

## 1 A Detailed Task Suite

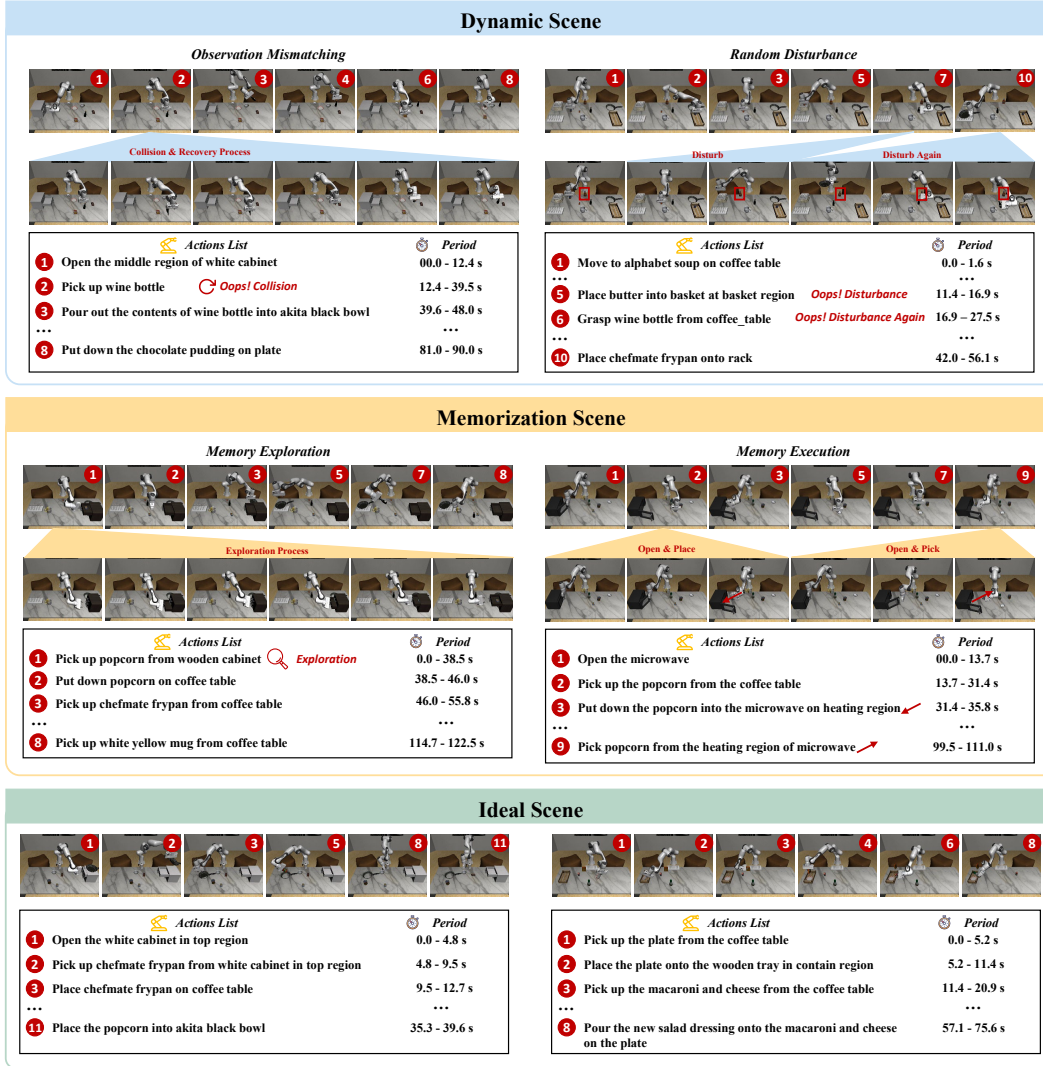


Figure 1: Example from RoboCerebra for different tasks.

- Fig. 1 illustrates concrete examples of our defined sub-task categories using the RoboCerebra benchmark. These scenarios are designed to evaluate the capabilities of System 2 models in memory retention, adaptive planning, and disturbance recovery within long-horizon manipulation settings. We group these into three broader categories for analysis:
- (1) Dynamic Scene includes Observation Mismatching and Random Disturbance sub-tasks. These emphasize robustness to environmental inconsistencies and unexpected changes. For example, Systems must recover from collisions or update plans after items are displaced mid-task.
  - (2) Memorization Scene covers both Memory Exploration and Memory Execution. In the exploration phase, VLMs actively probe the environment to build internal representations (e.g., checking cabinet contents). During execution, perceptual cues are removed and they rely on memory to complete the task correctly.
  - (3) Ideal Scene serves as a static, fully observable control condition. It reflects performance in the absence of memory or disturbance constraints and establishes a baseline for comparison.
- Each scenario is visualized as a sequence of image snapshots, with numbered action steps, annotated time segments, and accompanying descriptions of the behavior of System 2. These task compositions

17 holistically assess core abilities such as long-horizon memory, temporal reasoning, causal inference,  
 18 and goal-directed manipulation in evolving environments.

