This supplementary material provides additional details on the proposed method and used prompts that could not be included in the main manuscript due to page limitations.

Specifically, this appendix is organized as follows:

- Sec. A provides additional details on the model architecture and reinforcement learning training.
- Sec. B presents details on the prompts used for querying VLMs.
- Sec. C presents details for the experimental environment.

A. Model Architecture and Training Details

We use an object-centric transformer-based network architecture in our RL agent, similar to the one used in ExploRLLM [20]. As illustrated in Fig. 9, the visual embeddings, end-effector 6-D positions, and object bounding boxes are concatenated for each task-related object. As we concentrate on reward generation in this work, the bounding boxes used here are obtained from the simulators for all methods, following [20]. When computing rewards, we still utilizes SAM2 to obtain the bounding boxes without any supervision. Then, these features are input into the self-attention module to extract the overall features of the current frame. During the training phase, the convolutional neural networks (CNN) and the self-attention network are updated concurrently with the policy and value networks. We set the temperature α=0, discount factor $\gamma = 0.99$, critic update $\tau = 0.01$ in SAC. You can also find these details in our open-source code repository: https://github.com/nuomizai/T2VLM

B. Prompts for Querying VLMs

Chain-of-thought Task Relevant Object Detection. Detecting task-relevant objects poses a significant challenge that requires domain knowledge. Leveraging the strong visual captioning and reasoning abilities of VLM, we generate these objects using chain-of-thought (CoT) prompting, building on previous work [28, 33]. Below, we provide an example in Fig. 10 illustrating how to identify task-related objects with CoT prompts.

Prompt for Sub-goal Generation. As shown in Fig. 11, we provide the prompts for sub-goal generation. Specifically, we input task descriptions specified by the users and task-related object names obtained from Fig. 10 into user prompts for sub-goal generation. These prompts allow the VLM to generate spatial-aware sub-goals that can be tracked with Bayesian filters.

Typically, these prompts effectively guide the VLM to generate correct subgoals, provided the initial image captures all task-relevant objects. However, in challenging cases—such as when objects are overlapping or partially hidden—VLMs may fail to recognize key items, leading to false or incomplete subgoals. We find that a simple N-step re-check during initialization can effectively mitigate this issue. As shown in Fig. 8, in a real-world sorting task, the VLM initially merges the apple and blue bowl due to overlap, resulting in missing subgoals. A second query during random exploration—using SoM prompting [41] with numbered labels and bounding boxes from T^2-VLM —successfully recovers the blue bowl and updates the subgoals. This demonstrates the potential of our method to improve robustness in scenarios where VLMs may overlook relevant objects.

Prompt for Goal Completion Status Identification. The prompts for identifying the completion status of each subgoal are shown in Fig. 12. A complete subgoal completion status identification process for the Place-same-color task is visualized in Fig. 13.

C. Experiments

C.1. Environment Details

Training Environments. To support online RL training, we designed four robot manipulation tasks based on the CLIPort and CALVIN benchmarks. RL algorithms are trained for 400 episodes on CLIPort tasks and 2,000 episodes on CALVIN tasks, due to CALVIN's larger action space and longer execution steps. Training may end early due to task success or unrecoverable failures. In CLIPort, no unrecoverable failures occur. In CALVIN, an episode terminates if the object falls off the desk.

Testing Environments. For testing the baselines' recovery capability, we modified the training environments by introducing environment changes or robot execution failures. For all methods in Tab. 1, we measure the average recovery success rate and recovery meta steps over 20 trials.

