Completeness-Aware Task Planing and Execution using Pretrained Vision-Language Models in Manipulation Control

Anonymous Authors

***Abstract*— While Vision–Language Models (VLMs) have been successfully employed for task planning in manipulation control, they often struggle in long-horizon, context-rich scenarios, where an agent must decompose a task into a lengthy sequence of embodied actions while handling complex perception and visual grounding. In such settings, the generated plan must be semantically and logically coherent for embodied execution and must faithfully follow user instructions to complete the task. In this work, we propose a Completion-Aware Task Planning and Execution framework that analyzes complex queries and produces reliable and feasible plans for embodied execution. Specifically, we leverage pretrained VLMs as a Planner to generate task plans together with the required perceptual information from observations and user queries in a zero-shot prompting manner. An Evaluator then detects potential semantic and logical errors and prompts the Planner to revise the plan to ensure execution feasibility. Finally, an Executor carries out the embodied actions to fulfill the query. The Planner further incorporates a completeness detection mechanism at the end of each execution cycle, enabling verification of task fulfillment as well as replanning under incomplete execution or failure recovery scenarios. We evaluate the proposed framework across three distinct tabletop scenarios using a Franka Emika robotic arm, demonstrating its reliability and effectiveness in long-horizon, context-rich tasks [1]. Moreover, real-robot deployments highlight the adaptability of our approach in failure recovery and replanning settings.**

I. Introduction

The reliability, trustworthiness, and safety of artificial intelligence (AI) have emerged as critical challenges that significantly affect the usability and scalability of recent AI advancements across many domains of society [1]. At the same time, a growing body of research demonstrates that large language models (LLMs) and vision–language models (VLMs) [2], [3] can endow robots with advanced semantic planning and logical reasoning capabilities, enabling them to execute complex natural-language instructions provided directly by humans. Despite their strong vision–language understanding, VLMs can occasionally produce unreliable or inaccurate outputs [4], [5], which undermines their reliability and trustworthiness in real-world deployments. These challenges are further amplified in embodied AI settings, where both semantic and physical safety constraints must be enforced consistently from high-level planning to low-level control to ensure safe interaction with the physical world in human-centric environments. For instance, a robot cannot grasp an object unless its gripper is open, nor should it manipulate an object unnecessarily when the task requirements are already satisfied. Such semantic and geometric constraints substantially increase the complexity of generating executable plans using foundation VLMs. Moreover, long-horizon context-rich task planning remains particularly challenging for VLMs due to the increasing plan complexity and the tendency for context degradation over extended reasoning horizons [6].

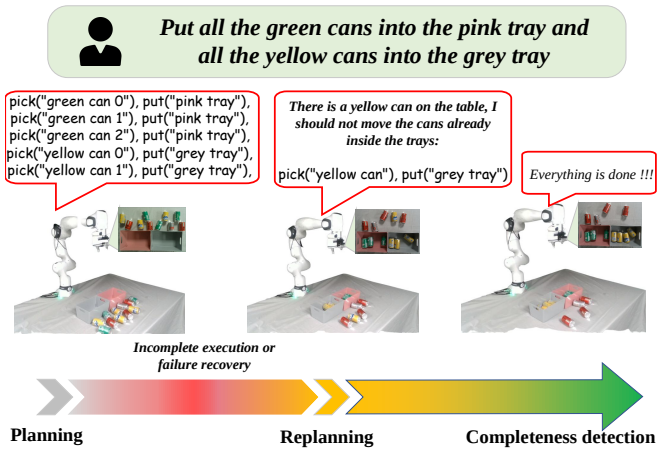


Fig. 1: Overview: Example of long-horizon, context-rich query poses challenge for agent in producing effective task plan. During execution, collisions may occur, requiring failure recovery and replanning. After finishing the execution, the embodied agent must evaluate task completion to determine whether additional replanning and execution are necessary to satisfy the query.

Prior work has demonstrated that, with appropriate grounding, foundation language models can generate executable robot plans under zero-shot and few-shot paradigms [2], [7], [8]. However, these approaches typically rely on large-scale robot action datasets for end-to-end training or fine-tuning, making data acquisition costly and time-consuming. Moreover, fine-tuning pretrained VLMs often sacrifices domain generalization, scalability, and usability, limiting their applicability across diverse tasks and environments [9], [10]. To address execution feasibility, other studies incorporate geometric constraints into language-based planning frameworks [6], [11]–[13]. For example, Guo et al. [11] propose the DoReMi framework, which jointly generates task plans and associated geometric constraints, and subsequently employs VLMs to detect constraint violations during execution. Similarly, Text2Motion [12] leverages symbolic goals to verify geometric dependencies and search for geometrically feasible plans. Despite their effectiveness, these methods primarily focus on geometric constraints, leaving other sources of failure unaddressed. The semantic inconsistencies, redundant actions, and safety violations may arise during planning, particularly when deploying pretrained

[1]We will publish our code once the paper is accepted

VLMs in a zero-shot setting.

