

914 A VLM Agent Training via Multi-Turn Reinforcement Learning

915 In this section, we detail our training pipeline, algorithm and system.

916 A.1 Problem Formulation: VLM Agent Training under a POMDP Formulation

917 The decision to frame the problem using a POMDP framework, rather than a simpler fully observable
918 Markov Decision Process (MDP), is deliberate and motivated by two principal considerations.

919 **Inherent Partial Observability in Environments.** A significant portion of the interactive environ-
920 ments we utilize, such as the Navigation, do not conform to an MDP setting. In these environments,
921 the agent does not have access to the complete, true state of the environment (s_t) at each timestep.
922 Instead, it receives an observation o_t (e.g., a first-person visual scene from its current vantage point)
923 which constitutes only a partial view of s_t . To gain a more comprehensive understanding of the
924 environment and to locate targets or critical information, the agent must actively perform actions,
925 such as rotating its viewpoint or moving to a new location. This necessity to act to disambiguate state
926 and gather information is a hallmark of POMDPs, where observations are explicitly distinguished
927 from the underlying states.

928 **Methodological Congruence with POMDPs.** Our proposed training methodology, particularly the
929 trajectory-based optimization strategy, is highly effective and naturally suited for POMDP settings.
930 By optimizing over entire trajectories, where multiple turns of interaction (sequences of observations,
931 actions, and reasoning steps) are considered collectively, our VLM agent learns to make decisions
932 based on the history of interactions. This is akin to a teacher-forcing approach during the update
933 phase, where the model is guided by the full sequence, encouraging it to learn dependencies on past
934 information. This inherent ability to leverage historical context to inform current actions is crucial
935 for effective decision-making under partial observability. The alignment between how trajectories
936 are rolled out (sequentially, with memory of the past) and how policy updates are performed (using
937 this historical sequence) makes our approach robust for tasks where the current observation alone is
938 insufficient.

939 Therefore, the POMDP formulation not only accurately reflects the nature of many of our experimental
940 environments but also aligns well with the strengths of our trajectory-based reinforcement learning
941 approach for VLM agents.

942 A.2 Multi-Turn Reinforcement Learning in VLM Agentic Tasks

943 Our multi-turn reinforcement learning framework for VLM agents is shown in Figure 5. At each turn
944 t , the VLM agent receives the current observation o_t (comprising a visual image and an optional
945 textual prompt). Based on o_t and its interaction history, the VLM generates a structured output
946 a_t . This output a_t includes an explicit reasoning component t_t (e.g., `<think><observation>...`
947 `<prediction>...</think>`) and an executable action component \bar{a}_t (e.g., `<answer>...</answer>`).
948 The executable action \bar{a}_t is parsed and sent to the environment. The environment then transitions to a
949 new state, providing the next observation o_{t+1} and a scalar reward r_t . This cycle repeats for N turns
950 to form a trajectory. The VLM’s parameters are updated using reinforcement learning, specifically
951 Proximal Policy Optimization (PPO), based on the collected trajectories.

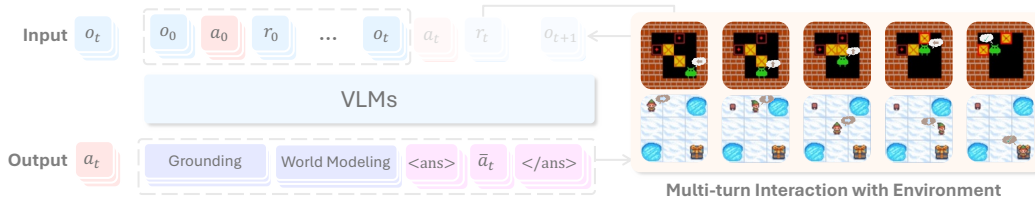


Figure 5: Multi-turn RL framework for VLM agents.

952 The detailed training algorithm for our multi-turn RL framework is presented in Algorithm 1.

953 The detailed reasoning templates is given in Table 4.

Algorithm 1 Multi-Turn Reinforcement Learning for VLM Agents

Input: Actor VLM π_θ , Environment, Critic V_ϕ , Reference VLM π_{ref}

Hyperparameters: Discount factor γ , GAE coefficient λ , PPO clip range ε , KL penalty coefficient β

Phase 1: Trajectory Collection

- 1: Initialize trajectory $\tau = []$ and sample initial observation o_0 from Environment
 - 2: **for** $t = 0, \dots, N - 1$ **do**
 - 3: Sample agent output $a_t \sim \pi_\theta(\cdot | o_0, a_0, \dots, a_{t-1}, o_t)$ autoregressively
 where a_t follows structure, e.g. `<think> t_t </think><answer> \bar{a}_t </answer>`
 with t_t as reasoning and \bar{a}_t as executable action
 - 4: Parse executable action \bar{a}_t from a_t
 - 5: Execute \bar{a}_t in Environment to receive next observation o_{t+1} and reward $r_{\text{env},t}$
 - 6: Compute format reward $r_{\text{format},t}$ (e.g., 0.5 if structured correctly, 0 otherwise)
 - 7: Set turn reward $r_{\text{turn},t} = r_{\text{env},t} + r_{\text{format},t}$
 - 8: Append $(o_t, a_t, r_{\text{turn},t})$ to trajectory τ
 - 9: **end for**
 - 10: Store token probabilities π_{old} for all tokens in actions a_0, \dots, a_{N-1}
-

Phase 2: Token-level Advantage Estimation

- 11: Let S_{tokens} be the concatenated sequence of all tokens from a_0, \dots, a_{N-1}
 - 12: Let T_{total} be the total number of tokens in S_{tokens}
 - 13: Let $k_{<i}$ denote the sequence prefix up to the i -th token in S_{tokens}
 - 14: Define mask M_i^{loss} where $M_i^{\text{loss}} = 1$ if token i is generated by π_θ , 0 otherwise
 - 15: **for** each token index $i = 0, \dots, T_{\text{total}} - 1$ in S_{tokens} **do**
 - 16: Calculate KL penalty: $r_i^{\text{KL}} = -\beta \cdot \text{KL}(\pi_\theta(\cdot | k_{<i}) \parallel \pi_{\text{ref}}(\cdot | k_{<i}))$
 - 17: **if** i is the final token ($T_{\text{total}} - 1$) of trajectory S_{tokens} **then**
 - 18: Calculate trajectory reward $R(\tau) = \sum_{t=0}^{N-1} r_{\text{turn},t}$
 - 19: Set token reward $r_i = r_i^{\text{KL}} + R(\tau)$
 - 20: **else**
 - 21: Set token reward $r_i = r_i^{\text{KL}}$
 - 22: **end if**
 - 23: **end for**
 - 24: Compute GAE advantages, proceeding backwards from $i = T_{\text{total}} - 1$ to 0:
 $\delta_i = r_i + \gamma V_\phi(k_{i+1}) - V_\phi(k_i)$ (where $V_\phi(k_{T_{\text{total}}}) = 0$ for terminal state)
 $A_i^{\text{GAE}} = \delta_i + \gamma \lambda A_{i+1}^{\text{GAE}}$ (where $A_{T_{\text{total}}}^{\text{GAE}} = 0$)
 - 25: Compute returns for critic update: $\hat{R}_i = A_i^{\text{GAE}} + V_\phi(k_i)$
-

Phase 3: Policy Update with PPO

- 26: **for** each token index $i = 0, \dots, T_{\text{total}} - 1$ in S_{tokens} **do**
 - 27: Compute probability ratio: $pr_i(\theta) = \frac{\pi_\theta(\text{token}_i | k_{<i})}{\pi_{\text{old}}(\text{token}_i | k_{<i})}$
 - 28: **end for**
 - 29: Compute PPO objective:
$$J^{\text{PPO}}(\theta) = \frac{\sum_{i=0}^{T_{\text{total}}-1} M_i^{\text{loss}} \cdot \min(pr_i(\theta) A_i^{\text{GAE}}, \text{clip}(pr_i(\theta), 1-\varepsilon, 1+\varepsilon) A_i^{\text{GAE}})}{\sum_{i=0}^{T_{\text{total}}-1} M_i^{\text{loss}}}$$
 - 30: Compute critic loss:
$$J^{\text{Critic}}(\phi) = \frac{\sum_{i=0}^{T_{\text{total}}-1} M_i^{\text{loss}} \cdot (V_\phi(k_i) - \hat{R}_i)^2}{\sum_{i=0}^{T_{\text{total}}-1} M_i^{\text{loss}}}$$
 - 31: Update parameters θ using $\nabla_\theta J^{\text{PPO}}(\theta)$ and ϕ using $\nabla_\phi J^{\text{Critic}}(\phi)$
 - 32: Return to **Phase 1** for next iteration
-

954 A.3 VAGEN Framework

955 To support our experiments, we developed VAGEN, a framework built upon VeRL[93]. VAGEN
956 specializes in multi-turn reinforcement learning (RL) training for Vision-Language Models (VLMs),
957 integrating various RL algorithms and environments. We adopt *env-as-service* design, which decou-
958 ples training from interacting with environments, enhances scalability.

959 A.4 Environments and Tasks for VLM Agents

960 **Sokoban** The action space and hyperparameters are given in Table 5 and Table 6.

Table 4: Reasoning strategy formats.

Strategy	Format
No-Think	<code><answer>\bar{a}_t</answer></code>
Free-Think	<code><think>t_t</think></code> <code><answer>\bar{a}_t</answer></code>
Grounding	<code><think><observation>s_t</observation></code> <code><reasoning>r_t</reasoning></think></code> <code><answer>\bar{a}_t</answer></code>
WorldModeling	<code><think><reasoning>r_t</reasoning></code> <code><prediction>s_{t+1}</prediction></think></code> <code><answer>\bar{a}_t</answer></code>
Grounding-WorldModeling	<code><think><observation>s_t</observation></code> <code><reasoning>r_t</reasoning></code> <code><prediction>s_{t+1}</prediction></think></code> <code><answer>\bar{a}_t</answer></code>

Table 5: Action space for the Sokoban environment.

Name	Description
Up	Move the agent one cell upward on the grid.
Left	Move the agent one cell to the left on the grid.
Right	Move the agent one cell to the right on the grid.
Down	Move the agent one cell downward on the grid.

961 **FrozenLake** The action space and hyperparameters are given in Table 7 and Table 8.

962 **Navigation** The action space and hyperparameters are given in Table 9 and Table 10.

963 **PrimitiveSkill** The action space and hyperparameters are given in Table 11 and Table 12.

964 **SVG Reconstruction** The action space and hyperparameters are given in Table 13 and Table 14.

965 A.5 Reward Assignment

966 **Sokoban** The reward structure for the Sokoban environment is presented in Table 15.

967 **FrozenLake** The reward structure for the FrozenLake environment is presented in Table 16.

968 **Navigation** The reward structure for the Navigation environment is presented in Table 17.

969 **PrimitiveSkill** The reward structure for the PrimitiveSkill environment is presented in Table 18.

970 **SVG** The reward structure for the SVG Reconstruction environment is presented in Table 19.

Table 6: Hyperparameters for the Sokoban environment.

Name	Value	Description
dim_room	(6, 6)	Dimensions of the Sokoban grid
max_steps	100	Maximum number of steps allowed per episode
num_boxes	1	Number of boxes to be pushed onto targets
min_actions_to_succeed	5	Minimum number of actions required to solve the puzzle
max_actions_per_step	3	Maximum number of actions the agent can take per turn
max_turns	3	Maximum number of turns the agent can interact with the environment

Table 7: Action space for the FrozenLake environment.

Name	Description
Up	Move the agent one cell upward on the grid.
Left	Move the agent one cell to the left on the grid.
Right	Move the agent one cell to the right on the grid.
Down	Move the agent one cell downward on the grid.

Table 8: Hyperparameters for the FrozenLake environment.

Name	Value	Description
desc	None	Environment map layout (if None, randomly generated)
is_slippery	False	Whether the frozen tiles are slippery
size	4	Size of the square grid
max_actions_per_step	3	Maximum number of actions the agent can take per turn
min_actions_to_succeed	5	Minimum number of actions required to reach the goal
max_turns	3	Maximum number of turns the agent can interact with the environment

Table 9: Action space for the Navigation environment.