## 19 B Results on Memory Tasks

Table 1: **Evaluation of different VLMs** under the Hierarchical Framework, including Memory Exploration Success Rate ( $SR_{Exp.}$ ), Exploration-only Success Rate ( $SR_{Exp.-only}$ ), Exploration Efficiency ( $\eta_{Exp.}$ ), Memory Execution Success Rate ( $SR_{Exe.}$ ), and Decision Accuracy ( $Acc_{Dec.}$ ).

VLM	$SR_{Exp.} \uparrow$	$SR_{Exp.-only} \uparrow$	$\eta_{Exp.} \uparrow$	$SR_{Exe.} \uparrow$	$Acc_{Dec.} \uparrow$
Qwen2.5-VL	3.54	50.0	0.17	12.39	10.0
GPT-4o	<b>9.06</b>	<b>80.0</b>	<b>0.32</b>	<b>17.83</b>	<b>30.0</b>

20 To further evaluate the role of System 2 reasoning in Memory-based manipulation tasks, we design a  
 21 set of fine-grained experiments to analyze how different reasoning capabilities affect performance  
 22 under a unified Hierarchical Framework. The detailed memory mechanism is illustrated in Alg. 1  
 23 Beyond the overall success rates of the Memory Exploration and Memory Execution tasks, we  
 24 introduce several intermediate metrics that assess the internal reasoning process:

25 To evaluate the ability of System 2 to discover the target object during the exploration phase, we  
 26 utilize the **Exploration-only Success Rate** ( $SR_{Exp.-only}$ ), which measures whether the VLMs  
 27 successfully locates the object regardless of overall task completion.

28 To assess the efficiency of object discovery during exploration, we utilize the **Exploration Efficiency**  
 29 ( $\eta_{Exp.}$ ), which jointly considers the correctness and conciseness of the predicted exploration plan.  
 30 Specifically, we define it based on the normalized overlap between the predicted plan  $\pi_G$  and the  
 31 ground truth plan  $\pi_{GT}$  for each task  $i$ , referred to as the completeness of the plan:

$$Comp_{Exp.} = \frac{|\pi_G \cap \pi_{GT}|}{|\pi_{GT}|} \quad (1)$$

32 The exploration efficiency is then computed by normalizing the completeness score by the length of  
 33 the predicted plan and averaging over all  $N$  tasks:

$$\eta_{Exp.} = \frac{1}{N} \sum_{i=1}^N \frac{Comp_{Exp.}}{|\pi_G|} \quad (2)$$

34 To measure the correctness of high-level decision-making during execution, we utilize the **Decision**  
 35 **Accuracy** ( $Acc_{Dec.}$ ), which quantifies the proportion of correct identifications of the target object  
 36 and appropriate plan selections.

---

### Algorithm 1 Task-Aware Memory Mechanism

---

- 1: **Input:** Task specification  $\mathcal{T}$ , current environment state  $s$
  - 2: **Output:** Success or Failure
  - 3: Determine whether  $\mathcal{T}$  requires memory (e.g., exploration, history tracking)
  - 4: Define goal condition  $G$  from  $\mathcal{T}$
  - 5: Identify goal-relevant steps  $\{s^{(1)}, s^{(2)}, \dots, s^{(k)}\}$
  - 6: Generate complete sub-plan  $\pi_G$  to achieve  $G$
  - 7: **for** each step  $a_t$  in  $\pi_G$  **do**
  - 8:   Execute  $a_t$  in environment
  - 9:   **if**  $\psi(s_t) = \text{True}$  **and**  $s_t$  satisfies goal  $G$  **then**
  - 10:     **return** Success
  - 11:   **end if**
  - 12: **end for**
  - 13: **return** Failure
-

## 37 C Prompt Details of Task Generation

38 To support structured generation of long-horizon manipulation tasks, we employ a multi-stage  
39 prompting pipeline to elicit detailed, consistent, and actionable representations from large language  
40 models. Each stage is designed to incrementally refine the task definitions, ensuring coherence,  
41 atomicity, and alignment with robotic capabilities. The full set of prompt templates used in our  
42 pipeline is shown in Fig. 2–6.

43 Specifically, Fig. 2 shows the Task and Steps Generation Prompt, which outlines the high-level goal  
44 and asks the model to break it down into discrete procedural steps. Following this, the Task and Steps  
45 Verification Prompt verifies the semantic and logical validity of these decomposed steps (Fig. 3).  
46 Once verified, we proceed with Primitive Actions Generation (Fig. 4), where each high-level step is  
47 further grounded into low-level robot-executable primitives. Subsequently, the Affordance Generation  
48 Prompt (Fig. 5) augments the plan with relevant object affordances, aiding visual grounding and  
49 interaction feasibility. Finally, a Format Verification Prompt (Fig. 6) ensures that the entire structured  
50 output adheres to the expected schema, enabling seamless downstream parsing and simulation.

