in Tables 8, 9, 4, and 10.

Effects of Images To further investigate the extent of leveraging visual perception information, we design ablation experiments for images and selected the best-performing model for testing, as shown in Table 4. The results show: (1) GPT-4o with 4 images achieves the best overall performance (highest AvgAcc of 85.43%); (2) the performance drops significantly when images are absent (*w/o images*, lowest AvgAcc of 75.69%); (3) incorporating text-described states improves performance compared to the image-absent setting, especially in failure detection, demonstrating the utility of the text modality in complementing visual inputs.

Effects of Prompts We also evaluate GPT-4o in the *independent* prompt mode, which removes the answers of previous tasks (e.g., remove "Based on your prior analysis, ..." in Figure 7). This means that in a single query, the model has no access to information from previous task stages and generates responses solely based on the current image and query. As shown in Table 4, there are significant performance drops in downstream tasks and overall average performance (-16.54%). This indicates that in failure-aware tasks, effective reasoning conditions enable the model to achieve better performance, rather than outputting answers based on a single query. Specifically, this cascaded reasoning paradigm facilitates performance gains in downstream tasks, where the magnitude of improvement is positively correlated with the accuracy of the provided auxiliary information.

Error Analysis By analyzing the statistical distribution of GPT-4o errors and error examples, we reveal the potential challenges embodied agents face in failure handling. Among the errors that occurred, GPT-4o's errors are mainly concentrated in failure detection with roughly equal "false_no" and "false_yes" (Figure 25 in the Appendix). And in the failure categorization task, GPT-4o often misclassifies failures as "find_failed", which might imply that the concept of "find_failed" has a relatively broad meaning (e.g., the model has insufficient understanding of scene-object relevance in long-horizon multi-room tasks). More error examples and analysis can be found in the Appendix 6.

6 Conclusion

In this paper, we introduce the `FAER-VirtualHome` pipeline for large-scale generation of failure-event-rich embodied task data, constructed the first FAER dataset focusing on high-level long-horizon task failures. We comprehensively evaluated 12 mainstream models to uncover their failure perception and handling limitations. We conclude that current VLMs have a limited ability to detect failures from visual input during embodied tasks. The FAER benchmark highlights this issue and offers an assessment tool and research direction for improving VLM robustness and environmental perception. Looking ahead, future work could include enhancing the interpretability of multi-modal decision-making in VLMs to better understand how visual and language cues are integrated for failure handling.

Limitations

FAER provides a structured benchmark for evaluating failure detection, failure categorization, failure description, and failure correction in long-horizon embodied tasks under multiple input modalities. Nevertheless, several limitations point to promising directions for future research:

Model coverage While our evaluation includes a broad range of current vision-language models (VLMs) and large language models (LLMs), it does not encompass every recently released model. Given the rapid pace of model development, we plan to regularly evaluate new models on FAER and its future iterations.

Input modalities and temporal context We evaluate three representative input settings: images, videos, and text containing environmental state information, each augmented with the full history of interaction actions. These settings enable controlled comparisons between perception-driven and state-based reasoning. However, they do not exhaust all possible information conditions relevant to failure perception. In long-horizon embodied tasks, failures may depend on fine-grained temporal cues and interaction history. Future work could systematically vary the amount and placement of temporal evidence and investigate more flexible mechanisms for integrating historical context into failure diagnosis.

Ethical Considerations

This research relies exclusively on publicly available datasets and open-source models (or publicly accessible APIs). The prompt templates used are also open and available. The visual data is synthetically generated within a simulated household setting. Consequently, our dataset does not contain any Personally Identifiable Information (PII) or biometric data of real human subjects and is intended solely for academic research purposes.

A data quality verification process was implemented to filter rendering artifacts. This process was conducted entirely by the co-authors of this paper, who are Master's students in Computer Science and Technology.

References

AI@Meta. 2024. Llama 3 model card.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. Qwen2.5-vl technical report. abs/2502.13923.

Sijie Cheng, Zhicheng Guo, Jingwen Wu, Kechen Fang, Peng Li, Huaping Liu, and Yang Liu. 2024. Egothink: Evaluating first-person perspective thinking capability of vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14291–14302.

Yinpei Dai, Jayjun Lee, Nima Fazeli, and Joyce Chai. 2024. Racer: Rich language-guided failure recovery policies for imitation learning. *arXiv preprint arXiv:2409.14674*.

Yan Ding, Xiaohan Zhang, Saeid Amiri, Nieqing Cao, Hao Yang, Andy Kaminski, Chad Esselink, and Shiqi Zhang. 2023. Integrating action knowledge and llms for task planning and situation handling in open worlds. *Auton. Robots*, 47(8):981–997.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, and 1 others. 2023. Palm-e: an embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, pages 8469–8488.

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*.

Zane Durante, Qiuyuan Huang, Naoki Wake, Ran Gong, Jae Sung Park, Bidipta Sarkar, Rohan Taori, Yusuke Noda, Demetri Terzopoulos, Yejin Choi, Katsushi Ikeuchi, Hoi Vo, Li Fei-Fei, and Jianfeng Gao. 2024. Agent ai: Surveying the horizons of multimodal interaction. abs/2401.03568.

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)*, pages 12124–12131. IEEE.

Baoxiong Jia, Ting Lei, Song-Chun Zhu, and Siyuan Huang. 2022. Egotaskqa: Understanding human tasks in egocentric videos. In *Advances in Neural Information Processing Systems*, volume 35, pages 3343–3360. Curran Associates, Inc.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, Aniruddha Kembhavi, Abhinav Gupta, and Ali Farhadi. 2017. Ai2-thor: an interactive 3d environment for visual ai. abs/1712.05474.

Mingcong Lei, Ge Wang, Yiming Zhao, Zhixin Mai, Qing Zhao, Yao Guo, Zhen Li, Shuguang Cui, Yatong Han, and Jinke Ren. 2025. Clea: Closed-loop embodied agent for enhancing task execution in dynamic environments. abs/2503.00729.

Boyi Li, Philipp Wu, Pieter Abbeel, and Jitendra Malik. 2025a. Interactive task planning with language models. *Transactions on Machine Learning Research*.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, Mona Anvari, Minjune Hwang, Manasi Sharma, Arman Aydin, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R Matthews, and 9 others. 2023. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 80–93. PMLR.

Chenxuan Li, Jiaming Liu, Guanqun Wang, Xiaoqi Li, Sixiang Chen, Liang Heng, Chuyan Xiong, Jiaxin Ge, Renrui Zhang, Kaichen Zhou, and Shanghang Zhang. 2024a. A self-correcting vision-language-action model for fast and slow system manipulation. abs/2405.17418.

Dongping Li, Tielong Cai, Tianci Tang, Wenhao Chai, Katherine Rose Driggs-Campbell, and Gaoang Wang. 2025b. Emmoe: A comprehensive benchmark for embodied mobile manipulation in open environments. abs/2503.08604.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li Erran Li, Ruohan Zhang, Weiyu Liu, Percy Liang, Li Fei-Fei, Jiayuan Mao, and Jiajun Wu. 2024b. Embodied agent interface: Benchmarking LLMs for embodied decision making. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 26296–26306.

Zeyi Liu, Arpit Bahety, and Shuran Song. 2023. Reflect: Summarizing robot experiences for failure explanation and correction. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 3468–3484. PMLR.

Weifeng Lu, Minghao Ye, Zewei Ye, Ruihan Tao, Shuo Yang, and Bo Zhao. 2025. Robofac: A comprehensive framework for robotic failure analysis and correction. *Preprint*, arXiv:2505.12224.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. 2018. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024a. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. abs/2409.12191.