Name	Description
MoveAhead	Move forward by some distance
MoveBack	Move backward by some distance
MoveRight	Move rightward by some distance
MoveLeft	Move leftward by some distance
RotateRight	Rotate to the right by 90 degrees
RotateLeft	Rotate to the left by 90 degrees
LookUp	Tilt the camera upward by 30 degrees
LookDown	Tilt the camera downward by 30 degrees

Table 10: Hyperparameters for the Navigation environment.

Name	Value	Description
resolution	255	Resolution of the rendered images
down_sample_ratio	1.0	Ratio for down-sampling images
fov	100	Field of view angle in degrees
multiview	False	Whether to use multiple camera views
max_actions_per_step	5	Maximum number of actions the agent can take per turn
success_threshold	1.0	Threshold for considering task successful
step_length	0.25	Distance traveled in a single movement action
max_turns	4	Maximum number of turns the agent can interact with the environment

Table 11: Action space for the PrimitiveSkill environment.

Name	Description
pick(x, y, z)	Grasp an object located at position (x, y, z) in the robot's workspace
place(x, y, z)	Place the object currently held by the robot's gripper at the target position (x, y, z)
push(x1, y1, z1, x2, y2, z2)	Push an object from position (x1, y1, z1) to position (x2, y2, z2)

Table 12: Hyperparameters for the PrimitiveSkill environment.

Name	Value	Description
max_actions_per_step	2	Maximum number of actions the agent can take per turn
max_turns	3	Maximum number of turns the agent can interact with the environment

Table 13: Action space for the SVG Reconstruction environment.

Name	Description
SVG Code	Open text format allowing for the specification of SVG markup code

A.6 Evaluation Metrics

We employ a range of metrics to evaluate agent performance across our diverse task environments. For trajectory-based evaluation, we define several key functions over a trajectory τ :

- $f(\tau) \in \{0, 1\}$: A binary success indicator function that equals 1 if the trajectory τ successfully completes the task, and 0 otherwise
- $g(\tau) \in [0, 1]$: The DreamSim similarity score between the target image and the final generated image in trajectory τ
- $h(\tau) \in [0, 1]$: The DINO similarity score between the target image and the final generated image in trajectory τ

Metrics for Task-Completion Environments For Sokoban, FrozenLake, Navigation, and PrimitiveSkill environments, we use the average success rate over a dataset \mathcal{D} of test trajectories:

$$\text{Success Rate} = \mathbb{E}_{\tau \sim \mathcal{D}}[f(\tau)] \quad (1)$$

A trajectory is considered successful ($f(\tau) = 1$) when the agent completes the specific task objectives for each environment.

Metrics for SVG Reconstruction For the SVG Reconstruction task, where success is measured by the similarity between generated and target images rather than discrete goal achievement, we report:

$$\text{DreamSim Score} = \mathbb{E}_{\tau \sim \mathcal{D}}[g(\tau)] \quad (2)$$

$$\text{DINO Score} = \mathbb{E}_{\tau \sim \mathcal{D}}[h(\tau)] \quad (3)$$

These complementary metrics capture different aspects of visual similarity:

- **DreamSim Score:** Measures perceptual similarity between images using a pre-trained diffusion model [58].
- **DINO Score:** Based on self-supervised visual features from a small version of the DINO model [2].

Table 14: Hyperparameters for the SVG Reconstruction environment.

Name	Value	Description
dataset_name	starvector/svg-icons-simple	Dataset used for SVG examples
max_turns	2	Maximum number of turns the agent can interact with the environment

Table 15: Reward structure for the Sokoban environment.

Reward Type	Value	Description
Success reward	10	Awarded when all boxes are placed on target locations
Failure penalty	-0.1	Applied each step when the task is not completed
Box placement reward	1	Granted for each box pushed onto a target location
Format reward	0.5	Provided at each turn to encourage visual state reasoning
Grounding reward weight	0.5	Weight applied to Grounding reward
World modeling reward weight	0.5	Weight applied to WorldModeling reward

Table 16: Reward structure for the FrozenLake environment.

Reward Type	Value	Description
Success reward	10	Awarded when the agent reaches the goal position
Failure penalty	-0.1	Applied each step when the task is not completed
Format reward	0.5	Provided at each turn to encourage visual state reasoning
Grounding reward weight	0.5	Weight applied to Grounding reward
World modeling reward weight	0.5	Weight applied to WorldModeling reward

Table 17: Reward structure for the Navigation environment.

Reward Type	Value	Description
Success reward	10	Awarded when the agent reaches the goal location
Failure penalty	-0.1	Applied each step when the task is not completed
Format reward	0.5	Provided at each turn to encourage visual state reasoning
Grounding reward weight	0.5	Weight applied to Grounding reward
World modeling reward weight	0.5	Weight applied to WorldModeling reward

Table 18: Reward structure for the PrimitiveSkill environment.

Reward Type	Value	Description
Success reward	10	Awarded when the manipulation task is completed
Failure penalty	-0.1	Applied each step when the task is not completed
Stage-based reward	$(\text{stage} + 1) \times 2$	Granted upon completing key manipulation subgoals (where stage is the highest successfully completed stage)
Format reward	0.5	Provided at each turn to encourage visual state reasoning
Grounding reward weight	0.5	Weight applied to Grounding reward
World modeling reward weight	0.5	Weight applied to WorldModeling reward

Table 19: Reward structure for the SVG Reconstruction environment.

Reward Type	Value	Description
Image similarity	Variable	Weighted DreamSim [58] and DINO [2] scores measuring similarity between generated and target images
Format reward	0.5	Provided at each turn to encourage visual state reasoning
Grounding reward weight	0.5	Weight applied to Grounding reward
World modeling reward weight	0.5	Weight applied to WorldModeling reward
DreamSim weight	5.0	Scaling factor applied to DreamSim similarity scores
Dino weight	0.0001	We only use DreamSim score for reward

B What Can We Reason About Visual States?

B.1 Bottleneck of Off-the-Shelf VLMs on Agentic Tasks

To understand the capabilities and limitations of existing VLMs on agentic tasks, we evaluate several state-of-the-art off-the-shelf VLMs across our proposed tasks. The prompts used for evaluating these models follow the Free-Think reasoning strategy format: $\langle \text{think} \rangle t_t \langle / \text{think} \rangle \langle \text{answer} \rangle \bar{a}_t \langle / \text{answer} \rangle$.

Sokoban Prompt for evaluating off-the-shelf VLMs

System

You are a Sokoban solver.
Sokoban Quick Guide
Goal: Push all boxes onto targets.
Symbols (If image is provided there are no symbols):
Wall | _ Floor | O Target | X Box | P You | * Box on Target | S You on Target
Rules:
1. Push boxes (can't pull).
2. Avoid walls.
Actions you can take: Left, Down, Right, Up.
You can take up to 3 action(s) at a time, separated by ,.
You should first give your reasoning, and then your answer.
Your response should be in the format of:
 $\langle \text{think} \rangle \dots \langle / \text{think} \rangle \langle \text{answer} \rangle \dots \langle / \text{answer} \rangle$
e.g. $\langle \text{think} \rangle \langle \text{reasoning} \rangle$ The box is one step below me, and the target is two steps below me, I need to go down then push the box down to the target. $\langle / \text{reasoning} \rangle \langle / \text{think} \rangle \langle \text{answer} \rangle$ Down, Down $\langle / \text{answer} \rangle$

User

[Initial Observation]:
 $\langle \text{image} \rangle$
Decide your next action(s).
You can take up to 3 action(s) at a time, separated by ,.
You should first give your reasoning, and then your answer.
Your response should be in the format of:
 $\langle \text{think} \rangle \dots \langle / \text{think} \rangle \langle \text{answer} \rangle \dots \langle / \text{answer} \rangle$

Assistant

$\langle \text{think} \rangle$ Since there are no boxes or targets visible, the first step is to move the character (P) to the wall to see if it will push the wall and open a path. This might also help in accessing more space to maneuver. $\langle / \text{think} \rangle$
 $\langle \text{answer} \rangle$ Left, Up, Right $\langle / \text{answer} \rangle$
rest of the turns ...

FrozenLake Prompt for evaluating off-the-shelf VLMs

System

You are a FrozenLake solver.
FrozenLake Quick Guide
Goal: Reach the goal (G).
Symbols (If image is provided there are no symbols):
_ Frozen | O Hole | G Goal | P Player | X Player fell into hole | * Player on goal
Rules:
1. Avoid falling into holes.
2. Frozen tiles are slippery, you may move perpendicular to your intended direction.
Actions you can take: Left, Down, Right, Up.

You can take up to 3 action(s) at a time, separated by ,.
You should first give your reasoning, and then your answer.

Your response should be in the format of:
`<think>...</think><answer>...</answer>`
 e.g. `<think>I can see the target is on my down left, I should go down then left to reach the target</think><answer>Down,Left</answer>`

User

[Initial Observation]:
`<image>`
 Decide your next action(s).

You can take up to 3 action(s) at a time, separated by ,.
 You should first give your reasoning, and then your answer.
 Your response should be in the format of:

`<think>...</think><answer>...</answer>`

Assistant

`<think>There is a present at the bottom right, which is likely the goal. Since the player is near the present and the goal is usually at the bottom right, I should move downwards to reach it.</think><answer>Down, Down, Down</answer>`
 rest of the turns ...

999

Navigation Prompt for evaluating off-the-shelf VLMs

System

You are a home robot and perform navigation tasks according to instructions. Actions you can take: moveahead, moveback, moveright, moveleft, rotateright, rotateleft, lookup, lookdown.

moveahead: Move forward by some distance

moveback: Move backward by some distance

moveright: Move rightward by some distance

moveleft: Move leftward by some distance

rotateright: Rotate to the right by 90 degrees

rotateleft: Rotate to the left by 90 degrees

lookup: Tilt the camera upward by 30 degrees

lookdown: Tilt the camera downward by 30 degrees

Rewards:

Format correct: +0.5

Achieve the human instruction: +10.0

The instruction will be provided with each observation. Look at the image carefully and navigate to complete the instruction.

Hints:

1. You can take multiple actions at a time, in most cases, if you find the target object is far away from you, you can call moveahead, moveleft and move right multiple times.
2. If you find yourself seems to be stuck, you can lookdown to see if there's any object above or below you, you can also rotate to see if there's any object behind you.

Example:

Round 1:

image_1

`<think>I can see the garbage can in the upper left corner of the image, next to the kitchen sink. To move there, we can go forward-left, but since there's a kitchen counter directly ahead, we should go left first. Following the strategy, I can go by first moving leftward.</think>`

`<answer>moveleft, moveleft</answer>`

Round 2:

Env_feedback: Last action is executed successfully.

image_2

`<think>From the scene, I see that by moving leftward, we are getting closer to the garbage can. Now, the garbage can is in front of me, slightly to the left. And there's a large area ahead of us. Following the strategy, I can go by first moving forward then moving leftward.</think>`

1000


```

<answer>moveahead, moveahead,moveahead,moveleft</answer>
Round 3:
Env_feedback: Last action is executed successfully.
image_3
<think>From the image we can see the garbage can is very close to us, still
to our front-left. Moving leftward might be blocked but i can see that there
is still space in front of me to get closer to the garbage can. Following
the strategy, we can take about two steps forward then one step left to
reach the garbage can.</think>
<answer>moveahead, moveahead,moveleft</answer>
Round 4:
Env_feedback: Success
You can take up to 5 action(s) at a time, separated by ','.
You should first give your thought process, and then your answer.
Your response should be in the format of:
<think>...</think><answer>...</answer>
e.g. <think>I can see from the sight the target object is right in the top
left of me, I will move forward, then move left to access
it.</think><answer>moveahead,moveahead,moveahead,moveleft,moveleft</answer>
User
[Initial Observation]:
<image>
Human Instruction: I need to dispose of some trash properly. Please navigate
to that object and stay near it.
Decide your next action(s).
You can take up to 5 action(s) at a time, separated by ','.
You should first give your thought process, and then your answer.
Your response should be in the format of:
<think>...</think><answer>...</answer>
Assistant
<think>From the image, I can see the trash can is in the bottom-right
corner. I need to navigate towards it. I will first move forward and then
turn to the right to reach it.</think><answer>moveahead, rotateright</answer>
rest of the turns ...

```

1001

PrimitiveSkill Prompt for evaluating off-the-shelf VLMs

System

You are an AI assistant controlling a Franka Emika robot arm. Your goal is to understand human instructions and translate them into a sequence of executable actions for the robot, based on visual input and the instruction.