**Stage 1: Task and Step Generation Prompt**

You are an imaginative robotic arm scene designer. You need to construct a long-sequence task based on the item list provided below, which includes multiple steps (no more than **8 steps**), with each step being **an action** (such as "open the xxx", "Pick up the xxx from xxx", "open the xxx" etc.). All the items were initially placed on the {workspace} or stored inside other objects. Note:

1. Except for items that offer executable actions, other items can only be picked up, moved, and put down. You are not allowed to open any food packaging boxes. If you wish to pour out any liquid from the container, simply pour it out—there's no need to open the packaging. The scene does not contain any real liquids.
2. This long sequence of tasks does not necessarily require a complete chain of events.
3. You don't have to use all the items—please prioritize ensuring that the constructed task logic is reasonable.
4. Can\_fit is a boolean value, which indicates whether the object can be stored in any storage container.
5. In 'Related object', list and only list all the NAME of objects involved in the step, including the object being operated on and the object being moved.
6. DONOT output any irrelevant replies. The format of the replies is as follows:  
Task: xxx  
Step: xxx  
Related Objects: xxx  
Step: xxx  
Related Objects: xxx  
....  
The item table is as follows:  
Item Category Action Affordance Can\_fit  
....

Figure 2: Task and Steps Generation Prompt.

## 51 D Prompt Details of Hierarchical Framework

52 To support reasoning over long-horizon manipulation tasks involving memory and dynamic scene  
53 understanding, we adopt a hierarchical prompting framework that decomposes planning into seman-  
54 tically modular components. This design enables the VLMs to perform goal decomposition, plan  
55 generation, and state tracking in a structured and interpretable manner. Fig. 7 to 9 illustrate the key  
56 prompts used in this hierarchical process.

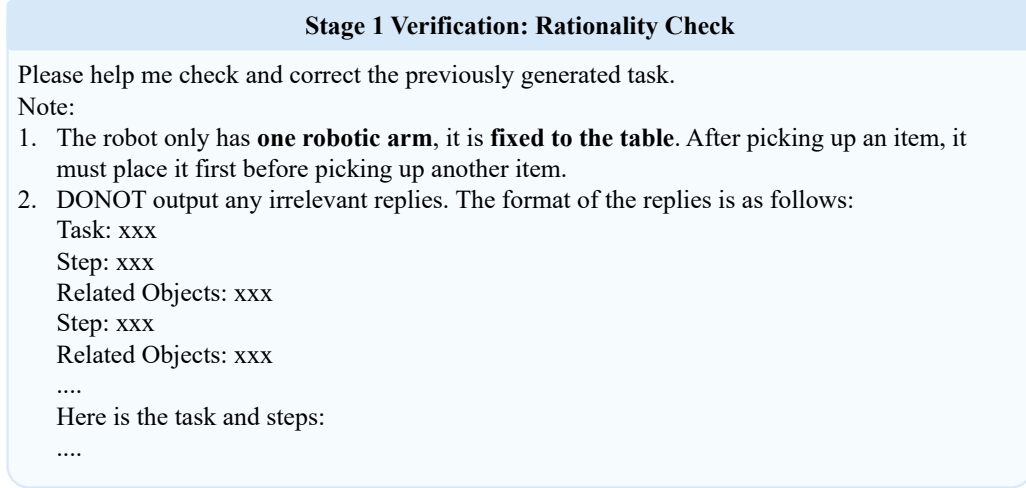


Figure 3: Task and Steps Verification Prompt.

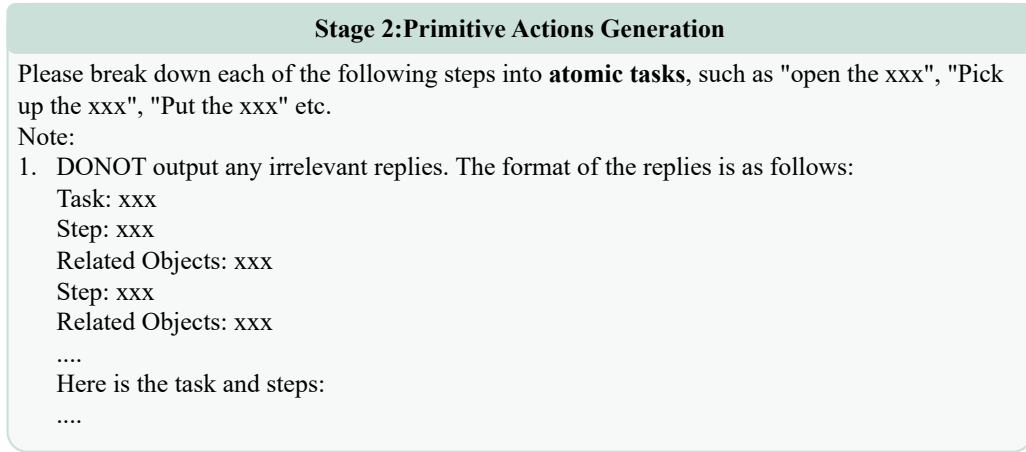


Figure 4: Primitive Actions Generation Prompt.