In this work, we propose a framework combining three components: **Planner**, **Evaluator**, and **Executor** that leverages pretrained VLMs to generate and execute reliable and feasible task plans for long-horizon, context-rich manipulation control. Specifically, the Planner analyzes the user query and produces a sequence of actions together with the required perceptual cues and visual grounding necessary for successful execution. The Evaluator then examines the generated plan, identifying potential issues such as formatting errors, semantic inconsistencies, or geometric violations that may hinder execution, and provides feedback to guide the Planner in refining the plan. This iterative process continues until the plan is approved by the Evaluator, after which an Executor carries out the actions to complete the task. Furthermore, we incorporate a post-execution completeness check that enables the framework to reason about task fulfillment and handle long-horizon, context-rich scenarios through replanning when necessary, thereby improving overall execution success. We evaluate the proposed framework across three experimental settings and compare it against multiple state-of-the-art baselines, demonstrating its effectiveness in reducing planning errors while employing pretrained VLMs for manipulation task planning.

II. Related Work

VLMs in Long-Horizon Task Planning for Manipulation Control. Task and Motion Planning (TAMP) using VLMs has been intensively studied, showing the remarkable achievements in breaking down the user's query and generating the feasible plan for execution [6], [14]–[18]. These frameworks incorporate off-the-shelf VLMs to enhance the high-level reasoning and low level planning of TAMP. Recently, ViLaIn [19] improves planning accuracy and scalability in long horizon task by combining vision–language models with classical symbolic planning by leveraging the Planning Domain Definition Language (PDDL) [20] to refine problem formulations through error-driven feedback. Additionally, DKPROMPT [21] automates vision–language model prompting by incorporating domain knowledge encoded in PDDL, resulting in improved task planning for open-world environments. Other studies [3], [22], [23] use VLMs to generate the code for low-level manipulation skills (code-as-policy) rather than the series of primitive actions in a zero-shot manner. However, long-horizon, context-rich task planing remains challenging to these approaches. Our framework introduces completeness assessment of the overall operation, allowing the replanning and execution to handle long-horizon context-rich task.

Output Improvement in pretrained VLMs. Output-improvement represents a promising strategy for enhancing the quality of VLMs' responses by enabling models to identify and amend mistakes in their outputs [24]. The most prominent work is Chain-of-Thought Prompting [25] where the agent is encourage to improve the reasoning capability by simply being prompted to think step-by-step. Another approach is to allow the model revises its outputs solely based on its inherent abilities and the provided input context, without any external feedback [26]–[30]. These works use self-consistency [30], self-evaluation [28], [31], or self-correction [29] of the self-generated data, mitigating factual flaws of the responses. Other studies leverage multi-agent scheme that contains multiple agent with different reasoning methods, allowing the model to look at different aspect of the query [32], [33]. Regarding to the self-improvement of VLMs in robotics, re-planning strategy is widely used [34]–[37] to adapt robot actions in cases where the initial plan does not successfully achieve the desired goal. Other works [6], [11]–[13] avoid the action plan that violate geometric constraints, eliminating the infeasible scenario of low level control. In this work, we employ pre-trained VLMs to construct a Planner that generates task plans, while an Evaluator assesses these plans using semantic, geometric, and embodied grounding criteria. The two agents iteratively generate and evaluate action plans to ensure accurate perception and feasible execution.

Task-replanning using VLMs. failure recovery and replanning capability of robot under the unexpected task execution errors has been investigated to enable the reliable robotic planning and execution [6], [34], [36], [38]–[41]. These works focus on recover to the most recent stable state of the robot and perform the task again while avoiding the failure repetition in the next execution. Specifically, Yang et al. [6] introduce VLM-TAMP that leverages VLM to generate semantically-meaningful and horizon-reducing intermediate subgoals to guide the planner. ReplanVLM [38] uses internal and external error correction mechanisms to correct errors when the task execution fails, producing the reliable replan to overcome the previous mistake. Pchelintsev et al. [34] developed LERa framework that revisits the generated scene description, identifies and explain the errors which later used to modify the plan accordingly. While these aforementioned works focus on previous mistake and modify it, our proposed approach uses the Evaluator to ensure the semantic and geometric correctness, and replans under the unexpected errors such as task incompleteness or collision, overcoming the difficulty of long-horizon context-rich query.

III. Method

In this section, we present our framework that is able to produce a reliable plan and execution in long-horizon context-rich task. As illustrated in Fig. 2, the framework comprising three components: Planner, Evaluator and Executor. We will elaborate each component in the following sections.