Action Space Guide

You can command the robot using the following actions:

1. pick(x, y, z) # To grasp an object located at position(x,y,z) in the robot's workspace.
2. place(x, y, z) # To place the object currently held by the robot's gripper at the target position (x,y,z).
3. push(x1, y1, z1, x2, y2, z2) # To push an object from position (x1,y1,z1) to (x2,y2,z2).

Hints:

1. The coordinates (x, y, z) are in millimeters and are all integers.
2. Please ensure that the coordinates are within the workspace limits.
3. The position is the center of the object, when you place, please consider the volume of the object. It's always fine to set z much higher when placing an item.
4. We will provide the object positions to you, but you need to match them to the object in the image by yourself. You're facing toward the negative

1002

x-axis, and the negative y-axis is to your left, the positive y-axis is to your right, and the positive z-axis is up.

Examples:

round1:

image1

Human Instruction: Put red cube on green cube and yellow cube on left target

Object positions:

[(62,-55,20),(75,33,20),(-44,100,20),(100,-43,0),(100,43,0)]

Reasoning: I can see from the picture that the red cube is on my left and green cube is on my right and near me.

Since I'm looking toward the negative x axis, and negative y-axis is to my left, (62,-55,20) would be the position of the red cube, (75,33,20) would be the position of the green cube and (-44,100,20) is the position of the yellow cube.

Also the (100,-43,0) would be the position of the left target, and

(100,43,0) would be the position of the right target.

I need to pick up red cube first and place it on the green cube, when placing, I should set z much higher.

Answer: pick(62,-55,20)|place(75,33,50)

round2:

image2

Human Instruction: Put red cube on green cube and yellow cube on left target

Object positions:

[(75,33,50),(75,33,20),(-44,100,20),(100,-43,0),(100,43,0)]

Reasoning: Now the red cube is on the green cube, so I need to pick up the yellow cube and place it on the left target.

Answer: pick(-44,100,20)|place(100,-43,50)

You can take up to 2 action(s) at a time, separated by |.

You should first give your thought process, and then your answer.

Your response should be in the format of:

<think>...</think><answer>...</answer>

e.g. e.g. <think>I need to pick the red_cube_pos at (10,20,30) and place it on the green_block_pos at

(50,60,40).</think><answer>pick(10,20,30)|place(50,60,70)</answer>

User

[Initial Observation]:

<image>

Human Instruction: Please align the cubes in the y-axis, which means the x-coordinates of both cubes should be 0 (+-10mm)

x_workspace_limit: (-100, 150)

y_workspace_limit: (-200, 200)

z_workspace_limit: (10, 200)

Object positions:

[(129, -119, 20), (108, 124, 20)]

Other information:

No other information needed

Decide your next action(s).

You can take up to 2 action(s) at a time, separated by |.

You should first give your thought process, and then your answer.

Your response should be in the format of:

<think>...</think><answer>...</answer>

Assistant

<think>First, I need to pick up the red cube at position (129, -119, 20) and place it at position (0, -119, 20).</think><answer>pick(129, -119, 20)|place(0, -119, 50)</answer>

rest of the turns ...

SVG Reconstruction Prompt for evaluating off-the-shelf VLMs

System

You are a precise SVG code generator.

SVG Quick Guide

Goal: Transform the provided image into precise SVG code that replicates the image.

Process:

1. First analyze the image carefully, identifying distinct visual elements
2. Identify colors, dimensions, positions, and relationships between elements
3. Generate accurate SVG code that reproduces the image, you can use path for better shape

Rewards:

- Overall visual similarity: +5.0
- Structural accuracy: +10.0

Example:

<think>I can see the image contains a red circle and a blue rectangle. The circle is positioned at the top-left, while the rectangle is at the bottom-right.</think>

```
<answer><svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg">
  <circle cx="25" cy="25" r="15" fill="red" />
  <rect x="60" y="60" width="30" height="20" fill="blue" />
</svg></answer>
```

You can take up to 1 action(s) at a time, separated by ~~.

You should first give your thought process, and then your answer.

Your response should be in the format of:

<think>...</think><answer>...</answer>

e.g. <think>I can see the image contains a red circle and a blue rectangle. The circle is positioned at the top-left, while the rectangle is at the bottom-right.</think>

```
<answer><svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg">
  <circle cx="25" cy="25" r="15" fill="red" />
  <rect x="60" y="60" width="30" height="20" fill="blue" />
</svg></answer>
```

User

[Initial Observation]:

<image>

Please carefully observe the image, and generate SVG code that reproduces it as accurately as possible.

Decide on your SVG code.

You can take up to 1 action(s) at a time, separated by ~~.

You should first give your thought process, and then your answer.

Your response should be in the format of:

<think>...</think><answer>...</answer>

Assistant

<think>Based on the image, I can see a circular shape with the letters "kr" inside it. The letters are positioned in the center of the circle.</think>

```
<answer><svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg">
  <circle cx="50" cy="50" r="40" fill="none" stroke="black" stroke-width="2" />
  <text x="50" y="70" font-size="40" font-family="Arial"
    fill="black">kr</text>
</svg></answer>
rest of the turns ...
```

1004

1005 The information of evaluated VLMs with corresponding hyperparameters are provided in Table 20.

Table 20: Off-the-shelf VLMs for evaluation.

Model Name	Model ID	Provider	Max Tokens	Temperature
VLM-R1-3B	omlab/VLM-R1-Qwen2.5VL-3B-Math-0305	omlab	150/400	0.7
Qwen2.5-VL-3B	Qwen/Qwen2.5-VL-3B-Instruct	Qwen	150/400	0.7
Qwen2.5-VL-7B	Qwen/Qwen2.5-VL-7B-Instruct	Qwen	150/400	0.7
Qwen2.5-VL-72B	Qwen/Qwen2.5-VL-72B-Instruct	Qwen	150/400	0.7
GPT-4o	gpt-4o	openai	150/400	0.7
Gemini 2.0	gemini-2.0-flash	gemini	150/400	0.7
Claude 3.7 Sonnet	claude-3-7-sonnet-20250219	claude	150/400	0.7

Table 21: Multi-turn RL training hyperparameters.

Parameter	Value	Description
Rollout Phase		
Top-p	0.95	Nucleus sampling parameter for action generation
Temperature	0.7	Sampling temperature for controlling randomness
Update Phase		
Advantage Estimator	masked_gae	Generalized Advantage Estimation with masking
Actor Model	Qwen/Qwen2.5-VL-3B-Instruct	Pre-trained model used for actor initialization
Critic Model	Qwen/Qwen2.5-VL-3B-Instruct	Pre-trained model used for critic initialization
γ_{token}	1.0	Discount factor for token-wise advantage calculation
KL Penalty Coefficient (β)	0.001	Coefficient for KL divergence penalty in PPO objective
Actor Learning Rate	1e-6	Learning rate for the actor network
Critic Learning Rate	1e-5	Learning rate for the critic network
Train Batch Size	128	Total batch size for training
PPO Mini Batch Size	32	Mini-batch size for PPO updates

B.2 Reasoning in Multi-turn RL Training

Following the training methodology described in Section 2.2, we use reinforcement learning to train Qwen2.5-VL-3B across all reasoning strategies presented in Table 4. Our experiments are conducted on servers equipped with 8xH100 GPUs, 104 CPUs, and 1.7TB of memory. Each server can run two experiments at the same time, with each training session requiring approximately 4-8 hours to complete.

The training hyperparameters used in our experiments are detailed in Table 21.

We also release the prompts used in the multi-turn RL training process. For different reasoning strategies, the structures of the prompts are largely identical, with minor differences in response formats. To avoid redundancy, we provide only the Grounding-WorldModeling prompts here. For Free-Think prompts, please refer to Appendix B.1. For other reasoning strategy prompts, please see our provided code or infer them from the existing prompts.

Sokoban Training Prompt for Grounding-WorldModeling Strategy

System

You are a Sokoban solver.
Sokoban Quick Guide
Goal: Push all boxes onto targets.
Symbols (If image is provided there are no symbols):
Wall | _ Floor | O Target | X Box | P You | * Box on Target | S You on Target
Rules:
1. Push boxes (can't pull).
2. Avoid walls.
Actions you can take: Left, Down, Right, Up.
You can take up to 3 action(s) at a time, separated by ,.
You should first give the description of your observation, then your reasoning, then predict the next state, and finally your answer.

Your response should be in the format of:
`<think><observation>...</observation><reasoning>...</reasoning>
<prediction>...</prediction></think><answer>...</answer>`
e.g. `<think><observation>The box is below the player and the target is below
the box</observation><reasoning>I need to go down then push the box down to
the target</reasoning><prediction>The player will be above the box, the
target and box will be at the same
place.</prediction></think><answer>Down,Down</answer>`
User
[Initial Observation]:
<image>
Decide your next action(s).
You can take up to 3 action(s) at a time, separated by ,.
You should first give the description of your observation, then your
reasoning, then predict the next state, and finally your answer.
Your response should be in the format of:
`<think><observation>...</observation><reasoning>...</reasoning>
<prediction>...</prediction></think><answer>...</answer>`
Assistant
`<think><observation>The player is at the bottom of the screen, and there is
a box to the right of the player. The target is to the left of the
box.</observation><reasoning>The player needs to push the box to the target
to complete the goal.</reasoning><prediction>The player will push the box to
the target, moving up, down, and to the
left.</prediction></think><answer>Up, Down, Left</answer>`
rest of the turns ...

1019

FrozenLake Training Prompt for Grounding-WorldModeling Strategy

System
You are a FrozenLake solver.
FrozenLake Quick Guide
Goal: Reach the goal (G).
Symbols (If image is provided there are no symbols):
_ Frozen | O Hole | G Goal | P Player | X Player fell into hole | * Player
on goal
Rules:
1. Avoid falling into holes.
2. Frozen tiles are slippery, you may move perpendicular to your intended
direction.
Actions you can take: Left, Down, Right, Up.

You can take up to 3 action(s) at a time, separated by ,.
You should first describe the observation, then your reasoning, then predict
the next state, and finally your answer.
Your response should be in the format of:
`<think><observation>...</observation><reasoning>...</reasoning>
<prediction>...</prediction></think><answer>...</answer>`
e.g. `<think><observation>The player is on the above the
target</observation><reasoning>I should go down then left to reach the
target</reasoning><prediction>The player will reach the
target</prediction></think><answer>Down,Left</answer>`
User
[Initial Observation]:
<image>
Decide your next action(s).

You can take up to 3 action(s) at a time, separated by ,.
You should first describe the observation, then your reasoning, then predict
the next state, and finally your answer.
Your response should be in the format of:

1020

```

<think><observation>...</observation><reasoning>...</reasoning>
<prediction>...</prediction></think><answer>...</answer>
Assistant
<think><observation>The player is on the right side of the
grid.</observation><reasoning>The player is on the right side of the grid,
which is indicated by the position on the grid.</reasoning><prediction>The
player will move to the left or down.</prediction></think><answer>Left,
Left, Down</answer>
rest of the turns ...

```

1021

Navigation Training Prompt for Grounding-WorldModeling Strategy

System

You are a home robot and perform navigation tasks according to instructions. Actions you can take: moveahead, moveback, moveright, moveleft, rotateright, rotateleft, lookup, lookdown.

moveahead: Move forward by some distance

moveback: Move backward by some distance

moveright: Move rightward by some distance

moveleft: Move leftward by some distance

rotateright: Rotate to the right by 90 degrees

rotateleft: Rotate to the left by 90 degrees

lookup: Tilt the camera upward by 30 degrees

lookdown: Tilt the camera downward by 30 degrees

Rewards:

Format correct: +0.5

Achieve the human instruction: +10.0

The instruction will be provided with each observation. Look at the image carefully and navigate to complete the instruction.

Hints:

1. You can take multiple actions at a time, in most cases, if you find the target object is far away from you, you can call moveahead, moveleft and move right multiple times.

2. If you find yourself seems to be stuck, you can lookdown to see if there's any object above or below you, you can also rotate to see if there's any object behind you.

