

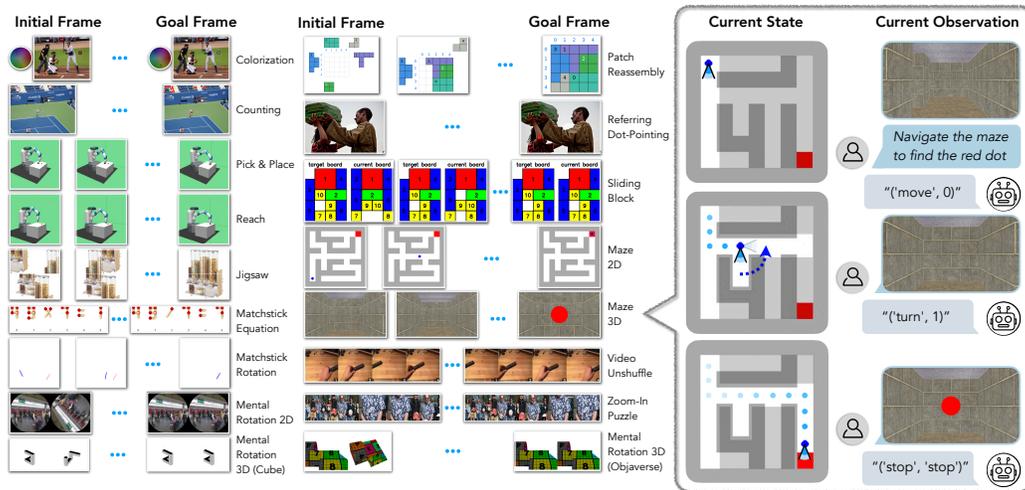
# VISGYM: DIVERSE, CUSTOMIZABLE, SCALABLE ENVIRONMENTS FOR MULTIMODAL AGENTS

Anonymous authors

Paper under double-blind review

## ABSTRACT

Modern Vision–Language Models (VLMs) remain poorly characterized in multi-step visual interactions, particularly in how they integrate perception, memory, and action over long horizons. We introduce VISGYM, a gymnasium of 17 environments for evaluating and training VLMs. The suite spans symbolic puzzles, real-image understanding, navigation, and manipulation, and provides flexible controls over difficulty, input representation, planning horizon, and feedback. We also provide multi-step solvers that generate structured demonstrations, enabling supervised finetuning. Our evaluations show that all frontier models struggle in interactive settings, achieving low success rates in both the easy (26.8%) and hard (12.6%) configurations. Our experiments reveal notable limitations: models struggle to effectively leverage long context, performing worse with an unbounded history than with truncated windows. Furthermore, we find that several text-based symbolic tasks become substantially harder once rendered visually. However, explicit goal observations, textual feedback, and exploratory demonstrations in partially observable or unknown-dynamics settings for supervised finetuning yield consistent gains, and solver-generated multi-step demonstrations generalize across tasks and domains. Together, these results position VISGYM as a principled and scalable foundation for diagnosing and training visually interactive agents.



**Figure 1: An overview of VISGYM.** (Left) VISGYM consists of 17 diverse, long-horizon environments designed to systematically evaluate, diagnose, and train VLMs on visually interactive tasks with different domains, levels of state observability, and types of observations. (Right) An example trajectory for the Maze 3D navigation task illustrates a partially observable environment consisting of non-structured synthetic renderings. Here, a VLM is prompted with (1) the task description (*simplified in the figure*) and (2) a set of available actions to use (*not shown in the figure for simplicity*). The agent must select each action conditioned on both its past actions and observation history for its decision-making.

## 1 INTRODUCTION

Humans navigate complex tasks in visually rich and interactive settings: manipulating objects, using devices, or exploring unfamiliar environments. Success in these settings hinges on the tight coupling

of perception, memory, and action over multiple steps (Gibson, 1979; Henderson, 2001). Foundation Vision–Language Models (VLMs) have made remarkable progress on static vision–language benchmarks (Yue et al., 2024; Lu et al., 2023; Wang et al., 2024) and on text-based multi-step tasks such as web browsing and coding (Sirdeshmukh et al., 2025; Wei et al., 2025; Jimenez et al., 2023). Yet when visual observations must be integrated into multi-step decision-making, their behavior remains far less understood. Recent evaluations across robotic manipulation, computer-use agents, and gaming agents highlight a range of challenges for visually interactive decision-making, including low task success rates, brittle visual grounding, and weak generalization (Zhang et al., 2025; Xie et al., 2024; Zhang et al., 2025; Liu et al., 2023; Hu et al., 2025; Chen et al., 2025; Shi et al., 2025; Liu et al., 2024). Although these insights are valuable, they tend to be domain-specific and observational, offering limited *systematic, controlled* diagnosis of how domain-agnostic factors such as context length, representation modality, feedback, or goal visibility affect model performance across tasks.

We introduce VISGYM, a highly diverse, scalable, and customizable gymnasium with 17 long-horizon environments designed to isolate what limits interactive decision-making across domains and to expose where current VLMs break down. The suite spans symbolic puzzles, real-image understanding, navigation, and manipulation tasks, each with distinct observability and dynamics and equipped with oracle multi-step solvers for supervised finetuning. Crucially, VISGYM provides fine-grained controls over input representation, difficulty, history length, planning horizon, and feedback, enabling domain-agnostic, systematic analysis of model behavior. Building on prior domain-specific studies, we conduct cross-domain controlled experiments that examine how these factors, together with module finetuning and data curation, affect performance in multi-step visual decision-making.

Across 11 state-of-the-art models, even the strongest achieve only 26.79% and 12.57% success in the easy and hard settings, respectively. Our analyses reveal several concrete, cross-domain failure modes: (1) models struggle to effectively leverage long-term context, showing a reversed-U relationship where performance degrades as the context grows unbounded; (2) VLMs struggle with low-level perceptual grounding, a limitation highlighted by symbolic variants of tasks being substantially easier than their visually rendered counterparts; (3) models struggle to infer task states and outcomes from purely visual transitions, consistently relying on explicit textual feedback to boost performance; (4) the benefit of providing explicit goal observations is brittle and can backfire: while explicit goals can yield large gains, limited visual perception can cause models to misidentify them and, paradoxically, perform worse than with no goal at all; (5) models fail to learn from standard demonstrations under partial observability or unknown dynamics, requiring information-revealing demonstrations that expose hidden states or clarify dynamics to significantly improve supervised finetuning outcomes.

Beyond diagnosis, we show that supervised finetuning on solver-generated multi-step demonstrations yields improvements that generalize across unseen tasks and transfer to downstream domains requiring visual interaction. Together, these findings establish VISGYM as a unified and extensible framework for diagnosing, understanding, and improving VLMs in visually interactive decision-making.

## 2 VISGYM

**VISGYM** contains 17 visually interactive environments. Each environment exposes initialization parameters that control task configuration and difficulty. We provide a high-level overview of the environments in Tab. 1 and detailed descriptions with visualizations in Sec. §C. VISGYM is built on top of the Gymnasium framework (Brockman et al., 2016; Towers et al., 2024), the same library underlying MuJoCo (Todorov et al., 2012) and Atari (Bellemare et al., 2013). Since vision–language agents can interpret images, read instructions, and produce free-form text, we extend Gymnasium with the following enhancements:

**Function-Conditioned Action Space.** Instead of the discrete or continuous action vectors used in standard Gymnasium environments, we represent actions as function calls with parameters (e.g., (`'swap'`, (1, 2)), (`'rotate'`, (30.5, 20.4, 15.1))). This abstraction allows models to leverage their function-calling capabilities and compose strategies across domains.

**Function Instructions.** Each task defines a set of functions and their parameter spaces. To enable zero-shot rollouts, we provide a natural-language description of these functions and their argument constraints as part of the initial prompt before the model takes its first action. Instructions for each task are shown in Sec. §C.

**Table 1: VISGYM environments.** For each environment, we specify (1) **Domain**: whether observations come from **Real** or **Synthetic** images, (2) **Observability (Obs.)**: **Full** or potentially **Partial**, (3) **Dynamics (Dyn.)**: **Known** vs. **Unknown** dynamics, (4) **Parameters (P)**: number of difficulty parameters, and (5) **Available Actions**.

Environment	Domain	Obs.	Dyn.	P.	Available Actions
Colorization (103)	Real	Full	Known	1	rotate( $\theta$ ); saturate( $\delta$ ); stop()
Counting (31)	Real	Full	Known	2	mark( $x, y$ ); undo(); guess( $N$ ); stop()
Jigsaw (28)	Real	Full	Known	2	swap( $(r_1, c_1), (r_2, c_2)$ ); reorder(...); stop()
Matchstick Equation (43)	Synthetic	Full	Known	1	move( $(i, s, j, t)$ ); undo(); stop()
Matchstick Rotation (45)	Synthetic	Full	Unknown	3	move( $(dx, dy, d\theta)$ ); stop()
Maze 2D (44)	Synthetic	Full	Known	2	move( $d$ ); stop()
Maze 3D (44)	Synthetic	Partial	Known	2	move( $d$ ); turn( $d$ ); stop()
Mental Rotation 2D (19)	Real	Full	Known	1	rotate( $\theta$ ); stop()
Mental Rotation 3D (CUBE) (67; 71)	Synthetic	Partial	Known	3	rotate( $(dx, dy, dz)$ ); stop()
Mental Rotation 3D (OBJVERSE) (71; 21)	Synthetic	Partial	Known	1	rotate( $(dx, dy)$ ); stop()
MuJoCo Fetch (PICK-AND-PLACE) (86)	Synthetic	Partial	Unknown	0	move( $(x, y, z)$ ); gripper( $g$ ); stop()
MuJoCo Fetch (REACH) (86)	Synthetic	Partial	Unknown	0	move( $(x, y, z)$ ); stop()
Patch Reassembly (29)	Synthetic	Full	Known	2	place( $p, r, c$ ); remove( $p$ ); stop()
Referring Dot-Pointing (40)	Real	Full	Known	0	mark( $x, y$ ); stop()
Sliding Block (76)	Synthetic	Full	Known	1	move( $(b, d)$ ); stop()
Video Unshuffle (30; 61)	Real	Full	Known	3	swap( $i, j$ ); reorder(...); stop()
Zoom-In Puzzle (6)	Real	Full	Known	5	swap( $i, j$ ); reorder(...); stop()

**Environment Feedback.** In addition to visual feedback as the model interacts with the environment, it optionally provides textual feedback describing the effect of each action (e.g., “invalid format,” “out of bounds,” “executed”). This helps models with weaker visual perception better ground their actions.

**Solver.** We implement heuristic multi-step solvers that complete each task using the available actions. The solver supports (1) multiple solving strategies and (2) optional stochasticity, enabling the generation of diverse demonstration trajectories for supervised fine-tuning. See Sec. §B for the solver design of each task.

### 3 EVALUATING FRONTIER MODELS WITH VISGYM

We evaluate 11 vision-language models spanning three categories: **proprietary** (Gemini 2.5 Pro (DeepMind, 2025), GPT-5 (OpenAI, 2025), Claude Sonnet 4 (Team, 2025), Grok 4 Fast (xAI, 2025), Qwen-VL-Max (Bai et al., 2025)); **open-weight** models (Qwen3-VL-235B-Instruct (Yang et al., 2025), GLM-4.5V (Hong et al., 2025), Llama-4-Maverick (Touvron et al., 2023), Qwen-2.5-VL-72B-Instruct (Bai et al., 2025), Gemma 3-27B-Instruct (Team et al., 2025)); and **specialized** models targeted at GUI/game environments (UI-Tars-1.5-7B (Qin et al., 2025)). We access all proprietary and hosted models through OpenRouter and thus ensure a consistent prompting interface and inference pipeline. We additionally evaluate models that we finetune on solver demonstrations. We provide details of the supervised finetuning setup in Sec. §5.1.

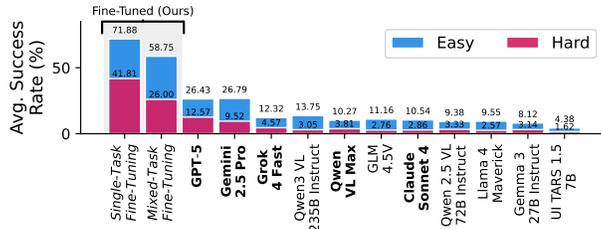
All models are evaluated in a multi-turn manner. At each step  $t$ , the model receives the full history

$$H_t = (I, \{(o_\tau, a_\tau, f_\tau)\}_{\tau < t}), \quad (1)$$

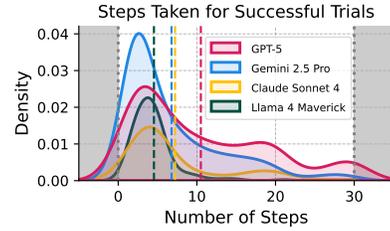
where  $I \in \mathbb{R}^{L_I}$  is the task instruction,  $o_\tau \in \mathbb{R}^{H \times W \times C}$  the observation,  $a_\tau \in \mathbb{R}^{L_a}$  the action, and  $f_\tau \in \mathbb{R}^{L_f}$  the environment feedback. The model then outputs an action  $a_t$ . If it outputs the stop action, the environment terminates and returns a binary reward indicating task success. In addition, we limit the number of interaction steps to 20 for the easy setting and 30 for the hard setting. All tasks are designed to be solvable within these limits, and the environment explicitly provides the number of remaining steps as part of its feedback. We also ensure that the length of interaction history is within models’ context window. We evaluate each model on 70 episodes per task and setting (i.e., easy, hard).

#### 3.1 RESULT AND ANALYSIS

**Frontier VLMs Fail on VISGYM.** We show the per-task success rate and the average task success rate of the frontier models in Figure 4 and Figure 2, respectively. The best-performing frontier model, GPT-5, achieves only 26.43% on VISGYM (Easy) and 12.57% on VISGYM (Hard), indicating that VISGYM poses a significant challenge for existing models.



**Figure 2: Average task success rate for frontier models and our finetuned models.** Proprietary models are in **bold** and our finetuned models are *italicized*.



**Figure 3: Density curve of steps taken for successful trajectories.** Colored dashed line marks each model’s mean number of steps.

	Task Success Rate per Model (%)																																		
9-Task Tuning	90.0	54.3	15.7	55.7	75.7	90.0	88.6	50.0	0.0	88.6	70.0	97.1	0.0	97.1	77.1	92.9	57.1	100.0	48.6	2.9	70.0	64.3	28.6	94.3	84.3	15.7	2.9	61.4	0.0	50.0	54.3				
5-Task Tuning	81.4	54.3	8.6	58.6	52.9	35.7	17.1	44.3	0.0	80.0	57.1	90.0	0.0	52.9	31.4	90.0	35.7	98.6	42.9	8.6	67.1	52.9	24.3	91.4	61.4	4.3	0.0	38.6	0.0	24.3	25.7				
<b>GPT-5</b>	67.1	40.0	24.3	50.0	30.0	27.1	17.1	77.1	41.4	58.6	52.9	18.6	1.4	18.6	4.3	11.4	4.3	7.1	17.1	0.0	4.3	0.0	0.0	11.4	4.3	5.7	1.4	11.4	1.4	1.4	1.4				
<b>emini 5 Pro</b>	71.4	51.4	20.0	50.0	40.0	24.3	11.4	48.6	1.4	31.4	22.9	27.1	0.0	28.6	10.0	30.0	14.3	10.0	14.3	5.7	2.9	4.3	0.0	32.9	12.9	1.4	1.4	1.4	0.0	1.4	0.0				
<b>Grok 4 Fast</b>	42.9	30.0	10.0	17.1	8.6	18.6	7.1	38.6	17.1	5.7	1.4	11.4	1.4	10.0	7.1	2.9	2.9	4.3	4.3	2.9	4.3	4.3	0.0	7.1	5.7	0.0	2.9	0.0	0.0	0.0	1.4	0.0			
<b>Qwen3 VL 235B Instruct</b>	90.0	40.0	14.3	27.1	10.0	12.9	8.6	0.0	0.0	0.0	0.0	18.6	0.0	8.6	2.9	11.4	2.9	1.4	8.6	0.0	4.3	0.0	1.4	0.0	0.0	1.4	1.4	0.0	0.0	0.0	0.0	0.0			
<b>Owen VL Max</b>	47.1	31.4	8.6	22.9	20.0	22.9	11.4	1.4	0.0	0.0	2.9	15.7	0.0	8.6	1.4	5.7	2.9	0.0	4.3	1.4	4.3	0.0	2.9	0.0	0.0	2.9	0.0	0.0	0.0	1.4	1.4	0.0			
<b>GLM 4.5V</b>	70.0	20.0	14.3	12.9	4.3	22.9	4.3	7.1	0.0	8.6	2.9	8.6	0.0	12.9	4.3	1.4	2.9	1.4	4.3	0.0	4.3	1.4	2.9	4.3	1.4	2.9	4.3	1.4	2.9	0.0	0.0	0.0	0.0		
<b>Claude Sonnet 4</b>	47.1	14.3	1.4	15.7	5.7	20.0	17.1	1.4	0.0	2.9	0.0	22.9	0.0	14.3	7.1	4.3	2.9	7.1	7.1	0.0	4.3	10.0	2.9	0.0	0.0	1.4	1.4	0.0	0.0	0.0	0.0	0.0	0.0		
<b>Owen 2.5 VL 72B Instruct</b>	31.4	31.4	14.3	17.1	5.7	17.1	18.6	1.4	0.0	7.1	1.4	24.3	0.0	4.3	0.0	2.9	2.9	10.0	1.4	0.0	4.3	1.4	2.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
<b>Llama 4 Maverick</b>	44.3	37.1	10.0	20.0	10.0	12.9	8.6	1.4	0.0	2.9	1.4	8.6	0.0	0.0	0.0	5.7	2.9	12.9	5.7	1.4	4.3	0.0	0.0	0.0	0.0	0.0	0.0	1.4	0.0	0.0	0.0	0.0	0.0	0.0	
<b>Gemma 3 27B Instruct</b>	31.4	31.4	14.3	17.1	8.6	10.0	7.1	0.0	0.0	1.4	0.0	2.9	0.0	4.3	1.4	2.9	2.9	4.3	7.1	0.0	4.3	17.1	8.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
<b>UI TARS 1.5 7B</b>	10.0	24.3	5.7	8.6	8.6	14.3	5.7	0.0	0.0	1.4	0.0	1.4	0.0	0.0	0.0	4.3	1.4	1.4	4.3	0.0	1.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.4	

**Figure 4: Task success rate of frontier and finetuned models.** Proprietary models are shown in **bold**, and our finetuned models in *italics*. (E) and (H) denote easy and hard task settings. Darker cells indicate higher success rates. Models are ordered by average task performance (top = better), and tasks by average model performance, excluding our finetuned ones (right = harder).

