

580 A Grounding DINO Detections and Prismatic Descriptions

581 We provide example scene descriptions provided by Prismatic VLM and bounding boxes provided by
 582 Grounding DINO in Fig. 7.

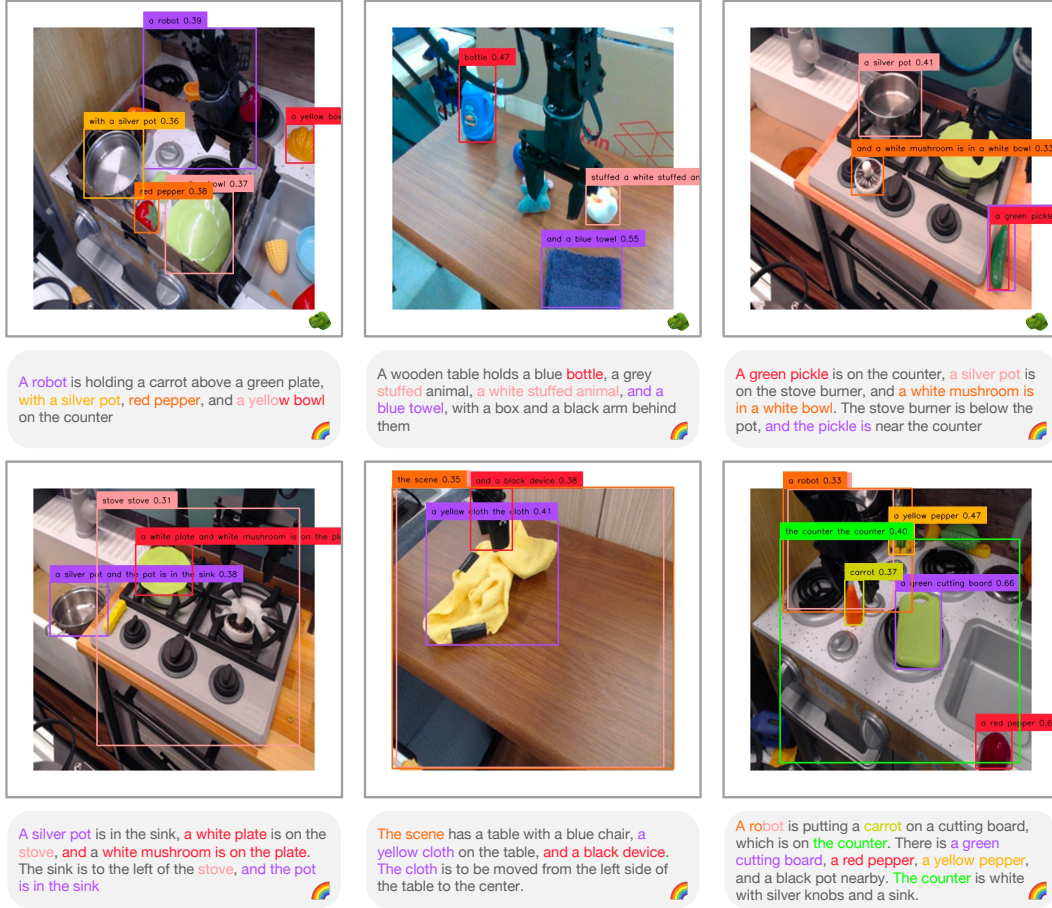


Figure 7: Examples of captions of observations from the Bridge dataset as generated by our Prismatic VLM, as well as associated bounding boxes generated by Grounding DINO.