Example:

Round 1:

image_1

```

<think><observation>There is a garbage can in the upper left corner of the
image, next to the kitchen sink. To move there, we can go forward-left, but
since there's a kitchen counter directly ahead, we should go left
first.</observation><reasoning>Following the strategy, I can go by first
moving leftward.</reasoning><prediction>I will be in front of the
garbage</prediction></think>

```

```

<answer>moveleft, moveleft</answer>

```

```

<answer>moveleft, moveleft</answer>

```

Round 2:

Env_feedback: Last action is executed successfully.

image_2

```

<think><observation>From the scene, I see that by moving leftward, we are
getting closer to the garbage can. Now, the garbage can is in front of me,
slightly to the left. And there's a large area ahead of
us.</observation><reasoning>Following the strategy, I can go by first moving
forward then moving leftward.</reasoning><prediction>I will be closer to the
garbage</prediction></think>

```

```

<answer>moveahead, moveahead, moveahead, moveleft</answer>

```

```

<answer>moveahead, moveahead, moveahead, moveleft</answer>

```

Round 3:

Env_feedback: Last action is executed successfully.

image_3

```

<think><observation>From the image we can see the garbage can is very close
to us, still to our front-left. Moving leftward might be blocked but i can

```

1022

see that there is still space in front of me to get closer to the garbage
 can.</observation><reasoning>Following the strategy, we can take about two
 steps forward then one step left to reach the garbage
 can.</reasoning><prediction>I will reach the garbage</prediction></think>
 <answer>moveahead, moveahead,moveleft</answer>
 Round 4:
 Env_feedback: Success
 You can take up to 5 action(s) at a time, separated by ','.
 You should first give your thought process with the your observation,
 reasoning, and prediction of next state, then your answer.
 Both the observation and prediction should describe what you see or expect
 to see in the environment.
 Your response should be in the format of:
 <think><observation>...</observation><reasoning>...</reasoning>
 <prediction>...</prediction></think><answer>...</answer>
 e.g. <think><observation>I am at the entrance of a bedroom. There is a bed
 to the left, a desk with a lamp on the right, and a closet straight ahead.
 The target object, a book, appears to be on the
 desk.</observation><reasoning>I need to move toward the desk to reach the
 book. I'll turn right and move forward.</reasoning><prediction>I am now
 standing in front of the desk. The desk has a lamp, a computer, and several
 books on it. The target book is within reach on the right side of the
 desk.</prediction></think><answer>rotateright,moveahead,moveahead</answer>
 User
 [Initial Observation]:
 <image>
 Human Instruction: I am looking for a luminous device on my desk to read
 books at night. Can you navigate to that object and stay close?
 Decide your next action(s).
 You can take up to 5 action(s) at a time, separated by ','.
 You should first give your thought process with the your observation,
 reasoning, and prediction of next state, then your answer.
 Both the observation and prediction should describe what you see or expect
 to see in the environment.
 Your response should be in the format of:
 <think><observation>...</observation><reasoning>...</reasoning>
 <prediction>...</prediction></think><answer>...</answer>
 Assistant
 <think><observation>There is a luminous device on the desk to the right of
 the chair. To reach it, I should move forward and then turn
 right.</observation><reasoning>Following the strategy, I can go by first
 moving forward then turning right to reach the luminous
 device.</reasoning><prediction>I will be close to the luminous
 device</prediction></think><answer>moveahead, moveahead, moveright</answer>
 rest of the turns ...

1023

PrimitiveSkill Training Prompt for Grounding-WorldModeling Strategy

System

You are an AI assistant controlling a Franka Emika robot arm. Your goal is to understand human instructions and translate them into a sequence of executable actions for the robot, based on visual input and the instruction.

Action Space Guide

You can command the robot using the following actions:

1. pick(x, y, z) # To grasp an object located at position(x,y,z) in the robot's workspace.
2. place(x, y, z) # To place the object currently held by the robot's gripper at the target position (x,y,z).

1024

3. push(x1, y1, z1, x2, y2, z2) # To push an object from position (x1,y1,z1) to (x2,y2,z2).

Hints:

1. The coordinates (x, y, z) are in millimeters and are all integers.
2. Please ensure that the coordinates are within the workspace limits.
3. The position is the center of the object, when you place, please consider the volume of the object. It's always fine to set z much higher when placing an item.
4. We will provide the object positions to you, but you need to match them to the object in the image by yourself. You're facing toward the negative x-axis, and the negative y-axis is to your left, the positive y-axis is to your right, and the positive z-axis is up.

Examples:

round1:

image1

Human Instruction: Put red cube on green cube and yellow cube on left target

Object positions:

[(62,-55,20),(75,33,20),(-44,100,20),(100,-43,0),(100,43,0)]

Reasoning: I can see from the picture that the red cube is on my left and green cube is on my right and near me.

Since I'm looking toward the negative x axis, and negative y-axis is to my left, (62,-55,20) would be the position of the red cube, (75,33,20) would be the position of the green cube and (-44,100,20) is the position of the yellow cube.

Also the (100,-43,0) would be the position of the left target, and

(100,43,0) would be the position of the right target.

I need to pick up red cube first and place it on the green cube, when placing, I should set z much higher.

Answer: pick(62,-55,20)|place(75,33,50)

round2:

image2

Human Instruction: Put red cube on green cube and yellow cube on left target

Object positions:

[(75,33,50),(75,33,20),(-44,100,20),(100,-43,0),(100,43,0)]

Reasoning: Now the red cube is on the green cube, so I need to pick up the yellow cube and place it on the left target.

Answer: pick(-44,100,20)|place(100,-43,50)

You can take up to 2 action(s) at a time, separated by |.

You should first give your thought process with reasoning and prediction of next state, and then your answer.

Your response should be in the format of:

<think><observation>...</observation><reasoning>...</reasoning>

<prediction>...</prediction>

</think><answer>...</answer>

e.g. <think><observation>The red cube is on the left at (100,100,40), the green cube is on the right</observation><reasoning>I need to pick the red cube at (100,100,40) and place it at (80,100,60)</reasoning><prediction>After executing this action, the red cube will be at (80,100,60)</prediction></think>

<answer>pick(100,100,40)|place(80,100,60)</answer>

User

[Initial Observation]:

<image>

Human Instruction: Please put the apple in the drawer and close the drawer.

x_workspace_limit: (-500, -100)

y_workspace_limit: (-600, 200)

z_workspace_limit: (10, 650)

Object positions:

[(-127, 192, 36), (-500, -537, 200)]

Other information:


```
{'drawer_open_value': 263}
Decide your next action(s).
You can take up to 2 action(s) at a time, separated by |.
You should first give your thought process with reasoning and prediction of
next state, and then your answer.
Your response should be in the format of:
<think><observation>...</observation><reasoning>...</reasoning>
<prediction>...</prediction></think><answer>...</answer>
Assistant
<think><observation>The apple is on the ground, right side of the
drawer</observation><reasoning>I need to pick up the apple first, as it is
on the floor. After placing the apple in the drawer, I will close the
drawer.</reasoning><prediction>The apple will be placed in the drawer and
the drawer will be closed.</prediction></think><answer>pick(-127, 192,
36)|place(-500, -537, 500)|push(-500, -537, 500, -500, -537, 650)</answer>
rest of the turns ...
```

1026

SVG Reconstruction Training Prompt for Grounding-WorldModeling Strategy

System

You are a precise SVG code generator.

SVG Quick Guide

Goal: Transform the provided image into precise SVG code that replicates the image.

Process:

1. First analyze the image carefully, identifying distinct visual elements
2. Identify colors, dimensions, positions, and relationships between elements
3. Generate accurate SVG code that reproduces the image, you can use path for better shape

Rewards:

- Overall visual similarity: +5.0
- Structural accuracy: +10.0

Example:

```
<think><observation>I can see an image containing a red circle positioned at
the top-left area of the canvas, approximately at coordinates (25, 25) with
a radius of 15 units. There is also a blue rectangle at the bottom-right
area, sized about 30x20 units and positioned at coordinates (60,
60).</observation><reasoning>Based on my observation, I need to create an
SVG that precisely matches these elements. The circle appears to be slightly
too far right, so I should adjust its x-coordinate to 20 instead of 25. The
rectangle could benefit from being slightly
wider.</reasoning><prediction>After implementing these adjustments, the
resulting SVG should more closely match the original image. I expect the
similarity score to improve to approximately 0.98, as the modified positions
and dimensions will better represent the original
graphic.</prediction></think>
<answer><svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg">
  <circle cx="20" cy="25" r="15" fill="red" />
  <rect x="60" y="60" width="35" height="20" fill="blue" />
</svg></answer>
```

You can take up to 1 action(s) at a time, separated by ~.

You should first give your thought process with the your observation and reasoning, then predict next state, and finally the answer.

Both the observation and prediction should describe what you see or expect to see in the environment.

Your response should be in the format of:

```
<think><observation>...</observation><reasoning>...</reasoning>
<prediction>...</prediction></think><answer>...</answer>
```

1027

e.g. <think><observation>I can see an image containing a red circle positioned at the top-left area of the canvas, approximately at coordinates (25, 25) with a radius of 15 units. There is also a blue rectangle at the bottom-right area, sized about 30x20 units and positioned at coordinates (60, 60).</observation><reasoning>Based on my observation, I need to create an SVG that precisely matches these elements. The circle appears to be slightly too far right, so I should adjust its x-coordinate to 20 instead of 25. The rectangle could benefit from being slightly wider.</reasoning><prediction>After implementing these adjustments, the resulting SVG should more closely match the original image. I expect the similarity score to improve to approximately 0.98, as the modified positions and dimensions will better represent the original graphic.</prediction></think>

```
<answer><svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg">
  <circle cx="20" cy="25" r="15" fill="red" />
  <rect x="60" y="60" width="35" height="20" fill="blue" />
</svg></answer>
```

User

[Initial Observation]:

<image>

Please carefully observe the image, and generate SVG code that reproduces it as accurately as possible.

Decide on your SVG code.

You can take up to 1 action(s) at a time, separated by ~~.

You should first give your thought process with the your observation and reasoning, then predict next state, and finally the answer.

Both the observation and prediction should describe what you see or expect to see in the environment.

Your response should be in the format of:

```
<think><observation>...</observation><reasoning>...</reasoning>
<prediction>...</prediction></think><answer>...</answer>
```

Assistant

```
<think><observation>I can see an image of the letters "kr" inside a circle.
The circle is centered at the top of the image. The letters are inside the
circle, with the circle having a consistent thickness around the
perimeter.</observation><reasoning>The SVG should accurately represent the
circle with a consistent thickness around it and the letters "kr" inside.
The circle should be centered and have a specific size, and the letters
should be placed inside the circle.</reasoning><prediction>The SVG will need
to accurately depict the circle, the letters "kr", and their positioning
within the circle. The circle's size and position should be precise, and the
letters should be correctly placed inside.</prediction></think>
```

```
<answer><svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg">
  <circle cx="50" cy="50" r="25" stroke="black" stroke-width="1" fill="none"
/>
```

```
  <text x="50" y="50" font-size="24" font-family="sans-serif"
text-anchor="middle" alignment-baseline="central">kr</text>
</svg></answer>
```

rest of the turns ...

1028

1029 C Is There A Better Representation For Visual States

1030 In this section, we release the details of different visual state representations. To further understand
 1031 visual state reasoning, we investigate how different visual state representations affect task performance
 1032 by prompting models to use the Grounding-WorldModeling reasoning strategy and requiring them
 1033 to output specific formats for the <observation> and <prediction> fields.

1034 We explore different visual state representations across our environments: Sokoban and Frozen-
 1035 Lake support three representation formats (Natural-Lanaguage, Symbolic, and Structured),
 1036 while PrimitiveSkill supports two formats (Natural-Lanaguage and Structured). For all

Table 22: Visual state representation formats across different environments.

Environment	Format	Type	Description
Sokoban	Natural-Lanaguage	Text	Free-form textual descriptions of the game state
	Symbolic	Grid	Environment-native grid-based symbols representing game entities
	Structured	Dictionary	JSON-like dictionary containing positions of game objects
FrozenLake	Natural-Lanaguage	Text	Free-form textual descriptions of the game state
	Symbolic	Grid	Environment-native grid-based symbols representing game entities
	Structured	Dictionary	JSON-like dictionary containing positions of game objects
PrimitiveSkill	Natural-Lanaguage	Text	Free-form textual descriptions of the manipulation state
	Structured	Dictionary	JSON-like dictionary containing 3D coordinates of task-relevant objects

other tasks, we use Natural-Lanaguage visual state representation as the default. Examples of Natural-Lanaguage visual state representations for all tasks can be found in the prompts provided in Appendix B.2.