Weiyun Wang, Zhe Chen, Wenhai Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Jinguo Zhu, Xizhou Zhu, Lewei Lu, Yu Qiao, and Jifeng Dai. 2024b. Enhancing the reasoning ability of multimodal large language models via mixed preference optimization. abs/2411.10442.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, and 1 others. 2025. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. In *Forty-second International Conference on Machine Learning*.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Akshara Prabhakar, Haolin Chen, Zhiwei Liu, Yihao Feng, Tulika Awalgaonkar, Rithesh Murthy, Eric Hu, Zeyuan Chen, Ran Xu, Juan Carlos Niebles, Shelby Heinecke, and 3 others. 2025. Xlam: A family of large action models to empower ai agent systems. pages 11583–11597.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Xuehui Wang, Yue Cao, Yangzhou Liu, Xingguang Wei, Hongjie Zhang, Haomin Wang, Weiye Xu, and 32 others. 2025. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models.

Appendix

Societal Impact and Potential Risks

Our work aims to enhance the robustness of embodied agents. While improving failure detection contributes to safer autonomous systems, we acknowledge several risks:

- *Safety and Sim-to-Real Gap:* Our evaluation is conducted in a simulation. A "successful correction" in FAER does not guarantee physical safety in the real world. Over-reliance on VLM reasoning without physical constraints could lead to real-world accidents if deployed directly onto hardware.

- *Model Biases:* Since we evaluate off-the-shelf VLMs trained on internet-scale data, the reasoning results may inherit societal biases.

Case Study

To illustrate how our failure augmentation and evaluation framework operates, we present a case study (Figure 4) based on the task "put cereal in the fridge." The original task script is shown in the figure. The system inserts a failure event, "[DROP] cereal", immediately after the action [GRAB] cereal. After the agent completes [FIND] fridge, the recovery sequence is generated, consisting of [FIND] cereal and [GRAB] cereal. This is necessary because the agent has moved away from the drop location during the search for the fridge and must re-approach the cereal before it can be re-grasped. The full augmented script with the injected failure and corresponding recovery actions is shown in the figure. We execute the updated script within the `FAER-VirtualHome` environment and collect multimodal data, including both first-person and third-person view images, the frame indices corresponding to the start and end of each action, and the complete environment state before and after each action (e.g., object positions, states). This data provides rich perceptual and semantic context for downstream tasks such as behavior understanding and failure evaluation.

Data Quality Control

To mitigate potential rendering artifacts, such as poor image quality or unnatural character deformation, we implemented a data verification process. The verification was performed by two volunteer annotators (graduate students) using a custom-built interface (Figure 5). The primary objective was to confirm that each image provides sufficient visual information to accurately reflect the specific failure type. Based on the reliability criteria detailed in Table 6, low-quality samples, particularly those suffering from severe occlusion or inadequate illumination, were filtered out.

Experiments

Input Configuration Our experiments were conducted within the Swift framework. We evaluated the existing models across three data modalities: images, video, and text. For the image modality, each model was provided with four frames uniformly sampled from the video segment corresponding to the anomaly event. For the video modality, we used the video processing engine from the Swift framework to handle the input, with a maximum of 32 frames. For the text modality, we supplied extracted textual descriptions of the environment states surrounding the anomaly event—specifically, those occurring before and after the event—as input to the models.

Prompts Figures 10, 11, 12, and 13 show the visual observations of the queries for failure detection, failure categorization, failure description, and failure correction, respectively.

We utilized text states from simulator interfaces (e.g., "character holds_rh remotecontrol" indicates a right-hand grip), with visual observations included for comparison. Figure 6, 11, 12, and 13 show the queries separately, with states in text for failure detection, failure categorization, failure description, and failure correction.

Results Table 8 presents an oracle experiment, feeding ground-truth from previous tasks as input to establish the models' performance upper-bound. Tables 9 and 10 respectively show the performance of different models on normal and abnormal samples, as well as a comparison of the models' capabilities in handling different types of failure events.

Other models on failure We also evaluated other models related to failure detection, such as RoboFAC (Lu et al., 2025). The experimental results for RoboFAC are presented in Table 7, showing mostly zero performance. On our benchmark, RoboFAC consistently outputs the same repetitive content, resulting in poor performance in the evaluation.

Tasks and Objects

All tasks and objects are listed in Table 11 and 12.

Results

We also analyze model architecture and scale impact on failure-aware reasoning (Table 3).

Reasoning-enhanced models consistently outperform their standard counterparts. Gemini-2.5-flash-thinking achieves an average accuracy of

61.10%, surpassing the standard Gemini-2.5-flash (56.72%) by a significant margin. This suggests that models optimized for chain-of-thought (CoT) reasoning are better equipped to handle the multi-step logic required for failure categorization and correction, effectively bridging the gap between perception and actionable planning.

Larger models demonstrate superior performance. Across both visual and textual modalities, increasing parameter size yields consistent gains. For instance, in the text modality, Qwen2.5-72B-Instruct (66.62%) outperforms its 7B variant (48.40%), and Meta-Llama-3-70B-Instruct (59.32%) significantly exceeds the 8B model (43.18%). This trend indicates that the emergent capabilities associated with larger parameters—such as robust world knowledge and instruction following—are essential for interpreting complex failure scenarios and generating precise corrective actions.

Error Analysis

Error Distribution

Error analysis for more models is provided in Figure 15-31. We defined the error types as follows:

- **Failure Detection**:
  - `false_no`: The model incorrectly predicts "no" (indicating no failure) when a failure actually exists.
  - `false_yes`: The model incorrectly predicts "yes"(indicating a failure) when no failure actually exists.
  - `parsing_error`: The model's prediction is neither "yes" nor "no", leading to a format parsing exception.
- **Failure Categorization**:
  - `false_nothing`: The model incorrectly predicts "nothing" (no specific failure type) when a failure actually exists.
  - `false_grab_failed`: The model misclassifies the failure as "grab_failed" when it belongs to another category.
  - `false_find_failed`: The model misclassifies the failure as "find_failed" when it belongs to another category.
  - `false_drop`: The model misclassifies the failure as "drop" when it belongs to another category.
  - `parsing_error`: The model's prediction is neither "yes" nor "no", leading to a format parsing exception.
- **Failure Description & Correction**:
  - `wrong_only_action`: The predicted action is incorrect, while the object prediction is correct.
  - `wrong_only_object`: The predicted object is incorrect, while the action prediction is correct.
  - `wrong_action_object`: Both the predicted action and object are incorrect.
  - `hallucination_error`: The predicted action is either outside the predefined `action_domain` or represents a fictional or unexpected behavior.
  - `parsing_error`: The model's prediction is neither "yes" nor "no", leading to a format parsing exception.

Error Examples

There are several error cases of GPT-4o in our benchmark, illustrating key limitations in its visual and temporal reasoning capabilities.

Figure 32 shows an error of GPT-4o related to limitations in fine-grained object recognition. In this example, the model fails to correctly identify the bread slice.

Figure 33 shows an error of GPT-4o related to difficulties in distinguishing visually similar objects. Although the agent successfully grabbed the "remotecontrol", the model fails to correctly identify it, likely due to confusion with visually similar objects such as a mobile phone, and outputs an incorrect "yes".

Figure 34 shows an error of GPT-4o related to deficiencies in understanding scene-object relevance. In this case, the character needs to find the mouse but moves to the living room, a location that is semantically unlikely to contain the target. The model fails to leverage spatial understanding and commonsense reasoning, and still outputs "no" when the target object is not visible in the third-person image.

Figure 35 shows an error of GPT-4o related to constraints in temporal perception and memory. Although the "remotecontrol" appears in an earlier input image, the model fails to retain and utilize this information, and incorrectly determines that a "find_failed" occurred.