57 As shown in Fig. 7, the VLM Planner Prompt guides a vision-language model to generate high-level  
58 plans based on multimodal inputs, including current visual observations and task descriptions. This  
59 prompt leverages the VLM’s capability to contextualize semantic goals within visual environments.

60 Fig. 8 presents the Memory-Related Goal Generation Prompt, which is used to infer intermediate or  
61 hidden sub-goals that depend on past interactions or occluded states. This is particularly important in  
62 scenarios where successful execution requires recalling previously explored regions or remembering  
63 the contents of closed containers.

64 Finally, the Plan and Memory Updating Prompt (Fig. 9) enables continual re-planning by integrating  
65 updated perceptions and internal memory states. This prompt ensures that the VLMs maintains  
66 coherence between its execution state and the evolving environment, allowing it to revise intentions  
67 and recover from deviations or external disturbances.

## 68 E Case Study on Planning

69 To better understand the planning capabilities of different vision-language models, we conduct  
70 a qualitative comparison across GPT-4o [1] (GPT), Qwen2.5-VL [2] (Qwen), and LLaVA-Next-  
71 Video [3] (LLaVA) on complex household manipulation tasks, as illustrated in Fig. 10 and Fig. 11.

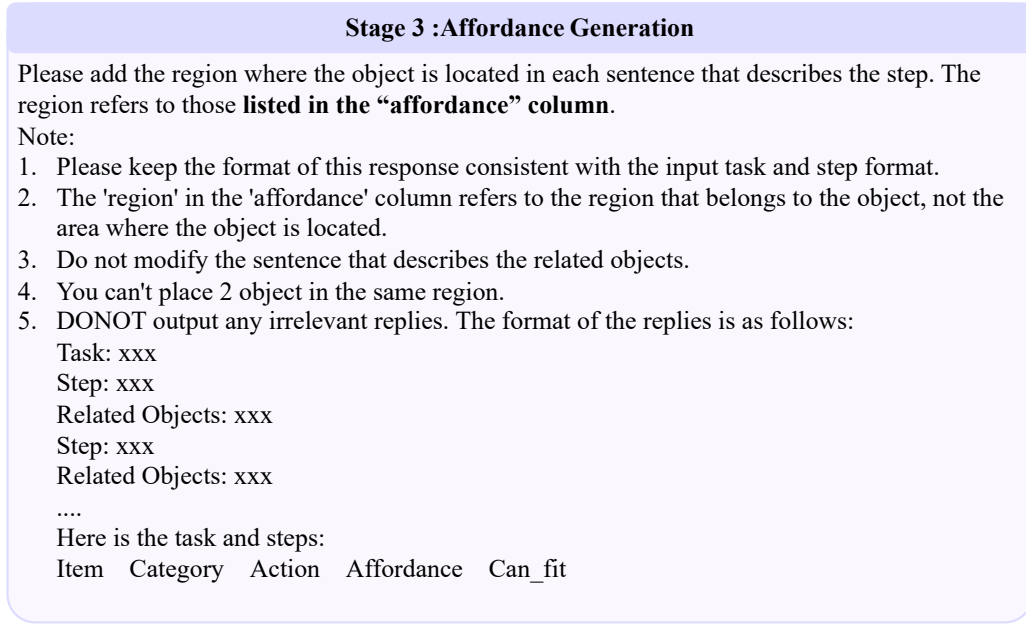


Figure 5: **Affordance Generation Prompt.**

72 These examples highlight notable differences in plan completeness, action granularity, and semantic  
73 correctness.

74 GPT consistently generates the most detailed and coherent plans across both tasks. In Fig. 10,  
75 GPT not only identifies the correct goal ("placing the plates between the knife and fork") but also  
76 incorporates contextual spatial cues and restores objects to their original positions (e.g., placing the  
77 wine bottle back to the far-left side), reflecting strong planning fidelity and environmental awareness.  
78 Similarly, in Fig. 11, GPT successfully decomposes a long-horizon task into logically ordered steps  
79 while preserving semantic consistency across diverse object types and destinations, demonstrating  
80 robustness in long-sequence planning.

81 Qwen produces generally correct but less detailed plans. In both examples, Qwen omits several con-  
82 textual elements, such as spatial descriptors or placement constraints, which reduces interpretability  
83 and precision. Notably, in Fig. 11, it fails to include important object transitions (e.g., placing the  
84 mug into the tray), indicating partial task understanding and missing intermediate steps.

85 LLaVA, in contrast, struggles with both semantic accuracy and action decomposition. It introduces  
86 several unnecessary or incorrect steps—such as manipulating objects irrelevant to the goal or reversing  
87 object trajectories. For example, it incorrectly places the white-yellow mug into the wooden tray from  
88 the wrong starting location and includes object movements not specified in the task. This suggests  
89 that LLaVA lacks grounded task representations and often fails to maintain consistency with the  
90 initial instructions.

## 91 **F Case Study on Sub-Plan**

92 Fig. 12 presents a comparison of sub-task decomposition quality between GPT and Qwen, focusing  
93 on the process of locating and retrieving butter from a multi-compartment cabinet. The results  
94 highlight distinct differences in the models’ ability to reason hierarchically and generate coherent  
95 sub-plans.