The different visual state representation formats are summarized in Table 22:

C.1 Symbolic Representations

For symbolic representations, we use environment-specific symbols to represent different entities in a grid format.

Sokoban Symbolic Format uses the following mapping:

Wall | _ Floor | O Target | X Box | P You | * Box on Target | S You on Target

An example of the symbolic representation:

```
#####
#_P__#
#_XO_#
###__#
#####
```

FrozenLake Symbolic Format uses the following mapping:

_ Frozen | O Hole | G Goal | P Player | X Player fell into hole | * Player on goal

An example of the symbolic representation:

```
___G
O___
__O_
_P__
```

C.2 Structured Representations

For structured representations, we use dictionary-based formats containing precise positional information of relevant objects.

Sokoban Structured Format contains player, box, and target positions along with grid dimensions.

An example of the symbolic representation:

```
{player_position: (4, 2), box_positions: [(3, 2)],
target_positions: [(3, 1)], grid_size: (6, 6)}
```

1068 **FrozenLake Structured Format** contains player, target, and hole positions along with grid dimen-
1069 sions.

1070 An example of the symbolic representation:

```
1071 {player_position: (3, 2), target_position: (3, 2),  
1072 hole_positions: [(1, 2)], grid_size: (4, 4)}
```

1073 **PrimitiveSkill Structured Format** contains 3D coordinates of task-relevant objects.

1074 An example of the symbolic representation:

```
1075 {red_cube: (100, 100, 40), green_cube: (200, 200, 60)}
```

1076 These different formats allow us to investigate how the choice of visual state representation affects the
1077 model’s ability to reason about and predict state transitions across different types of environments.

1078 **D How to improve Visual State Reasoning?**

1079 **D.1 Visual Reasoning Reward**

1080 To enhance visual state reasoning capabilities, we implement an LLM-as-Judge framework that
1081 evaluates the quality of the agent’s grounding and world modeling reasoning. The judge model used
1082 in our experiments is GPT-4.1 nano, which evaluates whether the agent’s descriptions and predictions
1083 accurately match the ground truth state information.

1084 Our evaluation system uses task-specific prompts for both grounding (current state description) and
1085 world modeling (next state prediction) assessment. The prompts are structured to guide the judge
1086 model through step-by-step reasoning before providing a binary YES/NO evaluation.

Sokoban Grounding Evaluation

Evaluate whether the description accurately captures the key position relationships in the Sokoban game state.
Answer YES if the directional relationships are correct, or NO if they contain directional errors.

Context

You are evaluating whether the description correctly identifies the directional relationships between:

1. The player and the box(es)
2. The box(es) and the target(s)

The description doesn't need to be perfectly precise or mention every detail
- it just needs to have the correct directional relationships (Up, Down, Left, Right).

Example:

Groundtruth Current State Information: ['box0 is at the same row and to the left of the player', 'target0 is above and on the left side of the player', 'target0 is above and on the left side of box0']

State Description: The player is below the box and the target is below the box.

- <think>The state description contains spatial relationship information, do further analysis. According to the ground truth data, box0 is at the same row and to the left of the player, target0 is above and on the left side of the player, target0 is above and on the left side of box0. The description states 'The player is below the box and the target is below the box.' The player is actually at the same row as the box (not below), and the target is actually above the box (not below). Both directional relationships are incorrectly identified.</think><answer>NO</answer>

Example:

1087

Groundtruth Current State Information: ['box0 is above and on the right side of the player', 'target0 is above and at the same column as the player', 'target0 is above and on the left side of box0']

State Description: The box is above the player and the target is to the left of the box

- <think>The state description contains spatial relationship information, do further analysis. According to the ground truth data, box0 is above and on the right side of the player, target0 is above and at the same column as the player, target0 is above and on the left side of box0. The description states 'The box is above the player and the target is to the left of the box.' It correctly identifies that the box is above the player (box0 is above and on the right side of the player). It also correctly identifies that the target is to the left of the box (target0 is above and on the left side of box0). Both key directional relationships are accurately described.</think><answer>YES</answer>

```
# Groundtruth Current State Information:
{state_information_dict}
```

```
# State Description:
{natural_language_description}
```

Think step by step:

1. Relative Relationship Requirements:
 - Must describe at least one relationships BETWEEN entities (player-box, player-target, box-target)
 - Absolute positions like "player is on the left side" are insufficient
 - Need relational descriptions like "player is left of target"
2. Essential Relationships to Check
 - Player-Target relationship (highest priority)
 - Player-Box relationship
 - Box-Target relationship
3. Equivalent Expression Recognition
 - "box is above player" = "player is below box"
 - "target is left of box" = "box is right of target"
 - Must recognize these as identical spatial relationships. Absolute position is not allowed

Your answer should be in the format of
<think>...</think><answer>YES</answer> or
<think>...</think><answer>NO</answer>.

1088

Sokoban World Modeling Evaluation

Evaluate whether the prediction correctly anticipates the key position relationships that will exist in the next Sokoban state.
Answer YES if the predicted directional relationships are correct, or NO if they contain directional errors.

Context

You are evaluating whether the prediction correctly identifies the directional relationships that will exist after the move:

1. The future position of the player relative to the box(es)
2. The future position of the box(es) relative to the target(s)

Important: The Prediction Comes First

Remember: The Next State Prediction is made BEFORE the Groundtruth Next State exists. Your task is to check if the prediction correctly anticipated what actually happened.

1089

If the box and target are at same position, this prediction is seen as success immediately (YES)

Example:

Groundtruth Next State Information: ['box0 is above and on the right side of the player', 'target0 is above and on the left side of the player', 'target0 is above and on the left side of box0']

Next State Prediction: The player will be to the left of the box, and the box will be to the right of the target.

- <think>The prediction state contains spatial relationship between player and target, do further analysis. According to the ground truth data, box0 is above and on the right side of the player, target0 is above and on the left side of the player, target0 is above and on the left side of box0. The description states 'The player will be to the left of the box, and the box will be to the right of the target.' It correctly identifies that the player is to the left of the box (since box0 is on the right side of the player). It also correctly identifies that the box is to the right of the target (since target0 is on the left side of box0). Therefore, this description correctly identifies the key directional relationships.</think><answer>YES</answer>

```
# Groundtruth Next State Information:
{state_information_dict}
```

```
# Next State Prediction:
{natural_language_description}
```

Think step by step:

1. Relative Relationship Requirements:
 - Must describe at least one relationships BETWEEN entities (player-box, player-target, box-target)
 - Absolute positions like "player is on the left side" are insufficient
 - Need relational descriptions like "player is left of target"
2. Essential Relationships to Check
 - Player-Target relationship (highest priority)
 - Player-Box relationship
 - Box-Target relationship
3. Equivalent Expression Recognition
 - "box is above player" = "player is below box"
 - "target is left of box" = "box is right of target"
 - Must recognize these as identical spatial relationships. Absolute position is not allowed

Your answer should be in the format of
<think>...</think><answer>YES</answer> or
<think>...</think><answer>NO</answer>.

1090

FrozenLake Grounding Evaluation

Evaluate whether the description accurately captures the key position relationships in the FrozenLake game state.

Answer YES if the directional relationships are correct, or NO if there are errors.

Context

You are evaluating whether the description correctly identifies:

1. The directional relationship between the player and the goal (MUST Have)
2. The directional relationship between the player and the hole (if present)

1091

The description doesn't need to be perfectly precise - it just needs to have the correct directional relationships between the player and target (Up, Down, Left, Right), and between the player and hole if applicable.

Groundtruth Current State Information:
{state_information_dict}

State Description:
{natural_language_description}

Think step by step:

1. Player relationship with Goal
 - Goal (Target) MUST include in state description, without target the description is automatically wrong (NO)
 - If there is no direction between player and goal, like "player is right to the target", the description is automatically wrong (NO)
 - This takes highest priority over all other considerations
2. Equivalent Expression Recognition
 - "goal is above player" = "player is below goal"
 - "target is left of box" = "box is right of target"
 - Must recognize these as identical spatial relationships. Absolute position is not allowed
3. Simple Judgment Rule
 - If player at goal -> YES
 - If direction aligns with needed movement -> YES
 - Otherwise -> NO

Your answer should be in the format of
<think>...</think><answer>YES</answer> or
<think>...</think><answer>NO</answer>.

1092

FrozenLake World Modeling Evaluation

Evaluate whether the prediction correctly anticipates the key aspects of the next FrozenLake state.

Answer YES if the prediction accounts for directional relationships and potential holes, or NO if it contains errors.

Context

You are evaluating whether the prediction correctly identifies:

1. The position relationship between the player and the goal after the prediction

Important: The Prediction Comes First

Remember: The Next State Prediction is made BEFORE the Groundtruth Next State exists. Your task is to check if the prediction correctly anticipated what actually happened.

The prediction doesn't need to perfectly describe every aspect of the next state - it just needs to correctly anticipate the directional relationships (Up, Down, Left, Right) or address any dangers from holes.

Groundtruth Next State Information:
{state_information_dict}

Next State Prediction:
{natural_language_description}

1093

Think step by step:

1. Player relationship with Goal

- If player is already at the goal position, the prediction is automatically correct (YES)
- Goal (Target) MUST include in prediction state, without target the prediction is automatically wrong (NO)
- If there is no direction between player and goal, like "player is right to the target", the prediction is automatically wrong (NO)
- This takes highest priority over all other considerations

2. Directional Correctness

- Evaluate if the predicted movement direction aligns with the relative position between player and goal
- For example, if player is left of goal, moving right is correct
- ****CRITICAL: Recognize equivalent expressions of the same spatial relationship****
 - * "player is above target" = "target is below player"
 - * "player is left of target" = "target is right of player"
 - * These are the SAME relationship expressed from different perspectives

3. Simple Judgment Rule

- If player at goal -> YES
- If direction aligns with needed movement -> YES
- Otherwise -> NO

Your answer should be in the format of

<think>...</think><answer>YES</answer> or
<think>...</think><answer>NO</answer>.

1094

Navigation Grounding Evaluation

Evaluate whether the description effectively communicates the spatial relationship between the agent and target object, even if the exact directional terms differ.

Answer YES if the overall spatial understanding is correct, or NO if it fundamentally misunderstands the spatial layout.

Context

You are evaluating whether the description effectively conveys where the target object is located relative to the agent. The exact directional terminology (left, right, ahead, etc.) may differ between the state information and the description, but the important factor is whether the description would lead to correct navigation.

Groundtruth Current State Information:

{state_information_dict}

State Description:

{natural_language_description}

Think step by step:

1. Check if the description contains spatial relationship between agent and target object

- If no spatial relationship is mentioned, answer NO

2. If spatial relationship exists, check if the predicted direction is consistent with the target direction

- "ahead/forward" = "ahead"
- "left" = "left"
- "right" = "right"
- Combined directions like "forward-left", "forward-right" are acceptable if they include the correct primary direction

1095

3. The prediction is correct if it mentions moving toward the target in a direction that reasonably aligns with the groundtruth direction

Your answer should be in the format of
`<think>...</think><answer>YES</answer>` or
`<think>...</think><answer>NO</answer>`.

Navigation World Modeling Evaluation

Evaluate whether the prediction effectively anticipates how the agent would navigate toward the target object, even if the exact directional terms differ.

Answer YES if the overall navigation plan is reasonable, or NO if it misunderstands or did not mention the spatial layout.

Context

You are evaluating whether the prediction effectively anticipates how the agent would move to reach the target object. The exact directional terminology (left, right, ahead, etc.) may differ between the state information and the prediction, but the important factor is whether the prediction would lead to successful navigation.

Important: The Prediction Comes First

Remember: The Next State Prediction is made BEFORE the Groundtruth Next State exists. Your task is to check if the prediction correctly anticipated what actually happened.