Figure 4: The case study of our dataset.

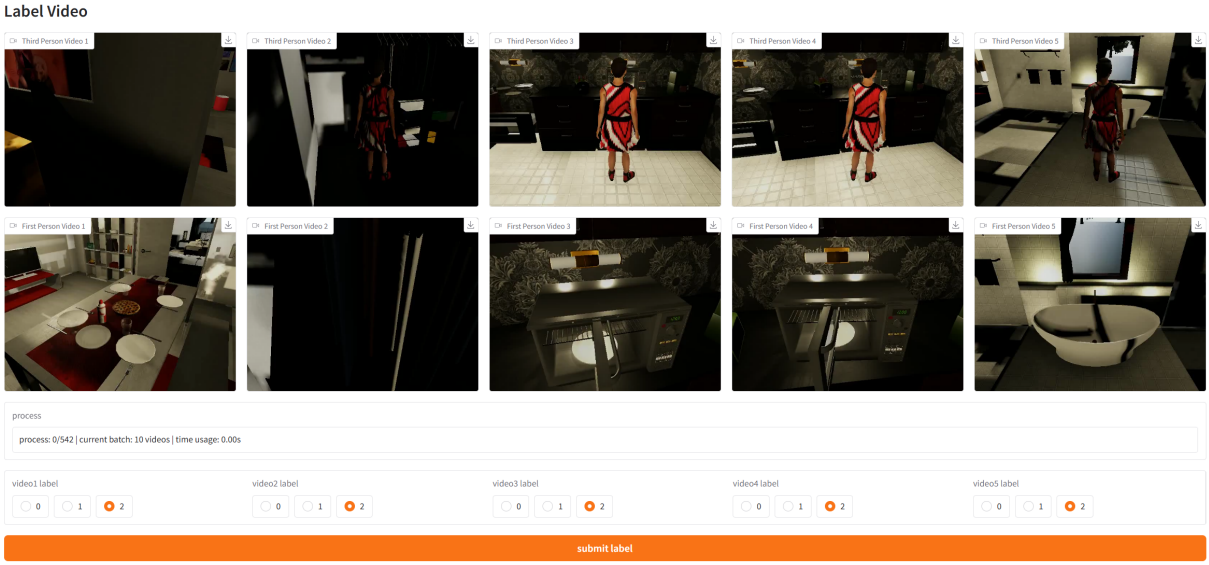

Figure 5: UI of the labeling Process. We designed a UI for manual labeling of data quality, categorizing the quality levels into three grades: 0, 1, and 2. Grade 0 indicates poor quality, grade 1 indicates average quality, and grade 2 indicates good quality.

Failure Detection Prompt Template

You are an intelligent household robot agent. You are completing the overall task: {instruction}.
The actions you have just executed is: {action_history}...
Note that the last excuted action is maybe not successfully completed. The {pov} images/video are what you are currently observing.
Above all, please analyze whether any unexpected situations were encountered during the execution of the overall task, based on the current observational analysis.
If unexpected situations were encountered, please answer "yes"; if not, please answer "no".
Please strictly answer only with "yes" or "no". Do not include any additional text, symbols, or explanations. Ensure your response is limited to these two options. Now please output your answer firstly.

Figure 6: The prompt template for failure detection in text modality.

Failure Categorization Prompt Template

You are an intelligent household robot agent, and you are completing the overall task: {instruction}.
The actions you have just completed is: {action_history}…
Note that the last excuted action is maybe not successfully completed.
Based on your prior analysis, {failure_detection_answer}…
Above all, what type of unexpected situation occurred in the {pov} images/video?
If the object in hand is accidentally dropped, please output 'drop'; If the target object is not found or the wrong position is walked to, please output 'find_failed';
If the object is not successfully picked up, please output 'grab_failed';
If nothing abnormal event happened, please output 'nothing'…
Now please output your answer firstly.

Figure 7: The prompt template for failure categorization in text modality.

Benchmark	Failure Detect from Vision	Failure Correct	Level	Perspective	Multi-room	Size
EgoThink (Cheng et al., 2024)	✗	✗	Plan	1^{st}	✓	700
ALFRED (Shridhar et al., 2020)	✗	✗	Plan	1^{st}	✓	25k
EAI (Li et al., 2024b)	✗	✗	Plan	1^{st}	✓	438
EgoTaskQA (Jia et al., 2022)	✗	✗	Plan	1^{st}	✓	40k
BEHAVIOR-1K (Li et al., 2023)	✗	✗	Plan	1^{st}	✓	1k
COWP (Ding et al., 2023)	✗	✓	Plan	1^{st}	✓	1085
EMMOE (Li et al., 2025b)	✗	✓	Plan	1^{st}	✓	966
AHA (Duan et al., 2025)	✓	✗	Mani.	3^{rd}	✗	45k
RoboFail (Liu et al., 2023)	✓	✓	Mani.	3^{rd}	✗	430
RoboFAC (Lu et al., 2025)	✓	✓	Mani.	3^{rd}	✗	78k
FAER (Ours)	✓	✓	Plan	1^{st} & 3^{rd}	✓	12k

Table 5: Comparison of considerations between existing benchmarks/datasets and our work. 'Mani' indicates 'Manipulation'. 'Multi-room' indicates whether the benchmark includes scenarios spanning multiple rooms.

Failure Type	Score 0	Score 1	Score 2
Drop	The object is too dark. The object is too small. The object is too thin. The object is mostly obscured by the character's body. The scene is too dark. The character is mostly obscured.	The object's drop can be basically seen but is hard to distinguish. The object's drop is partially obscured by the character's body.	The process of the object dropping and the item can be basically seen. The item is minimally obscured.
Find failure	The character barely moves. The character is mostly obscured. The scene is too dark. The camera is obscured for many frames. The camera angle rotates too much.	The camera is obscured for fewer frames. The character walks to the same room and is very close.	The character walks into another room. The character walks to the same room but is farther away. The obstruction is minimal.
Grab failure	The hand's state and the object's original position cannot be seen clearly after grabbing. The body is distorted. The body goes through walls. Using the hand that has already grabbed something to grab another thing. Inconsistency between third-person and first-person perspectives.	Only in the third-person perspective can the grabbing action be seen, but the hand is empty, this detail is too small, so it's not given a 2 (but for some difficulty, a few will be intentionally given a 2).	If either perspective can see all the information, it is labeled as 2. In the third-person perspective, the object is seen not moving and the hand is empty. In the first-person perspective, the object is seen not moving, and the hand is seen reaching out (the hand itself is enough), the third-person character is normal

Table 6: Items for Annotation. The leftmost column represents the type of Failure, and the top row represents the scores. These criteria are applied during the manual verification of data reliability, and only samples with a final score of 2 are retained.

Model	Failure Detection				Failure Categorization				Failure Description				Failure Correction				Avg. Acc.
	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1	
RoboFAC	50.01	0.00	0.00	0.00	00.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.25	0.00	0.00	0.00	–

Table 7: Quantitative evaluation of RoboFAC across four tasks: Failure Detection, Failure Categorization (Type Judge), Failure Description (What Happened), and Failure Correction (Next Action). Metrics include Accuracy (Acc.), Precision (Pre.), Recall (Rec.), and F1.