**Model Specialization.** We compare the 3 strongest models: Gemini 2.5 Pro, GPT-5, Qwen3-VL-235B Instruct. GPT-5 shows the best ability to handle long-context visual interactions. This is reflected in its stronger performance on matchstick rotation where the scale is unknown, its higher scores overall on the hard setting (Fig. 2), and its visibly longer tail in the number of steps taken to successfully solve tasks compared to the other models (Fig. 3). Gemini 2.5 Pro is good at low-level visual perception. This is reflected in its strongest performance on Jigsaw, Maze 2D, Zoom-In Puzzle, and Sliding Block, all of which demand tight spatial alignment, accurate correspondence of local patterns, and sensitivity to subtle visual cues. Qwen-3-VL is in particular capable of object localization (e.g., strongest in Referring Dot-Pointing).

Examining the step count distribution (smoothed density curve) for successful trajectories across models (Fig. 3), we found that most models (i.e., Gemini 2.5 Pro, Claude Sonnet 4, and Llama-4-Maverick) only peaked around 3-5 steps, followed by a sharp drop in successful trajectories when they spend more steps. This indicates limited capability in effectively handling long-context multi-step visual interactions.

**Common Failure Patterns.** We identify recurring failures using automated failure discovery methods (Dunlap et al., 2025; Lisa Dunlap et al., 2025), which employ a VLM annotator (GPT-4.1) to extract negative behaviors from each trajectory and cluster them into categories. This analysis reveals 4 failure types that appear consistently across multiple tasks (see Sec. §G for details):

(1) **Restricted action space and action looping:** models often rely on a single repeated operation or fixed-magnitude action, such as continually moving in the same direction in Fetch Pick & Place, using “swap” in Jigsaw instead of “reorder”, or rotating by the same angle in Mental Rotation 3D and Match Rotation rather than converging to an optimal magnitude.

(2) *State mismanagement*: models fail to maintain or update internal state across steps. They ignore textual or environmental feedback, revisit previously explored areas, or repeat illegal actions despite prior errors—for example, continuing to move into a wall after being told they have collided, or repeating invalid moves in the Match Equation, Sliding Block, and Toy Maze 2D tasks.

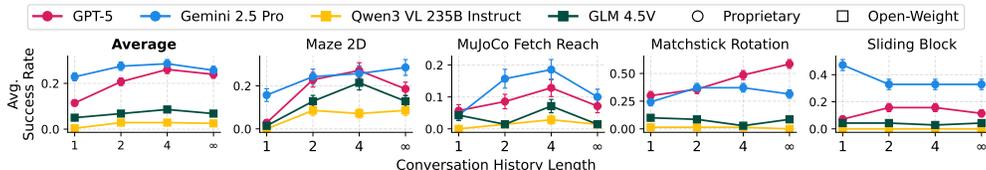
(3) *Early termination*: the model terminates before the max steps despite not reaching the goal.

(4) *Failure to use visual or spatial information*: models ignore the visual information provided, such as the target leaving the frame or the item being successfully aligned (e.g., Mental Rotation).

#### 4 DIAGNOSING FRONTIER MODELS WITH VISGYM

We perform diagnoses with the two best performing proprietary models, i.e., GPT-5 (OpenAI, 2025) and Gemini 2.5 Pro (DeepMind, 2025), and the two best performing open-weight models, i.e., Qwen3-VL-235B Instruct (Yang et al., 2025) and GLM-4.5V (Hong et al., 2025).

**Turns to Keep in Conversation History** Vision-language models are known to degrade with long visual context (Wang et al., 2025; Wu et al., 2024; Sharma et al., 2024). This creates a dilemma: while long histories provide more information about the environment (e.g., 3D layouts, unknown dynamics), they also introduce redundant observations that may harm performance. We study this trade-off in Maze2D, Sliding Block, MuJoCo Fetch Reach, and Matchstick Rotation, where history provides useful signals such as textual feedback (e.g., invalid actions) or correspondence between action magnitude and perceptual effect, but also introduces stale information.



**Figure 5: Effect of truncating conversational context on model performance.** The settings 1, 2, 4, and  $\infty$  correspond to retaining only the current turn, the current + previous turn, the current + previous 3 turns, and the full history, respectively. Error bars show the standard error of the mean.

As shown in Fig. 5, models benefit from including a limited number of previous turns up to roughly four, following a drop when given the full unbounded history. This indicates that expanding visual context helps multi-step visual decision-making only to a point, after which irrelevant or stale observations become detrimental. We also observe task-specific idiosyncrasies: Gemini 2.5 Pro scales well in Maze2D, GPT-5 scales well on Matchstick Rotation, while Sliding Block exhibits clear *reverse scaling* for Gemini 2.5 Pro.

**Representing Observation in Text** Inspired by prior work examining how different task representations affect agent performance (Hu et al., 2025; Shi et al., 2025; Ruoss et al., 2024), we select four symbolic tasks—Matchstick Equation, Maze 2D, Patch Reassembly, and Sliding Block—and implement alternate versions rendered entirely in ASCII (sample ASCII visualizations are provided in Sec. §E). This allows tasks to be solved without any visual encoding module.

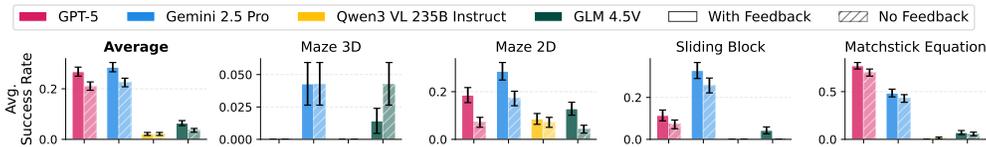


**Figure 6: Effect of visualizing observations with ASCII (text).** “Image” and “Text” denote the observation modalities. Error bars show the standard error of the mean.

The results in Fig. 6 show that GPT-5 substantially improves in most tasks, often achieving 3–4× higher success rates than in the visual setting, suggesting that its main bottleneck lies in visual grounding rather than long-horizon reasoning. Gemini 2.5 Pro shows mixed behavior: two tasks do not exhibit significant performance change, one task improves, and one task degrades, indicating possible limitations in both perception and planning. Open-weight models struggle across all tasks in both modalities, indicating general weaknesses in long-horizon decision-making regardless of representation. Interestingly, Matchstick Equation exhibits a *reverse* trend: all models perform substantially better with the visual representation than with ASCII, likely because the figlet-style

ASCII has irregular shapes and spacing that create distorted glyphs which models are known to struggle with (Stojanovski et al., 2025).

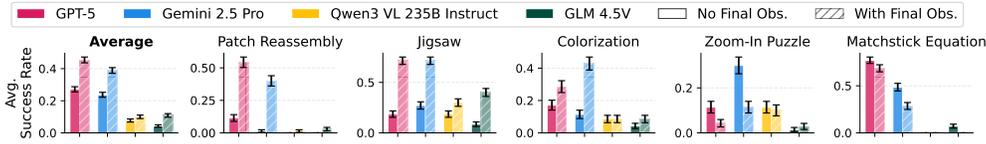
**Removal of Text-based Feedback** Humans can infer action consequences directly from visual changes (Michotte, 1963), but it remains unclear whether VLMs can do the same. To study this, we select four tasks—Maze 3D, Maze 2D, Sliding Block, and Matchstick Equation—in which the environment feedback  $f$  (see Equation (1)) provides not only formatting errors but also constraint violations (e.g., hitting a wall in Maze, sliding a block into an occupied cell). We remove this textual feedback and evaluate model using only visual state transitions; results are shown in Fig. 7.



**Figure 7: Effect of removing text-based environment feedback.** “With Feedback” includes environment feedback describing action execution at each turn; “No Feedback” removes this channel. Error bars show the standard error of the mean.

All models show consistent drops in average performance. This indicates that models struggle to infer action validity directly from visual transitions. These findings show that current VLMs depend heavily on text-based feedback during visually interactive decision-making and are less sensitive to pure visual feedback.

**Providing Final Goal at Beginning** Providing the solution image upfront simplifies the tasks to visually aligning current observations with a known target, shifting the difficulty from reasoning to visual perception and tool-calling. We test this on five tasks, Patch Reassembly, Jigsaw, Colorization, Zoom-In Puzzle, and Matchstick Equation, where constructing the goal observation involves significant effort. For these tasks, we augment the instruction with the ground-truth final observation  $o_{gt}$ , and show results in Fig. 8.



**Figure 8: Effect of providing the final goal observation at the beginning of the episode.** “No Final Obs.” and “With Final Obs.” denote settings without and with access to the goal observation at the start. Error bars show the standard error of the mean.

Across tasks, models improve substantially, indicating that a major bottleneck lies in *constructing or imagining the target state*. However, performance remains far from perfect, indicating additional limitations beyond reasoning, such as fine-grained visual perception and action calling. Surprisingly, GPT-5 and Gemini 2.5 Pro *underperform* on the Zoom-In Puzzle and Matchstick Equation when the final goal observation is provided, often terminating early despite visible misalignment. A follow-up test confirms this stems from visual misjudgment due to limited visual perception: we queried Gemini 2.5 Pro on 100 pairs of initial and final-goal observations with the prompt “Do the two images look exactly the same?” and it incorrectly judged images as identical 80% and 57% of the time for these tasks, versus only 18%, 2%, and 0% for Colorization, Jigsaw, and Patch Reassembly. This confirms that perception errors can *invert* the expected benefit of an explicit goal observation.

## 5 TRAINING WITH VISGYM

### 5.1 SUPERVISED FINE-TUNING EXPERIMENTS

We generate demonstration trajectories for supervised fine-tuning using the multi-step solver described in Sec. §2. We apply two preprocessing filters: (1) discarding trajectories that fail to complete the task, and (2) removing trajectories with initial states overlapping the test split to prevent data leakage. We evaluate two fine-tuning configurations: *single-task* and *mixed-task*. In the *single-task* setting, we fine-tune a separate model for each task, whereas in the *mixed-task* setting, a single model is trained jointly on all tasks. Notably, demonstrations are sourced exclusively from the easy difficulty level; thus, performance on the hard setting serves as a metric for difficulty generalization. All experiments employ Qwen2.5-VL-7B-Instruct (Bai et al., 2025) with full-parameter fine-tuning, a global batch

size of 64, a learning rate of  $1 \times 10^{-5}$ , and `bf16` precision. Models are trained for 1,500 steps in the single-task setting and 5,000 steps in the mixed-task setting. We utilize LlamaFactory (Zheng et al., 2024) for all data preprocessing and training orchestration.

As shown in Figs. 2 and 4, finetuned models achieve state-of-the-art performance on most tasks, validating both the learnability of our environments and the effectiveness of our multi-step solvers. These gains confirm that current VLMs can substantially benefit from structured, solver-generated demonstrations in visually grounded multi-step settings.

### 5.2 STRONGER BASE MODEL GENERALIZES BETTER

Existing work has discussed the limitations of supervised finetuning (Ross et al., 2011) and found that it exhibits limited generalization to task variants (Caccia et al., 2024; Deng et al., 2023; Jang et al., 2022). This motivates re-examining generalization in the context of modern VLMs, whose capabilities may shift the boundary of what supervised finetuning can or cannot retain. To this end, we select a set of environments where the easy-to-hard difficulty gap introduces substantial state changes (e.g., more views in the Zoom-In Puzzle, more patches in Patch Reassembly, larger maze sizes; details in Sec. §F). We finetune Qwen2.5-VL-7B-Instruct and Qwen3-VL-8B-Instruct on the same mixed-task training data using identical optimization hyperparameters (see Sec. §5.1), and report performance in Fig. 9.

As shown, both models achieve comparable performance on the easy variants they were trained on (e.g., 0.59 vs. 0.64), but the more recent Qwen3-VL generalizes substantially better to the harder variants, nearly doubling the success rate on average relative to Qwen2.5-VL. This trend highlights that newer VLMs provide stronger out-of-distribution generalization in multi-step visual decision-making despite being finetuned on an identical setup.

### 5.3 VISION AND LLM BOTH MATTER

Classic perception–action theories emphasize that fine-grained visual encoding and temporal integration are jointly necessary for interactive behavior (Gibson, 1979). We examine whether this holds for VLMs by fine-tuning variants that modify either the vision encoder or the LLM backbone to isolate each module’s contribution, where the vision encoder provides fine-grained perceptual features and the LLM performs temporal integration across steps.

As shown in Fig. 10, most tasks benefit from fine-tuning both components, with the LLM contributing the larger performance gain—particularly in tasks with partial observability or unknown environment dynamics. This highlights that temporal reasoning and history integration remain the primary bottlenecks for current VLMs, while strong fine-grained visual encoding is necessary (e.g., Zoom-In Puzzle primarily benefits from vision finetuning) but often not sufficient for multi-step decision-making.

### 5.4 IMPORTANCE OF INFORMATION-REVEALING BEHAVIORS FOR SFT CURATION

Not all experiences contribute equally to decision-making: trajectories that reveal hidden states or disambiguate perceptual aliasing are often far more valuable (McCallum, 1994; Fujii et al., 1998). We ask whether inducing such information-revealing behaviors during supervised finetuning helps

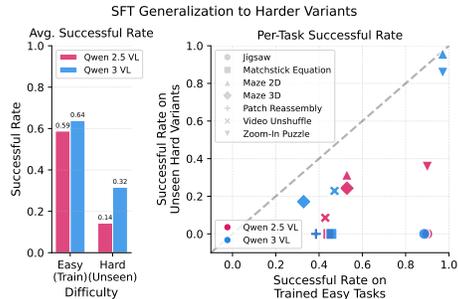


Figure 9: Generalization to unseen difficulty from mixed-task supervised finetuning. (Left)

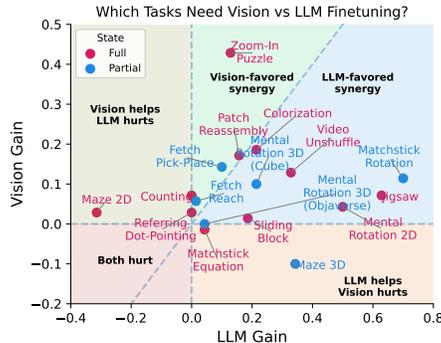
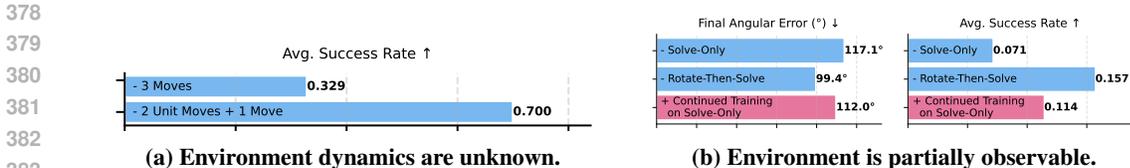


Figure 10: Tasks benefiting from finetuning different modules. “Vision Gain” and “LLM Gain” denote improvements from jointly finetuning both components, compared to finetuning only the LLM or the vision part. The dashed line ( $y = x$ ) divides vision-favored (above) and LLM-favored (below) synergy. “Full” and “Partial” denote whether observability and dynamics are fully known.



**Figure 11: Effect of data curation strategies on task performance.** (a) Numbers represent average task success (higher is better); “3 Moves” and “2 Unit Moves + 1 Move” are two curation strategies. (b) “Solve-Only” and “Rotate-Then-Solve” are two curation strategies, and “Continued Training on Solve-Only” denotes further finetuning on Solve-Only after training on Rotate-Then-Solve. (Left) Final angular error on the test set (lower is better). (Right) Average task success rate (higher is better).

VLMs form more accurate state representations. We evaluate this on two tasks, Matchstick Rotation (unknown dynamics) and Mental Rotation 3D Objaverse (partial observability), with results in Figs. 11a and 11b.

In Matchstick Rotation, the baseline demonstrations perform three stochastic moves toward the target. In contrast, the information-revealing demonstrations first perform two unit-scale steps to expose the correspondence between action magnitude and perceptual effect before executing the final aligning move. This structured exploration raises success from 32.9% to 70.0%.

In Mental Rotation, the baseline trajectories rotate along each principal axis once to reach the goal, while the information-revealing ones deliberately fully rotate along each axis to expose the full 3D geometry before settling on the target orientation. This strategy improves performance in both metrics. To verify that gains are not simply due to longer trajectories, we further continue training on baseline demonstrations starting from the model already finetuned with information-revealing data. Performance deteriorates in this setting, confirming that the observed improvements stem from the *informative structure* of the demonstrations rather than quantity or length. These results highlight that SFT effectiveness depends on whether demonstrations induce state-disambiguating behaviors, not merely on the number of examples.

### 5.5 GENERALIZATION ACROSS TASKS AND OTHER DOMAINS

While Sec. §5.2 shows that stronger models generalize across difficulty levels within the same task, we now evaluate whether the visual interaction priors learned from VisGym-generated episodes transfer to unseen tasks and to new domains that rely on such capabilities, including vision-language-action (VLA) models (Kim et al., 2024) for robotics.

We first study cross-task transfer via mixed-task fine-tuning (e.g., Sec. §5.1) on 16 environments, leaving one environment held out for zero-shot evaluation (e.g., Referring Dot-Pointing or Video Unshuffle). Results are reported in Tab. 2. We observe substantial performance gains on both held-out tasks compared to the base model (e.g., Qwen2.5-VL-7B-Instruct), indicating that visual interaction capabilities learned in VisGym can feasibly generalize across tasks.