Groundtruth Next State Information:
`{state_information_dict}`

Next State Prediction:
`{natural_language_description}`

Think step by step:

1. First, check if the prediction explicitly uses EXACT directional terms that appear in the groundtruth state: "ahead", "left", "right", "up", "down".
 - Terms like "move towards", "closer to", "near", "approaching", "in front of", "by", "at" DO NOT qualify
 - "Will be on the left/right/ahead" or "Will move left/right/forward" DO qualify
 - If no exact directional match to groundtruth is present, conclude with NO immediately
2. If explicit direction words exist, verify that they EXACTLY match the target object's direction in the groundtruth:
 - If target is "ahead", prediction must specify "ahead", "forward", "slightly left", OR "slightly right" (special case: we allow slightly left/right for ahead targets)
 - If target is "right", prediction must specify "right"
 - If target is "left", prediction must specify "left"
3. Even if the prediction mentions intermediate objects correctly, it MUST explicitly state the correct final direction to the target object
4. The prediction cannot substitute object references for directions (saying "move to X" instead of "move right")
5. Remember that the prediction was made BEFORE the groundtruth state was determined

Your answer should be in the format of
`<think>...</think><answer>YES</answer>` or
`<think>...</think><answer>NO</answer>`.
`\end{prompt}`

```

\begin{prompt}[PrimitiveSkill Grounding
Evaluation]{prompt:judge-maniskill-grounding}
Compare the description of the current state with the groundtruth current
state information.
Answer YES if the description reasonably matches the current state
information, or NO if it doesn't.

# Context
You are evaluating whether an agent's description accurately reflects the
actual state. The description should capture the meaningful relationships
and positions relevant for decision-making in the task.

Important evaluation criteria:
1. If the description includes coordinates, they don't need to be exact
matches with the groundtruth
2. For coordinate values, consider them correct if they are within these
error tolerances:
  - For x and y coordinates: within +10 or -10 units of groundtruth
  - For z coordinates: within +10 or -10 units of groundtruth
3. The overall spatial relationships and object identifications should be
correct
4. If the description includes a dict-formatted state information, that's
good but not required

# Groundtruth Current State Information:
{state_information_dict}

# State Description:
{natural_language_description}

Think step by step:
1. Identify the key objects and their positions in the groundtruth
information
2. Check if the description correctly identifies these objects
3. For any coordinates mentioned, check if they are within the acceptable
error range
4. Determine if the overall spatial understanding is correct, even if
specific numbers differ slightly
5. If coordinates in the description differ from groundtruth but are within
the error tolerance, consider them correct

Your answer should be in the format of
<think>...</think><answer>YES</answer> or
<think>...</think><answer>NO</answer>.

```

1098

PrimitiveSkill World Modeling Evaluation

```

Compare the prediction of the next state with the groundtruth next state
information.
Answer YES if the prediction reasonably matches the next state information,
or NO if it doesn't.

# Context
You are evaluating whether an agent's prediction of the next state is
accurate. The prediction should capture the meaningful changes and
relationships that will result from the planned actions.

# Important: The Prediction Comes First
Remember: The Next State Prediction is made BEFORE the Groundtruth Next
State exists. Your task is to check if the prediction correctly anticipated
what actually happened.

```

1099

```

Important evaluation criteria:
1. If the prediction includes coordinates, they don't need to be exact
   matches with the groundtruth
2. For coordinate values, consider them correct if they are within these
   error tolerances:
   - For x and y coordinates: within +10 or -10 units of groundtruth
   - For z coordinates: within +10 or -10 units of groundtruth
3. The overall predicted movements and resulting spatial relationships
   should be correct
4. If the prediction includes a dict-formatted state information, that's
   good but not required

# Groundtruth Next State Information:
{state_information_dict}

# Next State Prediction:
{natural_language_description}

Think step by step:
1. Identify the key objects and their positions in the groundtruth next
   state information
2. Check if the prediction correctly anticipates these object positions
3. For any coordinates mentioned, check if they are within the acceptable
   error range
4. Determine if the overall predicted movement and resulting state is
   correct, even if specific numbers differ slightly
5. If coordinates in the prediction differ from groundtruth but are within
   the error tolerance, consider them correct

Your answer should be in the format of
<think>...</think><answer>YES</answer> or
<think>...</think><answer>NO</answer>.

```

1100

1101 **Text-based State Information** To provide ground truth information for the judge model, we extract
 1102 and format state information differently for each environment:

1103 **Sokoban:** We extract the positions of players, boxes, and targets, then convert their relative positions
 1104 into natural language sentences. For example:

```

1105 ['box0 is above and at the same column as the player',
1106  'target0 is above and at the same column as the player',
1107  'target0 is above and at the same column as box0']

```

1108 **FrozenLake:** Similarly, we convert positional relationships into descriptive sentences:

```

1109 ['target is above and at the same column as the player',
1110  'hole0 is above and at the same column as the player',
1111  'hole1 is above and on the right side of the player',
1112  'hole2 is above and on the right side of the player',
1113  'hole3 is above and on the right side of the player',
1114  'hole4 is at the same row and to the right of the player']

```

1115 **Navigation:** We use dictionary-based information directly from the environment:

```

1116 {'target_obj_type': 'Toaster',
1117  'target_distance_to_player': 2.59,
1118  'target_direction_to_player': 'ahead',
1119  'visible_objects': [{'type': 'Cabinet',
1120                       'direction_to_player': 'left',
1121                       'distance_to_player': 0.94},
1122  {'type': 'Drawer',

```

```

1123     'direction_to_player': 'left',
1124     'distance_to_player': 1.18},
1125     {'type': 'CounterTop',
1126      'direction_to_player': 'left',
1127      'distance_to_player': 1.33},
1128     ...]}

```

1129 **PrimitiveSkill:** We provide coordinate-based position information:

```

1130 {'left_target_position': (80, -100, 0),
1131  'right_target_position': (80, 100, 0),
1132  'red_cube_position': (83, -102, 20),
1133  'green_cube_position': (-47, 95, 20)}

```

1134 D.2 Bi-Level General Advantage Estimation (GAE)

1135 This section provides the detailed pseudo-code for our Bi-Level GAE algorithm, which is shown in
 1136 Algorithm 2. The key modification from the base RL algorithm is in Phase 2, where we introduce a
 1137 two-stage advantage estimation process that operates at both turn and token levels.

1138 D.3 Visual Reasoning RL

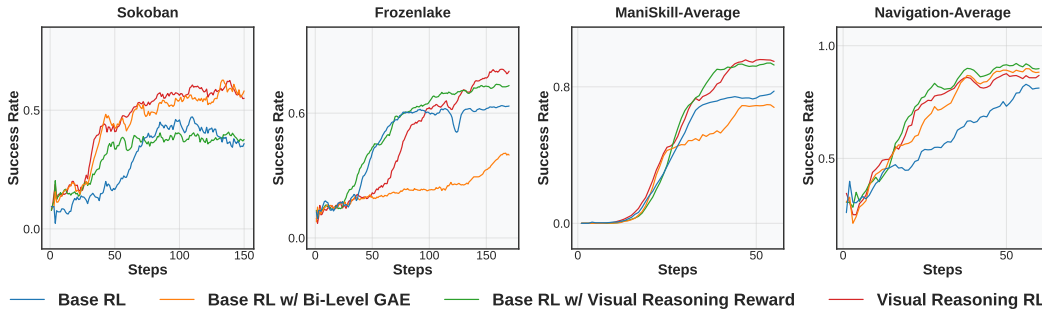


Figure 6: Training success rates for Base RL, Visual Reasoning RL and ablations.

1139 In this section, we study the individual contribution of Bi-Level GAE and Visual Reasoning Reward
 1140 to our Visual Reasoning RL approach. As shown in Figure 6, different components contribute
 1141 differently across tasks. The full Visual Reasoning RL approach (combining Bi-Level GAE with
 1142 Visual Reasoning Reward) consistently delivers the most stable and generally best performance
 1143 across all environments. However, we observe interesting patterns when examining the components
 1144 individually:

1145 **Bi-Level GAE Contribution.** Base RL with Bi-Level GAE shows significant improvements in
 1146 some environments (particularly in Sokoban) but demonstrates less stable performance in others
 1147 (such as FrozenLake and ManiSkill). This variability may stem from Bi-Level GAE’s sensitivity to
 1148 the design of intermediate rewards. Since most of our environments lack task-specific intermediate
 1149 rewards, Bi-Level GAE primarily responds to format rewards. While format correctness is relatively
 1150 easy to learn, this limited reward signal may lead to insufficient advantage gain in later training
 1151 stages. Without more nuanced rewards to guide exploration, this can potentially result in performance
 1152 plateaus or even degradation due to mode collapse.

1153 **Visual Reasoning Reward Contribution.** Base RL with Visual Reasoning Reward alone shows
 1154 consistent improvements over the Base RL across all environments, particularly in Navigation and
 1155 ManiSkill tasks. However, its performance is not as stable as the full Visual Reasoning RL approach.
 1156 This is likely because while the Visual Reasoning Reward provides valuable signals about the quality
 1157 of visual understanding, it is still constrained by the original trajectory-level reward structure, which
 1158 cannot effectively differentiate the quality of visual reasoning at each individual turn.

1159 **Combined Approach.** When Bi-Level GAE and Visual Reasoning Reward are combined in our full
 1160 Visual Reasoning RL approach, we observe the most consistent and robust performance improvements.
 1161 The Bi-Level GAE provides finer-grained credit assignment that helps the model accurately attribute
 1162 rewards to specific actions, while the Visual Reasoning Reward ensures the model develops and

Algorithm 2 Multi-Turn Reinforcement Learning with Bi-Level GAE for VLM Agents

Input: Actor VLM π_θ , Environment, Critic V_ϕ , Reference VLM π_{ref}

Hyperparameters: Token discount factor γ_{token} , Turn discount factor γ_{turn} , GAE coefficients $\lambda^{\text{token}}, \lambda^{\text{turn}}$, PPO clip range ε , KL penalty coefficient β

Phase 1: Trajectory Collection

- 1: Initialize trajectory $\tau = []$ and sample initial observation o_0 from Environment
 - 2: **for** $t = 0, \dots, N - 1$ **do**
 - 3: Sample agent output $a_t \sim \pi_\theta(\cdot | o_0, a_0, \dots, a_{t-1}, o_t)$ autoregressively
 where a_t follows structure, e.g. $\langle \text{think} \rangle_t \langle \text{think} \rangle \langle \text{answer} \rangle_{\bar{a}_t} \langle \text{answer} \rangle$
 with t_t as reasoning and \bar{a}_t as executable action
 - 4: Parse executable action \bar{a}_t from a_t
 - 5: Execute \bar{a}_t in Environment to receive next observation o_{t+1} and reward $r_{\text{env},t}$
 - 6: Compute format reward $r_{\text{format},t}$ and visual reasoning rewards $r_{\text{grounding},t}, r_{\text{worldmodeling},t}$
 - 7: Set turn reward $r_{\text{turn},t} = r_{\text{env},t} + r_{\text{format},t} + r_{\text{grounding},t} + r_{\text{worldmodeling},t}$
 - 8: Append $(o_t, a_t, r_{\text{turn},t})$ to trajectory τ
 - 9: **end for**
 - 10: Store token probabilities π_{old} for all tokens in actions a_0, \dots, a_{N-1}
-