96 GPT demonstrates a thorough and methodical decomposition strategy. It systematically explores all  
97 cabinet compartments in a top-down order, includes both opening and closing actions, and maintains  
98 logical sequencing. This behavior reflects robust sub-task planning and a clear understanding of both  
99 spatial structure and task completion integrity. In contrast, Qwen exhibits incomplete and less stable  
100 sub-plan generation. It assumes the butter is in the first compartment without exploration and omits

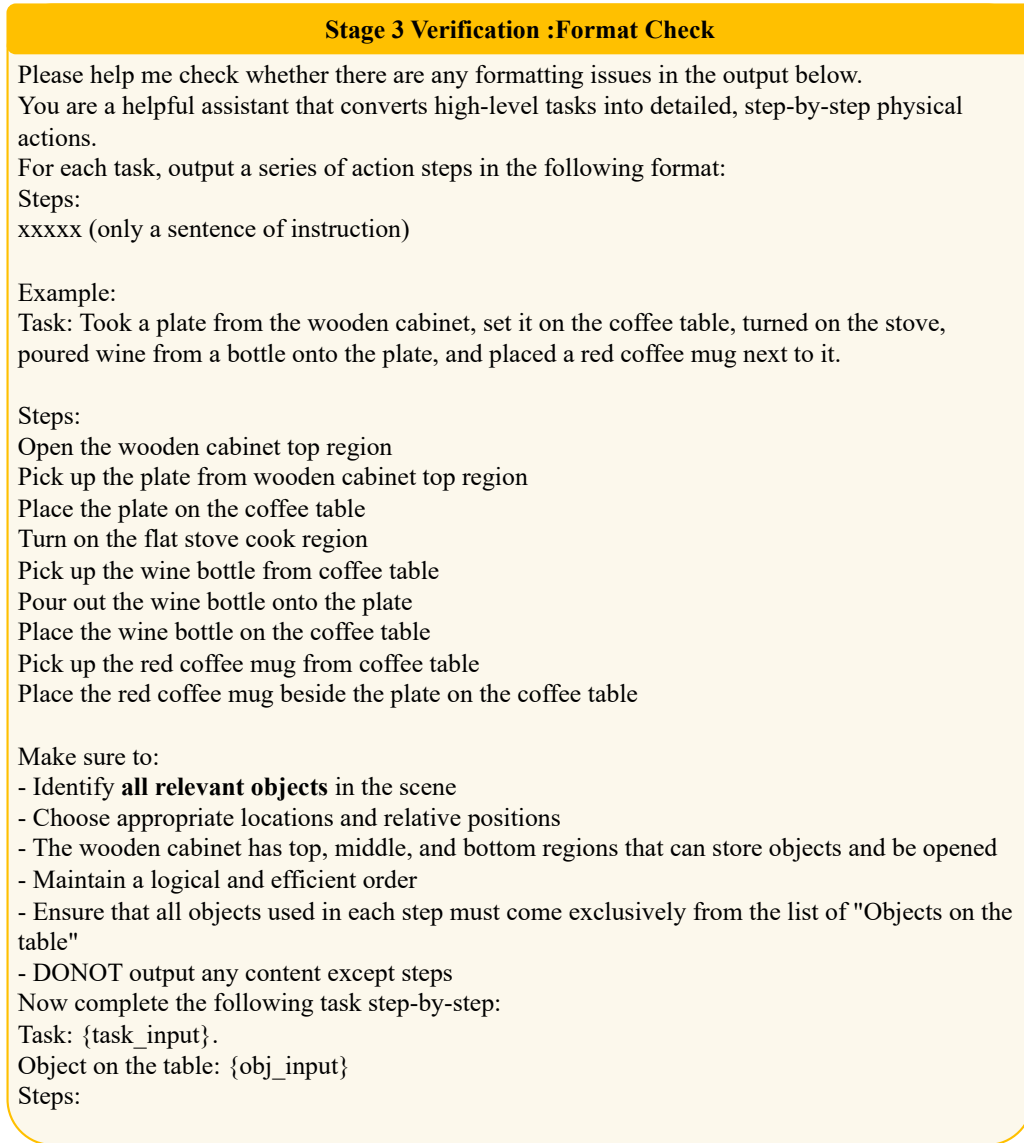


Figure 6: **Format Verification Prompt.**

101 intermediate or fallback steps entirely. This leads to a plan that may work in specific instances but  
 102 lacks generality and reliability in realistic or uncertain environments.

## 103 G Case Study on Memory Tasks

104 We evaluate models' capabilities to reason about memory-related goals and update task completion  
 105 status accordingly. As shown in Fig. 13 and 14, we compare GPT and Qwen on two representative  
 106 settings: memory exploration and memory execution.