A. Planner

Given the language instruction I from human as well as the visual observation O , the Planner $\mathcal{P}$ will generate a task plan $P = [p_0, p_1, \cdots, p_n]$ including multiple manipulation actions to accomplish the task, along with the task-assessment ω to determine the task-completeness. Mathematically:

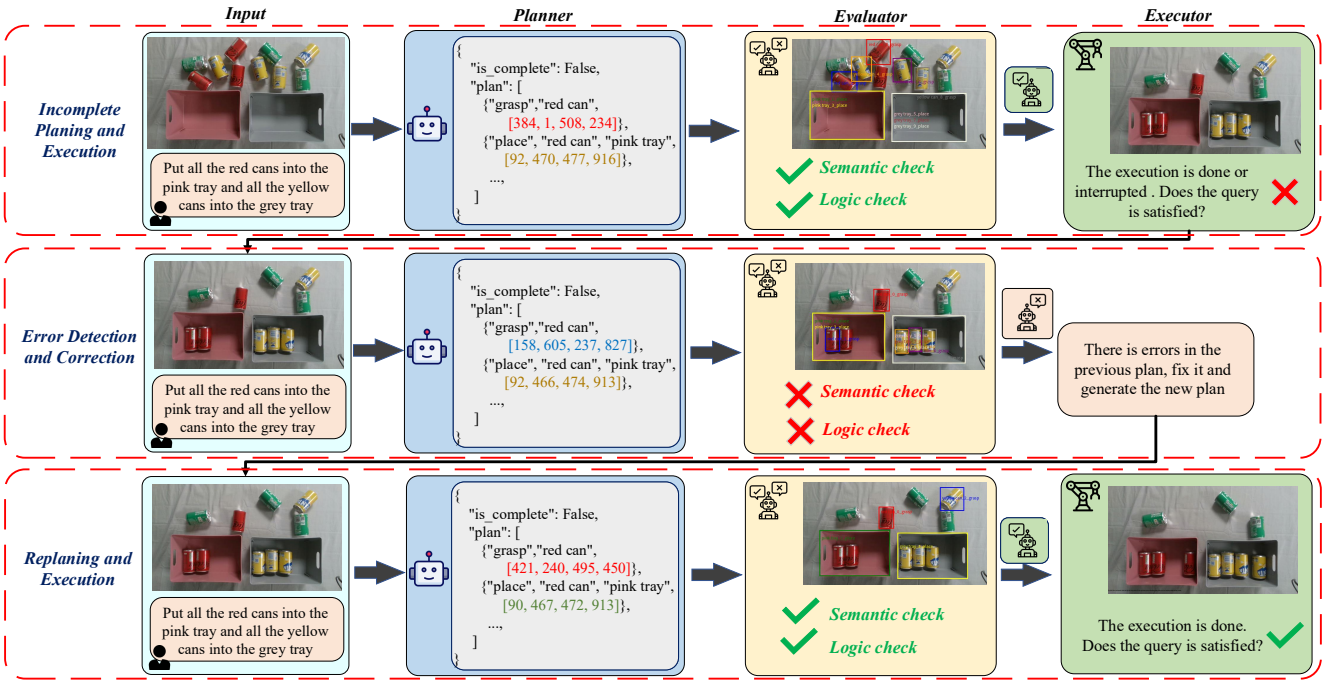


Fig. 2: Overall Architecture The Planner extracts task-relevant perceptual information and generates a sequence of actions to be executed sequentially in order to accomplish the task. The Evaluator then analyzes the generated plan to detect potential errors in visual grounding and semantic reasoning. If any issues are identified, feedback is provided to the Planner, which iteratively refines the action plan until it is approved by the Evaluator. Once the perception and plan are validated, the Executor uses the task-relevant perceptual information to carry out the actions and complete the task. After execution, the Planner assesses task completeness and initiates replanning for subsequent execution if necessary

$$[\omega, P] = \mathcal{P}(I, O) \tag{1}$$

The ω is used to assess the task-completeness, allowing the agent re-plan for the mistakes in previous execution. Meanwhile , each tuple p_i in the plan P contains (1) *primitive actions*, (2) *manipulated objects* and (3) *2D visual grounding* to facilitate the task evaluation and execution (Sec. III-B and III-C). The primitive actions are the primitive functions defined by the low level control interface of the robot, ensuring the feasibility of conversion between the generated plan and the real embodiment execution.

Primitive Action Definition

```
primitive_action = f"""
grasp(<target_object>):
    grasp the target_object,
place(<hold_object>, <target_object>):
    place the current hold_object on
    top of the target_object,
...
"""
```

Each primitive action targets a manipulated object, specified together with its 2D visual grounding (bounding box) in the observation. The tuple p_i encodes sufficient information for object identification and manipulation, and the sequential execution of $p_i \in P$ ensures task completion. The action set is extensible to more complex behaviors (e.g., stir and push), supporting scalability. Additionally, primitive actions are governed by logical constraints; for example, a `place` action is valid only if the corresponding `hold_object` has been previously grasped.

To construct the Planner, we prompt the pretrained VLM to analyze the instruction with the observation, breakdown the task into series of primitive actions as follows:

Instruction Prompt Definition

```
instruction_prompt = f"""
You are a Franka Emika Robot with one
    arm and a gripper.
Your task is to:
 1. Receive an user query and an image.
 2. Breakdown the task into a series of
     primitive actions.
 3. Reference the defined actions in {
    primitive_action}.
Report in the JSON format as follows:
{{
    "is_complete": boolean,
    "plan": [
        <primitive_action_1>,
        <primitive_action_2>,
        ...
    ]
}}
"""
```

While prompting the pretrained VLM maintains the domain generalization, usability and scalability of the framework, we observe that the output might suffers from various

problem such as semantic and logical errors or context-loss for long-horizon task. It often fails to assess the task-completeness in the context-rich scenarios. To resolve the issue, the plan need to be evaluated for ensuring the task-completion as well as the execution feasibility.