**Table 2:** Cross-task zero-shot transfer.

	RefDot	Unshuffle
Base.	20.0	4.3
<b>Ours</b>	<b>28.6</b>	<b>15.7</b>

Next, we examine whether these learned capabilities benefit domain-specific adaptation. We first perform mixed-task fine-tuning on all 17 VisGym environments for 500 steps. Starting from this VISGYM-trained VLM, we then follow recent VLM-to-VLA finetuning protocols (Zhang et al., 2026) and train a VLA model on the CALVIN (ABC-D) dataset (Mees et al., 2022) for 16k steps. We compare against an identical VLA training procedure initialized from the base VLM (e.g., Qwen2.5-VL-7B-Instruct). As shown in Tab. 3, VLA models initialized from VISGYM-trained VLM consistently outperform the baseline across all metrics, demonstrating that visual interaction priors learned via VISGYM transfer effectively to downstream robotic control tasks.

**Table 3:** CALVIN performance (Mees et al., 2022) by number of instructions chained.

	1	2	3	4	5	Total
Base.	0.817	0.639	0.501	0.394	0.312	2.663
<b>Ours</b>	<b>0.854</b>	<b>0.679</b>	<b>0.542</b>	<b>0.411</b>	<b>0.331</b>	<b>2.817</b>

Together, these results suggest that VisGym provides a principled and scalable framework for instilling transferable visual interaction capabilities, enabling models to generalize beyond individual tasks and domains, and serving as a strong foundation for training interactive multimodal agents.

## REFERENCES

- 432  
433
- 434 [1] Marwa Abdulhai, Isadora White, Charlie Snell, Charles Sun, Joey Hong, Yuexiang Zhai,  
435 Kelvin Xu, and Sergey Levine. Lmrl gym: Benchmarks for multi-turn reinforcement learning  
436 with language models. *arXiv preprint arXiv:2311.18232*, 2023.
- 437 [2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni  
438 Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4  
439 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- 440 [3] Ossama Ahmed, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Yoshua Bengio, Bernhard  
441 Schölkopf, Manuel Wüthrich, and Stefan Bauer. Causalworld: A robotic manipulation  
442 benchmark for causal structure and transfer learning. *arXiv preprint arXiv:2010.04296*, 2020.
- 443 [4] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson,  
444 Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual  
445 language model for few-shot learning. *Advances in neural information processing systems*, 35:  
446 23716–23736, 2022.
- 447 [5] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao Song, Kai Dang,  
448 Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint  
449 arXiv:2502.13923*, 2025.
- 450 [6] John C. Baird. *Psychophysical Analysis of Visual Space*. International Series of Mono-  
451 graphs in Experimental Psychology. Pergamon Press / Elsevier, 1970. ISBN 978-0-08-  
452 013876-3. URL [https://www.sciencedirect.com/book/9780080138763/  
453 psychophysical-analysis-of-visual-space](https://www.sciencedirect.com/book/9780080138763/psychophysical-analysis-of-visual-space).
- 454 [7] Mohamad Ballout, Okajevo Wilfred, Seyedalireza Yaghoubi, Nohayr Muhammad Abdel-  
455 moneim, Julius Mayer, and Elia Bruni. Can you splice it together? a human curated benchmark  
456 for probing visual reasoning in vlms. *arXiv preprint arXiv:2509.24640*, 2025.
- 457 [8] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich  
458 Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab.  
459 *arXiv preprint arXiv:1612.03801*, 2016.
- 460 [9] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment:  
461 An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:  
462 253–279, jun 2013.
- 463 [10] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan,  
464 Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation  
465 model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.
- 466 [11] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo  
467 Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al.  $\pi$ 0: A vision-language-action flow  
468 model for general robot control. corr, abs/2410.24164, 2024. doi: 10.48550. *arXiv preprint  
469 ARXIV.2410.24164*, 2024.
- 470 [12] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang,  
471 and Wojciech Zaremba. Openai gym, 2016.
- 472 [13] Massimo Caccia, Megh Thakkar, Léo Boisvert, Thibault Le Sellier de Chezelles, Alexan-  
473 dre Piché, Nicolas Chapados, Alexandre Drouin, Maxime Gasse, and Alexandre Lacoste.  
474 Fine-tuning web agents: It works, but it’s trickier than you think. In *NeurIPS 2024 Work-  
475 shop on Open-World Agents*, 2024. URL [https://openreview.net/forum?id=  
476 SkwtxEkst2](https://openreview.net/forum?id=SkwtxEkst2).
- 477 [14] Guanqun Cao, Ryan Mckenna, Erich Graf, and John Oyekan. Learn from the past:  
478 Language-conditioned object rearrangement with large language models. *arXiv preprint  
479 arXiv:2501.18516*, 2025.
- 480  
481  
482  
483  
484  
485

- 486 [15] Liang Chen, Hongcheng Gao, Tianyu Liu, Zhiqi Huang, Flood Sung, Xinyu Zhou, Yuxin Wu,  
487 and Baobao Chang. G1: Bootstrapping perception and reasoning abilities of vision-language  
488 model via reinforcement learning. *arXiv preprint arXiv:2505.13426*, 2025.  
489
- 490 [16] Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan,  
491 Jiaqi Wang, Yu Qiao, Dahua Lin, et al. Are we on the right way for evaluating large vision-  
492 language models? *Advances in Neural Information Processing Systems*, 37:27056–27087,  
493 2024.
- 494 [17] Jae-Woo Choi, Youngwoo Yoon, Hyobin Ong, Jaehong Kim, and Minsu Jang. Lota-  
495 bench: Benchmarking language-oriented task planners for embodied agents. *arXiv preprint*  
496 *arXiv:2402.08178*, 2024.
- 497 [18] João Coelho, Jingjie Ning, Jingyuan He, Kangrui Mao, Abhijay Paladugu, Pranav Setlur, Jiahe  
498 Jin, Jamie Callan, João Magalhães, Bruno Martins, et al. Deepresearchgym: A free, transparent,  
499 and reproducible evaluation sandbox for deep research. *arXiv preprint arXiv:2505.19253*,  
500 2025.
- 501 [19] Lynn A. Cooper and Roger N. Shepard. Chronometric studies of the rotation of mental images.  
502 In WILLIAM G. CHASE, editor, *Visual Information Processing*, pages 75–176. Academic  
503 Press, 1973. ISBN 978-0-12-170150-5. doi: [https://doi.org/10.1016/B978-0-12-170150-5.](https://doi.org/10.1016/B978-0-12-170150-5.50009-3)  
504 50009-3. URL [https://www.sciencedirect.com/science/article/pii/](https://www.sciencedirect.com/science/article/pii/B9780121701505500093)  
505 [B9780121701505500093](https://www.sciencedirect.com/science/article/pii/B9780121701505500093).  
506
- 507 [20] Google DeepMind. Gemini 2.5 Pro Technical Report. Technical report, Google DeepMind,  
508 2025. URL [https://modelcards.withgoogle.com/assets/documents/](https://modelcards.withgoogle.com/assets/documents/gemini-2.5-pro.pdf)  
509 [gemini-2.5-pro.pdf](https://modelcards.withgoogle.com/assets/documents/gemini-2.5-pro.pdf). Accessed: 2025-11-06.
- 510 [21] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt,  
511 Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe  
512 of annotated 3d objects. In *Proceedings of the IEEE/CVF conference on computer vision and*  
513 *pattern recognition*, pages 13142–13153, 2023.
- 514 [22] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and  
515 Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information*  
516 *Processing Systems*, 36:28091–28114, 2023.
- 517 [23] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle,  
518 Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd  
519 of models. *arXiv e-prints*, pages arXiv–2407, 2024.
- 520 [24] Lisa Dunlap, Krishna Mandal, Trevor Darrell, Jacob Steinhardt, and Joseph E Gonzale-  
521 z. Vibecheck: Discover and quantify qualitative differences in large language models.  
522 In *International Conference on Learning Representations (ICLR)*, 2025. URL [https://](https://lisadunlap.github.io/VibeCheck/)  
523 [lisadunlap.github.io/VibeCheck/](https://lisadunlap.github.io/VibeCheck/).  
524
- 525 [25] Kiana Ehsani, Winson Han, Alvaro Herrasti, Eli VanderBilt, Luca Weihs, Eric Kolve, Anirud-  
526 dha Kembhavi, and Roozbeh Mottaghi. Manipulathor: A framework for visual object man-  
527 ipulation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern*  
528 *recognition*, pages 4497–4506, 2021.
- 529 [26] Atsushi Fujii, Kentaro Inui, Takenobu Tokunaga, and Hozumi Tanaka. Selective sampling for  
530 example-based word sense disambiguation. *Computational Linguistics*, 24(4):573–597, 1998.  
531 URL <https://aclanthology.org/J98-4002/>.  
532
- 533 [27] James J. Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin, Boston,  
534 MA, 1979. ISBN 0-395-27049-9.
- 535 [28] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning  
536 by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.
- 537 [29] Solomon W. Golomb. *Polyominoes: Puzzles, Patterns, Problems, and Packings*. Princeton  
538 University Press, Princeton, NJ, 2nd edition, 1994. ISBN 978-0-691-02444-8.  
539

- 540 [30] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne  
541 Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag,  
542 et al. The "something something" video database for learning and evaluating visual common  
543 sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–  
544 5850, 2017.
- 545 [31] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance  
546 segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern  
547 recognition*, pages 5356–5364, 2019.
- 548 [32] John M. Henderson. A sensorimotor account of vision and visual consciousness. *The  
549 Behavioral and brain sciences*, 24 5:939–73; discussion 973–1031, 2001. URL <https://api.semanticscholar.org/CorpusID:22606536>.
- 550 [33] Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale  
551 Cheng, Ji Qi, Junhui Ji, Lihang Pan, et al. Glm-4.1 v-thinking: Towards versatile multimodal  
552 reasoning with scalable reinforcement learning. *arXiv e-prints*, pages arXiv–2507, 2025.
- 553 [34] Lanxiang Hu, Mingjia Huo, Yuxuan Zhang, Haoyang Yu, Eric P Xing, Ion Stoica, Tajana  
554 Rosing, Haojian Jin, and Hao Zhang. Igame-bench: How good are llms at playing games?  
555 *arXiv preprint arXiv:2505.15146*, 2025.
- 556 [35] Huang Huang, Fangchen Liu, Letian Fu, Tingfan Wu, Mustafa Mukadam, Jitendra Malik, Ken  
557 Goldberg, and Pieter Abbeel. Otter: A vision-language-action model with text-aware feature  
558 extraciton. *arXiv preprint arXiv:2503.03734*, 2025.
- 559 [36] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot  
560 learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):  
561 3019–3026, 2020.
- 562 [37] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey  
563 Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning.  
564 In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.
- 565 [38] Lawrence Jang, Yinheng Li, Dan Zhao, Charles Ding, Justin Lin, Paul Pu Liang, Rogerio  
566 Bonatti, and Kazuhito Koishida. Videowebarena: Evaluating long context multimodal agents  
567 with video understanding web tasks. *arXiv preprint arXiv:2410.19100*, 2024.
- 568 [39] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and  
569 Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues?  
570 *arXiv preprint arXiv:2310.06770*, 2023.
- 571 [40] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. ReferItGame: Referring  
572 to objects in photographs of natural scenes. In Alessandro Moschitti, Bo Pang, and Walter  
573 Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural  
574 Language Processing (EMNLP)*, pages 787–798, Doha, Qatar, October 2014. Association for  
575 Computational Linguistics. doi: 10.3115/v1/D14-1086. URL <https://aclanthology.org/D14-1086/>.
- 576 [41] Mukul Khanna, Ram Ramrakhya, Gunjan Chhablani, Sriram Yenamandra, Theophile Gervet,  
577 Matthew Chang, Zsolt Kira, Devendra Singh Chaplot, Dhruv Batra, and Roozbeh Mot-  
578 taghi. Goat-bench: A benchmark for multi-modal lifelong navigation. In *Proceedings of the  
579 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16373–16383,  
580 2024.
- 581 [42] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair,  
582 Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source  
583 vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.
- 584 [43] Günther Knoblich, Stellan Ohlsson, Hilde Haider, and Detlef Rhenius. Constraint relaxation  
585 and chunk decomposition in insight problem solving. *Journal of Experimental Psychology:  
586 Learning Memory and Cognition*, 25(6):1534–1555, November 1999. ISSN 0278-7393. doi:  
587 10.1037/0278-7393.25.6.1534.

- 594 [44] Anurag Koul. maze-world: Random maze environments with different size and complex-  
595 ity for reinforcement learning and planning research. [https://koulanurag.dev/  
596 maze-world/index.html](https://koulanurag.dev/maze-world/index.html), 2024. Accessed: 2025-11-07.
- 597 [45] Russell Kruger, Sheelagh Carpendale, Stacey D. Scott, and Anthony Tang. Fluid integration  
598 of rotation and translation. In *Proceedings of the SIGCHI Conference on Human Factors  
599 in Computing Systems*, CHI '05, page 601–610, New York, NY, USA, 2005. Association  
600 for Computing Machinery. ISBN 1581139985. doi: 10.1145/1054972.1055055. URL  
601 <https://doi.org/10.1145/1054972.1055055>.
- 602 [46] Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying  
603 Shan. Seed-bench: Benchmarking multimodal large language models. In *Proceedings of the  
604 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13299–13308,  
605 2024.
- 607 [47] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto  
608 Martín-Martín, Chen Wang, Gabriel Levine, Wensi Ai, Benjamin Martinez, et al. Behavior-  
609 1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic  
610 simulation. *arXiv preprint arXiv:2403.09227*, 2024.
- 611 [48] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-  
612 image pre-training for unified vision-language understanding and generation. In *International  
613 conference on machine learning*, pages 12888–12900. PMLR, 2022.
- 614 [49] Lisa Dunlap, Trevor Darrell, Jacob Steinhardt, and Joseph Gonzalez. Stringsight: Automating  
615 analysis of model outputs. <https://www.stringsight.com/>, 2025.
- 617 [50] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone.  
618 Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural  
619 Information Processing Systems*, 36:44776–44791, 2023.
- 620 [51] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In  
621 *NeurIPS*, 2023.
- 623 [52] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning.  
624 *Advances in neural information processing systems*, 36:34892–34916, 2023.
- 625 [53] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang  
626 Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *The Twelfth  
627 International Conference on Learning Representations*, 2023.
- 628 [54] Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Yifan Xu, Xixuan Song, Shudan Zhang, Hanyu  
629 Lai, Xinyi Liu, Hanlin Zhao, et al. Visualagentbench: Towards large multimodal models as  
630 visual foundation agents. *arXiv preprint arXiv:2408.06327*, 2024.
- 631 [55] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike  
632 Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an  
633 all-around player? In *European conference on computer vision*, pages 216–233. Springer,  
634 2024.
- 635 [56] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao  
636 Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical  
637 reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*, 2023.
- 638 [57] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni,  
639 Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning  
640 from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*,  
641 2021.
- 642 [58] R. Andrew McCallum. Instance-based state identification for reinforcement  
643 learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in  
644 Neural Information Processing Systems*, volume 7. MIT Press, 1994. URL  
645 [https://proceedings.neurips.cc/paper\\_files/paper/1994/file/  
646 d2ed45a52bc0edfa11c2064e9edee8bf-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/1994/file/d2ed45a52bc0edfa11c2064e9edee8bf-Paper.pdf).