107 In the memory exploration task (Fig. 13), the VLMs need to track visual progress and determine  
 108 when a goal—retrieving butter from a cabinet—has been fulfilled. GPT demonstrates clear task  
 109 completion awareness, identifying in the third step that the butter has been found inside the open  
 110 cabinet and marking the goal as completed. This reflects its ability to connect visual observations  
 111 with memory-dependent goals and dynamically update task status. In contrast, Qwen remains task  
 112 completion-unaware, repeatedly denying goal fulfillment despite the presence of the butter in view.  
 113 This suggests difficulty in grounding visual evidence against prior memory constraints.

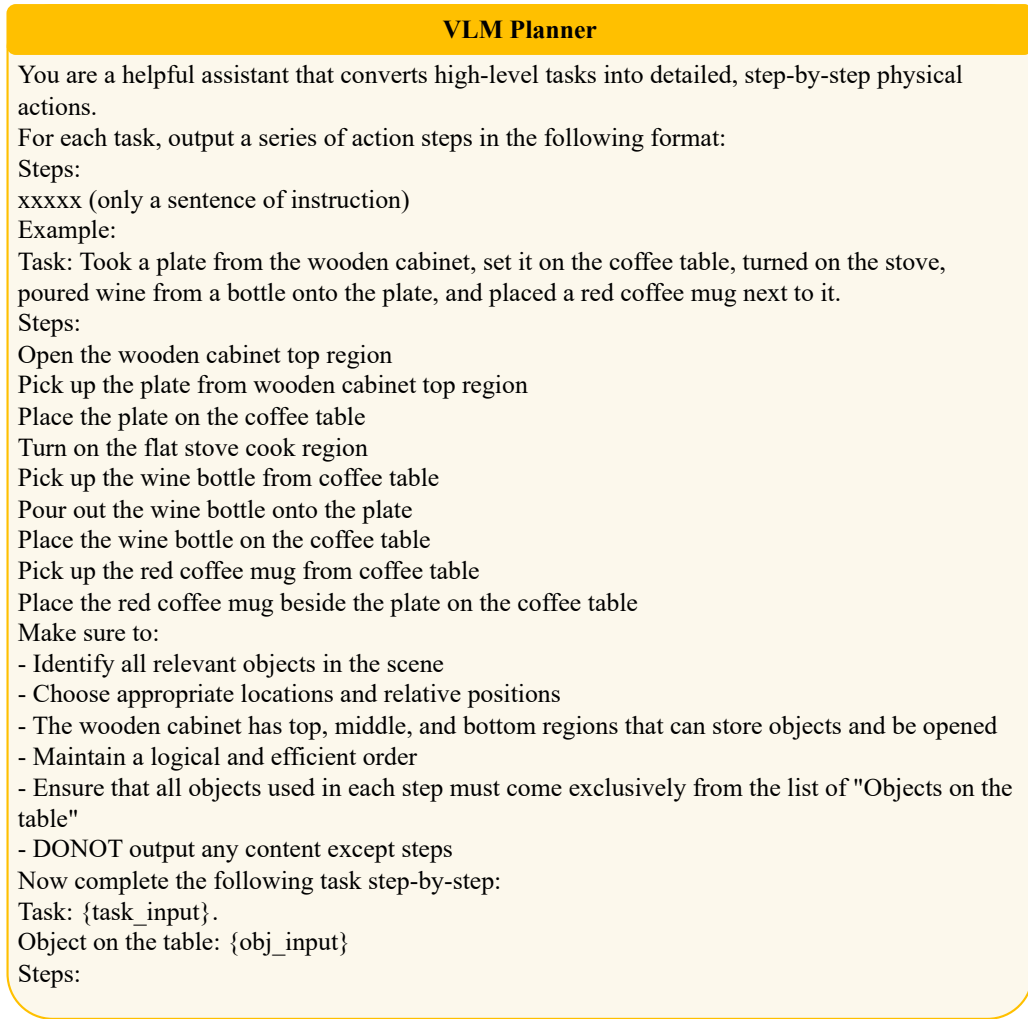


Figure 7: VLM planner Prompt.

114 In the memory execution task (Fig. 14), the goal is to check whether a previously interacted cabinet  
 115 contains any remaining objects. GPT again shows strong reasoning, correctly concluding the goal is  
 116 satisfied when the cabinet is observed to be empty. Its output highlights an understanding of absence  
 117 as valid evidence. On the other hand, Qwen fails to interpret the empty cabinet scene as sufficient for  
 118 task completion, citing uncertainty and lack of confirmation. This highlights a limitation in negative  
 119 inference, where models must reason not just over what is visible, but also over what is not.

## 120 References

- 121 [1] OpenAI. GPT-4o system card, 2024.
- 122 [2] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao Song, Kai Dang, Peng Wang, Shijie  
 123 Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.
- 124 [3] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next:  
 125 Improved reasoning, ocr, and world knowledge, January 2024.

**Hierarchical Framework: Memory Related Goal Generation**

You are a helpful assistant that can decide whether a task requires memory to finish.

Generate a goal that requires memory to complete — for example, it might involve searching through multiple compartments or drawers to find an object, where the agent needs to remember which areas have already been searched to avoid redundant actions.