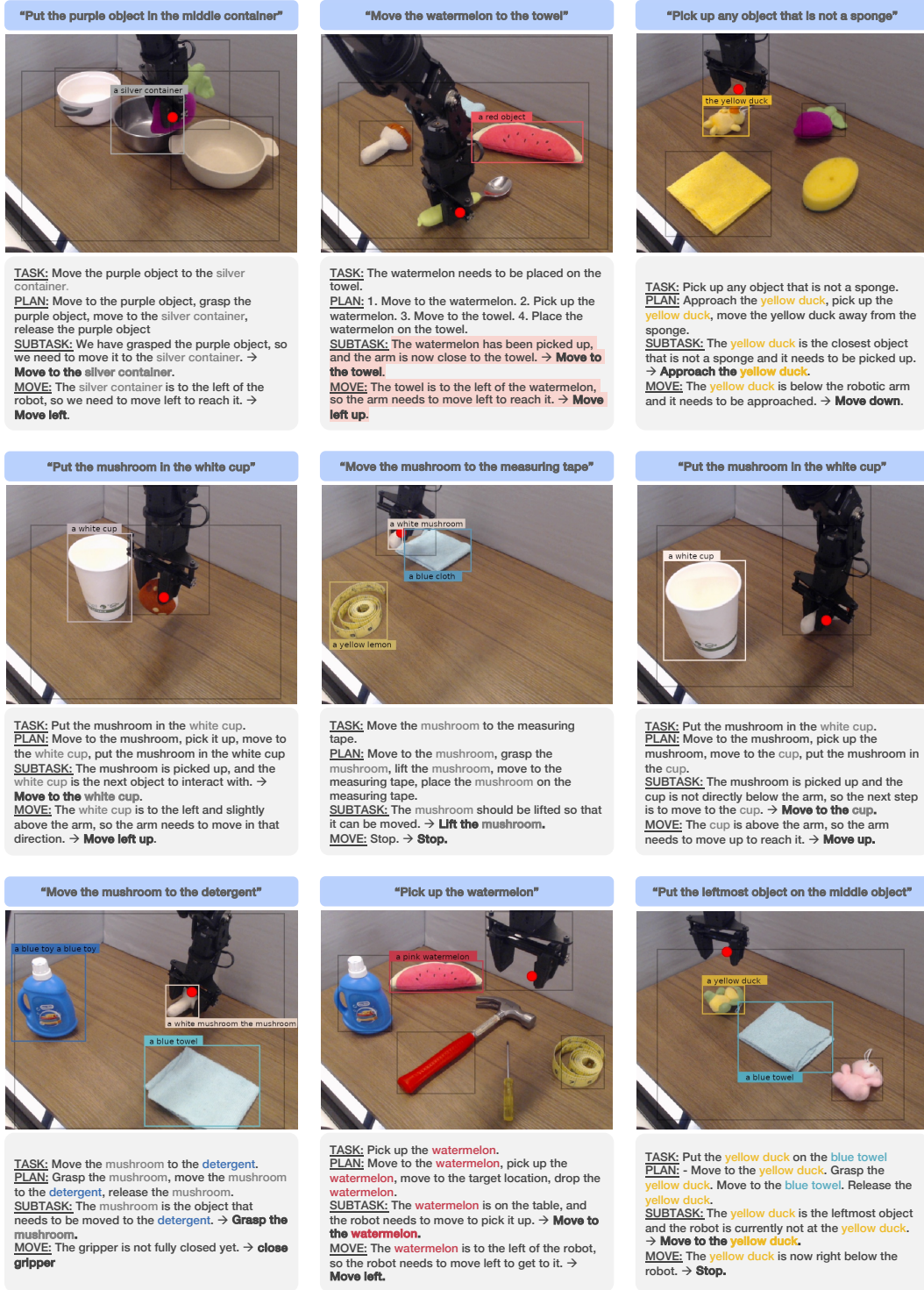


Figure 8: More qualitative examples of successful and failed chain-of-thought reasonings.