Model	Failure Detection				Failure Categorization				Failure Description				Failure Correction				Avg. Acc.
	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1	
Image																	
Qwen2.5-VL-7B-Instruct	50.13	**100.00**	0.23	0.46	51.37	**100.00**	2.71	5.28	42.87	45.07	65.40	53.36	75.30	83.70	62.80	71.76	54.92
InternVL3-8B-Instruct	49.11	**100.00**	0.18	0.35	69.62	92.71	45.44	60.99	35.14	39.39	55.25	45.99	60.91	63.37	51.67	56.92	56.87
Llava-v1.6-mistral-7B	55.69	53.65	83.51	65.33	55.47	23.87	11.48	15.50	20.15	20.16	23.88	21.86	55.03	70.52	17.24	27.71	45.28
Llama-3.2-11B-Vision	51.43	50.78	**92.27**	65.51	52.13	93.28	25.61	40.19	36.54	36.69	52.13	43.07	40.30	38.94	34.20	36.42	46.18
Gemini-2.5-flash	50.07	56.52	0.75	1.48	76.22	**100.00**	52.40	68.77	63.93	58.78	93.25	72.11	94.49	91.27	98.38	94.69	71.18
Gemini-2.5-flash-thinking	55.57	56.95	45.31	50.47	70.73	96.55	43.50	59.98	64.77	64.77	91.17	75.74	97.29	96.54	98.10	97.31	73.58
GPT-4o-mini	51.52	82.05	3.70	7.08	73.27	72.07	75.94	73.95	58.06	**95.10**	72.10	82.02	71.14	96.66	81.96	88.71	66.69
GPT-4o	**70.62**	70.08	71.90	**70.98**	**88.47**	88.39	**88.54**	**88.46**	**82.92**	75.40	**97.75**	**85.13**	**99.71**	**99.83**	**99.60**	**99.71**	**85.43**
Video																	
Qwen2.5-VL-7B-Instruct	49.24	33.33	0.12	0.24	53.27	**98.57**	7.96	14.73	**42.98**	**45.45**	**62.17**	**52.51**	**81.81**	**94.41**	**68.17**	**79.17**	**54.92**
InternVL3-8B-Instruct	**51.05**	**82.14**	**2.65**	**5.13**	**70.94**	92.71	**45.44**	**60.99**	35.14	39.39	55.25	45.99	60.91	63.37	51.67	56.92	54.51
Text																	
Qwen2.5-7B-Instruct	55.87	**75.83**	17.19	28.02	81.35	88.69	71.91	79.42	49.52	49.52	82.99	62.03	71.35	92.96	76.43	83.89	66.26
Llama-3-8B-Instruct	61.52	56.92	**94.64**	71.08	78.73	**98.92**	58.07	73.18	47.45	48.31	73.47	58.29	82.68	80.91	85.52	83.15	67.59
Qwen2.5-72B-Instruct	73.62	74.71	71.40	73.02	88.33	98.10	76.87	86.20	**52.54**	**52.54**	**96.71**	**68.09**	**97.25**	**96.44**	**99.88**	**98.13**	**78.68**
Llama-3-70B-Instruct	**73.97**	74.34	73.18	**73.76**	**93.53**	97.75	**90.25**	**93.85**	50.55	50.54	96.48	66.33	89.47	82.32	98.56	89.71	76.70
Avg.	57.10	69.09	39.79	50.49	71.67	88.69	49.72	63.71	48.75	51.51	72.71	60.30	76.97	82.23	73.16	77.43	63.62

Table 8: Quantitative evaluation of 12 models across image, video, and text modalities on four failure-related tasks. All experiments are conducted under the *groundtruth-cascade* setting, where ground truth from upstream tasks is used as input for the current task to isolate reasoning capabilities. We report Accuracy (Acc.), Precision (Pre.), Recall (Rec.), and F1 score. Best results are in **bold**, and second best are underlined within each modality.

	Model	Failure Detection		Failure Categorization		Failure Description		Failure Correction	
		normal	abnormal	normal	abnormal	normal	abnormal	normal	abnormal
Image	Qwen2.5-VL-7B-Instruct	100.00	0.23	100.00	2.71	20.34	65.39	87.78	62.80
	InternVL3-8B-Instruct	100.00	0.18	95.91	45.30	13.25	66.72	73.76	64.70
	Llava-v1.6-mistral-7B-hf	27.89	83.50	100.00	11.47	5.47	23.87	92.79	17.24
	Llama-3.2-11B-Vision-Instruct	10.60	92.27	98.15	25.60	10.08	52.13	46.39	34.19
	Gemini-2.5-flash	99.42	0.75	100.00	52.39	34.60	93.25	90.59	98.38
	Gemini-2.5-flash-thinking	65.80	45.31	97.80	43.50	50.46	91.16	96.48	98.09
	GPT-4o-mini	99.19	3.69	70.60	75.93	31.35	95.09	60.80	96.65
	GPT-4o	69.33	71.89	88.40	88.54	68.10	97.75	99.82	99.59
Video	Qwen2.5-VL-7B-Instruct	99.76	0.11	99.88	7.95	23.25	65.16	70.14	51.67
	InternVL3-8B-Instruct	99.42	2.65	96.42	45.44	15.04	55.24	95.84	68.14
Text	Qwen2.5-7B-Instruct	94.52	17.18	90.77	71.91	15.44	82.98	49.74	92.96
	Meta-Llama-3-8B-Instruct	28.41	94.63	99.36	58.07	21.44	73.47	79.82	85.52
	Qwen2.5-72B-Instruct	75.85	71.39	99.76	76.87	12.68	96.71	96.31	99.88
	Meta-Llama-3-70B-Instruct	74.75	73.18	95.90	90.25	5.64	96.48	78.84	98.55

Table 9: Evaluation results categorized by sample type (normal vs. abnormal). The table presents accuracy scores of 14 models across image, video, and text modalities—including open-source VLMs (e.g., Qwen2.5-VL, InternVL3) and closed-source models (e.g., Gemini-2.5, GPT-4o)—on four failure tasks. Notably, most models exhibit significant performance imbalance between sample types: many achieve high accuracy on normal samples but struggle with abnormal ones (e.g., Gemini-2.5-flash, Qwen2.5-VL-7B-Instruct and InternVL3-8B-Instruct). In contrast, GPT-4o shows the most balanced performance, with strong accuracy on both normal and abnormal samples across all tasks.

	Model	Failure Detection			Failure Categorization			Failure Description			Failure Correction		
		Drop	FF	GF	Drop	FF	GF	Drop	FF	GF	Drop	FF	GF
Image	Qwen2.5-VL-7B-Instruct	51.07	50.69	48.59	51.34	53.17	48.59	40.99	40.84	47.28	70.83	73.83	80.65
	InternVL3-8B-Instruct	50.87	49.52	46.94	48.49	77.48	73.90	41.66	38.61	40.93	82.48	55.96	80.45
	Llava-v1.6-mistral-7B-hf	53.09	56.13	56.82	50.53	62.71	48.59	18.41	11.72	16.63	64.11	54.13	50.09
	Llama-3.2-11B-Vision-Instruct	49.73	50.27	54.39	**89.38**	57.46	49.62	34.67	31.11	28.59	59.13	31.17	41.30
	Gemini-2.5-flash	50.40	50.94	48.49	70.69	73.41	**81.30**	55.37	77.54	71.12	88.29	98.24	92.98
	Gemini-2.5-flash-thinking	57.73	58.82	49.01	63.44	67.69	80.28	53.36	68.64	63.98	93.14	98.73	97.94
	GPT-4o-mini	57.81	50.51	48.69	63.79	81.69	66.82	57.93	70.39	55.79	**100**	**99.69**	**99.53**
	GPT-4o	**74.05**	**75.51**	**60.65**	84.81	**94.79**	81.25	**70.02**	**87.49**	**84.84**	**100**	**99.69**	**99.53**
Video	Qwen2.5-VL-7B-Instruct	49.17	49.84	48.35	49.17	58.25	48.45	16.02	44.19	59.41	80.52	79.25	86.59
	InternVL3-8B-Instruct	52.15	50.81	50.65	54.03	78.73	70.65	32.66	31.96	41.77	80.10	46.64	69.62
Text	Qwen2.5-7B-Instruct	51.88	62.53	48.31	56.72	92.50	81.21	46.63	**59.87**	34.48	65.18	78.97	63.83
	Meta-Llama-3-8B-Instruct	66.80	66.16	**50.65**	56.04	77.03	**97.10**	53.09	46.52	44.95	82.66	78.61	88.97
	Qwen2.5-72B-Instruct	72.31	**90.33**	48.69	75.53	**99.63**	79.71	55.24	54.92	**53.92**	**96.63**	**98.79**	**98.03**
	Meta-Llama-3-70B-Instruct	**84.27**	87.97	45.14	**77.68**	97.94	96.26	**55.51**	48.64	51.68	87.63	90.21	87.10
	Avg.	58.67	60.72	50.38	63.69	76.61	71.70	45.11	50.89	49.67	82.19	77.42	81.19