- 648 [59] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark  
649 for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE*  
650 *Robotics and Automation Letters*, 7(3):7327–7334, 2022.
- 651 [60] Albert Michotte. *The Perception of Causality*. Routledge, 1 edition, 1963. doi: 10.4324/  
652 9781315519050. URL <https://doi.org/10.4324/9781315519050>.
- 653 [61] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning  
654 using temporal order verification. In *European conference on computer vision*, pages 527–544.  
655 Springer, 2016.
- 656 [62] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan  
657 Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint*  
658 *arXiv:1312.5602*, 2013.
- 659 [63] Dantong Niu, Yuvan Sharma, Giscard Biamby, Jerome Quenum, Yutong Bai, Baifeng Shi,  
660 Trevor Darrell, and Roei Herzig. Llarva: Vision-action instruction tuning enhances robot  
661 learning. *arXiv preprint arXiv:2406.11815*, 2024.
- 662 [64] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep  
663 Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yun-  
664 liang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey  
665 Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science*  
666 *and Systems*, Delft, Netherlands, 2024.
- 667 [65] OpenAI. GPT-5 System Card. System card, OpenAI, August 2025. URL [https://cdn.](https://cdn.openai.com/gpt-5-system-card.pdf)  
668 [openai.com/gpt-5-system-card.pdf](https://cdn.openai.com/gpt-5-system-card.pdf). Accessed: 2025-11-06.
- 669 [66] Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang,  
670 Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with  
671 native agents. *arXiv preprint arXiv:2501.12326*, 2025.
- 672 [67] Santhosh Kumar Ramakrishnan, Erik Wijmans, Philipp Kraehenbuehl, and Vladlen Koltun.  
673 Does spatial cognition emerge in frontier models? *arXiv preprint arXiv:2410.06468*, 2024.
- 674 [68] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and  
675 structured prediction to no-regret online learning. In *Proceedings of the fourteenth international*  
676 *conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and  
677 Conference Proceedings, 2011.
- 678 [69] Anian Ruoss, Fabio Pardo, Harris Chan, Bonnie Li, Volodymyr Mnih, and Tim Genewein.  
679 Lmact: A benchmark for in-context imitation learning with long multimodal demonstrations.  
680 *arXiv preprint arXiv:2412.01441*, 2024.
- 681 [70] Aditya Sharma, Michael Saxon, and William Yang Wang. Losing visual needles in image  
682 haystacks: Vision language models are easily distracted in short and long contexts. In  
683 Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for*  
684 *Computational Linguistics: EMNLP 2024*, pages 5429–5451, Miami, Florida, USA, November  
685 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.312.  
686 URL <https://aclanthology.org/2024.findings-emnlp.312/>.
- 687 [71] R. N. Shepard and J. Metzler. Mental rotation of three-dimensional objects. *Science*, 171  
688 (3972):701–703, 1971. doi: 10.1126/science.171.3972.701.
- 689 [72] Jiajun Shi, Jian Yang, Jiaheng Liu, Xingyuan Bu, Jiangjie Chen, Junting Zhou, Kaijing Ma,  
690 Zhoufutu Wen, Bingli Wang, Yancheng He, et al. Korgym: A dynamic game platform for llm  
691 reasoning evaluation. *arXiv preprint arXiv:2505.14552*, 2025.
- 692 [73] Modi Shi, Li Chen, Jin Chen, Yuxiang Lu, Chiming Liu, Guanghui Ren, Ping Luo, Di Huang,  
693 Maoqing Yao, and Hongyang Li. Is diversity all you need for scalable robotic manipulation?  
694 *arXiv preprint arXiv:2507.06219*, 2025.
- 695  
696  
697  
698  
699  
700  
701

- 702 [74] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh  
703 Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded  
704 instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer  
705 vision and pattern recognition*, pages 10740–10749, 2020.
- 706 [75] Ved Sirdeshmukh, Kaustubh Deshpande, Johannes Mols, Lifeng Jin, Ed-Yeremai Cardona,  
707 Dean Lee, Jeremy Kritz, Willow Primack, Summer Yue, and Chen Xing. Multichallenge: A  
708 realistic multi-turn conversation evaluation benchmark challenging to frontier llms. *arXiv  
709 preprint arXiv:2501.17399*, 2025.
- 710 [76] Ruben Spaans. Solving sliding-block puzzles. Specialisation project report, Norwegian  
711 University of Science and Technology (NTNU), 2009. URL [https://www.pvv.org/  
712 ~spaans/spec-cs.pdf](https://www.pvv.org/~spaans/spec-cs.pdf).
- 713 [77] Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia,  
714 Kent Elliott Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, Karen Liu, et al. Behavior:  
715 Benchmark for everyday household activities in virtual, interactive, and ecological environ-  
716 ments. In *Conference on robot learning*, pages 477–490. PMLR, 2022.
- 717 [78] Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean  
718 Kaddour, and Andreas Köpf. Reasoning gym: Reasoning environments for reinforcement  
719 learning with verifiable rewards. *arXiv preprint arXiv:2505.24760*, 2025.
- 720 [79] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner,  
721 Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al.  
722 Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in neural information  
723 processing systems*, 34:251–266, 2021.
- 724 [80] Claude Team. Claude 4 sonnet, 2025. URL [https://www.anthropic.com/claude/  
725 sonnet](https://www.anthropic.com/claude/sonnet).
- 726 [81] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu  
727 Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family  
728 of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- 729 [82] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montser-  
730 rat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza,  
731 Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint  
732 arXiv:2503.20020*, 2025.
- 733 [83] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona  
734 Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma  
735 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.
- 736 [84] Kimi Team, Angang Du, Bohong Yin, Bowei Xing, Bowen Qu, Bowen Wang, Cheng Chen,  
737 Chenlin Zhang, Chenzhuang Du, Chu Wei, et al. Kimi-vl technical report. *arXiv preprint  
738 arXiv:2504.07491*, 2025.
- 739 [85] TRI LBM Team, Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau,  
740 Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, Masha  
741 Itkina, Naveen Kuppuswamy, Kuan-Hui Lee, Katherine Liu, Dale McConachie, Ian McMahon,  
742 Haruki Nishimura, Calder Phillips-Grafflin, Charles Richter, Parth Shah, Krishnan Srinivasan,  
743 Blake Wulfe, Chen Xu, Mengchao Zhang, Alex Alspach, Maya Angeles, Kushal Arora,  
744 Vitor Campagnolo Guizilini, Alejandro Castro, Dian Chen, Ting-Sheng Chu, Sam Creasey,  
745 Sean Curtis, Richard Denitto, Emma Dixon, Eric Dusel, Matthew Ferreira, Aimee Goncalves,  
746 Grant Gould, Damrong Guoy, Swati Gupta, Xuchen Han, Kyle Hatch, Brendan Hathaway,  
747 Allison Henry, Hillel Hochsztein, Phoebe Horgan, Shun Iwase, Donovan Jackson, Siddharth  
748 Karamcheti, Sedrick Keh, Joseph Masterjohn, Jean Mercat, Patrick Miller, Paul Mitiguy,  
749 Tony Nguyen, Jeremy Nimmer, Yuki Noguchi, Reko Ong, Aykut Onol, Owen Pfannenstiehl,  
750 Richard Poyner, Leticia Priebe Mendes Rocha, Gordon Richardson, Christopher Rodriguez,  
751 Derick Seale, Michael Sherman, Mariah Smith-Jones, David Tago, Pavel Tokmakov, Matthew  
752 Tran, Basile Van Hoorick, Igor Vasiljevic, Sergey Zakharov, Mark Zolotas, Rares Ambrus,

- 756 Kerri Fetzer-Borelli, Benjamin Burchfiel, Hadas Kress-Gazit, Siyuan Feng, Stacie Ford, and  
757 Russ Tedrake. A careful examination of large behavior models for multitask dexterous  
758 manipulation. *arXiv preprint arXiv:2507.05331*, 2025. URL [https://arxiv.org/abs/  
759 2507.05331](https://arxiv.org/abs/2507.05331).
- 760 [86] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based  
761 control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages  
762 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.
- 763 [87] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timo-  
764 thée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open  
765 and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- 766 [88] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu,  
767 Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A  
768 standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*,  
769 2024.
- 770 [89] Hengyi Wang, Haizhou Shi, Shiwei Tan, Weiyi Qin, Wenyuan Wang, Tunyu Zhang, Akshay  
771 Nambi, Tanuja Ganu, and Hao Wang. Multimodal needle in a haystack: Benchmarking  
772 long-context capability of multimodal large language models. In *Proceedings of the 2025  
773 Conference of the Nations of the Americas Chapter of the Association for Computational  
774 Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3221–3241,  
775 2025.
- 776 [90] Zifu Wang, Junyi Zhu, Bo Tang, Zhiyu Li, Feiyu Xiong, Jiaqian Yu, and Matthew B Blaschko.  
777 Jigsaw-r1: A study of rule-based visual reinforcement learning with jigsaw puzzles. *arXiv  
778 preprint arXiv:2505.23590*, 2025.
- 779 [91] Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang,  
780 Xindi Wu, Haotian Liu, Sadhika Malladi, et al. Charxiv: Charting gaps in realistic chart  
781 understanding in multimodal llms. *Advances in Neural Information Processing Systems*, 37:  
782 113569–113697, 2024.
- 783 [92] Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won  
784 Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet  
785 challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025.
- 786 [93] Tsung-Han Wu, Giscard Biamby, Jerome Quenum, Ritwik Gupta, Joseph E Gonzalez, Trevor  
787 Darrell, and David M Chan. Visual haystacks: A vision-centric needle-in-a-haystack bench-  
788 mark. *arXiv preprint arXiv:2407.13766*, 2024.
- 789 [94] xAI. Grok 4 Model Card. Model card, xAI, August 2025. URL [https://data.x.ai/  
790 2025-08-20-grok-4-model-card.pdf](https://data.x.ai/2025-08-20-grok-4-model-card.pdf). Accessed: 2025-11-06.
- 791 [95] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J  
792 Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal  
793 agents for open-ended tasks in real computer environments. *Advances in Neural Information  
794 Processing Systems*, 37:52040–52094, 2024.
- 795 [96] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu,  
796 Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint  
797 arXiv:2505.09388*, 2025.
- 798 [97] Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng  
799 Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, et al. Embodiedbench:  
800 Comprehensive benchmarking multi-modal large language models for vision-driven embodied  
801 agents. *arXiv preprint arXiv:2502.09560*, 2025.
- 802 [98] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn,  
803 and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta  
804 reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

- 810 [99] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens,  
811 Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multi-  
812 modal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF*  
813 *Conference on Computer Vision and Pattern Recognition*, pages 9556–9567, 2024.
- 814 [100] Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan  
815 Sun, Botao Yu, Ge Zhang, Huan Sun, et al. Mmmu-pro: A more robust multi-discipline  
816 multimodal understanding benchmark. In *Proceedings of the 63rd Annual Meeting of the*  
817 *Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15134–15186,  
818 2025.
- 819 [101] Alex L Zhang, Thomas L Griffiths, Karthik R Narasimhan, and Ofir Press. Videogamebench:  
820 Can vision-language models complete popular video games? *arXiv preprint arXiv:2505.18134*,  
821 2025.
- 822 [102] Jianke Zhang, Xiaoyu Chen, Qiuyue Wang, Mingsheng Li, Yanjiang Guo, Yucheng Hu, Jiajun  
823 Zhang, Shuai Bai, Junyang Lin, and Jianyu Chen. Vlm4vla: Revisiting vision-language-models  
824 in vision-language-action models. *arXiv preprint arXiv:2601.03309*, 2026.
- 825 [103] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European*  
826 *conference on computer vision*, pages 649–666. Springer, 2016.
- 827 [104] Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue  
828 Yin, Zuxuan Wu, Yu-Gang Jiang, et al. Vlabench: A large-scale benchmark for language-  
829 conditioned robotics manipulation with long-horizon reasoning tasks. In *Proceedings of the*  
830 *IEEE/CVF International Conference on Computer Vision*, pages 11142–11152, 2025.
- 831 [105] Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyang Luo, Zhangchi Feng, and  
832 Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In  
833 *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*  
834 *(Volume 3: System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational  
835 Linguistics. URL <http://arxiv.org/abs/2403.13372>.
- 836 [106] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng,  
837 Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for  
838 building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.
- 839 [107] Zhongyi Zhou, Yichen Zhu, Junjie Wen, Chaomin Shen, and Yi Xu. Vision-language-action  
840 model with open-world embodied reasoning from pretrained knowledge. *arXiv preprint*  
841 *arXiv:2505.21906*, 2025.
- 842 [108] Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian,  
843 Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time  
844 recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.
- 845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863

864	TABLE OF CONTENTS	
865		
866	<b>1 Introduction</b>	<b>1</b>
867		
868	<b>2 VISGYM</b>	<b>2</b>
869		
870	<b>3 Evaluating Frontier Models with VISGYM</b>	<b>3</b>
871	3.1 Result and Analysis . . . . .	3
872		
873	<b>4 Diagnosing Frontier Models with VISGYM</b>	<b>5</b>
874		
875	<b>5 Training with VISGYM</b>	<b>6</b>
876	5.1 Supervised Fine-Tuning Experiments . . . . .	6
877	5.2 Stronger Base Model Generalizes Better . . . . .	7
878	5.3 Vision and LLM Both Matter . . . . .	7
879	5.4 Importance of Information-Revealing Behaviors for SFT Curation . . . . .	7
880	5.5 Generalization Across Tasks and Other Domains . . . . .	8
881		
882	<b>A Related Work</b>	<b>2</b>
883		
884	<b>B Solver Design</b>	<b>3</b>
885		
886	<b>C Environment Episode Progression</b>	<b>5</b>
887		
888	<b>D VISGYM Interface</b>	<b>5</b>
889		
890	<b>E ASCII-based Observation Visualization</b>	<b>6</b>
891		
892	<b>F Configuration of Environments</b>	<b>7</b>
893		
894	<b>G Analyzing Model Failures</b>	<b>8</b>
895		
896	<b>H Additional Performance Analysis</b>	<b>11</b>
897		
898		
899		
900		
901		
902		
903		
904		
905		
906		
907		
908		
909		
910		
911		
912		
913		
914		
915		
916		
917		

## A RELATED WORK

The development of foundation models (Achiam et al., 2023; Team et al., 2023; Team, 2025; Dubey et al., 2024; Yang et al., 2025), particularly vision-language models (VLMs) (Hong et al., 2025; Bai et al., 2025; Team et al., 2025; Zhu et al., 2025; Team et al., 2025; Liu et al., 2023; Alayrac et al., 2022; Li et al., 2022) and vision-language-action models (VLAs) (Black et al., 2024; Bjorck et al., 2025; Team et al., 2025; Kim et al., 2024; Octo Model Team et al., 2024; Huang et al., 2025; Team et al., 2025; Zhou et al., 2025; Niu et al., 2024; Shi et al., 2025), has reshaped how AI agents perceive, make decisions, and act across physical and simulated environments. To properly assess the capabilities and limitations of the models, a plethora of benchmarks have been developed.

Early benchmarks such as Atari, OpenAI Gym, and DeepMind Lab (Mnih et al., 2013; Brockman et al., 2016; Towers et al., 2024; Beattie et al., 2016) were developed to evaluate vision-based control and decision-making in fully observable environments. These platforms laid the groundwork for reinforcement learning but focused primarily on low-level motor control. Subsequent efforts extended these ideas to robotic manipulation and navigation, introducing partially observable, multi-task, and long-horizon settings that better reflect real-world complexity (Yu et al., 2020; Ahmed et al., 2020; Shridhar et al., 2020; Li et al., 2024; Cao et al., 2025; Liu et al., 2023; Khanna et al., 2024; Choi et al., 2024; Yang et al., 2025; Ehsani et al., 2021; Szot et al., 2021; Srivastava et al., 2022; Mees et al., 2022; Mandlkar et al., 2021; James et al., 2020). These modern suites enable training and evaluation of multi-task imitation learning and meta-learning policies across diverse embodiment and task horizons.

Concurrently, many VLM benchmarks have been developed to probe models’ cognitive and perceptual limits. Early efforts focused on visual question answering—first as multiple-choice tasks and later as open-ended reasoning (Yue et al., 2024; 2025; Liu et al., 2024; Li et al., 2024; Chen et al., 2024; Ballout et al., 2025). As visual grounding and reasoning improved, newer benchmarks began representing actions through text instead of fixed, predefined action spaces, enabling studies of the interplay between perception, reasoning, and control (Wang et al., 2025; Jang et al., 2024; Liu et al., 2023; Shi et al., 2025; Stojanovski et al., 2025; Abdulhai et al., 2023; Coelho et al., 2025). For example, G1 (Chen et al., 2025) introduces VLM-Gym, a suite of visual game environments with unified interfaces and adjustable difficulty. Broader evaluation suites such as VisualAgentBench (Liu et al., 2024), EmbodiedBench (Yang et al., 2025), and WebArena (Zhou et al., 2023) aggregate tasks across embodied control, graphical interfaces, and visual reasoning, challenging agents with multi-step planning and tool use.

VISGYM unifies reasoning and control under an RL-style “gym” paradigm, combining 17 multimodal tasks spanning visual puzzles, spatial reasoning, manipulation, and grounding. Each environment includes an oracle solution to ensure solvability and allow synthetic trajectory generation for post-training. Moreover, VISGYM introduces controllable difficulty along with targeted diagnostics—such as history utilization, representation variants, feedback specificity, and perception–action causality—allowing researchers to examine not only whether models fail but also what causes failures. We hope these designs enable more systematic analysis of VLMs and VLAs across domains and levels of interactivity.

## 972 B SOLVER DESIGN

973 This section provides detailed descriptions of the multi-step solvers introduced in Sec. §2 and used  
974 for supervised finetuning across all environments.

977 **Colorization.** The solver computes how far the current hue and saturation are from the target,  
978 breaks those differences into small incremental steps, and outputs a sequence of rotate and saturate  
979 actions that move steadily toward the correct color. If a target number of steps is requested or if the  
980 color is already close enough, it fills the sequence with reversible rotate/saturate pairs that cancel out  
981 and don't change the final state.

982 **Counting.** *mark\_all strategy:* The solver places a dot at the center of each target instance, then  
983 submits the correct total count and stops. *guess\_only strategy:* The solver directly submits the correct  
984 total count and stops, without placing any dots.

986 **Jigsaw.** *reorder strategy:* The solver computes a single permutation payload that, when applied  
987 via the reorder' action, instantly rearranges the current pieces into their correct target positions. *swap*  
988 *strategy:* The solver generates a minimal sequence of swap' actions by repeatedly finding a misplaced  
989 piece and swapping it with the piece at its correct target location. If a target number of steps is  
990 requested, it pads this sequence with reversible pairs of swaps (*e.g.*, swapping two pieces and then  
991 immediately swapping them back) until the desired length is reached.

992 **Matchstick Equation.** *bfs strategy:* The solver finds the shortest possible sequence of move'  
993 actions to correct the equation using a Breadth-First Search (BFS) and then stops. *dfs strategy:* The  
994 solver finds a solution using a Depth-First Search (DFS), producing a sequence of move' actions  
995 and undo' actions that represent its full exploratory and backtracking process before stopping. *sos*  
996 *strategy:* The solver first finds the shortest solution path (via BFS), then pads this path by inserting  
997 random, reversible detours. Before an optimal step, it takes one or more random move' actions and  
998 immediately undo'es them, returning to the optimal path before proceeding.

999 **Matchstick Rotation.** The solver first performs one or more translation-only move' actions, which  
1000 are typically unit-length moves in the general direction of the target. It then executes a final move'  
1001 action that applies the entire required rotation and corrects any remaining translation error, before  
1002 stopping.

1003 **Maze 2D.** The solver uses a graph search algorithm to find the optimal coordinate path from the  
1004 agent to the target, which is converted into the shortest sequence of move' actions. If a target number  
1005 of steps is requested, the solver pads this optimal sequence by inserting random, reversible move'  
1006 pairs (*e.g.*, move up' followed by move down') at valid locations along the path until the desired  
1007 length is met, before stopping.

1008 **Maze 3D.** The solver uses a graph search algorithm to find the optimal coordinate path from  
1009 the agent's location to the target. It then converts this path into the shortest sequence of turn' (left,  
1010 right, or around) and move' actions required to follow that path, accounting for the agent's current  
1011 orientation. If a target number of steps is requested, the solver pads this optimal sequence by inserting  
1012 random, reversible turn' pairs (*e.g.*, turn left' followed by turn right') at locations along the path until  
1013 the desired length is met, before stopping.