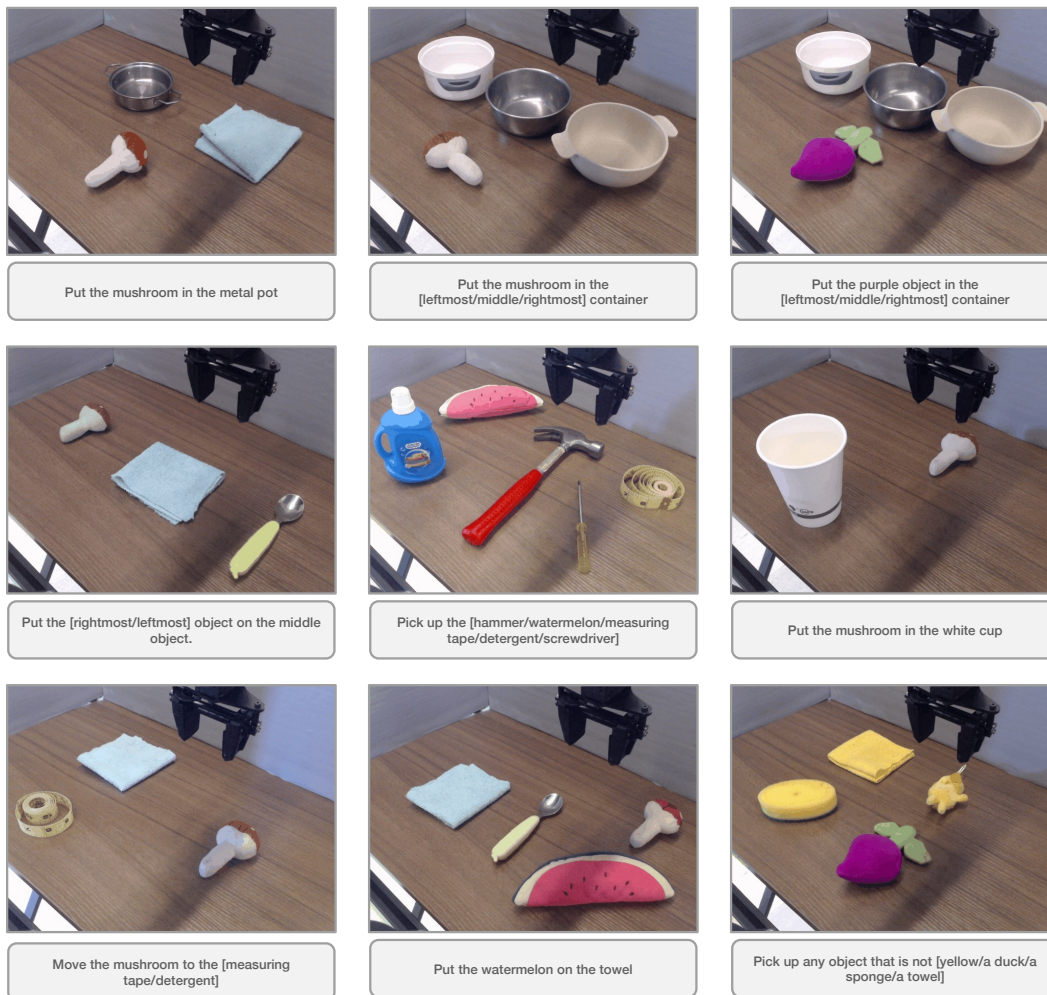


Figure 9: Example starting scenes and associated prompt for all task types.

B List of Movement Primitives

To classify a movement, we take the difference between the current state of the robot and its position four steps ahead. Based on the axes where the difference exceeds a threshold of 0.03, we assign it a label of the following form:

move [forward/backward] [left/right] [up/down], tilt [up/down], rotate
[clockwise/counterclockwise], [close/open] gripper

Whenever the movement in a certain axis is below the threshold, we omit its block for simplicity. For instance, if the robot is just moving left, the label is move left. If no movement is detected, the label is stop.

While technically it results in $3^6 = 729$ possible labels, only 54 are used in more than 0.1% of cases:

1. stop (26.9%)
2. close gripper (10.8%)
3. open gripper (7.2%)
4. move down (6.8%)
5. move left (6.6%)
6. move right (6.1%)
7. move up (5.7%)
8. move forward (3.0%)
9. move backward (2.4%)
10. move up, open gripper (2.1%)
11. move forward right (1.1%)
12. move up, close gripper (1.0%)
13. move backward left (1.0%)
14. move forward left (0.9%)
15. move left down (0.8%)
16. move down, close gripper (0.8%)
17. move right down (0.8%)
18. move left up (0.8%)
19. move right up (0.8%)
20. move right, rotate clockwise (0.8%)
21. move left, rotate counterclockwise (0.8%)
22. move backward right (0.8%)
23. rotate counterclockwise (0.7%)
24. move down, open gripper (0.7%)
25. rotate clockwise (0.7%)
26. move forward down (0.7%)
27. move up, rotate clockwise (0.5%)
28. move up, rotate counterclockwise (0.5%)
29. move backward up (0.5%)
30. move left, rotate clockwise (0.3%)
31. move backward down (0.3%)
32. move right, open gripper (0.3%)
33. move forward up (0.3%)
34. move left, open gripper (0.3%)
35. move right, rotate counterclockwise (0.3%)
36. move backward, open gripper (0.2%)
37. move down, rotate clockwise (0.2%)
38. move down, rotate counterclockwise (0.2%)
39. move forward, rotate counterclockwise (0.2%)
40. move forward, rotate clockwise (0.2%)
41. move forward, open gripper (0.2%)
42. move right, close gripper (0.2%)
43. move backward, rotate clockwise (0.2%)
44. move backward, rotate counterclockwise (0.2%)
45. move left, close gripper (0.2%)
46. move backward right, rotate clockwise (0.1%)
47. move backward left, rotate counterclockwise (0.1%)
48. move right up, open gripper (0.1%)
49. move right up, close gripper (0.1%)
50. move backward, close gripper (0.1%)
51. rotate clockwise, close gripper (0.1%)
52. rotate counterclockwise, close gripper (0.1%)
53. move left up, open gripper (0.1%)
54. move forward right, rotate clockwise (0.1%)

C Prompts

We now provide all the prompts used for data generation and policy language conditioning.