B. Evaluator

We introduce the Evaluator $\mathcal{E}$ that analyzes the plan P and verify it with the embodied reasoning according to the following criteria:

- The primitive actions only manipulate the task-required objects. This ensures the efficiency as well as correctness for re-planing and failure recovery.
- The primitive actions satisfy the semantic and geometric constraints. This criterion imposes the constraints for execution feasibility.
- The task completion detection and the plan should be logically coherent.

These criteria define a set of checking rules used by the Evaluator to systematically assess the generated plan P, with the goal of ensuring grounding correctness as well as semantic and logical consistency across all actions. Specifically, the Evaluator analyzes the plan by iterating through each primitive action, verifying that the output format is valid and that the semantic meaning and logical dependencies between consecutive actions are coherent (Fig. 2).

When inconsistencies or errors are detected, the Evaluator generates structured feedback $f_{feedback}$ describing the nature of the problem and communicates it to the Planner. The current plan P, together with the corresponding Evaluator feedback $f_{feedback}$, is appended to the instruction I to prompt the Planner to revise and improve its response, as formalized in Eq. 1. This closed-loop interaction between the Planner and Evaluator is performed iteratively until the generated plan satisfies all evaluation criteria and receives approval.

In addition to semantic and logical verification, the Evaluator addresses inaccuracies in 2D visual grounding produced by pretrained VLMs, which may arise from generation bias or stochastic decoding. To mitigate this issue, the Evaluator refines object grounding using an open-vocabulary object detection and segmentation model, producing precise 2D bounding boxes and pixel-level masks aligned with the scene observations. Once the plan is validated and grounding is refined, the approved plan is forwarded to the Executor, which performs low-level control to complete the task (Sec. III-C).

C. Executor

For each tuple $p_i \in P$, the Executor converts to the low level control of the gripper to manipulate the targeted object. First, we use 2D bounding box to segment the object and identify the 3D coordinates (x_c, y_c, z_c) of the centroid of the object respective to the camera. The coordinates is then convert to the gripper coordinates to identify the displacement needed for the gripper to manipulate the object. Finally, the agent will execute the Plan sequentially to accomplish the task. After finishing the plan execution, the Planner will assess task-completeness mentioned in Sec. III-A, and continue the pipeline until the task-completion is confirmed.

Overall. Our proposed approach constructs a reliable pipeline for for task planning and embodied execution. The Planner is responsible for generating the necessary information for evaluation and execution, while the information are verified by the Evaluator, ensuring the correctness for Execution. Furthermore, our proposed approach enable the re-planing for failure recovery or incomplete execution, improving the performance in the long-horizon context-rich task. We depict our framework in Fig. 2 and summarize the overall code flow in Al. 1.

Algorithm 1: VLM-guided Manipulation Control

Input: I_h : Initial human instruction
O : Visual observation
$\mathcal{P}$: Planner
$\mathcal{E}$: Evaluator

function `VLMManipulationControl(`I_h`)`
  $\omega \leftarrow$ `False` // Task completion flag
  $I \leftarrow I_h$ // Current instruction
  while $\omega =$ `False` **do**
    $e \leftarrow$ `False` // Initialize evaluation flag
    Update(O) // Update visual observation
    while $e =$ `False` **do**
      $[\omega, P] \leftarrow \mathcal{P}(I, O)$ // Generate plan
      $(f_{\text{feedback}}, e) \leftarrow \mathcal{E}(P)$ // Evaluate plan
      $I \leftarrow [I, P, f_{\text{feedback}}]$ // Update instruction
    for $p \in P$ **do**
      Execute(p) // Execute action

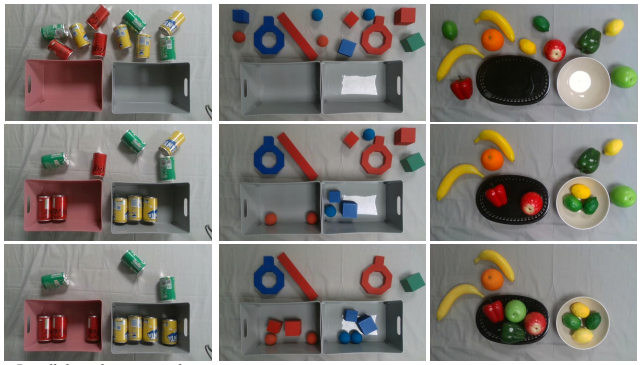
IV. Experimental Results

In this section, we design the experiments to demonstrate the effectiveness of the proposed approach in improving the reliability and the completeness of the VLM-guided Manipulation Control.