1014 **Mental Rotation 2D.** The solver first calculates the shortest total rotation angle required to align  
1015 the current image with the target. If the requested number of steps is 1, it outputs a single rotate'  
1016 action for that total angle. If a larger number of steps is requested, it stochastically divides the total  
1017 rotation into that many smaller rotate' actions, which are executed sequentially and sum to the correct  
1018 total angle, before stopping.

1019 **Mental Rotation 3D (CUBE).** The solver decomposes the total required rotation into its yaw, pitch,  
1020 and roll components. It then corrects each component sequentially. Before applying the corrective  
1021 'rotate' action for a specific axis (*e.g.*, yaw), it first executes a padding sequence of four 90-degree  
1022 rotations around that same axis. After this 360-degree padding, it applies the single action to correct  
1023 the yaw. It repeats this pad-then-correct process for the pitch and roll axes, then stops.

1024 **Mental Rotation 3D (OBJAVERSE).** The same as Mental Rotation 3D (CUBE).  
1025

1026 **MuJoCo Fetch (PICK-AND-PLACE).** The solver is a state-machine-based oracle. It follows a  
1027 sequence: (0) move the gripper to a safe height above the object, (1) open the gripper, (2) descend to  
1028 the object, (3) close the gripper to grasp. (4) Once grasped, it moves the object directly toward the  
1029 3D goal position using a greedy, per-axis strategy (correcting the axis with the largest error at each  
1030 step). (7) Finally, it holds the object at the target location and stops.

1031 **MuJoCo Fetch (REACH).** The solver is a greedy, per-axis oracle. At each step, it identifies the  
1032 single axis (x, y, or z) with the largest error between the gripper and the goal. It then outputs a ‘move’  
1033 action along that single axis to reduce the error, repeating this process until the goal is reached, at  
1034 which point it stops.

1035 **Patch Reassembly.** The solver uses a backtracking search to find the optimal sequence of ‘place’  
1036 actions that perfectly tile the grid. If a target number of steps is requested, it pads this sequence by  
1037 repeatedly inserting “mistake-and-correct” actions: it finds a correct ‘place’ action in the solution,  
1038 finds a valid wrong location for that piece, and inserts this “mistake” ‘place’ action immediately  
1039 before the “correct” ‘place’ action. If no valid mistakes can be found, it falls back to inserting a  
1040 ‘remove’ and a duplicate ‘place’ action. This repeats until the desired number of ‘place’ actions is  
1041 met.

1042 **Referring Dot-Pointing.** The solver first samples a random pixel from within the target object’s  
1043 segmentation mask and also calculates the mask’s center of mass. It then generates a sequence of  
1044 ‘mark’ actions by linearly interpolating from the random starting point to the center of mass over the  
1045 requested number of steps. The final action in this sequence places a mark at the exact center of mass,  
1046 which is then followed by a ‘stop’ action.

1047 **Sliding Block.** The solver uses a Breadth-First Search (BFS) to find the shortest sequence of  
1048 ‘move’ actions from the current board state to the target configuration. If a target number of steps  
1049 is requested, it pads this optimal path by first reconstructing all intermediate board states. At each  
1050 state, it identifies all valid “back-and-forth” moves (*e.g.*, move block 1 right, then move block 1 left).  
1051 It then randomly samples from these opportunities and inserts the required number of ‘move’ and  
1052 ‘reverse-move’ pairs into the solution path until the desired length is met, before stopping.

1053 **Video Unshuffle.** *reorder strategy:* The solver computes a single permutation payload that,  
1054 when applied via the ‘reorder’ action, instantly rearranges the shuffled frames into their correct  
1055 chronological order, then stops. *swap strategy:* The solver generates a minimal sequence of ‘swap’  
1056 actions to sort the frames. It iterates through the positions, and if a frame is in the wrong place, it  
1057 finds the correct frame and swaps it into its target position, repeating until all frames are sorted, then  
1058 stops.

1059 **Zoom-In Puzzle.** The same as Video Unshuffle.

1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079

1080	C ENVIRONMENT EPISODE PROGRESSION	
1081		
1082	Referenced in Tab. 1 and Sec. §2, this section presents detailed episode progressions for each	
1083	environment. A summary index with page numbers is provided below:	
1084	<b>Colorization</b>	B. 2
1085		
1086	<b>Counting</b>	B. 4
1087		
1088	<b>Jigsaw</b>	B. 6
1089		
1090	<b>Matchstick Equation</b>	B. 8
1091		
1092	<b>Matchstick Rotation</b>	B. 10
1093		
1094	<b>Maze 2D</b>	B. 12
1095		
1096	<b>Maze 3D</b>	B. 14
1097		
1098	<b>Mental Rotation 2D</b>	B. 16
1099		
1100	<b>Mental Rotation 3D (CUBE)</b>	B. 18
1101		
1102	<b>Mental Rotation 3D (OBJAVERSE)</b>	B. 20
1103		
1104	<b>MuJoCo Fetch PICK-AND-PLACE</b>	B. 22
1105		
1106	<b>MuJoCo Fetch REACH</b>	B. 24
1107		
1108	<b>Patch Reassembly</b>	B. 26
1109		
1110	<b>Referring Dot-Pointing</b>	B. 28
1111		
1112	<b>Sliding Block</b>	B. 30
1113		
1114	<b>Video Unshuffle</b>	B. 32
1115		
1116	<b>Zoom-In Puzzle</b>	B. 34
1117		

## D VISGYM INTERFACE

In this section, we present the pseudocode for the `step` function (Algorithm 1) used in VISGYM (*i.e.*, Sec. §2). The function initializes the reward and both termination flags, then parses the model’s output string into an action name and payload. If parsing fails, it immediately returns an observation with “invalid format” as feedback.

If the parsed action name is supported and its payload is valid for the corresponding action space, the function calls `Apply`, which executes the action and returns the environment feedback. Otherwise, it ends early with “invalid action” as feedback.

Termination and truncation are determined inside `Apply`. If the action triggers termination (*e.g.*, `stop`), the function computes the final reward based on the environment state. Thus, the returned reward is always zero for non-terminal transitions and the final score upon termination.

Finally, the function returns the new observation, reward, termination, and truncation flags, and the feedback describing the action outcome.

---

**Algorithm 1** Generic Step Function (Sec. §D). Symbols:  $\rho$  = reward,  $\tau$  = terminated,  $v$  = truncated,  $\varphi$  = feedback,  $\alpha$  = action name,  $\pi$  = payload,  $\iota$  = info.

---

```

function STEP( $a$ )
   $\rho \leftarrow 0$ 
   $(\tau, v) \leftarrow (false, false)$ 
  Parse  $a \rightarrow (\alpha, \pi)$ 
  if invalid format then
    return
    ( $obs()$ , 0,  $\tau$ ,  $v$ ,  $\iota$ (“invalid format”))
  end if
  if  $\alpha \in \mathcal{A}$  and  $\pi \in \mathcal{A}[\alpha]$  then
     $(\varphi, \tau, v) \leftarrow \text{Apply}(\alpha, \pi)$ 
  else
    return
    ( $obs()$ , 0,  $\tau$ ,  $v$ ,  $\iota$ (“invalid action”))
  end if
  if  $\tau = true$  then
     $\rho \leftarrow \text{ComputeReward}()$ 
  end if
  return ( $obs()$ ,  $\rho$ ,  $\tau$ ,  $v$ ,  $\iota(\varphi)$ )
end function

```

---

## E ASCII-BASED OBSERVATION VISUALIZATION

In this section, we present example episode variants rendered in text, as discussed in Sec. §4, for the Sliding Block, Maze 2D, Patch Reassembly, and Matchstick Equation environments in Fig. 12.

Note that the instructions are slightly adapted to fit the text-based format (*e.g.*, in the visual version of Patch Reassembly, we describe the *anchor* as “the cell that shows the patch’s ID number,” while in the text version we note that “the anchor cell for each parked patch is marked with a ‘\*’ instead of its ID number”).

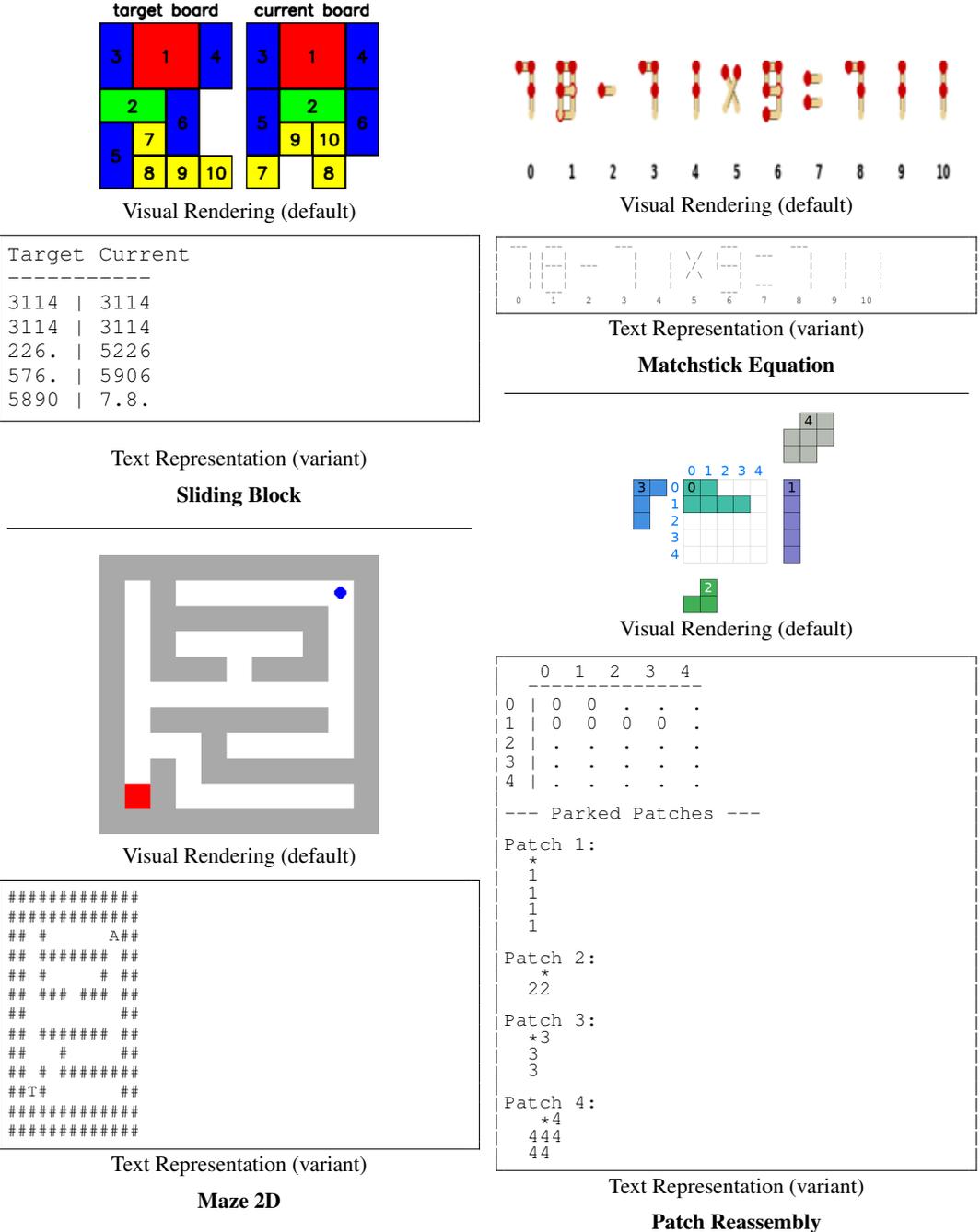


Figure 12: Visual and text representations across four environments

## F CONFIGURATION OF ENVIRONMENTS

In this section, we provide the tunable parameters for each task that determine its difficulty. Note that in some environments, higher difficulty primarily increases the complexity of the input observations (e.g., Sec. §5.2), while in others it tightens the reward criteria and requires more precise control. Details are in Tab. 4.

**Table 4: Configuration summary for tasks:** tunable parameters, difficulty, and source datasets.

Task	Tunable Difficulty Parameters	Easy	Hard	Src. Dataset
Colorization	Accuracy radius $ar$ (precision required for hue and saturation match).	$ar = 11$	$ar = 16$	LLaVA Liu et al. (2023)
Counting	Minimum and maximum count range $c_{min}, c_{max}$ .	$c_{min} = 2,$ $c_{max} = 20$	$c_{min} = 5,$ $c_{max} = 30$	LVIS Gupta et al. (2019)
Jigsaw	number of rows and columns $nr, nc$ .	$nr = 2, nc = 2$	$nr = 3, nc = 3$	LLaVA Liu et al. (2023)
Matchstick Equation	Number of break moves $bm$ (corruptions to fix).	$bm = 1$	$bm = 2$	—
Matchstick Rotation	Hidden scale range $sr$ , position tolerance $pt$ , angular tolerance $at$ .	$pt = 10,$ $at = 15$	$pt = 5,$ $at = 10$	—
Maze 2D	Maze width and height $mw, mh$ .	$mw = 9, mh = 9$	$mw = 11, mh = 11$	—
Maze 3D	Maze width and height $mw, mh$ .	$mw = 7, mh = 7$	$mw = 9, mh = 9$	—
Mental Rotation 2D	Angular tolerance $at$ .	$at = 10.0$	$at = 5.0$	LLaVA Liu et al. (2023)
Mental Rotation 3D (CUBE)	Number of segments $ns$ , length range $lr$ , angular tolerance $at$ .	$ns = 4$	$ns = 6$	—
Mental Rotation 3D (OBJVERSE)	Angular tolerance $at$ .	$at = 15.0$	$at = 5.0$	Objaverse Deitke et al. (2023)
MuJoCo Fetch PICK-AND-PLACE	No user-tuned difficulty parameters (standardized task).	Standard	Standard	—
MuJoCo Fetch REACH	No user-tuned difficulty parameters (standardized task).	Standard	Standard	—
Patch Reassembly	Grid size $gs$ , number of patches $np$ .	$gs = (6, 6),$ $np = 5$	$gs = (8, 8),$ $np = 6$	—
Referring DOT-POINTING	No user-tuned difficulty parameters (standardized task).	Standard	Standard	RefCOCO Kazemzadeh et al. (2014)
Sliding Block	Number of shuffle moves $sm$ .	$sm = 30$	$sm = 90$	—
Video Unshuffle	Number of frames $nf$ , sampling strategy $ss$ , minimum frame-diff threshold $mfd$ .	$nf = 4$	$nf = 5$	SS2 Goyal et al. (2017)
Zoom-In Puzzle	Zoom gap $zg$ , zoom variability $zs$ , minimum zoom $mz$ , num. of views $zv$ , nested crop $nest$ .	$zv = 4$	$zv = 5$	LLaVA Liu et al. (2023)

Table 5: Example clusters discovered by StringSight.

Cluster Description
<b>MuJoCo Fetch (Pick-and-Place)</b>
<ul style="list-style-type: none"> <li>The model issues repetitive movement commands without adapting based on task progress or environment feedback, resulting in oscillatory behaviors like moving up and down, left and right, or in a single direction with no meaningful progress toward grasping or placing the cube. For example, it may move forward repeatedly even after overshooting the target or oscillate without ever attempting a grasp.</li> <li>Ignores visual feedback from images and does not adjust its actions in response to clear cues about the state or position of the cube, gripper, or target marker. This results in the model issuing irrelevant or counterproductive actions, such as continuing to move when the cube has not been grasped.</li> <li>Issues the “stop” command prematurely before the end-effector is even close to the target marker, ending the task early without justification or clear success criteria.</li> </ul>
<b>Mental Rotation 3D (Cube)</b>
<ul style="list-style-type: none"> <li>The model repeatedly issues the same rotation action, often on a single axis, for many steps without adapting based on visual feedback, resulting in rigid loops or unbroken patterns.</li> <li>The model oscillates between a small set of orientations, alternating or repeating similar rotations (e.g., <math>+45^\circ</math>, <math>-45^\circ</math>), causing the object to cycle endlessly without making progress toward the target alignment.</li> </ul>
<b>Zoom-In Puzzle</b>
<ul style="list-style-type: none"> <li>The model finalizes the arrangement immediately without performing any swaps, reordering, or verification, accepting the initial sequence as correct regardless of its accuracy. For example, it issues a “stop” command on the first step even if the order is incorrect.</li> </ul>
<b>Matchstick Rotation</b>
<ul style="list-style-type: none"> <li>The model issues fixed or monotonically decreasing movement and rotation magnitudes without attempting to estimate the unknown scale or using exploratory actions to resolve scale ambiguity, proceeding as if the appropriate step size is already known.</li> <li>Action sequences do not adapt based on feedback or observed outcomes; the model follows a predetermined or repetitive strategy without checking if moves are effective or responding to evidence from the environment.</li> </ul>
<b>Maze 2D</b>
<ul style="list-style-type: none"> <li>The model repeatedly issues the same invalid movement commands, such as trying to move into walls, even after receiving explicit feedback that these actions are not possible. For example, it continues to try moving left into a wall after being told each time that the move is blocked.</li> <li>Does not build, update, or use any internal map or memory of previous moves or environmental feedback, resulting in repeated visits to the same locations, blocked paths, and inefficient looping navigation.</li> </ul>
<b>Maze 3D</b>
<ul style="list-style-type: none"> <li>Movement decisions are based exclusively on immediate sensory feedback, without building or referencing internal memory or a map of previously explored locations, leading to repeated wall collisions, revisiting dead ends, and inefficient navigation.</li> <li>Frequently issues long sequences of consecutive turning actions—such as alternating left and right—without forward movement or meaningful progress toward the goal, resulting in wasted steps and inefficient pathfinding.</li> </ul>

## G ANALYZING MODEL FAILURES

We run StringSight Dunlap et al. (2025); Lisa Dunlap et al. (2025), a pipeline for automatically uncovering failure cases and comparing models. It uses a VLM annotator (GPT-4.1) to extract behaviors from each trace (e.g., “uses `move(1, 1)` for all 20 steps”) and clusters these behaviors into higher-level patterns (e.g., “repeats the same action”). Examples of discovered cluster descriptions are shown in Table 5. We then manually examine the top failure cases for each task and identify four common failure modes across all tasks.