C.2. Action Space

In the CLIPort benchmark, following the settings of SayCan [1], the RL agent needs to simultaneously determine the objects for picking and placing. Therefore, the action space in CLIPort is $\mathcal{A} = [a_1, a_2]$, where a_1 is the total number of picking objects and a_2 is the total number of placing objects. In the CALVIN benchmark, the action space consists of a list of finite compositions combining skills and objects. Specifically, the action space of each task is shown as follows:

- Cleanup-desk task: $\mathcal{A}$=[[PICK, RED], [PICK, BLUE], [PLACE, DRAWER], [PULL, HANDLE], [PUSH, HANDLE]]
- Make-line task: $\mathcal{A}$=[[PICK, BLUE], [PLACE2LEFT, BLUE], [PLACE2RIGHT, BLUE], [PICK, RED], [PLACE2LEFT, RED], [PLACE2RIGHT, RED],[PICK,

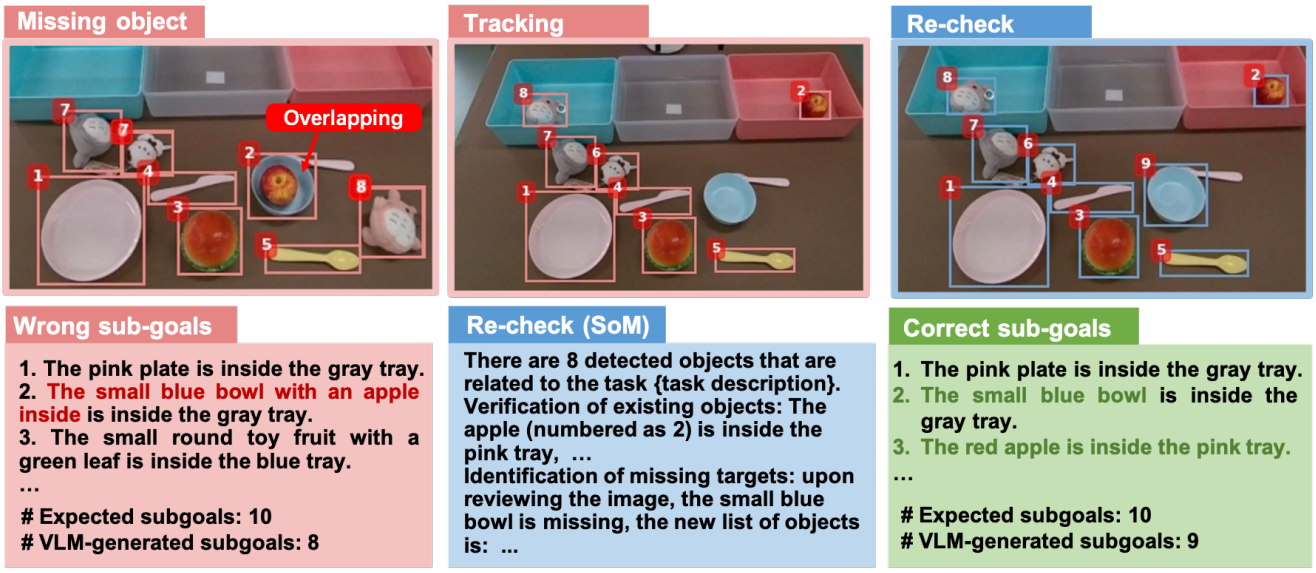


Figure 8. **Re-check VLM initialization with Set-of-Mark (SoM) prompting in a long-horizon real-world task.**