Table 10: Quantitative evaluation of 14 models across image, video, and text modalities, including open-source VLMs (e.g., Qwen2.5-VL, InternVL3, Llama-3.2-Vision) and closed-source models (Gemini-2.5, GPT-4o). In the Failure Detection task, models exhibit relatively weaker perception capabilities for Drop and GF events. However, in reasoning tasks such as Failure Categorization, Failure Description, and Failure Correction, the opposite trend is often seen, where models tend to perform better in handling Drop and GF events.

Tasks		
Cook dinner and set the table	Turn on TV	Bake cupcakes and decorate them
Prepare a fruit salad	Have breakfast and read news	Take medicine with water
Watch TV	Clean the entire house	Put groceries in the fridge
Wash dishes and clean the kitchen	Drink	Wash teeth
Make toast and have breakfast	Write an email	Have breakfast
Bake a cake	Bake a pie	Clean the dishes after dinner
Set up table	Make coffee and read a book	Work on the computer and drink coffee
Put groceries in the fridge and make dinner	Watch a movie and eat food	Browse the internet
Put groceries in Fridge	Cook some food	Bake bread and make toast
Prepare coffee and watch TV	Prepare breakfast	Have a picnic in the living room
Set up a home office	Make coffee and read the newspaper	Prepare to wash teeth
Read book	Water the orchid and open the curtains	Charge the cellphone and send a message
Fix a meal and watch a movie	Prepare a meal and set the table	Organize clothes in the closet
Work	Watch TV and relax on the sofa	Turn off all lights before bed
Wash dishes with dishwasher	Organize the bookshelf	Organize the kitchen cabinets
Make a cup of tea	Browse the internet and watch TV	Switch off the lights
Clean the living room	Bake a cake and serve it	Listen to music while working
Make toast and eat breakfast	Relax on sofa	Work on the computer
Listen to music while cooking	Light candles	Prepare to paint a picture at the desk
Change TV channel	Have a movie marathon with snacks	Teach a language lesson online
Browse internet	Prepare coffee and watch tv	Set up a movie night with friends
Pick up phone	Read a book	

Table 11: All tasks involved.

Objects					
plate	dishbowl	sink	faucet	nightstand	orchid
kitchentable	sofa	mouse	chips	tvstand	crackers
stove	computer	keyboard	coffeetable	chocolatesyrup	stall
cutleryfork	salmon	dishwasher	peach	cookingpot	cabinet
tv	kitchencabinet	washingsponge	poundcake	banana	bathroom
bellpepper	book	desk	fryingpan	juice	tablelamp
microwave	waterglass	radio	cellphone	cupcake	floor
kitchen	fridge	candle	kitchencounterdrawer	oventray	clothespile
remotecontrol	chair	wineglass	plum	bed	powersocket
cutleryknife	apple	pillow	bookshelf	lightswitch	mincedmeat
toaster	livingroom	whippedcream	bench	lime	milk
breadslice	bananas	paper	bedroom	candybar	hanger
mug	pie	chicken	dishwashingliquid	toothbrush	rug
coffeemaker	kitchencounter	cereal	cutlets	painkillers	

Table 12: All objects involved.

Failure Description Prompt Template

You are an intelligent household robot agent, and you are completing the overall task: {instruction}.
The actions you have just completed is: {action_history}…
Note that the last excuted action is maybe not successfully completed.
Based on your prior analysis, {failure_detection_answer}…, and {failure_categorization_answer}…
The {pov} images/video are what you are currently observing.
Above all, what happened in the images while you was performing your overall task?
{action_domain_prompt}...
{additional_few_shots_prompt}…
Now please output your answer firstly.

Figure 8: The prompt template for failure description in text modality.

Failure Correction Prompt Template

You are an intelligent household robot agent responsible for performing a series of tasks to complete household chores. You are completing the overall task: {instruction} Your decisions are based on previously executed actions, the current environmental images/video ({pov} perspective), and recommendations provided by a planner.
The actions you have just completed is: {action_history}…
Note that the last excuted action is maybe not successfully completed.
Based on your prior analysis, , {failure_detection_answer}…, and {failure_categorization_answer}…, and {failure_description_answer}…
Above all, make your action selection.
If the previous action executed successfully, adopt the planner's recommended next action and output: {next_action}. Otherwise, if the previous action failed, perform appropriate corrective actions based on the type of issue:
1. Dropped item: execute an action to pick up the item again, for example, output "grab salmon".
2. Grasping failure: attempt to grasp the item again, for example, output "grab salmon".
3. Target item not found: execute an action to search for the item again, for example, output "find tv"…
Now please output your answer firstly.

Figure 9: The prompt template for failure correction in text modality.

Failure Detection Prompt Template

You are an intelligent household robot agent. You are completing the overall task: {instruction}.
The actions you have just executed is: {action_history}...
Note that the last excuted action is maybe not successfully completed. The {pov} images/video are what you are currently observing.
Above all, please analyze whether any unexpected situations were encountered during the execution of the overall task, based on the current observational analysis.
If unexpected situations were encountered, please answer "yes"; if not, please answer "no".
Please strictly answer only with "yes" or "no". Do not include any additional text, symbols, or explanations. Ensure your response is limited to these two options. Now please output your answer firstly.

Figure 10: The prompt template for failure detection in visual modality.

Failure Categorization Prompt Template

You are an intelligent household robot agent, and you are completing the overall task: {instruction}.
The actions you have just completed is: {action_history}…
Note that the last excuted action is maybe not successfully completed.
Based on your prior analysis, {failure_detection_answer}…
Above all, what type of unexpected situation occurred in the {pov} images/video?
If the object in hand is accidentally dropped, please output 'drop'; If the target object is not found or the wrong position is walked to, please output 'find_failed';
If the object is not successfully picked up, please output 'grab_failed';
If nothing abnormal event happened, please output 'nothing'…
Now please output your answer firstly.

Figure 11: The prompt template for failure categorization in visual modality.

Failure Description Prompt Template

You are an intelligent household robot agent, and you are completing the overall task: {instruction}.
The actions you have just completed is: {action_history}…
Note that the last excuted action is maybe not successfully completed.
Based on your prior analysis, {failure_detection_answer}…, and {failure_categorization_answer}…
The {pov} images/video are what you are currently observing.
Above all, what happened in the images while you was performing your overall task?
{action_domain_prompt}...
{additional_few_shots_prompt}…
Now please output your answer firstly.

Figure 12: The prompt template for failure description in visual modality.