*(1) Restricted action space and action looping:* models often rely on a single repeated operation or fixed-magnitude action, such as continually moving in the same direction in Fetch Pick & Place, using “swap” in Jigsaw instead of “reorder”, or rotating by the same angle in Mental Rotation 3D and Match Rotation rather than converging to an optimal magnitude.

*(2) State mismanagement:* models fail to maintain or update internal state across steps. They ignore textual or environmental feedback, revisit previously explored areas, or repeat illegal actions despite prior errors—for example, continuing to move into a wall after being told they have collided, or repeating invalid moves in the Match Equation, Sliding Block, and Toy Maze 2D tasks.

1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349

### System Prompt: Labeling failure modes in traces

You are an expert model behavior analyst. Your task is to meticulously analyze the trace of a large language model to identify whether it contains any of the following behaviors:

- **Restricted action space and action looping:** The model keeps repeating the same or nearly identical action without making progress. Look for consecutive turns with the same command or sequence of commands, movements by the same amount, or the same tool being used even when it is ineffective.
- **State mismanagement:** The model forgets or ignores what it already learned in earlier steps. It may revisit old states, contradict past reasoning, or repeat mistakes it was corrected for (e.g. being told it hit a wall and then continuing to move forward in the same direction). Do not include if this is simply action looping, where the model is repeating the same action without making progress; this is specifically when the model is ignoring feedback or not adjusting its behavior based on its previous actions, but is still issuing different commands.
- **Early termination:** The model stops too early. Early means terminating before the maximum number of steps is reached.
- **Failure to use visual or spatial information:** The model ignores visible or spatial cues. This applies to traces where either an image or ASCII art is provided, and the model does not react to changes in the scene. For example, if the object leaves the frame, but the model continues to move towards it. Look for actions that contradict what's visually or spatially clear. Do not include if the model is simply action looping, where the model is repeating the same action without making progress; this is specifically when the model is not utilizing the visual information when it is available. If the trace does not provide visual information (either images or ASCII), do not include this label.

If the trace contains any of the behaviors, return a list of objects with the following structure. If a trace has more than one behavior, return a list of objects with the structure below for each behavior. If the trace contains none of the behaviors, return an empty list.

#### JSON Output Structure

```
[
  {
    "property_description": "which behavior is present in the
trace",
    "reason": "an explanation of the exact behaviors in the
trace that fall under the property_description
(1-2 sentences)",
    "evidence": "What exactly in the trace exhibits this
property? Include quotes/tool calls/actions when possible."
  }
]
```

(3) *Early termination:* the model terminates the episode before the maximum steps, despite not reaching the goal.

(4) *Failure to use visual or spatial information:* models ignore the visual information provided, such as the target leaving the frame or the item being successfully aligned (e.g., Mental Rotation).

Finally, we quantify the prevalence of each failure mode by having a VLM annotator (GPT-4.1) label each trace for these behaviors (a trace may exhibit multiple behaviors).

**Frequency of failures.** Figure 13 shows the proportion of traces that contain each failure. We see that action looping is very common, occurring in more than 60% of traces, followed in frequency by early termination, state mismanagement, and failure to use visual or spatial information. Looking at how the frequency of the behaviors changes compared across tasks, we see in Figure 14 (b) that certain tasks, like Matchstick Equation and Sliding Block, result in a particularly large amount of action repetition and state mismanagement failures, likely due to the difficulty of the task and the frequency of invalid moves. We additionally see that tasks like the Maze task, which provide clear



and state mismanagement, which is surprising given that overall performance decreases without feedback. This suggests the presence of additional failure modes not captured by our taxonomy, which we leave for future work.

*Ground truth state given at the beginning (Figure 15c):* When given the ground truth state at the start of the task, the model is less likely to “guess” or give up early, reflected in lower rates of action looping and early termination.

*Image vs. text representation (Figure 15d):* For tasks aside from Matchstick Equation, models process visual information more effectively when it is presented as text rather than an image. The large reduction in action looping suggests that this text-based representation provides clearer guidance for selecting actions.

## G.2 FAILURE TRAJECTORIES VISUALIZATION

Using StringSight, we visualize the trajectory for each failure type. In each trajectory, we show the prompt, the image, the models’ raw output, and the action parsed from the raw output. We also show the output from StringSight for each trajectory, tagged “Reason” and “Evidence” at the top, where “Reason” stands for StringSight’s reason for classifying this trajectory into a specific failure category, and “Evidence” stands for the evidence in the trajectory that leads to the conclusion.

(1) *Restricted action space and action looping:* As in Sec. §F.0.2, we show a case of action looping of GPT-5 on the Jigsaw task. The model repeatedly takes the same action (“swap”, (0, 0), (0, 1))”, resulting in looping behaviors without making any progress.

(2) *State mismanagement:* As in Sec. §F.0.4, we show a case of Claude Sonnet 4 on Maze 2D. In Observation 7, the model takes action (“move”, 2)”, which leads to the environment feedback “Cannot move into a wall.” However, at Observation 16, the model is in the exact same state, disregards the previous feedback, and takes the same action (“move”, 2)” again.

(3) *Early termination:* We show in Sec. §F.0.6 a case of Gemma 3 27B Instruct on Matchstick Equation, where the model decides to give up and terminate at step 13, while the model is allowed to take 30 steps in total.

(4) *Failure to use visual or spatial information:* As in Sec. §F.0.8, we show a case of Gemini 2.5 Pro on Mental Rotation 3D (Cube). In the last three steps, after rotating in the wrong direction, the model does not take the visual information into account and continues to rotate in the same direction, which moves the object even farther away from the target position.

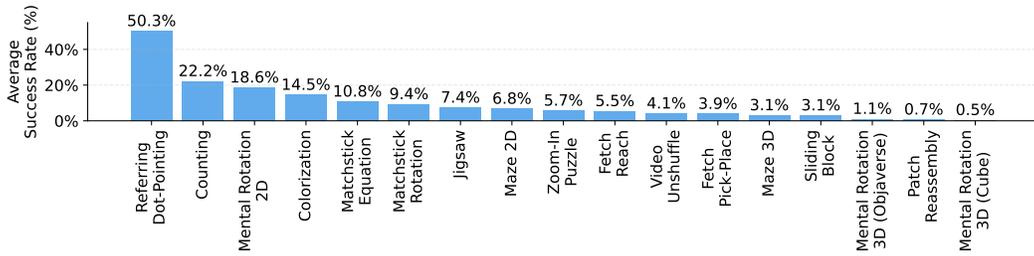
## H ADDITIONAL PERFORMANCE ANALYSIS

**Difficulty of Each Task.** In Fig. 16, we compute the average accuracy across models for each task and sort tasks from easiest to hardest based on these averages. In general, we found that Referring Dot-Pointing and Counting are the easiest, with models achieving over 20% accuracy on average, whereas Mental Rotation 3D (Cube), Patch Reassembly, and Mental Rotation 3D (Objaverse) are the hardest with an accuracy around 1%. Sliding Block, Maze 3D, Fetch Pick-Place, and Video Unshuffle also pose significant challenges for the models, with less than 5% on average. This suggests that tasks requiring memory and long-horizon planning, or strong 3D spatial understanding, remain the most difficult for current models.

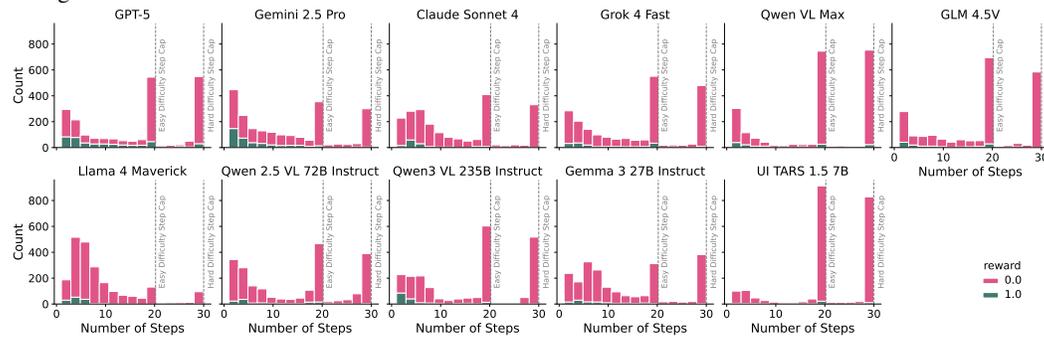
**Number of Steps.** In Fig. 17, we calculate the number of steps taken on all trajectories for each model and calculate the number of correct trajectories (green) and the number of incorrect trajectories (red). There is a clear cutoff on steps 20 (maximum steps allowed for Easy setting) and 30 (maximum steps allowed for Hard setting), indicating that all models tend to reach the maximum number of steps. We also observed a “U-shaped” trend over the steps for all models, where they tend to either terminate early or continue until the final step.

**Easy to Hard Performance Drop.** In Fig. 18, we calculate the average accuracy in Easy and Hard, respectively, on all models, and then visualize the performance gap between easy and hard on each task. The biggest Easy to Hard performance drops occur on

1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511



**Figure 16: Average success rate across frontier models on each task.** The easiest tasks are Referring Dot-Pointing, and Counting, with over 20% accuracy on average across all models, while the hardest tasks are Mental Rotation 3D (Cube), Patch Reassembly, and Mental Rotation, with the average accuracy less than 2% on average.



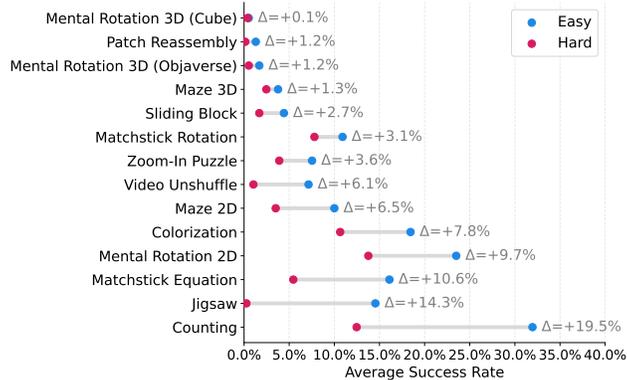
**Figure 17: The Number of Steps each Model Takes Over all Tasks.** We calculate the number of steps over all trajectories for each model and visualize the correct trajectories (green) and the incorrect trajectories (red).

Counting and Jigsaw. For Counting, accuracy drops sharply as the number of objects increases. For Jigsaw, performance drops to near zero as the puzzle changes from 2x2 to 3x3, suggesting that this task can be further scaled to even more difficult  $n \times n$  configurations.

For some tasks (e.g., Patch Reassembly, Sliding Block, Video Unshuffle), the absolute gap is smaller, likely because Easy performance is already very low ( $\approx 0$ ). These tasks are also naturally scalable in terms of difficulty. For example, increasing the number of patches for Patch Reassembly, the number of blocks for Sliding Block, or the number of frames for Video Unshuffle. As VLMs improve and begin to reliably solve the Easy settings, we expect to see larger easy-to-hard gaps on these tasks, and that our gym can be correspondingly scaled to provide harder task variants.

**Model Rankings.**

In Fig. 19, we plot each model’s ranking on every task. When multiple models tie on a task, we assign them the worse (higher) numerical rank. For example, if two models are tied for the best score, we label both as rank 2, so that ties do not overstate how clearly a model is separated from others. While these mostly align with the global leaderboard, they also reveal clear task-specific strengths and weaknesses. For example, on specialization, Qwen2.5 VL 72B Instruct performs well on Counting (rank 2), Gemma 3 27B Instruct performs well on Maze 3D (rank 3), and Qwen3 VL 235B Instruct performs well on Referring Dot-Pointing. On weakness, despite strong average performance, GPT-5 performs poorly on Video Unshuffle and Maze 3D, while Gemini 2.5 Pro performs poorly on Jigsaw (Hard) and Patch Reassembly.



**Figure 18: Easy → Hard Performance Drop.** For each task, we calculate the average accuracy on Easy and Hard, respectively, over all models, and then visualize the performance drop between Easy and Hard.



## F.0.2 SAMPLE TRAJECTORY FOR "ACTION LOOPING" (BY GPT-5) (Back to Table of Contents)

**Reason**

The model repeatedly issues the same swap command between the same two pieces without making any progress towards solving the puzzle, as evidenced by identical actions over multiple consecutive steps.

**Evidence**

"('swap', ((0, 0), (0, 1)))" is issued in steps 1, 2, 3, 4, 5, and 6, with no change in strategy or progress.

**Prompt  $p_0$** 

You are solving a 2x2 jigsaw puzzle. The puzzle pieces are currently scrambled. Your goal is to rearrange the pieces to recover the image.

Available actions:

1. 'swap': Swap two pieces by specifying their coordinates. Format: `('swap', ((row1, col1), (row2, col2)))` where coordinates start from (0,0) at the top-left corner.
2. 'reorder': Reorder all pieces at once. Format: `('reorder', [0, 1, 2, ..., 3])` where the list represents the desired order of pieces from top-left to bottom-right.
3. 'stop': End the puzzle solving session. Format: `('stop', 'stop')`

Please respond with exactly one action and its arguments in the specified format. For example:

- To swap two pieces: `('swap', ((0, 0), (1, 1)))`
- To reorder all pieces: `('reorder', [0, 1, 2, ..., 3])`
- To stop: `('stop', 'stop')`

Index-to-cell mapping (0-based rows/cols):

- Index = row \* 2 + col.
  - Row 0 covers indices 0..1, row 1 covers 2..3, etc.
- Example for 2x2: (0,0)->0, (0,1)->1, (1,0)->2, (1,1)->3.

This is step 1. You are allowed to take 19 more steps.

**Observation  $o_0$** **Action  $a_0$** 

('swap', ((0, 0), (0, 1)))

**VLM Raw Output:**

('swap', ((0, 0), (0, 1)))

**Observation  $o_1$** **Prompt  $p_1$** 

This is step 2. You are allowed to take 18 more steps.

**Action  $a_1$** 

('swap', ((0, 0), (0, 1)))

**VLM Raw Output:**

('swap', ((0, 0), (0, 1)))

1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673

**Observation  $o_2$**



**Prompt  $p_2$**

This is step 3. You are allowed to take 17 more steps.

**Action  $a_2$**

('swap', ((0, 0), (0, 1)))

**VLM Raw Output:**

('swap', ((0, 0), (0, 1)))

**Observation  $o_3$**



**Prompt  $p_3$**

This is step 4. You are allowed to take 16 more steps.

**Action  $a_3$**

('swap', ((0, 0), (0, 1)))

**VLM Raw Output:**

('swap', ((0, 0), (0, 1)))

**Observation  $o_4$**



**Prompt  $p_4$**

This is step 5. You are allowed to take 15 more steps.

**Action  $a_4$**

('swap', ((0, 0), (0, 1)))

**VLM Raw Output:**

('swap', ((0, 0), (0, 1)))

**Observation  $o_5$**



**Prompt  $p_5$**

This is step 6. You are allowed to take 14 more steps.

**Action  $a_5$**

('swap', ((0, 0), (0, 1)))

**VLM Raw Output:**

('swap', ((0, 0), (0, 1)))

1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727

**Observation  $o_6$**



**Prompt  $p_6$**

This is step 7. You are allowed to take 13 more steps.

**Action  $a_6$**

('stop', 'stop')

**VLM Raw Output:**

('stop', 'stop')

F.0.4 **SAMPLE TRAJECTORY FOR “STATE MISMANAGEMENT” (BY CLAUDE SONNET 4)** ([Back to Table of Contents](#))

**Reason**

The model revisits previously blocked directions and fails to remember which moves have already been proven impossible, leading to repeated failed actions instead of seeking new routes.

**Evidence**

After being told ‘Cannot move into a wall.’ for left and up, the model continues to try those directions in subsequent steps, e.g., after failing to move left at step 9, it tries left again at steps 11, 13, and 17; after failing to move up at step 10, it tries up again at steps 15, 16, and 18.

**Prompt  $p_0$**

You are navigating a 9x9 maze. The maze consists of walls (gray) and paths (white). You are represented by a blue circle, and your goal is to reach the red target square.

Available actions:

1. ‘move’: Move in one of four directions. Format: `\('move', direction)\`` where direction is an integer:
  - 0=right, 1=up, 2=left, 3=down
2. ‘stop’: End the navigation session. Format: `\('stop', 'stop')\``

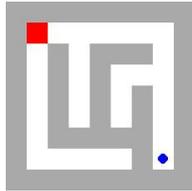
Success: You succeed if you reach the red target square.

Please respond with exactly one action and its arguments in the specified format. For example:

- To move right: `\('move', 0)\``
- To move up: `\('move', 1)\``
- To move left: `\('move', 2)\``
- To move down: `\('move', 3)\``
- To stop: `\('stop', 'stop')\``

This is step 1. You are allowed to take 19 more steps.

**Observation  $o_0$**



**Action  $a_0$**

`\('move', 2)\``

**VLM Raw Output:**

I need to analyze the maze to find a path from my current position (blue circle) to the red target square.

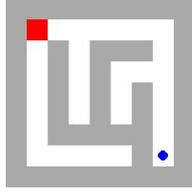
Looking at the maze, I can see:

- I’m currently positioned in the bottom right area (blue circle)
- The red target is in the top left corner
- There are gray walls and white paths
- I need to navigate through the available white paths

From my current position, I can see there’s a white path that I can move through. Let me start by moving left to begin navigating toward the target.

`\('move', 2)\``

1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835

**Observation  $o_1$** **Prompt  $p_1$** 

Environment feedback: Cannot move into a wall.

This is step 2. You are allowed to take 18 more steps.

**Action  $a_1$** 

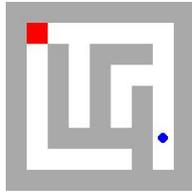
`('move', 1)`

**VLM Raw Output:**

I see that I cannot move left from my current position because there's a wall there. Let me reassess my surroundings.