Phase 2: Bi-Level Advantage Estimation

- 11: Let S_{tokens} be the concatenated sequence of all tokens from a_0, \dots, a_{N-1}
 - 12: Let T_{total} be the total number of tokens in S_{tokens}
 - 13: Let $k_{<i}$ denote the sequence prefix up to the i -th token in S_{tokens}
 - 14: Let p_0, p_1, \dots, p_{N-1} denote the positions of terminal tokens for each action
 - 15: Define mask M^{loss} where $M_i^{\text{loss}} = 1$ if token i is generated by π_θ , 0 otherwise
 - 16: **for** each turn $n = N - 1, \dots, 0$ (backwards) **do**
 - 17: Calculate turn-level TD error:
 $\delta_n^{\text{turn}} = r_{\text{turn},n} + \gamma_{\text{turn}} V_\phi(k_{p_{n+1}}) - V_\phi(k_{p_n})$
 (where $V_\phi(k_{p_N}) = 0$ for terminal state)
 - 18: **if** $n = N - 1$ (final turn) **then**
 - 19: $A_n^{\text{turn}} = \delta_n^{\text{turn}}$
 - 20: **else**
 - 21: $A_n^{\text{turn}} = \delta_n^{\text{turn}} + \gamma_{\text{turn}} \lambda^{\text{turn}} A_{n+1}^{\text{turn}}$
 - 22: **end if**
 - 23: **end for**
 - 24: **for** each turn $n = 0, \dots, N - 1$ **do**
 - 25: Let J_n be the number of tokens in action a_n
 - 26: **for** each token $j = J_n - 1, \dots, 0$ (backwards within turn) **do**
 - 27: Calculate token index $i = p_n - (J_n - 1 - j)$
 - 28: Calculate KL penalty: $r_i^{\text{KL}} = -\beta \cdot \text{KL}(\pi_\theta(\cdot | k_{<i}) \parallel \pi_{\text{ref}}(\cdot | k_{<i}))$
 - 29: Calculate token-level TD error:
 $\delta_{n,j}^{\text{token}} = r_i^{\text{KL}} + \gamma_{\text{token}} V_\phi(k_{i+1}) - V_\phi(k_i)$
 - 30: **if** $j = J_n - 1$ (terminal token of turn) **then**
 - 31: $A_{n,j}^{\text{token}} = \delta_{n,j}^{\text{token}} + A_n^{\text{turn}}$
 - 32: **else**
 - 33: $A_{n,j}^{\text{token}} = \delta_{n,j}^{\text{token}} + \gamma_{\text{token}} \lambda^{\text{token}} A_{n,j+1}^{\text{token}}$
 - 34: **end if**
 - 35: Set final advantage: $A_i^{\text{GAE}} = A_{n,j}^{\text{token}}$
 - 36: **end for**
 - 37: **end for**
 - 38: Compute returns for critic update: $\hat{R}_i = A_i^{\text{GAE}} + V_\phi(k_i)$ for all tokens i
-

Phase 3: Policy Update with PPO

- 39: **for** each token index $i = 0, \dots, T_{\text{total}} - 1$ in S_{tokens} **do**
 - 40: Compute probability ratio: $pr_i(\theta) = \frac{\pi_\theta(\text{token}_i | k_{<i})}{\pi_{\text{old}}(\text{token}_i | k_{<i})}$
 - 41: **end for**
 - 42: Compute PPO objective:
$$J^{\text{PPO}}(\theta) = \frac{\sum_{i=0}^{T_{\text{total}}-1} M_i^{\text{loss}} \cdot \min(pr_i(\theta) A_i^{\text{GAE}}, \text{clip}(pr_i(\theta), 1-\varepsilon, 1+\varepsilon) A_i^{\text{GAE}})}{\sum_{i=0}^{T_{\text{total}}-1} M_i^{\text{loss}}}$$
 - 43: Compute critic loss:
$$J^{\text{Critic}}(\phi) = \frac{\sum_{i=0}^{T_{\text{total}}-1} M_i^{\text{loss}} \cdot (V_\phi(k_i) - \hat{R}_i)^2}{\sum_{i=0}^{T_{\text{total}}-1} M_i^{\text{loss}}}$$
 - 44: Update parameters θ using $\nabla_\theta J^{\text{PPO}}(\theta)$ and ϕ using $\nabla_\phi J^{\text{Critic}}(\phi)$
 - 45: Return to **Phase 1** for next iteration
-

maintains strong visual grounding and world modeling capabilities throughout training. This synergy creates a more stable learning dynamic that consistently outperforms other variations across different environments.

E Case Study

In this section, we present a comprehensive case study examining the behavioral patterns and learning dynamics of multi-turn VLM agents across three distinct environments: FrozenLake, Navigation, and Sokoban. Through systematic analysis of agent responses during different training phases, we investigate how visual state reasoning capabilities evolve and identify key phenomena that emerge during the learning process.



Figure 7: Visual State Reasoning (Upper Case) Enhances Spatial Understanding in Frozenlake

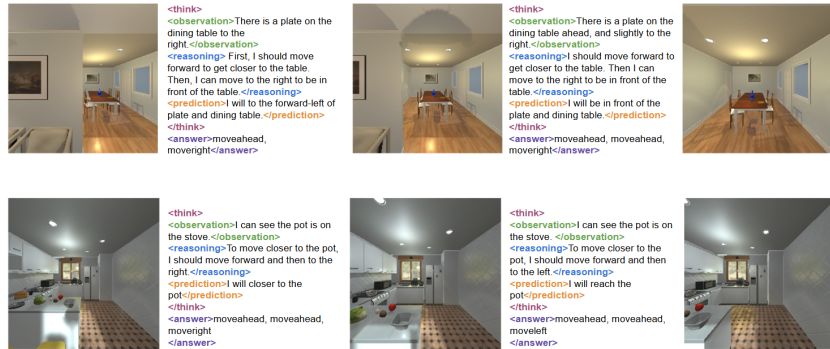


Figure 8: Visual State Reasoning (Upper Case) Enhances Multi-Step Planning in Navigation

E.1 Visual State Reasoning Enhances Spatial Understanding and Multi-Step Planning

Our analysis reveals that incorporating explicit visual state reasoning into multi-turn VLM agents significantly improves their spatial understanding and planning capabilities across different environments.

The integration of Grounding and WorldModeling steps enables agents to better understand relative positions between objects. In FrozenLake (Figure 7), agents trained with visual reasoning demonstrate improved ability to identify spatial configurations between the player, target, and holes. This allows them to effectively navigate around obstacles while maintaining progress toward objectives, contrasting with untrained agents that often make suboptimal moves without proper spatial consideration.

Visual state reasoning enables sophisticated multi-step planning. In Navigation tasks (Figure 8), agents develop the ability to not only identify target objects and their positions but also recognize blocking objects that impede direct paths. This leads to:

- **Accurate spatial relationship identification** between targets, obstacles, and agent position

- **Recognition of blocking constraints** that prevent direct paths
- **Multi-step sequence planning** that accounts for both immediate and longer-term goals

E.2 LLM-as-Judge Evaluation Framework

To ensure robust evaluation of visual reasoning quality, we designed a comprehensive LLM-as-Judge framework with carefully crafted evaluation templates that enable the judge model to fully understand task requirements and maintain high accuracy in assessment. Our evaluation protocol incorporates task-specific criteria that go beyond surface-level format checking, requiring the judge to evaluate the semantic correctness of spatial relationships, the logical consistency of predicted outcomes, and the appropriateness of reasoning given the current state configuration, which are reflected in Case 1 and Case 2.

LLM as Judge: Yes

Sokoban

State:

['target is below and at the same column as the player', 'hole0 is above and on the right side of the player', 'hole1 is at the same row and to the right of the player']

Response:

The player is positioned above the goal (G) with a hole directly in front of them.

Judge Model Reasoning:

<think>First, check the relationship between the player and the goal (target). The ground truth states: 'target is below and at the same column as the player.' This means the goal is below the player, or equivalently, the player is above the goal. The description says: "The player is positioned above the goal," which matches the ground truth. Since the description correctly states that the player is above the goal, the first key relationship is accurately captured. Next, consider the relationship between the player and the hole. The ground truth indicates: 'hole0 is above and on the right side of the player,' and 'hole1 is at the same row and to the right of the player.' The description mentions: "a hole directly in front of them," but does not specify the position relative to the player. Since the description does not specify the position of the hole relative to the player, and the key focus is on the goal, the main relationship (player to goal) is correctly captured. Given that the critical relationship between the player and the goal is correctly described, and the description does not contain errors regarding the key positional relationships, the overall description is accurate. Therefore, the answer is YES.</think><answer>YES</answer>

Navigation

State:

{'target_obj_type': 'DeskLamp', 'target_distance_to_player': 2.89, 'target_direction_to_player': 'ahead', 'visible_objects': [{'type': 'Floor', 'direction_to_player': 'left', 'distance_to_player': 0.98}, {'type': 'Statue', 'direction_to_player': 'ahead', 'distance_to_player': 1.3}, {'type': 'Painting', 'direction_to_player': 'ahead', 'distance_to_player': 1.72}, {'type': 'Painting', 'direction_to_player': 'ahead', 'distance_to_player': 1.92}, {'type': 'CD', 'direction_to_player': 'ahead', 'distance_to_player': 2.14}]}

Response:

There is a lamp on the desk to the left. To reach the lamp, I should move forward and then slightly to the left.

Judge Model Reasoning:

<think>The description states that the lamp is "on the desk to the left" and suggests moving "forward and then slightly to the left" to reach it. The groundtruth indicates the target is "ahead" of the agent. Moving forward and then slightly left aligns with approaching the target in the "ahead" direction, and the description does mention a spatial relationship ("to the

left") relative to the lamp. Since the primary movement is forward, which matches the target direction, and the description includes a slight left adjustment, it effectively communicates the spatial relationship and the approximate direction toward the target object. Therefore, the overall spatial understanding is correct.</think><answer>YES</answer>

1197

LLM as Judge: No

Sokoban

State:

['box0 is above and on the left side of the player', 'target0 is above and on the left side of the player', 'target0 is above and at the same column as box0']

Response:

The player is to the left of the box, and the target is to the right of the box.

Judge Model Reasoning:

<think>According to the ground truth, the box0 is above and on the left side of the player, and target0 is also above and on the left side of the player, with target0 being at the same column as box0. This indicates that the box and target are both positioned above and to the left relative to the player, and the target is aligned vertically with the box. The description states: "The player is to the left of the box, and the target is to the right of the box." - "Player is to the left of the box" implies player is on the left side of the box, which conflicts with the ground truth that the box is above and on the left side of the player (meaning the player is below or to the right of the box). - "Target is to the right of the box" conflicts with the ground truth that target0 is above and on the left side of the box, which would mean the target is generally to the left and above the box, not to the right. Therefore, the description incorrectly describes the directional relationships between the entities based on the ground truth data. Answer: NO</think><answer>NO</answer>

Navigation

State:

```
{'target_obj_type': 'CellPhone', 'target_distance_to_player': 2.59,
'target_direction_to_player': 'ahead', 'visible_objects': [{'type':
'GarbageCan', 'direction_to_player': 'ahead', 'distance_to_player': 2.2},
{'type': 'CD', 'direction_to_player': 'ahead', 'distance_to_player': 2.55},
{'type': 'CellPhone', 'direction_to_player': 'ahead', 'distance_to_player':
2.59}, {'type': 'CreditCard', 'direction_to_player': 'ahead',
'distance_to_player': 2.6}, {'type': 'DeskLamp', 'direction_to_player':
'ahead', 'distance_to_player': 2.6}]}
```

Response:

There is a table with a cell phone on it in the middle of the room. The cell phone is on the table, close to the center of the room.

Judge Model Reasoning:

<think>The description states that the cell phone is on a table in the middle of the room and is "close to the center of the room." It does not specify any directional relationship between the agent and the cell phone, such as "ahead," "left," or "right." Since no explicit spatial relationship is provided, the description does not effectively communicate the position of the target object relative to the agent. Therefore, the overall spatial understanding is not conveyed accurately.</think><answer>NO</answer>

1198

1199 E.3 Response Convergence and Reduced Exploration

1200 We observe a notable pattern of response evolution across different environments. As shown in
 1201 Cases 3 through 6, agents exhibit dramatically different response behaviors between early and late
 1202 training phases.

1203 During early training, agents demonstrate strong exploratory tendencies with diverse response patterns.
1204 In Sokoban tasks, early responses show varied reasoning approaches, creative problem-solving
1205 strategies, and rich descriptive language when analyzing spatial configurations. Similarly, Navigation
1206 tasks initially elicit detailed environmental descriptions and multi-faceted planning considerations.
1207 As training progresses, agents develop more streamlined and consistent communication patterns.
1208 Late-stage responses become notably more concise and structured, with standardized phrasing that pri-
1209 marily varies in directional or specific action tokens while maintaining clear, efficient expression. This
1210 evolution toward templated sentence structures occurs across different environments and reasoning
1211 qualities, suggesting that agents learn to optimize their communication for task effectiveness.

Sokoban - Early Training

World Modeling

State:

['box0 is at the same row and to the right of the player', 'target0 is at the same row and to the right of the player', 'target0 is at the same place as box0']

Response:

The player will move down, then to the right, pushing the box to the target.

State:

['box0 is above and at the same column as the player', 'target0 is above and at the same column as the player', 'target0 is above and at the same column as box0']

Response:

Push the box down and then move to the right. These actions would bring the box closer to the target.