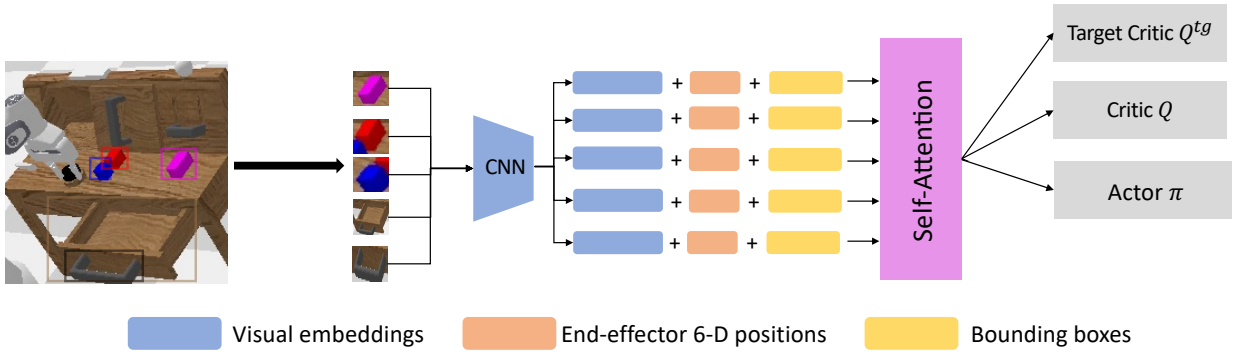


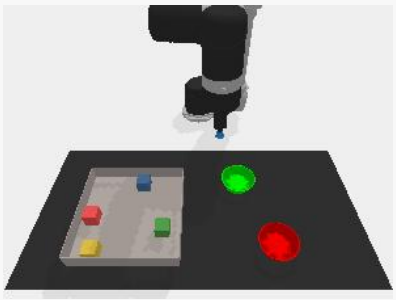
Figure 9. **The network architecture for RL agent.**

GREEN], [PLACE2LEFT, GREEN], [PLACE2RIGHT, GREEN]]

Primitive Skill Parameterization. The primitive skill considered in simulators is parameterized as SKILL[$\boldsymbol{p}$], where $\boldsymbol{p}$=[x, y, z] specifies the 3D coordinates of the target position.

Prompts for chain-of-thought object detection

User input Task Description:
Place the cube from gray tray into matching colored bowl; cubes without a matching bowl should stay in place.
VLM Input:



This is a task related to robot arm manipulation. Pay a comprehensive attention to the provided picture and detect all the objects in detail shown in this image.

VLM Output:
Let's identify all the objects present:

- Robot Arm. The black robotic arm is positioned above the gray tray, indicating it is used for picking and placing objects.
- Gray Tray. The gray tray contains four cubes of different colors:
  - Red Cube
  - Blue Cube
  - Green Cube
  - Yellow Cube
- Colored Bowls. There are three colored bowls on the table:
  - Green Bowl: Positioned to the right of the gray tray.
  - Red Bowl: Positioned further to the right of the green bowl.
  - No Blue or Yellow Bowls: These colors are missing, meaning there are no bowls for the blue and yellow cubes.
- Table Surface. The objects (gray tray and bowls) are placed on a flat, dark surface which serves as the workspace for the robot arm.

VLM Input:
Assume you are an expert of robot arm control filed with high intelligence. Your goal is to achieve this task: Place the cube from the gray tray into the matching colored bowl; cubes without a matching bowl should stay in place. Refer to the previous description for the objects shown in image, If the robot arm wants to achieve this task, what task-related objects need to be considered? Please print each object name with specific color like "white cube". Robot arm could be ignore. Your answer needs to be as detailed as possible to avoid confusion. The output format is a list string: ["XX", ...]

VLM Output:
["a red cube", "a green cube", "a blue cube", "a yellow cube", "a gray tray", "a green bowl", "a red bowl"]

Figure 10. **Prompts for chain-of-thought object detection.**

Prompts for sub-goals generation

User Input Task Description:
Place the cube from gray tray into matching colored bowl; cubes without a matching bowl should stay in place.

System Prompt:

You are an intelligent vision-language assistant agent operating within a virtual environment. Your primary objective is to achieve specified goals by breaking down complex tasks into manageable sub-goals that describe the status of objects at various stages.

When you receive a task description, decompose it into a set of clear and concise sub-goals that will enable you to accomplish the task. Ensure that each sub-goal is complete and accurate to prevent task failure due to missing information about the status of relevant objects.

To describe the state of target objects accurately, use the spatial relationship terms like: *inside, on top of, on the bottom of, on the left of, on the right of* and so on. Additionally, use the motion terms like *open, closed* and so on to describe object status.