Failure Correction Prompt Template

You are an intelligent household robot agent responsible for performing a series of tasks to complete household chores. You are completing the overall task: {instruction} Your decisions are based on previously executed actions, the current environmental images/video ({pov} perspective), and recommendations provided by a planner.
The actions you have just completed is: {action_history}…
Note that the last excuted action is maybe not successfully completed.
Based on your prior analysis, , {failure_detection_answer}…, and {failure_categorization_answer}…, and {failure_description_answer}…
Above all, make your action selection.
If the previous action executed successfully, adopt the planner's recommended next action and output: {next_action}. Otherwise, if the previous action failed, perform appropriate corrective actions based on the type of issue:
1. Dropped item: execute an action to pick up the item again, for example, output "grab salmon".
2. Grasping failure: attempt to grasp the item again, for example, output "grab salmon".
3. Target item not found: execute an action to search for the item again, for example, output "find tv"…
Now please output your answer firstly.

Figure 13: The prompt template for failure correction in visual modality.

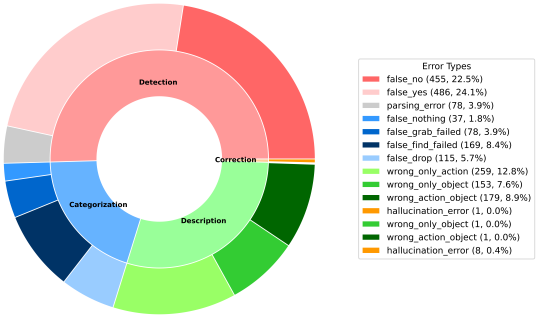

Figure 14: The error distribution of GPT-4o.

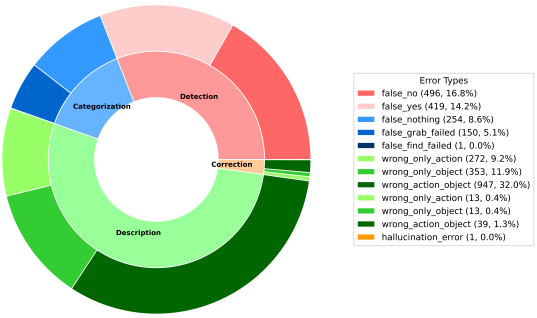

Figure 17: The error distribution of Qwen-2.5-72B-Instruct.

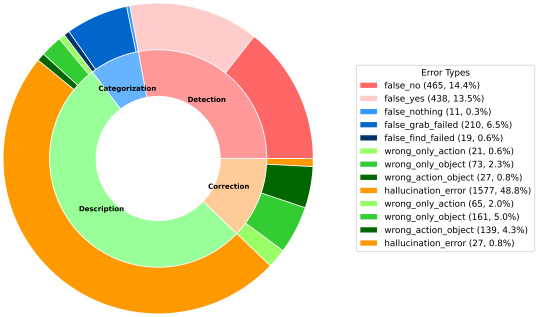

Figure 15: The error distribution of Meta-Llama-3-70B-Instruct.

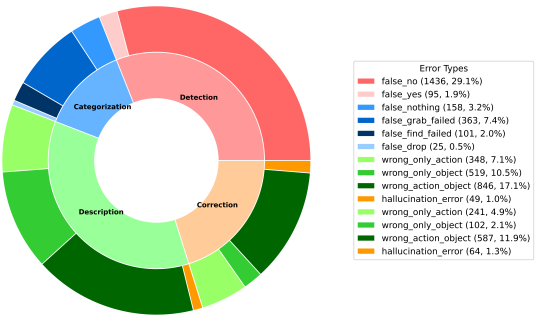

Figure 18: The error distribution of Qwen-2.5-7B-Instruct.

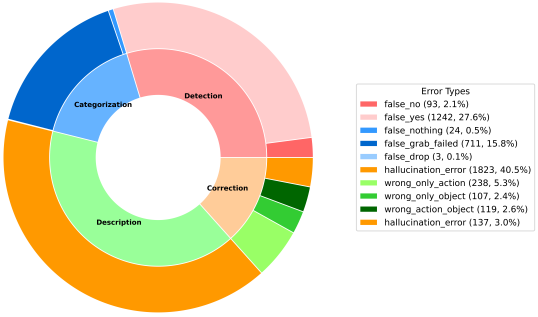

Figure 16: The error distribution of Meta-Llama-3-8B-Instruct.

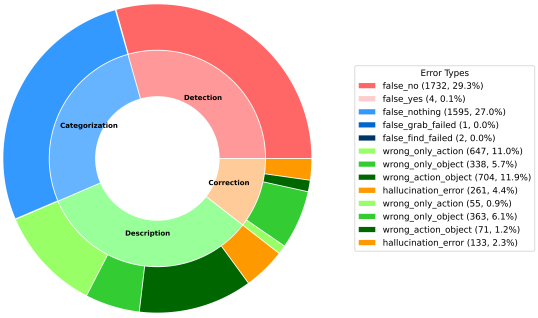

Figure 19: The error distribution of Qwen-2.5-VL-7B-Instruct.

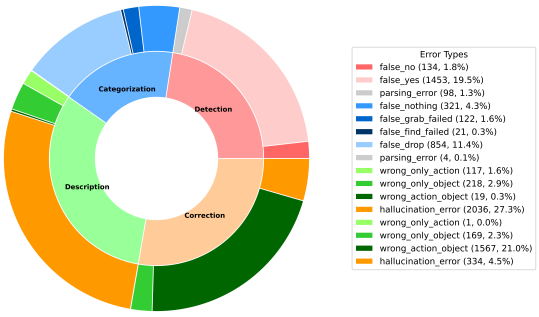

Figure 20: The error distribution of Llama-3.2-11B-Vision-Instruct.

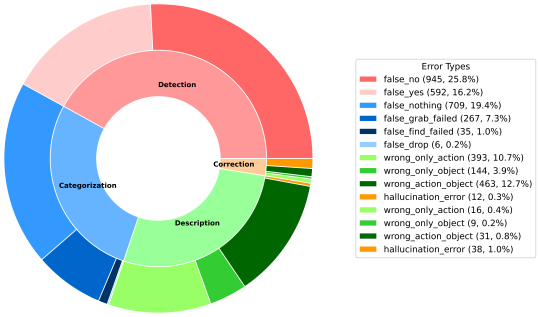

Figure 21: The error distribution of Gemini-2.5-Flash-Lite-Preview-06-17 (Thinking).

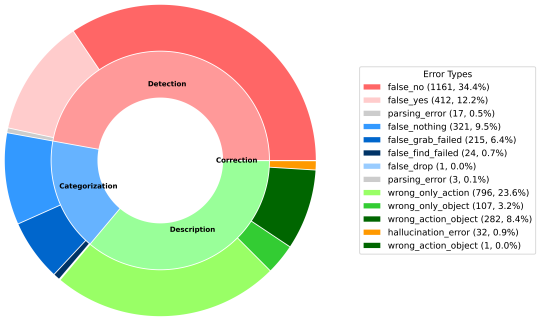

Figure 22: The error distribution of GPT-4o (0 image).

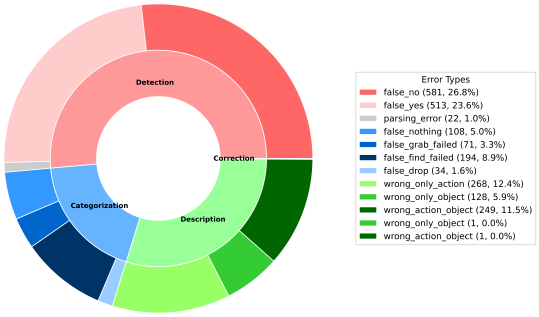

Figure 23: The error distribution of GPT-4o (2 images).