Looking at my current position (blue circle), I need to find which directions have white paths available. From what I can see, I should try moving up to find an available path.

`('move', 1)`

**Observation  $o_2$** **Prompt  $p_2$** 

Environment feedback: Action executed successfully.

This is step 3. You are allowed to take 17 more steps.

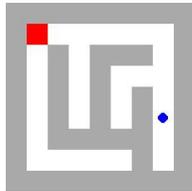
**Action  $a_2$** 

`('move', 1)`

**VLM Raw Output:**

Good! I successfully moved up. Now I can see I'm in a vertical white path. Looking at the maze structure, I need to continue moving up to find a way to navigate toward the red target in the top left corner.

`('move', 1)`

**Observation  $o_3$** **Prompt  $p_3$** 

Environment feedback: Action executed successfully.

This is step 4. You are allowed to take 16 more steps.

**Action  $a_3$** 

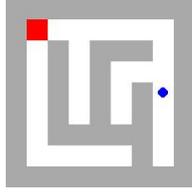
`('move', 1)`

**VLM Raw Output:**

Great! I've moved up again and I'm continuing along the white path. I can see that I'm in a vertical corridor on the right side of the maze. I need to keep moving up to find a horizontal path that will allow me to move toward the left side where the red target is located.

`('move', 1)`

1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889

**Observation  $o_4$** **Prompt  $p_4$** 

Environment feedback: Action executed successfully.

This is step 5. You are allowed to take 15 more steps.

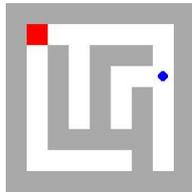
**Action  $a_4$** 

`('move', 1)`

**VLM Raw Output:**

Excellent! I've moved up again and I'm getting closer to the top of the maze. I can see that I'm still in the vertical white path on the right side. I need to continue moving up to reach a point where I can turn left and start moving toward the red target in the top left corner.

``('move', 1)``

**Observation  $o_5$** **Prompt  $p_5$** 

Environment feedback: Action executed successfully.

This is step 6. You are allowed to take 14 more steps.

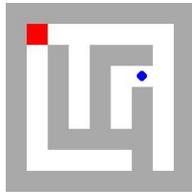
**Action  $a_5$** 

`('move', 2)`

**VLM Raw Output:**

Perfect! I've moved up again and I can see that I'm now at a position where there appears to be a horizontal white path available to my left. This should allow me to start moving toward the red target. Let me try moving left now.

``('move', 2)``

**Observation  $o_6$** **Prompt  $p_6$** 

Environment feedback: Action executed successfully.

This is step 7. You are allowed to take 13 more steps.

**Action  $a_6$** 

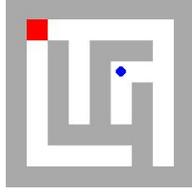
`('move', 2)`

**VLM Raw Output:**

Great! I successfully moved left and I'm now in a horizontal white path. I can see I'm making progress toward the left side of the maze where the red target is located. I should continue moving left to get closer to the target.

``('move', 2)``

1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943

**Observation  $o_7$** **Prompt  $p_7$** 

Environment feedback: Action executed successfully.

This is step 8. You are allowed to take 12 more steps.

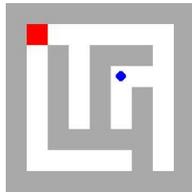
**Action  $a_7$** 

`('move', 2)`

**VLM Raw Output:**

Excellent! I've moved left again and I'm getting closer to the left side of the maze. I can see that I'm still in a white path. Looking at the maze structure, I need to continue navigating to reach the red target in the top left corner. Let me try moving left again to continue in this direction.

``('move', 2)``

**Observation  $o_8$** **Prompt  $p_8$** 

Environment feedback: Cannot move into a wall.

This is step 9. You are allowed to take 11 more steps.

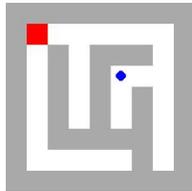
**Action  $a_8$** 

`('move', 1)`

**VLM Raw Output:**

I see that I can't move left anymore because there's a wall blocking that direction. Looking at my current position, I need to find another route. Since I can't go left, let me try moving up to see if there's a path that will eventually lead me to the red target.

``('move', 1)``

**Observation  $o_9$** **Prompt  $p_9$** 

Environment feedback: Cannot move into a wall.

This is step 10. You are allowed to take 10 more steps.

**Action  $a_9$** 

`('move', 3)`

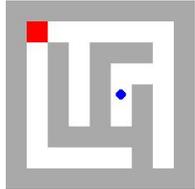
**VLM Raw Output:**

I can't move up either because there's a wall there. Let me try moving down to see if there's an alternative path that will allow me to navigate around this obstacle and find another route to the red target.

``('move', 3)``

1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997

**Observation  $o_{10}$**



**Prompt  $p_{10}$**

Environment feedback: Action executed successfully.

This is step 11. You are allowed to take 9 more steps.

**Action  $a_{10}$**

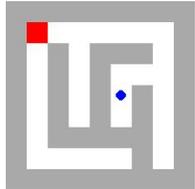
`('move', 2)`

**VLM Raw Output:**

Good! I successfully moved down. Now I'm in a different position and I need to find a way to continue toward the red target. Looking at the maze structure, I should try moving left again to see if there's now a path available in that direction.

``('move', 2)``

**Observation  $o_{11}$**



**Prompt  $p_{11}$**

Environment feedback: Cannot move into a wall.

This is step 12. You are allowed to take 8 more steps.

**Action  $a_{11}$**

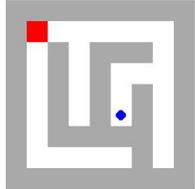
`('move', 3)`

**VLM Raw Output:**

I can't move left from this position either. Let me try moving down again to continue exploring the available paths and find a route that will eventually lead me to the red target.

``('move', 3)``

**Observation  $o_{12}$**



**Prompt  $p_{12}$**

Environment feedback: Action executed successfully.

This is step 13. You are allowed to take 7 more steps.

**Action  $a_{12}$**

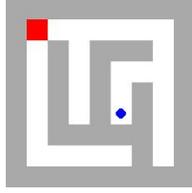
`('move', 2)`

**VLM Raw Output:**

Good! I moved down successfully. Now I'm in a lower position in the maze. Let me try moving left from this new position to see if I can find a path that leads toward the red target.

``('move', 2)``

1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051

**Observation  $o_{13}$** **Prompt  $p_{13}$** 

Environment feedback: Cannot move into a wall.

This is step 14. You are allowed to take 6 more steps.

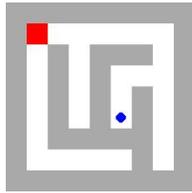
**Action  $a_{13}$** 

`('move', 3)`

**VLM Raw Output:**

I can't move left from here either. Let me continue moving down to explore more of the maze and find an alternative path to the red target.

``('move', 3)``

**Observation  $o_{14}$** **Prompt  $p_{14}$** 

Environment feedback: Cannot move into a wall.

This is step 15. You are allowed to take 5 more steps.

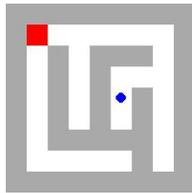
**Action  $a_{14}$** 

`('move', 1)`

**VLM Raw Output:**

I can't move down anymore because there's a wall. Let me try moving up to go back and explore a different route.

``('move', 1)``

**Observation  $o_{15}$** **Prompt  $p_{15}$** 

Environment feedback: Action executed successfully.

This is step 16. You are allowed to take 4 more steps.

**Action  $a_{15}$** 

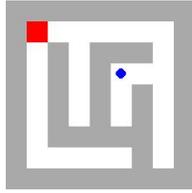
`('move', 1)`

**VLM Raw Output:**

Good! I moved up successfully. Now I'm back to a previous position. Let me try moving up again to continue exploring and find a path that will lead me toward the red target.

``('move', 1)``

2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105

**Observation  $o_{16}$** **Prompt  $p_{16}$** 

Environment feedback: Action executed successfully.

This is step 17. You are allowed to take 3 more steps.

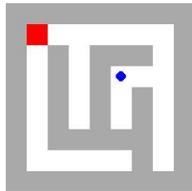
**Action  $a_{16}$** 

`('move', 2)`

**VLM Raw Output:**

Good! I'm back to this position. I'm running low on steps (only 3 left), so I need to think carefully. From what I can see of the maze structure, let me try moving left again to see if I can find a path that leads to the red target.

``('move', 2)``

**Observation  $o_{17}$** **Prompt  $p_{17}$** 

Environment feedback: Cannot move into a wall.

This is step 18. You are allowed to take 2 more steps.

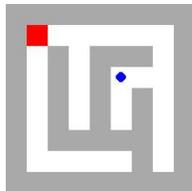
**Action  $a_{17}$** 

`('move', 1)`

**VLM Raw Output:**

I'm running out of steps and haven't found a clear path to the red target yet. Looking at the maze structure, I need to find a way to get to the left side. Let me try moving up to see if there's a path at a higher level that I might have missed.

``('move', 1)``

**Observation  $o_{18}$** **Prompt  $p_{18}$** 

Environment feedback: Cannot move into a wall.

This is step 19. You are allowed to take 1 more steps.

**Action  $a_{18}$** 

`('move', 3)`

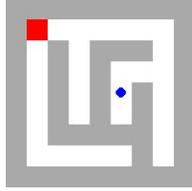
**VLM Raw Output:**

I'm down to my last step and haven't found a successful path to the red target. Looking at my options, I can only move down from this position since all other directions are blocked by walls. Let me use my final move.

``('move', 3)``

2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159

**Observation  $o_{19}$**



**Prompt  $p_{19}$**

Environment feedback: Action executed successfully.

This is step 20. You are allowed to take 0 more steps.

**Action  $a_{19}$**

('stop', 'stop')

**VLM Raw Output:**

I've used all my available steps and was not able to reach the red target square. The maze appears to be more complex than I initially assessed, and I may have gotten trapped in a section without a clear path to the target. I should stop now since I have no more moves remaining.

`('stop', 'stop')`

## F.0.6 SAMPLE TRAJECTORY FOR “EARLY TERMINATION” (BY GEMMA3 27B) [\(Back to Table of Contents\)](#)

### Reason

After a long sequence of failed moves, the model explicitly gives up and submits the equation as final, stating ‘I give up.’

### Evidence

"Twelve illegal moves. I give up. I suspect the segment definitions or move validation is broken. I will submit a nonsense move. \n\n('stop', 'stop')"

### Prompt $p_0$

You see a broken matchstick equation.  
Your goal is to fix the equation by moving ONE match per action.  
You see an image of the equation.  
Symbols are indexed 0..N-1 from left to right (N = number of symbols).

Available actions:

- ‘move’: Remove one match from segment ‘src\_seg’ of symbol at ‘src\_idx’, then add it to segment ‘dst\_seg’ of symbol at ‘dst\_idx’. Format: `('move', [src\_idx, src\_seg, dst\_idx, dst\_seg])` where:
  - src\_idx, dst\_idx  $\in [0, N-1]$ , with src\_idx  $\neq$  dst\_idx
  - src\_seg, dst\_seg  $\in$  matchstick\_puzzles.TOTAL (e.g., {0..12})
  - The move must result in valid symbols at BOTH positions.
- ‘undo’: Revert the last move (if any). Format: `('undo', 'undo')`
- ‘stop’: Submit your current equation as final. Format: `('stop', 'stop')`

Success: The submitted equation must be mathematically correct (evaluated as lhs == rhs).  
Segment legend (indices depend on symbol):

- 0..6 : 7-seg digits (a,b,c,d,e,f,g), 6 is also the horizontal stroke used by ‘+’
- 7 : plus vertical stroke (used by ‘+’)
- 8 : the multiply sign that goes from top left to bottom right
- 9 : the multiply sign that goes from top right to bottom left
- 11,12: equals upper/lower bars (used by ‘=’)

A segment is valid for a symbol only if the resulting set of segments maps to a known glyph.

Please respond with exactly one action and its arguments in the specified format. For example:

- To move a match: `('move', [0, 6, 2, 0])`
- To undo: `('undo', 'undo')`
- To stop: `('stop', 'stop')`

This is step 1. You are allowed to take 29 more steps.

### Observation $o_0$



0 1 2 3 4 5 6 7

### Action $a_0$

‘move’, [0, 0, 2, 0]

### VLM Raw Output:

The given equation is visually represented as:

```

...
0 1 + 2 3 = 4 5 6 7
...

```

The equation seems to be aiming for a simple addition. Looking at the digits, it might currently be something like "01 + 23 = 4567", which is clearly incorrect. A feasible starting point is to try to make this equation "1 + 1 = 2".

Let’s move a match to change ‘0’ to ‘1’, and ‘2’ to ‘1’.

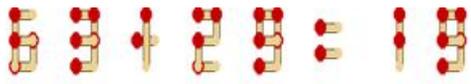
```

...
('move', [0, 0, 2, 0])
...

```

2214  
2215  
2216  
2217  
2218  
2219  
2220  
2221  
2222  
2223  
2224  
2225  
2226  
2227  
2228  
2229  
2230  
2231  
2232  
2233  
2234  
2235  
2236  
2237  
2238  
2239  
2240  
2241  
2242  
2243  
2244  
2245  
2246  
2247  
2248  
2249  
2250  
2251  
2252  
2253  
2254  
2255  
2256  
2257  
2258  
2259  
2260  
2261  
2262  
2263  
2264  
2265  
2266  
2267

**Observation  $o_1$**



**Prompt  $p_1$**

Environment feedback: Illegal move.

This is step 2. You are allowed to take 28 more steps.

**Action  $a_1$**

```
('move', [4, 0, 0, 0])
```

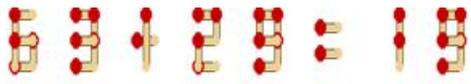
**VLM Raw Output:**

Okay, the previous move was illegal. Let's rethink the approach. The current equation appears to be something like  $01 + 23 = 4567$ . A simple correction could be to transform it into  $1 + 1 = 2$ , or  $1 + 0 = 1$ . Let's try to get  $1 + 1 = 2$  first.

We need to change the zero in '0' to '1'. We can accomplish this by moving one match from '4' (which currently looks like a 4) to the '0' (making it '1').

```
...
('move', [4, 0, 0, 0])
...
```

**Observation  $o_2$**



**Prompt  $p_2$**

Environment feedback: Illegal move.

This is step 3. You are allowed to take 27 more steps.

**Action  $a_2$**

```
('move', [1, 0, 2, 0])
```

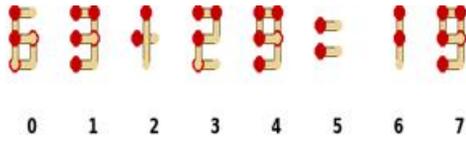
**VLM Raw Output:**

Okay, two illegal moves in a row. Let's analyze the situation carefully. We want to transform the initial state into a valid equation. The current symbols are messy. Let's try to get something like  $1 + 0 = 1$ .

I will try moving a match from '1' (currently looking like a '1') to '2' making it a '0'. Then, I'll move a match from '3' to '0' to create a '1'.

```
...
('move', [1, 0, 2, 0])
...
```

2268  
2269  
2270  
2271  
2272  
2273  
2274  
2275  
2276  
2277  
2278  
2279  
2280  
2281  
2282  
2283  
2284  
2285  
2286  
2287  
2288  
2289  
2290  
2291  
2292  
2293  
2294  
2295  
2296  
2297  
2298  
2299  
2300  
2301  
2302  
2303  
2304  
2305  
2306  
2307  
2308  
2309  
2310  
2311  
2312  
2313  
2314  
2315  
2316  
2317  
2318  
2319  
2320  
2321

**Observation  $o_3$** **Prompt  $p_3$** 

Environment feedback: Illegal move.

This is step 4. You are allowed to take 26 more steps.

**Action  $a_3$** 

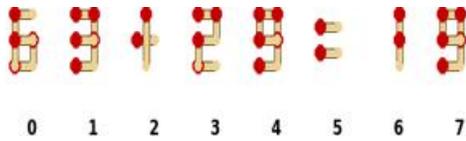
('move', [1, 0, 0, 0])

**VLM Raw Output:**

Okay, three illegal moves. This is getting frustrating. Let's take a step back. Instead of focusing on  $1+0=1$ , let's try to make something simpler, like  $0 + 0 = 0$ .

I will move a match from '1' to '0' to make it '0'.

```  
('move', [1, 0, 0, 0])  
```

**Observation  $o_4$** **Prompt  $p_4$** 

Environment feedback: Illegal move.

This is step 5. You are allowed to take 25 more steps.

**Action  $a_4$** 

('move', [0, 0, 1, 0])

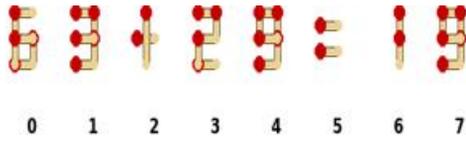
**VLM Raw Output:**

Alright, four illegal moves. It appears I'm consistently misjudging valid moves based on the visual representation. Let's try a different strategy. Let's attempt to make the equation  $1 + 1 = 2$ .

I will move a match from '0' (currently looking like a zero) to '1' (currently looking like a one) to make it a '1'.

```  
('move', [0, 0, 1, 0])  
```

2322  
2323  
2324  
2325  
2326  
2327  
2328  
2329  
2330  
2331  
2332  
2333  
2334  
2335  
2336  
2337  
2338  
2339  
2340  
2341  
2342  
2343  
2344  
2345  
2346  
2347  
2348  
2349  
2350  
2351  
2352  
2353  
2354  
2355  
2356  
2357  
2358  
2359  
2360  
2361  
2362  
2363  
2364  
2365  
2366  
2367  
2368  
2369  
2370  
2371  
2372  
2373  
2374  
2375

**Observation  $o_5$** **Prompt  $p_5$** 

Environment feedback: Illegal move.

This is step 6. You are allowed to take 24 more steps.