A. Settings

a) Environment: A diverse set of real-robot environments is constructed to validate the effectiveness of the proposed framework in real-world applications. We meticulously collect 150 samples for evaluation in the three distinct scenarios (Fig. 3). For each sample, the instruction requires the Planner to decompose the task into smaller, hierarchical steps and to perform multiple execution cycles. The collected samples span three stages of inference: initial planning, incomplete replanning, and final task completion, enabling a comprehensive evaluation of the framework's semantic understanding capabilities as well as its ability to assess and ensure task completeness.

b) Models: We select Qwen3-VL [42] as the Foundation VLM to construct the Planner due to its advancement int 2D grouding capability as well as spatial and semantic reasoning. We also observe that other foundation families such as Gemma3 [43] or MiniCPM-V [44] could generate the plan well but failed in providing accurate 2D grounding.



Put all the reds cans into the pink tray, and all the yellow cans into the grey tray | *Put all the red cubes and balls to the left tray, and all the blue cubes and balls to the right tray* | *Put all the bell peppers and apples into the tray, and all the lemons and limes into the bowl*

Fig. 3: Examples of the dataset in our experiments. *Top*: The **Initial** state of the experiment, these samples require long-horizon context-rich task planning. *Middle*: The **Incomplete** state of the experiment, these samples simulate the failure recovery state or incomplete execution state. The Planner is required to understand the semantic of the objects that are already satisfied the query. *Bottom*: The **Complete** state of the experiment, these samples are used to evaluate the completeness detection of the planner.

Besides, SAM3 [45] is used for open-vocabulary object detection and segmentation.

c) Baselines: We evaluate the proposed framework with the State-Of-The-Art approaches targeting in improving the performance of pretrained-VLM including normal prompting, Chain-of-thought (CoT) [23] and Compositional Chain-of-Thought (CCoT) [46]. We empirically compare the performance of these baselines with various sizes of the Qwen3-VL model.

B. Results

We firstly evaluate the semantic correctness of the generated plan. The semantic correctness is defined as a proportion of executable plan satisfying the geometry constraint as well as correct targets (in terms of spatial, categorical and color references) over the total number of samples.

TABLE I: Semantic Correctness comparison of the generated plan of proposed framework and the baselines. We report the average of the metric over three aforementioned scenario.

Model \ State	Initial	Incomplete	Complete	Average
4B	0.56	0.08	0.14	0.26
4B - CoT	0.62	0.10	0.14	0.29
4B - CCoT	0.58	0.14	0.18	0.3
4B - Ours	**0.78**	**0.76**	**0.86**	**0.80**
8B	0.60	0.14	0.18	0.31
8B - CoT	0.64	0.16	0.14	0.31
8B - CCoT	0.66	0.18	0.16	0.33
8B - Ours	**0.88**	**0.88**	**0.9**	**0.89**
32B	0.58	0.16	0.2	0.31
32B - CoT	0.6	0.18	0.18	0.32
32B - CCoT	0.62	0.16	0.16	0.31
32B - Ours	**0.86**	**0.88**	**0.92**	**0.89**

As shown in Table I, our approach significantly outperforms all baselines across all evaluation settings. On average, our method achieves 89% semantic correctness for both the 8B and 32B model variants, whereas the baseline methods attain only around 30%. Furthermore, our approach demonstrates consistent effectiveness across both model scales, owing to its ability to iteratively identify and correct errors through feedback from a foundation model. This substantial performance gain arises from the interaction between the Planner and Evaluator, which collaboratively detect and rectify semantic errors.

Secondly, we compare the task-completeness of our framework compared with the baselines. A task completeness is defined as the sufficiency of primitive actions to manipulate all the objects to satisfy the user query.

TABLE II: Completeness comparison of the generated plan of proposed framework and the baselines. For the completed samples, the plan is considered as correct if it could detect the task-completeness, while the other samples need to generate the plan to complete the task. We report the average of the metric over three aforementioned scenario.

Model \ State	Initial	Incomplete	Complete	Average
4B	0.08	0.24	0.14	0.15
4B - CoT	0.12	0.32	0.16	0.20
4B - CCoT	0.10	0.26	0.12	0.16
4B - Ours	0.10	0.82	0.86	0.59
8B	0.10	0.24	0.16	0.17
8B - CoT	0.10	0.26	0.18	0.18
8B - CCoT	0.12	0.32	0.16	0.20
8B - Ours	0.10	0.82	90	0.61
32B	0.12	0.30	0.16	0.19
32B - CoT	0.10	0.28	0.16	0.18
32B - CCoT	0.12	0.26	0.18	0.19
32B - Ours	0.12	0.86	0.92	0.63