Make sure that:

- Identify all relevant objects in the scene
- The wooden cabinet has top, middle, and bottom regions that can store objects and be opened

Example1:

-Task:  
Take the white bowl from the wooden cabinet, pour in the chocolate pudding and cookies, and put it on top of the wooden cabinet.

-Output Format:  
MEMORY RELATED GOAL:Take the white bowl in the cabinet.

-Reason for Output(DO NOT output this part):  
This instruction guides the user to complete a goal-oriented task. To achieve the goal, the user must search through the top, middle, and bottom compartments of the wooden cabinet. Since the red mug is located in only one of the compartments, the user needs to avoid opening the same compartment twice. This encourages careful planning and memory use to efficiently complete the task.

Example2:

-Task:  
Task: After putting the butter and mug on the rack, pour in the milk, and check the wooden cabinet top region for any previously handled objects, placing them on the table if found.

-Output Format:  
MEMORY RELATED GOAL:Check the wooden cabinet top region for any previously handled objects, placing them on the table if found.

-Reason for Output(DO NOT output this part):  
To achieve the goal, the user must remember previously handled objects.

Please answer and ONLY ANSWER MEMORY RELATED GOAL!DO NOT OUTPUT REASON!

Here is the task:  
{{task}}

Figure 8: Memory Related Goal Generation Prompt.



### Hierarchical Framework: Updating Plan & Memory

Analyze this image and determine if the following Memory related goal has been completed:

Current step: {step\_text}

Previously completed steps: {completed\_steps\_text}

Memory related goal: {memory\_related\_goal}

Based on the image, has the Memory related goal been completed? Answer with 'Yes' or 'No' and explain your reasoning.

Note:

1. When you check cabinet top region for any previously handled objects, an object is considered previously handled if it also appears on the table.
2. When you check cabinet top region for any previously handled objects, consider the task complete if no objects are found inside the cabinet.
3. As long as you see any object in the opened cabinet, you can consider that you have found the item you need.
4. The butter looks like a red box.
5. The cream cheese looks like a blue box.

Figure 9: Plan & Memory Updating Prompt.

### Comparison of Plan Quality

**Task:** Place the plates neatly between the knife and fork, and pour all the snacks and drinks into the plates.

**GPT:**

...

Step: Place the popcorn box back to the far-right side of the table

Step: Pick up the wine bottle from the far-left side of the table

Step: Pour wine onto the plate positioned between the knife and fork

Step: Place the wine bottle back to the far-left side of the table ✓ *More complete action descriptions*

**Qwen:**

...

Step: Place the popcorn box back on the table.

Step: Pick up the wine bottle.

Step: Pour the wine into the plate.

**LLaVA:**

Step: Pick up the dining set from the wooden cabinet. ✗ *Unnecessary steps*

Step: Place the dining set on the table. ✗ *Unnecessary steps*

...

Step: Arrange the plates neatly between the knife and fork. ✗ *Incomplete decomposition*

Figure 10: Comparison of Planning Quality.