For using generating scene descriptions with Prismatic (step 1 in Fig. 4), we use the prompt: “Briefly describe the things in this scene and their spatial relations to each other.” We prepend “The robot task is: [TASK].” if the given demonstration trajectory contains a corresponding task instruction (where we ensure that said instruction contains at least one space character to remove noisy instructions).

We provide the prompt for Gemini data labeling (step 5 in Fig. 4) in Fig. 10.

The prompts used for our language-conditioned policies are provided in Fig. 9, along with example starting scenes for the associated tasks. For the OpenVLA-based policies, said prompts are inserted into the template provided by the original authors [7]: “A chat between a curious user and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the user’s questions. USER: What action should the robot take to [PROMPT]? ASSISTANT:”. The agent then generates reasoning text (if trained to do so) and an action.

We provide the prompt used for human interventions with ChatGPT in Fig. 11.

Annotate the training trajectory with reasoning

Specification of the experimental setup

You're an expert reinforcement learning researcher. You've trained an optimal policy for controlling a robotic arm. The robot successfully completed a task specified by the instruction: "unfold the cloth from top right to bottom left". For that purpose, the robotic arm executed a sequence of actions. Consecutive states that were visited can be characterized by the following features:

```

“python
trajectory_features = {
    0: "stop"
    1: "stop"
    2: "move_forward_left"
    3: "move_forward_down"
    4: "move_forward_down"
    ...
    36: "stop"
}
“

```

Each entry in that dictionary corresponds to a single step on the trajectory and describes the move that is about to be executed.

Scene description

The robot is operating in the following environment. A black and red toy stove with a yellow banana in a silver pot, a blue toy brush, and a purple towel on the counter, surrounded by white tiled walls and a grey sink.

Your objective

I want you to annotate the given trajectory with reasoning. That is, for each step, I need to know not only which action should be chosen, but importantly what reasoning justifies that action choice. I want you to be descriptive and include all the relevant information available. The reasoning should include the task to complete, the remaining high-level steps, the high-level movements that should be executed and why they are required, the premises that allow inferring the direction of each move, including the locations of relevant objects, possible obstacles or difficulties to avoid, and any other relevant justification.

Begin by describing the task

Start by giving an overview of the task. Make it more comprehensive than the simple instruction. Include the activity, the objects the robotic arm interacts with, and their relative locations in the environment. Then, describe the high-level movements that were most likely executed, based on the task that was completed and the primitive movements that were executed. Then, for each high-level movement write the interval of steps that movement consists of. Also, for each high-level movement write a justification for why it should be executed. Write an answer for this part using markdown and natural language. Be descriptive and highlight all the relevant details, but ensure that your description is consistent with the trajectory that was executed, specified by the features listed above in the 'trajectory_features' dictionary.

List the reasonings for each step

Finally, for each step describe the reasoning that allows to determine the correct action. For each step describe the remaining part of the objective, the current progress, the objects that are still relevant for determining the plan, and the plan for the next steps, based on the available features. Start the reasoning from a high level and gradually add finer features. I need you to be descriptive and very precise. Ensure that the reasoning is consistent with the task and the executed trajectory. Write the answer for this part as a Python-executable dictionary. For every step in the initial trajectory there should be exactly one separate item of the form <step id>:<reasoning>. Do not group the answers. The final dictionary should have exactly the same set of integer keys as the dictionary of features provided in the 'trajectory_features' dictionary above. The reasoning should be a single string that describes the reasoning in natural language and includes all the required features.

Each reasoning string should have the following form:

- Describe the full task that remains to be completed (but only describe what remains), and place it inside a tag <task>.
- Describe the complete high-level plan for completing the remaining task (the list of remaining high-level steps), and place it inside a tag <plan>.
- Describe the high-level step that should be executed now (chosen from the list of high-level steps), and place it inside a tag <subtask>.
- Describe why the chosen high-level step should be executed now, which features of the current environment influence that decision, and how it should be done. Place it within a tag <subtask_reason>.
- Describe the current primitive movement of the arm that needs to be executed, and place it inside a tag <move>.
- Describe why the chosen movement should be executed now and which features of the current environment influence that decision. Place it inside a tag <move_reason>.