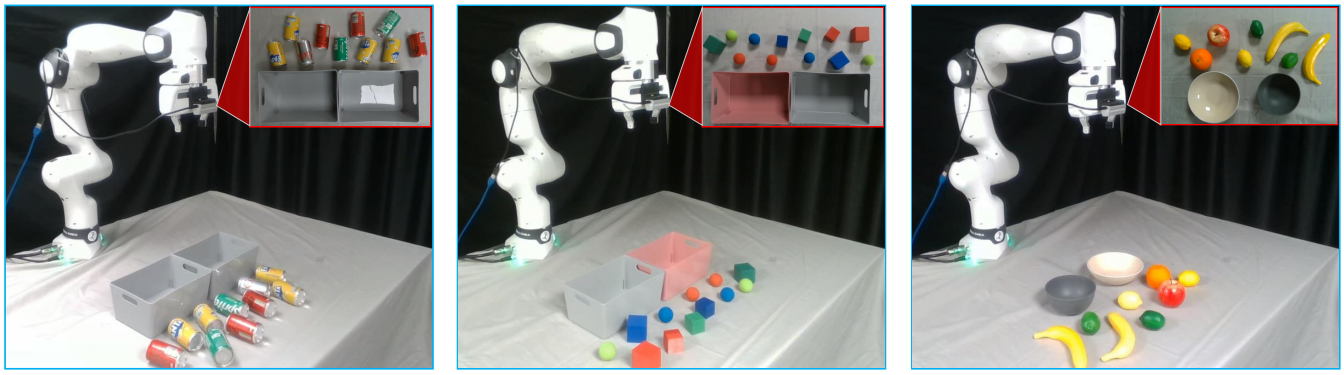
All algorithms exhibit limited performance on the **Initial** samples due to the inherent complexity of long-horizon, context-rich task planning (Table II). However, our proposed approach significantly outperforms the baselines in both the **Incomplete** and **Complete** settings, effectively mitigating failure cases during overall execution. These results demonstrate the effectiveness of the Evaluator in identifying and eliminating semantic and geometric errors, thereby improving task completeness. Moreover, by explicitly detecting incomplete plans and executions, our agent can recover from initial failures and progressively improve task completion through iterative replanning and execution.

V. Real-Robot Experiments

In this section, we investigate the performance of the proposed framework on real robotic tasks with Franka Emika Robot Arm. Specifically, we investigate on the performance of the proposed framework in failure recovery and dynamic environment.

A. Experiment Setup

An Intel RealSense D435i RGB-D camera is attached to the gripper, providing the visual input of the table top environment (Fig 4). The Planner and Evaluator run on a local Workstation with RTX 5090 GPU, producing a reliable plan. Meanwhile, the Executor is implemented via `franky`



User prompt: *Put all the yellow cans into the left tray and all the red cans into the right tray*

a) Cans

User prompt: *Put all the red objects into the pink tray and all the blue objects into the grey tray*

b) Cubes and balls

User prompt: *Put all the limes into the big bowl and all the lemons into the small bowl*

c) Kitchen

Fig. 4: Experimental settings of three distinct tasks including (a) `Cans`, (b) `Cubes and Balls` and (c) `Kitchen`. In each task, we intentionally create the collision or tamper the environment during execution, requiring the robot re-plan after the initial plan completion. **Note:** the scenario in these experiments are not included in Sec. IV.

library, allowing the low level control of primitive actions in the plan sequentially. Due to the limit of hardware, we use Qwen3-VL-8B to run for all of the experiments in this section. After each plan completion, the Planner is required to check the completeness of the task. The three tasks, including (a) `cans`, (b) `cubes and balls` and (c) `kitchen`, are evaluated five times with randomized initial location of the objects. Each task requires eight to six-teen primitive actions, and various semantic and spatial planing such as spatial location (left or right), colors, shape, and size. Moreover, we intentionally create the re-planing scenario such as collision or environment alteration during execution. We anonymously published our videos in the website.

Due to limited opensource work on replanning during execution due to collision or errors, we re-implement the ReplanVLM [38] and integrate to our framework. Specifically, we only develop ReplanVLM with task plan generation and use the same executor for low-level control. We also use Qwen3-VL instead of GPT4-V as in the original work.

B. Results

TABLE III: Real-robot Performance on Long-Horizon Context-Rich Task. We run ten times for each experiment scenario and report the average successful rate.

Model \ Settings	Cans	Cubes & balls	Kitchen	Average
8B - ReplanVLM	0.1	0.1	0.2	0.13
8B - Ours	**0.6**	**0.8**	**0.9**	**0.77**

As shown in Table III, our method significantly outperforms ReplanVLM across all evaluation settings, achieving an average performance of 77% compared to only 13% for ReplanVLM. This improvement can be attributed to our enhanced handling of incomplete state detection, as well as the role of the Evaluator in correcting plans to successfully complete long-horizon, context-rich tasks. While ReplanVLM has demonstrated effectiveness on short-horizon tasks, we observe that it struggles with semantic understanding under incomplete state information, leading to compounding errors and making task completion after failure recovery considerably more challenging.

VI. Conclusions

In this work, we propose a Completeness-Aware Task Planning and Execution framework that leverages pretrained vision–language models (VLMs) for long-horizon, context-rich manipulation control. Our approach generates task plans and iteratively detects and rectifies multiple types of planning and execution errors until the plans become feasible for low-level robot control. We further introduce a completeness-detection and replanning mechanism to address long-horizon, context-rich tasks, enabling the agent to continuously plan and execute actions until task completion. Experimental results show that our framework outperforms baseline methods on both semantic and completeness metrics, achieving an average success rate of 77% across three distinct tabletop scenarios. In future work, we aim to extend the framework to more dynamic environments, such as those involving moving target objects or physically grounded planning.

References

[1] A. Raza, I. A. Sumra, and A. Sattar, "Ai safety and trustworthiness: A survey," *Journal of Computing & Biomedical Informatics*, vol. 10, no. 01, 2025.