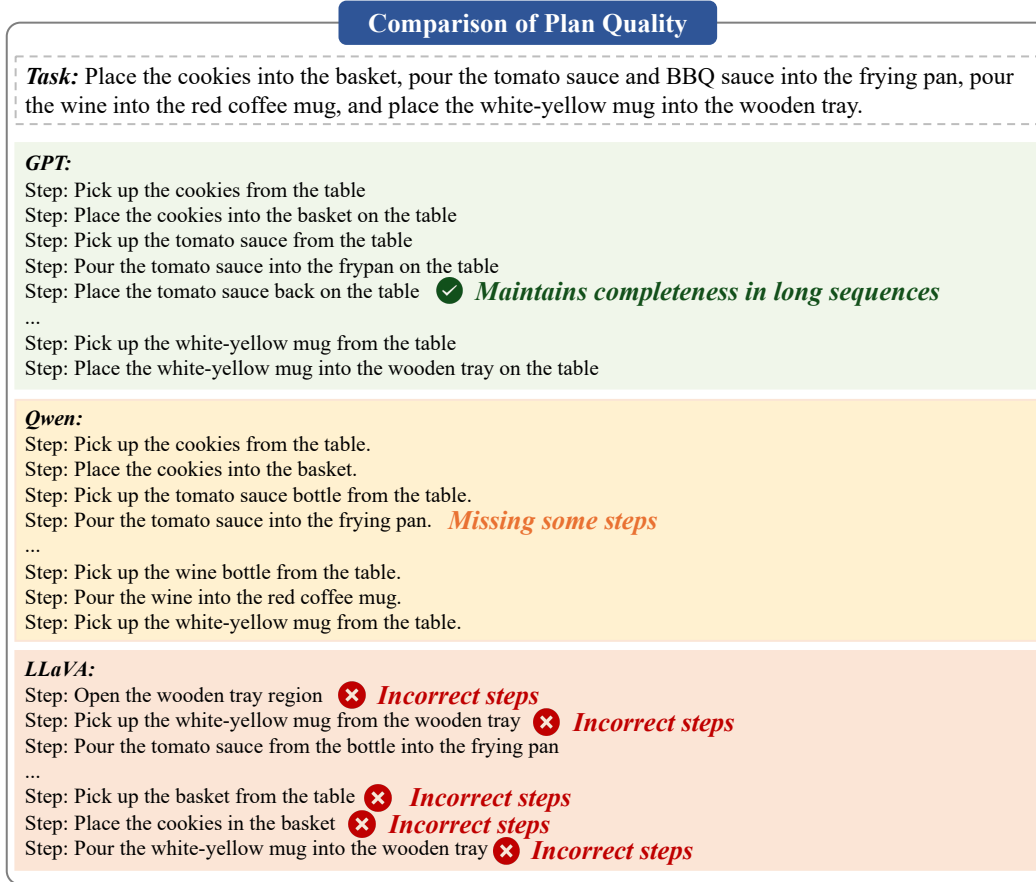


Figure 11: Comparison of Planning Quality.

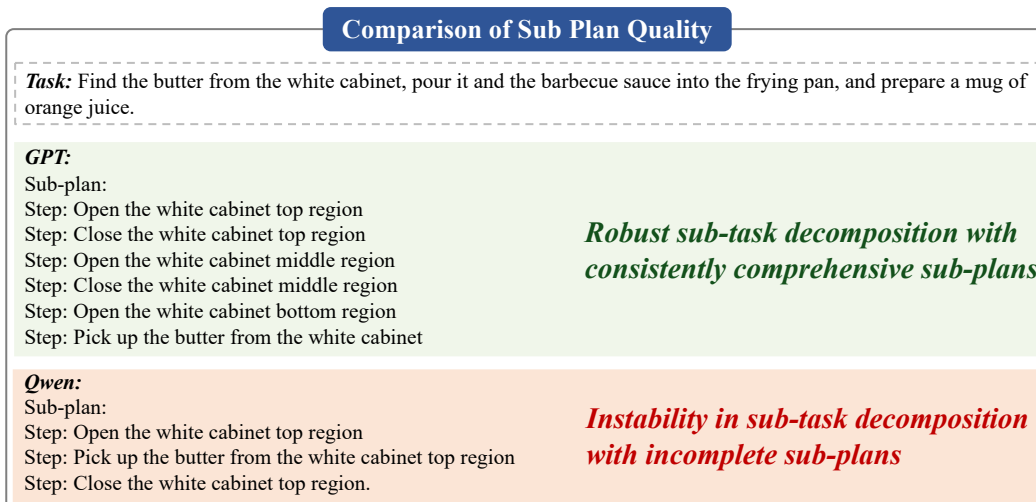
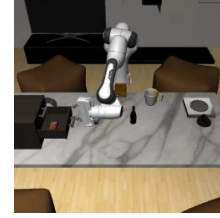
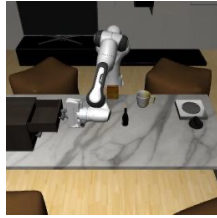


Figure 12: Comparison of Sub Plan Quality

### Comparison of Task Completion Update Capabilities in Memory Exploration Task

**Task:** Take the butter from the wooden cabinet, melt it in a Chefmate frypan, and serve wine in a white-and-yellow mug.



**GPT:**

Step 1:

No, the Memory related goal has not been completed.

...

Step 2:

No, the Memory related goal has not been completed.

...

Step 3:  **Task completion-aware**

Yes, the Memory related goal has been completed. In the image, the wooden cabinet middle region is open, and there is a red box inside the cabinet, which matches the description of the butter. Therefore, the butter has been found and the goal achieved.

**Qwen:**

Step1:

No

...

- Therefore, the Memory related goal "Take the butter from the wooden cabinet" has not been completed.

Step 2:

No

...

Since no objects are found inside the open cabinet, the goal has not been completed.

Step 3:  **Task completion-unaware**

No

Reasoning:

...

Since one of the key elements to completing the goal (finding the butter) has not been visually confirmed in the current image, the goal has not been fully completed.

Figure 13: Comparison of Task Completion Update Capabilities in Memory Exploration Task.

### Comparison of Task Completion Update Capabilities in Memory Execution Task



**Task:** After putting the butter and mug on the rack, pour in the milk, and check the wooden cabinet top region for any previously handled objects, placing them on the table if found.

**GPT:**

Step 1: ✅ **Correct inference in empty-cabinet scenarios**

Yes, the Memory related goal has been completed. The cabinet top region appears to be open, and there are no objects visible inside the cabinet. The task is considered complete if no objects are found inside the cabinet, which is the case here.

**Qwen:**

Step 1: ❌ **Difficulty recognizing empty-cabinet scenarios**

No

Reasoning:

...

- However, we do not know if the check for the top region of the cabinet was completed. There are no explicit signs, such as an empty cabinet or the placement of objects, to support the claim that the check was finished.

...

Therefore, based on the information provided in the image, the Memory related goal cannot be conclusively stated as completed.

Figure 14: Comparison of Task Completion Update Capabilities in Memory Execution Task.