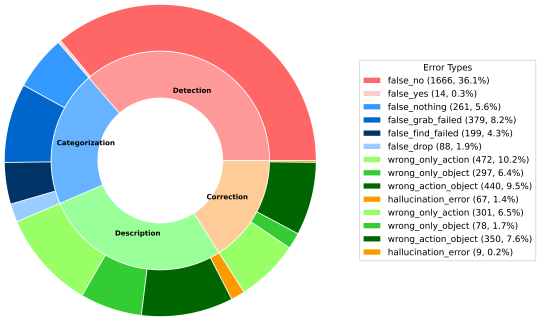

Figure 24: The error distribution of GPT-4o-mini.

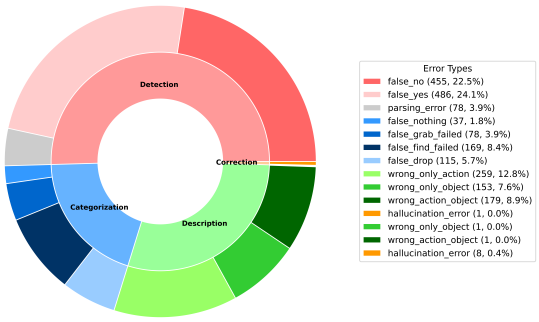

Figure 25: The error distribution of `GPT-4o`.

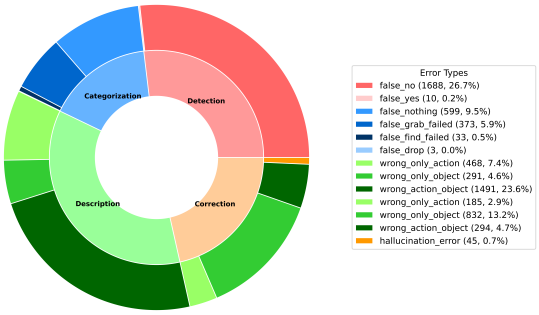

Figure 26: The error distribution of InternVL-3-8B-Instruct.

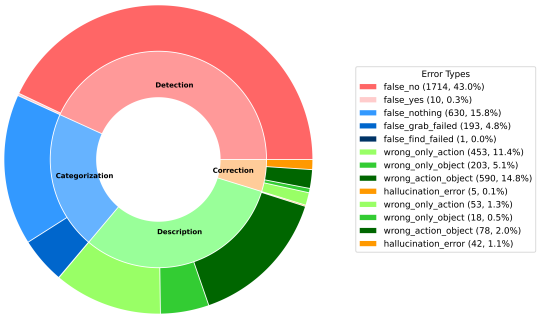

Figure 29: The error distribution of Gemini-2.5-Flash-Lite-Preview-06-17.

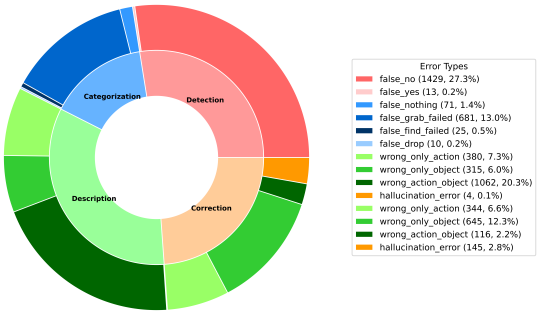

Figure 27: The error distribution of InternVL-3-8B-Instruct (Text).

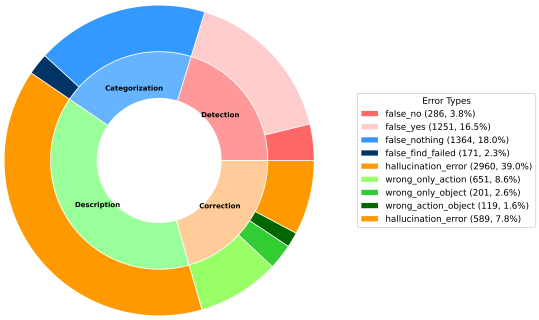

Figure 30: The error distribution of LlaVA-v1.6-Mistral-7B-HF.

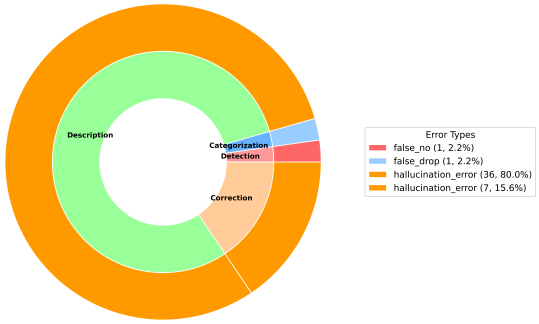

Figure 28: The error distribution of `Llama-3 LLaVA-Next 8B HF`.

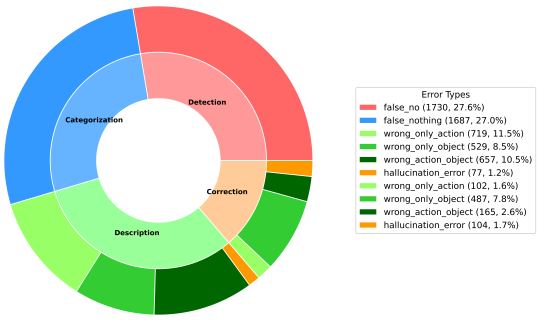

Figure 31: The error distribution of Qwen-2.5-VL-7B-Instruct (No Image).

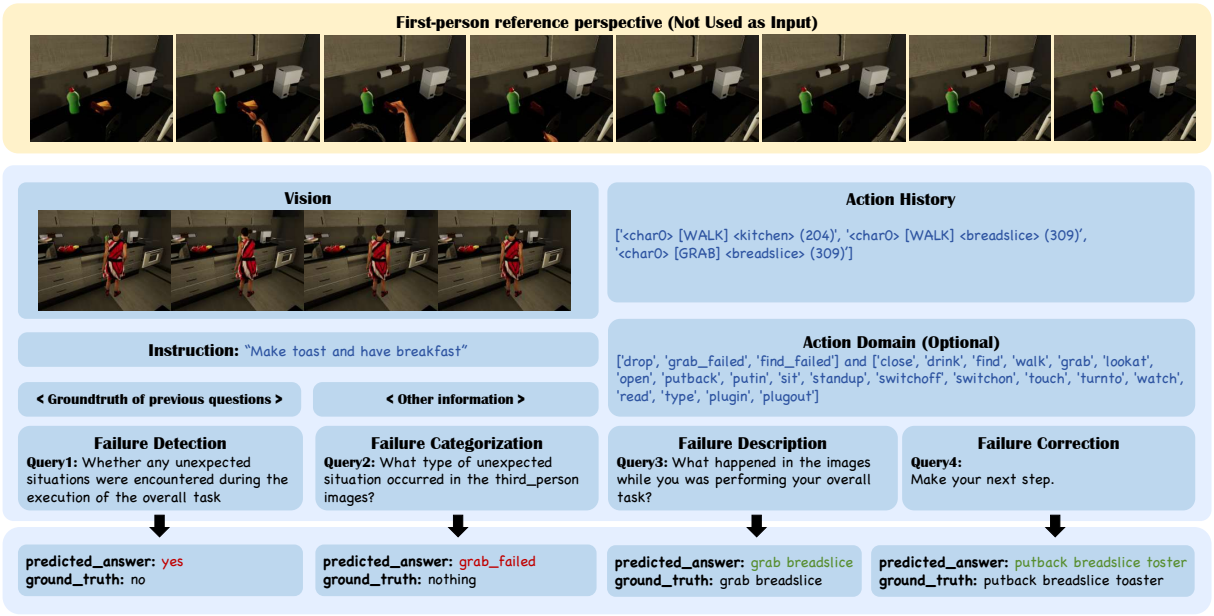

Figure 32: An error sample of GPT-4o due to small size of object.