[2] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

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

[4] J. Zhang, D. Yao, R. Pi, P. P. Liang, and Y. R. Fung, "Vlm2-bench: A closer look at how well vlms implicitly link explicit matching visual cues," *arXiv preprint arXiv:2502.12084*, 2025.

[5] Z. He, S. Polisetty, Z. Fan, Y. Huang, S. Wu, and Y. R. Fung, "Mm-boundary: Advancing mllm knowledge boundary awareness through reasoning step confidence calibration," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025, pp. 16 427–16 444.

[6] Z. Yang, C. Garrett, D. Fox, T. Lozano-Pérez, and L. P. Kaelbling, "Guiding long-horizon task and motion planning with vision language models," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 16 847–16 853.

[7] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," *arXiv preprint arXiv:2201.07207*, 2022.

[8] A. Z. Ren, A. Dixit, A. Bodrova, S. Singh, S. Tu, N. Brown, P. Xu, L. Takayama, F. Xia, J. Varley *et al.*, "Robots that ask for help: Uncertainty alignment for large language model planners," *arXiv preprint arXiv:2307.01928*, 2023.

[9] V. B. Parthasarathy, A. Zafar, A. Khan, and A. Shahid, "The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities," *arXiv preprint arXiv:2408.13296*, 2024.

[10] X.-K. Wu, M. Chen, W. Li, R. Wang, L. Lu, J. Liu, K. Hwang, Y. Hao, Y. Pan, Q. Meng *et al.*, "Llm fine-tuning: Concepts, opportunities, and challenges," *Big Data and Cognitive Computing*, vol. 9, no. 4, p. 87, 2025.

[11] Y. Guo, Y.-J. Wang, L. Zha, and J. Chen, "Doremi: Grounding language model by detecting and recovering from plan-execution misalignment," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 12 124–12 131.

[12] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: From natural language instructions to feasible plans," *Autonomous Robots*, vol. 47, no. 8, pp. 1345–1365, 2023.

[13] Y. Xie, C. Yu, T. Zhu, J. Bai, Z. Gong, and H. Soh, "Translating natural language to planning goals with large-language models," *arXiv preprint arXiv:2302.05128*, 2023.

[14] P. Chen, Z. Wu, J. Sun, D. Wang, P. Zhou, N. Cao, Y. Ding, B. Zhao, X. Li *et al.*, "Alignbot: Aligning vlm-powered customized task planning with user reminders through fine-tuning for household robots," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 12 549–12 556.

[15] X. Fu, M. Zhang, P. Han, H. Zhang, L. Shi, H. Tang *et al.*, "What can vlms do for zero-shot embodied task planning?" in *ICML 2024 Workshop on LLMs and Cognition*, 2024.

[16] H. Guo, F. Wu, Y. Qin, R. Li, K. Li, and K. Li, "Recent trends in task and motion planning for robotics: A survey," *ACM Computing Surveys*, vol. 55, no. 13s, pp. 1–36, 2023.

[17] N. Wake, A. Kanehira, K. Sasabuchi, J. Takamatsu, and K. Ikeuchi, "Gpt-4v (ision) for robotics: Multimodal task planning from human demonstration," *IEEE Robotics and Automation Letters*, 2024.

[18] P. Zhi, Z. Zhang, Y. Zhao, M. Han, Z. Zhang, Z. Li, Z. Jiao, B. Jia, and S. Huang, "Closed-loop open-vocabulary mobile manipulation with gpt-4v," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 4761–4767.

[19] K. Shirai, C. C. Beltran-Hernandez, M. Hamaya, A. Hashimoto, S. Tanaka, K. Kawaharazuka, K. Tanaka, Y. Ushiku, and S. Mori, "Vision-language interpreter for robot task planning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 2051–2058.

[20] Y.-q. Jiang, S.-q. Zhang, P. Khandelwal, and P. Stone, "Task planning in robotics: an empirical comparison of pddl-and asp-based systems," *Frontiers of Information Technology & Electronic Engineering*, vol. 20, no. 3, pp. 363–373, 2019.

[21] X. Zhang, Z. Altaweel, Y. Hayamizu, Y. Ding, S. Amiri, H. Yang, A. Kaminski, C. Esselink, and S. Zhang, "Dkprompt: Domain knowledge prompting vision-language models for open-world planning," *arXiv preprint arXiv:2406.17659*, 2024.

[22] M. G. Arenas, T. Xiao, S. Singh, V. Jain, A. Ren, Q. Vuong, J. Varley, A. Herzog, I. Leal, S. Kirmani *et al.*, "How to prompt your robot: A promptbook for manipulation skills with code as policies," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 4340–4348.

[23] C. Li, J. Liang, A. Zeng, X. Chen, K. Hausman, D. Sadigh, S. Levine, L. Fei-Fei, F. Xia, and B. Ichter, "Chain of code: Reasoning with a language model-augmented code emulator," *arXiv preprint arXiv:2312.04474*, 2023.

[24] R. Kamoi, Y. Zhang, N. Zhang, J. Han, and R. Zhang, "When can llms actually correct their own mistakes? a critical survey of self-correction of llms," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 1417–1440, 2024.