Task summary

Here is a breakdown of what needs to be done:

- Describe the task.
- Describe the high-level movements that were executed, based on the completed task and the listed features.
- Describe the plan for the solution that allowed the robot to complete the task successfully.
- For each step on the trajectory, describe the reasoning that leads to determining the correct action. The reasoning should be descriptive and precise. You should provide exactly one reasoning string for each step on the trajectory specified by 'trajectory_features'.
- At the very end of the response, write a single label FINISHED to indicate that the answer is complete.

Figure 10: Prompt used for Gemini to generate plans, subtasks, and movement labels.

```

# Objective

You're an expert reinforcement learning researcher. You've trained a policy for controlling a robotic arm. The policy
computes the correct action based on a reasoning that leads to it, which includes the task that remains to be completed,
the plan for completing that task, and the subtask that currently needs to be done. I want you to prepare such a reasoning,
based on a feedback from a user of that robot.

The reasoning must have the following elements:
- TASK: the task that remains to be done.
- PLAN: a list of high-level steps that need to be executed.
- SUBTASK REASONING: reasoning that determines the current subtask.
- SUBTASK REASONING: reasoning that determines the current subtask.
- SUBTASK: the current subtask that should be executed.
- MOVE REASONING: reasoning that determines the current move.
- MOVE: the current move that should be executed

Write the answer as a python string. It will be used as an additional input for the policy, so keep the format exactly as
described.

# Examples

Given the task "Put the tomato inside the pot on the left burner" and feedback "you are too low, move up", the reasoning
should be "TASK: Put the tomato inside the pot on the left burner. PLAN: Go to the tomato, grasp it, transport it to the
stove, position it in the pot. SUBTASK REASONING: The tomato is grasped. The tomato is already near the pot, but below its
edge. SUBTASK: Position the tomato in the pot. MOVE REASONING: The pot is above current position. Move the arm up. MOVE:
Move up."

Given the task "place the silver lid on the silver pot on the upper right of the table" and feedback "move to the pot", the
reasoning should be "TASK: The lid needs to be placed on a silver pot on the upper right part of the scene. PLAN: First
move to the lid, then grip it, then move to the pot, then place the lid on the pot. SUBTASK REASONING: The lid is gripped,
so it should be moved to the pot. SUBTASK: Move to the pot."

Given the task "move the fork to the bottom left side of the counter" and feedback "move down to grasp the fork", the
reasoning should be "TASK: Pick up the fork and move it to the bottom left side of the counter. PLAN: 1. Move to the fork.
2. Pick up the fork. 3. Move to the bottom left side of the counter. 4. Put down the fork. SUBTASK REASONING: The fork
is the first object that needs to be reached. SUBTASK: 1. Move to the fork. MOVE REASONING: The fork is still downward
from the current position of the arm, so the arm continues to move that direction. MOVE: move down"

Given the task "remove the cylinder from the green cube and place it on top of the red cube" and feedback "close", the
reasoning should be "TASK: Remove the cylinder from the green cube and place it on top of the red cube. PLAN: Approach the
green cube, close the gripper around the cylinder, move the cylinder towards the red cube, open the gripper to place the
cylinder on top of the red cube. SUBTASK REASONING: The arm is now in contact with the cylinder, so it should close the
grripper to grab it. SUBTASK: Close the gripper MOVE REASONING: The cylinder has already been reached and the gripper is
closing. MOVE: Close gripper"

Given the task "pick up the towel" and feedback "go right", the reasoning should be "TASK: Pick up the towel. PLAN: Move to
the towel, Grasp the towel, Pick up the towel SUBTASK REASONING: The arm should reach the towel first. SUBTASK: Move to the
towel. MOVE REASONING: The towel is to the right, so the arm should move right. MOVE: move right"

# The current task

The policy generated the following reasoning: "TASK: Put the mushroom in the white cup. PLAN: Move to the mushroom, pick
it up, move to the cup, put the mushroom in the cup. SUBTASK REASONING: The mushroom is picked up, and the cup is the next
object to interact with. SUBTASK: Move to the cup. MOVE REASONING: The cup is positioned below the arm. MOVE: Move down.
GRIPPER POSITION: [111, 61] VISIBLE OBJECTS: a white cup [124, 25, 176, 113], a wooden table [13, 21, 241, 248], a wooden
table [10, 21, 249, 250]"

Given the task "put the mushroom in the white cup" and feedback "no, the cup is actually in front of the gripper", what
should be the reasoning?

```

Figure 11: Prompt used for ChatGPT during human intervention experiments