Grounding

State:

['box0 is above and at the same column as the player', 'target0 is above and on the left side of the player', 'target0 is above and on the left side of box0']

Response:

The player is above the box, and the player is below the target, and there is a wall between the player and the target. The box is above the player

State:

['box0 is above and on the right side of the player', 'target0 is above and on the right side of the player', 'target0 is above and on the right side of box0']

Response:

From the image, the player is at the bottom of the wall, with the box directly in front of the player to the right and the target directly above the box.

1212

Sokoban - Late Training

World Modeling

State:

['box0 is at the same row and to the left of the player', 'target0 is above and on the left side of the player', 'target0 is above and at the same column as box0']

Response:

The player will move to the left, then push the box left to the target.

State:

['box0 is at the same row and to the right of the player', 'target0 is above and on the right side of the player', 'target0 is above and at the same column as box0']

1213

1214

Response:

The player will move to the right, then push the box right to the target.

Grounding

State:

['box0 is above and at the same column as the player', 'target0 is above and on the right side of the player', 'target0 is above and on the right side of box0']

Response:

The player is below the box, and the box is below the target.

State:

['box0 is below and on the left side of the player', 'target0 is below and on the left side of the player', 'target0 is at the same row and to the left of box0']

Response:

The player is to the right of the box, and the box is to the right of the target.

Navigation - Early Training

World Modeling

State:

```
{'target_obj_type': 'Laptop',  
 'target_distance_to_player': 2.58,  
 'target_direction_to_player': 'right',  
 'visible_objects': [{'type': 'Bed',  
   'direction_to_player': 'ahead',  
   'distance_to_player': 1.22},  
 {'type': 'Pillow',  
   'direction_to_player': 'left',  
   'distance_to_player': 1.35},  
 ...]}
```

Response:

I can see a laptop on the desk. I will move forward and right to get closer to the laptop.

State:

```
{'target_obj_type': 'Laptop',  
 'target_distance_to_player': 2.31,  
 'target_direction_to_player': 'ahead',  
 'visible_objects': [{'type': 'Drawer',  
   'direction_to_player': 'ahead',  
   'distance_to_player': 0.85},  
 {'type': 'Television',  
   'direction_to_player': 'ahead',  
   'distance_to_player': 1.3},  
 ...]}
```

Response:

My next move will be to approach the table where the laptop is placed. The laptop is placed on the right side of table in front of me.

Grounding

State:

```
{'target_obj_type': 'CellPhone',  
 'target_distance_to_player': 2.59,  
 'target_direction_to_player': 'ahead',  
 'visible_objects': [{'type': 'GarbageCan',  
   'direction_to_player': 'ahead',  
   'distance_to_player': 2.2},  
 {'type': 'CD',
```

1215

```
    'direction_to_player': 'ahead',
    'distance_to_player': 2.55},
    ...]]
```

Response:

There is a table with a cell phone on it in the middle of the room. The cell phone is on the table, close to the center of the room, right beside a trash can.

State:

```
{'target_obj_type': 'Statue',
 'target_distance_to_player': 2.54,
 'target_direction_to_player': 'ahead',
 'visible_objects': [{'type': 'Drawer',
   'direction_to_player': 'ahead',
   'distance_to_player': 1.99},
 {'type': 'Drawer',
   'direction_to_player': 'ahead',
   'distance_to_player': 1.99},
 ...]]
```

Response:

There is a statue on a table in the room. I need to move forward in order to reach the statue. The floor is smooth, so I can move forward without obstacles.

1216

Navigation - Late Training

World Modeling

State:

```
{'target_obj_type': 'DeskLamp',
 'target_distance_to_player': 1.71,
 'target_direction_to_player': 'right',
 'visible_objects': [{'type': 'Chair',
   'direction_to_player': 'ahead',
   'distance_to_player': 1.08},
 {'type': 'Laptop',
   'direction_to_player': 'ahead',
   'distance_to_player': 1.19},
 ...]]
```

Response:

I will move forward-right to reach the lamp.

State:

```
{'target_obj_type': 'Pot',
 'target_distance_to_player': 2.54,
 'target_direction_to_player': 'left',
 'visible_objects': [{'type': 'Bowl',
   'direction_to_player': 'ahead',
   'distance_to_player': 0.88},
 {'type': 'Potato',
   'direction_to_player': 'ahead',
   'distance_to_player': 1.2},
 ...]]
```

Response:

I will move forward-left to reach the pot.

Grounding

State:

```
{'target_obj_type': 'Bowl',
 'target_distance_to_player': 1.52,
 'target_direction_to_player': 'ahead',
 'visible_objects': [{'type': 'Book',
```

1217

```

'direction_to_player': 'right',
'distance_to_player': 1.18},
{'type': 'Floor',
'direction_to_player': 'back',
'distance_to_player': 1.3},
...]}

```

Response:

There is a round container on the floor. To reach the round container, I need to move forward.

State:

```

{'target_obj_type': 'GarbageCan',
'target_distance_to_player': 1.71,
'target_direction_to_player': 'ahead',
'visible_objects': [{'type': 'GarbageBag',
'direction_to_player': 'ahead',
'distance_to_player': 1.42},
{'type': 'GarbageCan',
'direction_to_player': 'ahead',
'distance_to_player': 1.71},
...]}

```

Response:

There is a trash can in front of the table to the right. To reach the trash can, I need to move forward-right.

1218

1219 E.4 Reward Hacking in LLM-as-Judge Evaluation

1220 We observe interesting patterns of reward optimization behavior, particularly in spatially sensitive
 1221 environments like FrozenLake where agents learn to adapt their responses to the LLM-as-Judge
 1222 evaluation mechanism. As demonstrated in Cases 7 and 8, agents develop strategic approaches to
 1223 maximize reward signals, though not always in ways that reflect deeper reasoning improvements.

1224 The most notable optimization pattern involves agents converging toward generic, broadly applicable
 1225 responses that incorporate multiple directional possibilities and general spatial relationships. Rather
 1226 than providing highly specific, state-dependent analysis, agents learn to produce consistent reasoning
 1227 templates that reliably satisfy the reward mechanism's evaluation criteria. For instance, responses like
 1228 "The player will be at the position of the gift" emerge as universal patterns that can be applied across
 1229 diverse state configurations, effectively balancing reward acquisition with computational efficiency
 1230 while potentially sacrificing reasoning specificity.

1231 This optimization behavior becomes more pronounced under Bi-Level training configurations. The
 1232 enhanced optimization method and turn-aware reward structure enable agents to become more
 1233 sensitive to reward patterns and develop sophisticated strategies for consistent reward acquisition.
 1234 The Bi-Level GAE's expanded exploration capabilities and improved credit assignment allow agents
 1235 to efficiently identify and adapt to the evaluation criteria, demonstrating the system's effectiveness
 1236 in learning reward-maximizing behaviors, though this may sometimes diverge from the intended
 1237 reasoning improvements.

FrozenLake - Early Training

World Modeling

State:

```

['target is at the same row and to the left of the player', 'hole0 is at the
same place as the player', 'hole1 is below and on the left side of the
player', 'hole2 is below and on the right side of the player']

```

Response:

The player will either go left or right to avoid the hole and continue moving up to reach the icy blocks.

State:

1238

['target is above and on the right side of the player', 'hole0 is above and on the right side of the player']

Response:

The player will be in the bottom center after moving right, and then move down to the center of the gift.

Grounding

State:

['target is below and on the left side of the player', 'hole0 is below and on the left side of the player']

Response:

The player is standing on a block, there is a hole directly in front of the player, and there is a goal at the left corner of the player.

State:

['target is above and on the left side of the player', 'hole0 is above and on the left side of the player', 'hole1 is below and on the left side of the player', 'hole2 is below and at the same column as the player']

Response:

The player is now at the right side of the grid, adjacent to a hole. The goal (G) is on the left side of the player.

1239

FrozenLake - Late Training

World Modeling (Hacking)

State:

['target is at the same place as the player', 'hole0 is below and on the right side of the player']

Response:

The player will be at the position of the gift.

State:

['target is above and on the right side of the player']

Response:

The player will be at the position of the gift.

Grounding (Hacking)

State:

['target is below and on the right side of the player', 'hole0 is above and on the right side of the player', 'hole1 is below and on the right side of the player']

Response:

The player is at the top left, the gift is at the bottom right, and there are holes at the top and bottom right corners.

State:

['target is at the same row and to the left of the player']

Response:

The player is in the center of the grid, the gift is on the left side, and there is a hole on the right side.

1240

1241 F Summary of Findings

1242 Our investigation into reinforcing visual state reasoning for multi-turn VLM agents, facilitated by
1243 the VAGEN training framework, has yielded several key findings regarding the efficacy of explicit
1244 reasoning, optimal state representations, specialized RL techniques, and observed agent behaviors
1245 during training. These insights are crucial for developing more robust and capable VLM agents

1246 that can understand and interact with dynamic visual environments. Below is the summary of our
1247 findings.

Finding 1: Explicit Visual State Reasoning is Crucial for Multi-Turn VLM Agents

Vanilla VLMs struggle with multi-turn agentic tasks requiring visual state understanding. Integrating explicit visual state reasoning steps—specifically **Grounding** and **WorldModeling**—into the VLM’s thinking process during RL training significantly enhances task performance. The combined Grounding-WorldModeling strategy, in particular, demonstrates strong and stable performance, enabling a trained open-source VLM to outperform its un-trained counterpart and even surpass benchmarked proprietary models.

1248

Finding 2: Optimal Visual State Representation is Task-Dependent

The choice of representation for visual states during explicit reasoning significantly impacts performance.

- **Natural Language:** Performs consistently well, especially when structured information must be inferred from raw visual input.
- **Structured Formats:** Excel in manipulation-heavy tasks (e.g., PrimitiveSkill) where object-centric state abstractions are readily available.
- **Symbolic Representations:** Proved less effective due to the model’s limited prior interpretability from visual input.

1249

Finding 3: Visual Reasoning RL with Targeted Rewards and Bi-Level GAE Enhances Reasoning Quality and Task Success

To specifically improve visual state reasoning, Visual Reasoning RL incorporates:

- **Turn-level Visual Reasoning Reward:** An LLM-as-Judge assesses the accuracy of the VLM’s explicit state descriptions and predictions, effectively supervising reasoning.
- **Bi-Level General Advantage Estimation (GAE):** Estimates advantages at both turn and token levels, providing finer-grained reward signals and improving credit assignment.

This approach consistently outperforms Base RL, leading to improved reasoning quality, higher task success rates, and better generalization.

1250

Finding 4: Emergent Reasoning Patterns and Challenges

Beyond quantitative measurements, we qualitatively analyzed how agents learn to reason:

- **Reasoning Stability Varies by Task:** While reasoning in tasks like Navigation and PrimitiveSkill (and often Sokoban) appears relatively normal and beneficial with explicit rewards, tasks like FrozenLake show more erratic reasoning patterns, potentially correlating with its lower performance and the difficulty of its visual state reasoning.
- **Potential for Reward Hacking:** Instances of "reward hacking" were observed, particularly with certain reward configurations (e.g., Bi-Level WorldModeling in some contexts). Agents might learn to generate reasoning-like text that satisfies the reward mechanism (e.g., format reward, or even LLM-judge for simple cases) without genuinely reflecting deep understanding or accurate future prediction. This suggests the LLM-as-Judge mechanism, while an improvement, is not infallible and can be gamed.
- **Bi-Level GAE as a Double-Edged Sword:** While Bi-Level GAE can improve credit assignment, its interaction with visual reasoning rewards might sometimes allow for more "divergent" or less grounded thinking if the reasoning reward itself can be easily hacked. Well-designed, hard-to-game reasoning rewards are crucial for Bi-Level GAE to be consistently beneficial; otherwise, it might amplify the effects of a poor intermediate reward.
- **Convergence to Standardized Phrasing:** Regardless of initial hacking or reasoning quality, agents across different environments tend to converge towards using a more uniform, templated sentence structure for their reasoning and actions over prolonged training, primarily varying only the directional or specific action tokens. This might be an efficiency learned for tasks ultimately requiring discrete actions but could also indicate a reduction in diverse or creative reasoning.
- **Rule-Based Filtering as a Potential Mitigation:** For simpler forms of reward hacking where reasoning outputs fail basic semantic checks (e.g., not mentioning valid actions in a grid world), simple rule-based filtering before reward assignment could be a pragmatic interim solution.

These observations underscore that while explicit reasoning and rewards are beneficial, the design of these rewards must be robust against exploitation, and continuous monitoring of reasoning quality is essential.

1251