[25] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[26] X. Wang, J. Chen, Z. Wang, Y. Zhou, Y. Zhou, H. Yao, T. Zhou, T. Goldstein, P. Bhatia, T. Kass-Hout *et al.*, "Enhancing visual-language modality alignment in large vision language models via self-improvement," in *Findings of the Association for Computational Linguistics: NAACL 2025*, 2025, pp. 268–282.

[27] J. Huang, S. Gu, L. Hou, Y. Wu, X. Wang, H. Yu, and J. Han, "Large language models can self-improve," in *Proceedings of the 2023 conference on empirical methods in natural language processing*, 2023, pp. 1051–1068.

[28] X. Zhang, B. Peng, Y. Tian, J. Zhou, L. Jin, L. Song, H. Mi, and H. Meng, "Self-alignment for factuality: Mitigating hallucinations in llms via self-evaluation," *arXiv preprint arXiv:2402.09267*, 2024.

[29] A. Kumar, V. Zhuang, R. Agarwal, Y. Su, J. D. Co-Reyes, A. Singh, K. Baumli, S. Iqbal, C. Bishop, R. Roelofs *et al.*, "Training language models to self-correct via reinforcement learning, 2024," *URL https://arxiv. org/abs/2409.12917*, vol. 2, no. 3, p. 4.

[30] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," *arXiv preprint arXiv:2203.11171*, 2022.

[31] J. Huang, Z. He, Y. Huang, S. Polisetty, Q. Wang, and Y. R. Fung, "Mac-tuning: Llm multi-compositional problem reasoning with enhanced knowledge boundary awareness," in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025, pp. 663–676.

[32] Y. Liu, J. Cao, Z. Li, R. He, and T. Tan, "Breaking mental set to improve reasoning through diverse multi-agent debate," in *The Thirteenth International Conference on Learning Representations*, 2025.

[33] D. Zhang, J. Lei, J. Li, X. Wang, Y. Liu, Z. Yang, J. Li, W. Wang, S. Yang, J. Wu *et al.*, "Critic-v: Vlm critics help catch vlm errors in multimodal reasoning," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 9050–9061.

[34] S. Pchelintsev, M. Patratskiy, A. Onishchenko, A. Korchemnyi, A. Medvedev, U. Vinogradova, I. Galuzinsky, A. Postnikov, A. K. Kovalev, and A. I. Panov, "Lera: Replanning with visual feedback in instruction following," *arXiv preprint arXiv:2507.05135*, 2025.

[35] X. Zhang, Y. Ding, S. Amiri, H. Yang, A. Kaminski, C. Esselink, and S. Zhang, "Grounding classical task planners via vision-language models," *arXiv preprint arXiv:2304.08587*, 2023.

[36] M. Skreta, Z. Zhou, J. L. Yuan, K. Darvish, A. Aspuru-Guzik, and A. Garg, "Replan: Robotic replanning with perception and language models," *arXiv preprint arXiv:2401.04157*, 2024.

[37] S. Wang, M. Han, Z. Jiao, Z. Zhang, Y. N. Wu, S.-C. Zhu, and H. Liu, "Llmˆ 3: Large language model-based task and motion planning with motion failure reasoning," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 12 086–12 092.

[38] A. Mei, G.-N. Zhu, H. Zhang, and Z. Gan, "Replanvlm: Replanning robotic tasks with visual language models," *IEEE Robotics and Automation Letters*, 2024.

[39] A. Lager, G. Spampinato, A. V. Papadopoulos, and T. Nolte, "Task roadmaps: speeding up task replanning," *Frontiers in Robotics and AI*, vol. 9, p. 816355, 2022.

[40] A. Pasricha, Y.-S. Tung, B. Hayes, and A. Roncone, "Pokerrt: Poking as a skill and failure recovery tactic for planar non-prehensile manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4480–4487, 2022.

[41] T. Pan, A. M. Wells, R. Shome, and L. E. Kavraki, "Failure is an option: task and motion planning with failing executions," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1947–1953.

[42] S. B. et al., "Qwen3-vl technical report," *arXiv preprint arXiv:2511.21631*, 2025.

[43] G. Team, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin, T. Matejovicova, A. Ramé, M. Rivière *et al.*, "Gemma 3 technical report," *arXiv preprint arXiv:2503.19786*, 2025.

[44] Y. Yao, T. Yu, A. Zhang, C. Wang, J. Cui, H. Zhu, T. Cai, H. Li, W. Zhao, Z. He *et al.*, "Minicpm-v: A gpt-4v level mllm on your phone," *arXiv preprint arXiv:2408.01800*, 2024.

[45] N. Carion, L. Gustafson, Y.-T. Hu, S. Debnath, R. Hu, D. Suris, C. Ryali, K. V. Alwala, H. Khedr, A. Huang *et al.*, "Sam 3: Segment anything with concepts," *arXiv preprint arXiv:2511.16719*, 2025.

[46] C. Mitra, B. Huang, T. Darrell, and R. Herzig, "Compositional chain of thought prompting for large multimodal models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024.