Make sure that each description captures the essential intermediate status needed to complete the task.

User Prompt:

Examples:

Task: Place the red cube inside the red bowl.
Relevant Objects: Red cube, red bowl.
Sub-Goals: 1. The red cube is inside the red bowl.

Task: Stack three cubes in the order of red, black, and blue, with the red cube on top.
Relevant Objects: Red cube, black cube, blue cube.
Sub-Goals: 1. The black cube is on top of the blue cube.
2. The red cube is on top of the black cube.

Now, decompose the following task using the provided visual information:
Task: Place the cube from the gray tray into the matching colored bowl; cubes without a matching bowl should stay in place.
Relevant Objects: a red cube, a green cube, a blue cube, a yellow cube, a gray tray, a green bowl, a red bowl
Sub-Goals:

VLM Output:

1. The green cube is inside the green bowl.
2. The red cube is inside the red bowl.
3. The blue cube is on top of the gray tray.
4. The yellow cube is on top of the gray tray.

Figure 11. **Prompts for sub-goals generation in the setup stage.**

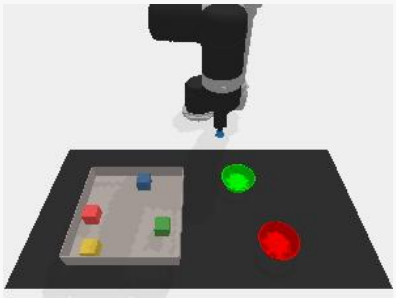
Prompts for identifying the completion status of each subgoal

System Prompt:

You assist in controlling a high-intelligence robotic arm that determine the spatial relationship between the objects in the image. Print Yes or No given the task description and queried subgoal.

User Prompt:

Task Description: Place cubes from the gray tray into the matching colored bowl, cubes without a matching bowl should stay in place.



Subgoal: The red cube is in the red bowl.
Please determine if the current status meets the subgoal.

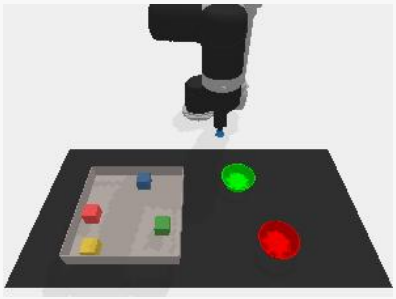
VLM output: No.

Figure 12. **Prompts for goal completion status queried VLMs.**

An example for completion status identification of sub-goals

Task Description:

Place the cube from the gray tray into the matching colored bowl; cubes without a matching bowl should stay in place.



Chain-of-thought Object Detection:

blue cube, red cube, green cube, yellow cube, red bowl, green bowl

Sub-goal Generation:

1. The blue cube remains on the gray tray.
2. The red cube is inside the red bowl.
3. The green cube is inside the green bowl.
4. The yellow cube remains on the gray tray.

Completion Status Identification of Sub-goals:

System prompt: Detailed in Fig. 12.

Task description: Place the cube from the gray tray into the matching colored bowl; cubes without a matching bowl should stay in place.

Case 1. Sub-goal matching for *The blue cube remains on the gray tray.*

Sub-goal: The blue cube remains on the gray tray.

Please determine if the current status meets the sub-goal.

VLM Output: Yes.

Case 2. Sub-goal matching for: *The red cube is inside the red bowl.*

Sub-goal: The red cube is inside the red bowl.

Please determine if the current status meets the sub-goal.

VLM Output: No.

Case 3. Sub-goal matching for: *The green cube is inside the green bowl.*

Sub-goal: The green cube is inside the green bowl.

Please determine if the current status meets the sub-goal.

VLM Output: No.

Case 4. Sub-goal matching for: *The yellow cube remains on the gray tray.*

Sub-goal: The yellow cube remains on the gray tray.

Please determine if the current status meets the sub-goal.

VLM Output: Yes.

Figure 13. **An example of goal completion status identification.**