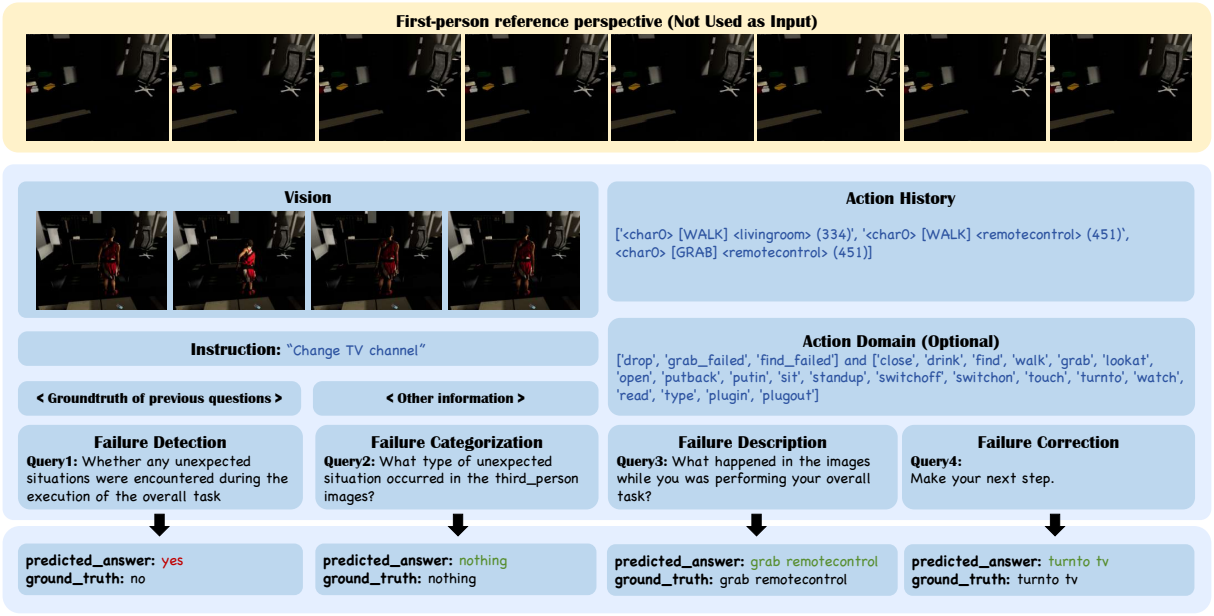

Figure 33: An error sample of GPT-4o due to visual misleading or small object.

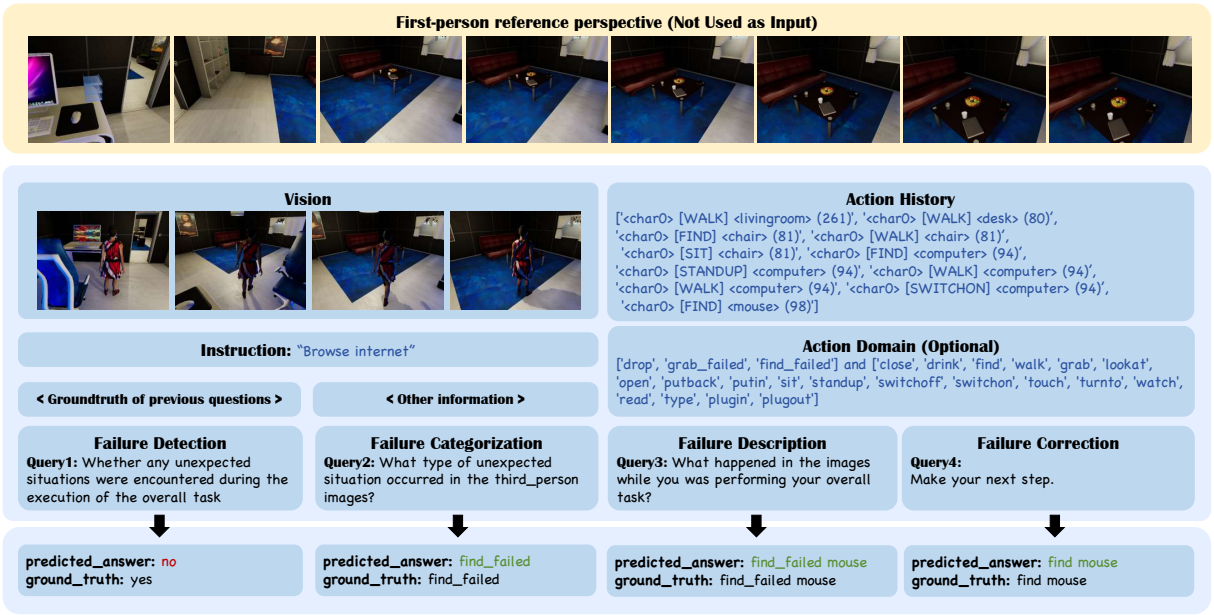

Figure 34: An error sample of GPT-4o about scene-object relevance.

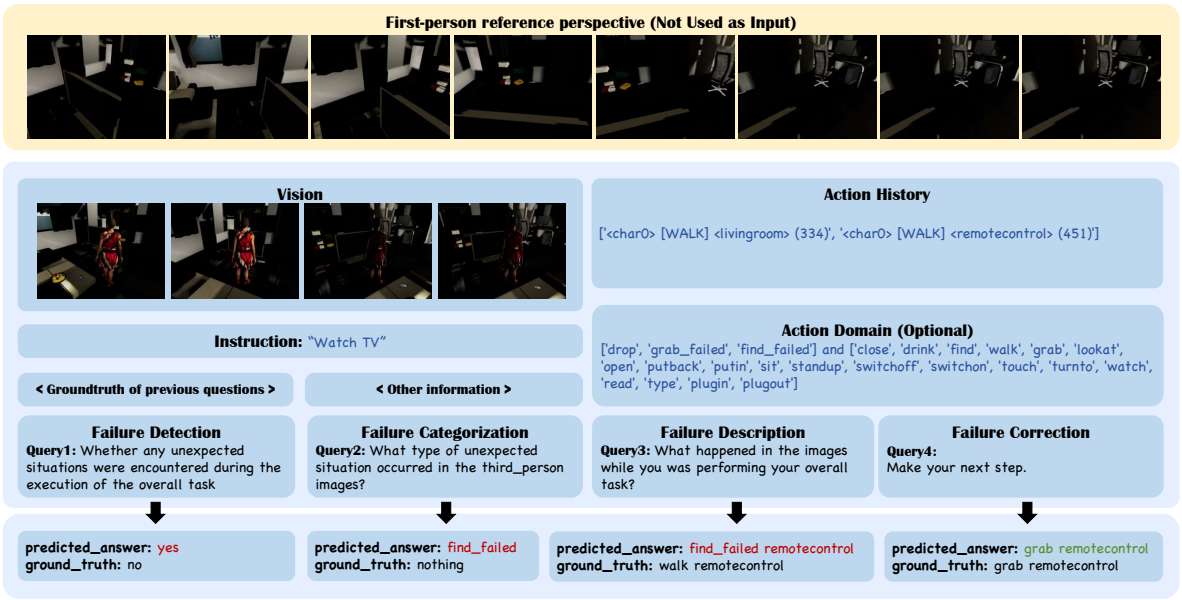

Figure 35: An error sample of GPT-4o about timely perception.