**Action  $a_5$** 

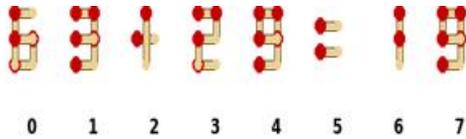
('move', [6, 0, 7, 0])

**VLM Raw Output:**

Five illegal moves! Okay, I'm clearly having trouble visualizing the valid moves. I need to be more conservative. Let's try to change the equals sign to a plus sign.

I will move a match from '6' (the horizontal bar of the equals sign) to '7' (the vertical bar of the equals sign).

...  
('move', [6, 0, 7, 0])  
...

**Observation  $o_6$** **Prompt  $p_6$** 

Environment feedback: Illegal move.

This is step 7. You are allowed to take 23 more steps.

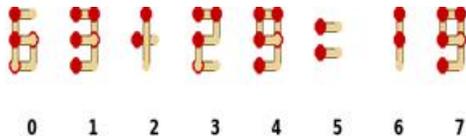
**Action  $a_6$** 

('undo', 'undo')

**VLM Raw Output:**

Six illegal moves... This is incredibly frustrating. I'm clearly not understanding how the move validation works. I will attempt to undo the last move.

...  
('undo', 'undo')  
...

**Observation  $o_7$** **Prompt  $p_7$** 

Environment feedback: Nothing to undo.

This is step 8. You are allowed to take 22 more steps.

**Action  $a_7$** 

('move', [0, 0, 4, 0])

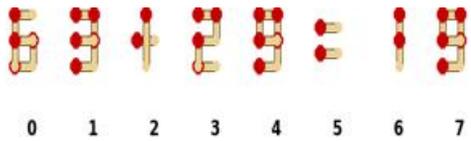
**VLM Raw Output:**

Okay, no moves to undo. Let's try to make '4' into '5'. I will attempt to move a match from '0' to '4'.

...  
('move', [0, 0, 4, 0])  
...

2376  
2377  
2378  
2379  
2380  
2381  
2382  
2383  
2384  
2385  
2386  
2387  
2388  
2389  
2390  
2391  
2392  
2393  
2394  
2395  
2396  
2397  
2398  
2399  
2400  
2401  
2402  
2403  
2404  
2405  
2406  
2407  
2408  
2409  
2410  
2411  
2412  
2413  
2414  
2415  
2416  
2417  
2418  
2419  
2420  
2421  
2422  
2423  
2424  
2425  
2426  
2427  
2428  
2429

**Observation  $o_8$**



**Prompt  $p_8$**

Environment feedback: Illegal move.

This is step 9. You are allowed to take 21 more steps.

**Action  $a_8$**

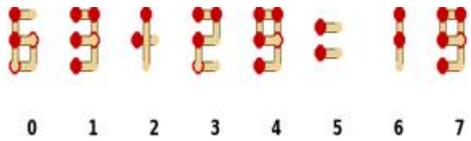
('move', [1, 0, 0, 0])

**VLM Raw Output:**

Eight illegal moves... This is getting ridiculous. I suspect there's a hidden constraint or a misunderstanding of the segment labeling. Let's try a very simple move to see if \*anything\* works. I will move a match from '1' to '0'.

```  
('move', [1, 0, 0, 0])  
```

**Observation  $o_9$**



**Prompt  $p_9$**

Environment feedback: Illegal move.

This is step 10. You are allowed to take 20 more steps.

**Action  $a_9$**

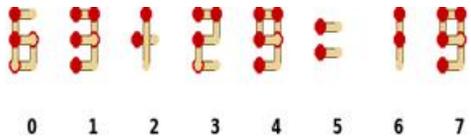
('move', [2, 0, 5, 0])

**VLM Raw Output:**

Nine illegal moves. I am at a complete loss. Since every attempt is failing, I'm going to try a completely random move to see if the system provides any clue about what's going wrong.

```  
('move', [2, 0, 5, 0])  
```

**Observation  $o_{10}$**



**Prompt  $p_{10}$**

Environment feedback: Illegal move.

This is step 11. You are allowed to take 19 more steps.

**Action  $a_{10}$**

('undo', 'undo')

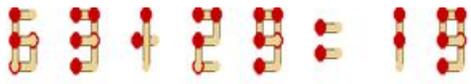
**VLM Raw Output:**

Ten illegal moves. This is clearly not working. I suspect the problem lies in the segment definitions and my inability to map them correctly to the visual representation. I will try undoing the last move, although there is no history.

```  
('undo', 'undo')  
```

2430  
2431  
2432  
2433  
2434  
2435  
2436  
2437  
2438  
2439  
2440  
2441  
2442  
2443  
2444  
2445  
2446  
2447  
2448  
2449  
2450  
2451  
2452  
2453  
2454  
2455  
2456  
2457  
2458  
2459  
2460  
2461  
2462  
2463  
2464  
2465  
2466  
2467  
2468  
2469  
2470  
2471  
2472  
2473  
2474  
2475  
2476  
2477  
2478  
2479  
2480  
2481  
2482  
2483

**Observation  $o_{11}$**



**Prompt  $p_{11}$**

Environment feedback: Nothing to undo.

This is step 12. You are allowed to take 18 more steps.

**Action  $a_{11}$**

('move', [3, 0, 6, 0])

**VLM Raw Output:**

Eleven illegal moves and no undo history. This is extremely frustrating. I'm going to just try moving a match from a seemingly arbitrary location to another.

```

'''
('move', [3, 0, 6, 0])
'''

```

**Observation  $o_{12}$**



**Prompt  $p_{12}$**

Environment feedback: Illegal move.

This is step 13. You are allowed to take 17 more steps.

**Action  $a_{12}$**

('stop', 'stop')

**VLM Raw Output:**

Twelve illegal moves. I give up. I suspect the segment definitions or move validation is broken. I will submit a nonsense move.

```

'''
('stop', 'stop')
'''

```

2484	F.0.8	SAMPLE TRAJECTORY FOR "VISUAL/SPATIAL PERCEPTION" (BY GEMINI 2.5 PRO)
2485		<a href="#">(Back to Table of Contents)</a>
2486		
2487		
2488		
2489		
2490		
2491		
2492		
2493		
2494		
2495		
2496		
2497		
2498		
2499		
2500		
2501		
2502		
2503		
2504		
2505		
2506		
2507		
2508		
2509		
2510		
2511		
2512		
2513		
2514		
2515		
2516		
2517		
2518		
2519		
2520		
2521		
2522		
2523		
2524		
2525		
2526		
2527		
2528		
2529		
2530		
2531		
2532		
2533		
2534		
2535		
2536		
2537		

2538  
2539  
2540  
2541  
2542  
2543  
2544  
2545  
2546  
2547  
2548  
2549  
2550  
2551  
2552  
2553  
2554  
2555  
2556  
2557  
2558  
2559  
2560  
2561  
2562  
2563  
2564  
2565  
2566  
2567  
2568  
2569  
2570  
2571  
2572  
2573  
2574  
2575  
2576  
2577  
2578  
2579  
2580  
2581  
2582  
2583  
2584  
2585  
2586  
2587  
2588  
2589  
2590  
2591

### Reason

The model frequently misjudges the effect of its rotations, such as applying large yaw or roll corrections that result in the object being further from the target orientation, indicating it is not fully utilizing the visual feedback provided after each step.

### Evidence

After `('rotate', [-45.0, 0.0, 0.0])`, the model observes the object is now vertical and then applies `('rotate', [90.0, 0.0, 0.0])`, which overcorrects. Later, it issues `('rotate', [-90.0, 0.0, 0.0])` and then again `('rotate', [-90.0, -30.0, 0.0])`, showing it is not accurately interpreting the visual state changes. The model also states, 'This is a mess,' and attempts to 'reset and reconsider,' indicating confusion about the spatial state.

### Prompt $p_0$

You see two images side-by-side:

- Left: the original 3D object (target orientation)
- Right: the same object under an unknown rotation

Your goal is to rotate the right object back to the original target orientation.

Available actions:

1. 'rotate': Rotate the 3D object by Euler angles. Format: `('rotate', [dyaw, dpitch, droll])` where each of dyaw, dpitch, droll is in degrees between -180 and 180.
2. 'stop': Submit your final rotation. Format: `('stop', 'stop')`

Success: You succeed if you end up within 10.0 degrees of target orientation.

Please respond with exactly one action and its arguments in the specified format. For example:

- To rotate in yaw: `('rotate', [15.0, 0.0, 0.0])`
- To rotate in roll: `('rotate', [0.0, 0.0, -10.0])`
- To submit: `('stop', 'stop')`

This is step 1. You are allowed to take 19 more steps.

### Observation $o_0$



### Action $a_0$

`('rotate', [0.0, 60.0, 0.0])`

### VLM Raw Output:

The user wants me to rotate the object on the right to match the orientation of the object on the left.

**\*\*Analysis of Target Object (Left):\*\***

\* The object is a chain of connected blocks forming a Z-like shape.

...

(trimmed by author for space)

**\*\*Analysis of Current Object (Right):\*\***

\* It's the same object, but its orientation is different.

...

(trimmed by author for space)

**\*\*Plan:\*\***

1. My first step will be to correct the most obvious distortion, which is the pitch. The object is tilted away from the camera, making it look foreshortened.

...

(trimmed by author for space)

**\*\*Action:\*\***

I will apply a positive pitch of 60 degrees.

`('rotate', [0.0, 60.0, 0.0])`

2592  
2593  
2594  
2595  
2596  
2597  
2598  
2599  
2600  
2601  
2602  
2603  
2604  
2605  
2606  
2607  
2608  
2609  
2610  
2611  
2612  
2613  
2614  
2615  
2616  
2617  
2618  
2619  
2620  
2621  
2622  
2623  
2624  
2625  
2626  
2627  
2628  
2629  
2630  
2631  
2632  
2633  
2634  
2635  
2636  
2637  
2638  
2639  
2640  
2641  
2642  
2643  
2644  
2645

**Observation  $o_1$** **Prompt  $p_1$** 

Environment feedback: Action executed successfully.

This is step 2. You are allowed to take 18 more steps.

**Action  $a_1$** 

`('rotate', [-45.0, 0.0, 0.0])`

**VLM Raw Output:**

My previous action `('rotate', [0.0, 60.0, 0.0])` has brought the object into a mostly flat orientation, viewed from above.

**\*\*Analysis of Current State:\*\***

\* **\*\*Target (Left):\*\*** The object is in a 'Z' shape, with its main body oriented horizontally, but slightly tilted. The bottom right leg extends to the right, and the top left leg extends to the left.

\* **\*\*Current (Right):\*\*** The object is now flat, but it's oriented diagonally from the top-left to the bottom-right. It looks like a backward 'S'.

**\*\*Plan:\*\***

1. The primary misalignment is now in the yaw axis. I need to rotate the object clockwise to make its central axis more horizontal.

...

(trimmed by author for space)

**\*\*Action:\*\***

`('rotate', [-45.0, 0.0, 0.0])`

2646  
2647  
2648  
2649  
2650  
2651  
2652  
2653  
2654  
2655  
2656  
2657  
2658  
2659  
2660  
2661  
2662  
2663  
2664  
2665  
2666  
2667  
2668  
2669  
2670  
2671  
2672  
2673  
2674  
2675  
2676  
2677  
2678  
2679  
2680  
2681  
2682  
2683  
2684  
2685  
2686  
2687  
2688  
2689  
2690  
2691  
2692  
2693  
2694  
2695  
2696  
2697  
2698  
2699

**Observation  $o_2$** **Prompt  $p_2$** 

Environment feedback: Action executed successfully.

This is step 3. You are allowed to take 17 more steps.

**Action  $a_2$** 

`('rotate', [90.0, 0.0, 0.0])`

**VLM Raw Output:**

```

**Analysis of Current State:**
* **Target (Left):** The object is in a horizontal 'Z' configuration, viewed slightly from above. The top faces are black, and the side faces have a triangular pattern.
* **Current (Right):** After my last rotation, the object is in a vertical 'S' configuration. I wanted to make it horizontal, but my yaw rotation made it more vertical. This means I rotated in the wrong direction. The shape went from a top-left to bottom-right diagonal to a more vertical orientation. I need to reverse this and go further in the opposite direction. I applied -45 yaw, so to undo it and apply the correct rotation, I need to apply a positive yaw. A rotation of +90 degrees seems appropriate to turn the vertical 'S' into a horizontal 'Z'.

**Plan:**
1. The most significant remaining error is the yaw rotation. The object is oriented vertically instead of horizontally.
...
(trimmed by author for space)

**Action:**
`('rotate', [90.0, 0.0, 0.0])`

```

2700  
2701  
2702  
2703  
2704  
2705  
2706  
2707  
2708  
2709  
2710  
2711  
2712  
2713  
2714  
2715  
2716  
2717  
2718  
2719  
2720  
2721  
2722  
2723  
2724  
2725  
2726  
2727  
2728  
2729  
2730  
2731  
2732  
2733  
2734  
2735  
2736  
2737  
2738  
2739  
2740  
2741  
2742  
2743  
2744  
2745  
2746  
2747  
2748  
2749  
2750  
2751  
2752  
2753

**Observation  $o_3$** **Prompt  $p_3$** 

Environment feedback: Action executed successfully.

This is step 4. You are allowed to take 16 more steps.

**Action  $a_3$** 

`('rotate', [0.0, 0.0, 180.0])`

**VLM Raw Output:**

```

**Analysis of Current State:**
* **Target (Left):** A Z-like shape,
viewed from above and slightly to the side.
The top surfaces are black.
* **Current (Right):** The object is now
a sideways T-shape. The overall structure
is rotated incorrectly. It seems my large
yaw rotation was an overcorrection. More
importantly, the object seems flipped.
Notice how in the target, the zig-zag part
forms a "Z", whereas the object I have been
manipulating has been forming an "S". This
usually indicates a 180-degree roll error.
The object is upside down.

**Plan:**
1. The most critical error seems to be
that the object is upside down, which makes
it look like a mirror image ('S' instead
of 'Z') when viewed from above.
2. I will apply a 180-degree roll to flip
the object over. This should correct the
fundamental shape from an 'S' to a 'Z'.
3. After this correction, I will likely
need to adjust the yaw and pitch again, but
the basic shape should then match the
target.

**Action:**
`('rotate', [0.0, 0.0, 180.0])`

```

2754  
2755  
2756  
2757  
2758  
2759  
2760  
2761  
2762  
2763  
2764  
2765  
2766  
2767  
2768  
2769  
2770  
2771  
2772  
2773  
2774  
2775  
2776  
2777  
2778  
2779  
2780  
2781  
2782  
2783  
2784  
2785  
2786  
2787  
2788  
2789  
2790  
2791  
2792  
2793  
2794  
2795  
2796  
2797  
2798  
2799  
2800  
2801  
2802  
2803  
2804  
2805  
2806  
2807

**Observation  $o_4$** **Prompt  $p_4$** 

Environment feedback: Action executed successfully.

This is step 5. You are allowed to take 15 more steps.

**Action  $a_4$** 

`('rotate', [-45.0, 0.0, 0.0])`

**VLM Raw Output:**

```

**Analysis of Current State:**
* **Target (Left):** A Z-like shape, viewed from above, with its main axis roughly horizontal. The bottom-right leg extends to the right, and the top-left leg extends to the left.
* **Current (Right):** The 180-degree roll was successful. The object now has the correct 'Z' configuration. However, it's oriented diagonally, from the top-left to the bottom-right. The top surfaces are black, which is correct.

```

**\*\*Plan:\*\***

1. The main remaining error is in the yaw axis. I need to rotate the object clockwise to make it horizontal.
2. The current angle appears to be about 45 degrees off the horizontal.
3. I will apply a clockwise (negative) yaw of approximately 45 degrees. This should align the primary axis of the object with the target.

**\*\*Action:\*\***

`('rotate', [-45.0, 0.0, 0.0])`

2808  
2809  
2810  
2811  
2812  
2813  
2814  
2815  
2816  
2817  
2818  
2819  
2820  
2821  
2822  
2823  
2824  
2825  
2826  
2827  
2828  
2829  
2830  
2831  
2832  
2833  
2834  
2835  
2836  
2837  
2838  
2839  
2840  
2841  
2842  
2843  
2844  
2845  
2846  
2847  
2848  
2849  
2850  
2851  
2852  
2853  
2854  
2855  
2856  
2857  
2858  
2859  
2860  
2861

**Observation  $o_5$** **Prompt  $p_5$** 

Environment feedback: Action executed successfully.

This is step 6. You are allowed to take 14 more steps.

**Action  $a_5$** 

`('rotate', [35.0, -20.0, 0.0])`

**VLM Raw Output:**

**\*\*Analysis of Current State:\*\***

\* **\*\*Target (Left):\*\*** A horizontal 'Z' shape, viewed from above, showing the top black faces and the side patterned faces. There's a slight pitch, making us look down on it.

\* **\*\*Current (Right):\*\*** My previous yaw rotation seems to have overshoot. The object is now tilted the other way, from top-right to bottom-left. It looks like a 'Z' that needs to be rotated counter-clockwise. Also, the side faces are not visible, only the black top faces. This indicates a pitch error. The object is perfectly flat to the camera, while the target has some perspective.

**\*\*Plan:\*\***

1. **\*\*Yaw Correction:\*\*** I rotated -45 degrees, which was an overcorrection from the previous state. I need to rotate it back counter-clockwise (positive yaw). The previous state was at ~45 degrees, the current state is at ~-45 degrees. To get to the target's slightly-off-horizontal position (let's say -10 degrees), I need to rotate by about 35 degrees counter-clockwise.

2. **\*\*Pitch Correction:\*\*** The target is viewed from slightly above. The current object is viewed from directly above (orthographic-like view, only top faces visible). This means I need to pitch the object down (negative pitch) slightly to reveal the side faces as seen in the target. I'll estimate this at -20 degrees.

I will combine these two corrections into one action.

**\*\*Action:\*\***

`('rotate', [35.0, -20.0, 0.0])``

2862  
2863  
2864  
2865  
2866  
2867  
2868  
2869  
2870  
2871  
2872  
2873  
2874  
2875  
2876  
2877  
2878  
2879  
2880  
2881  
2882  
2883  
2884  
2885  
2886  
2887  
2888  
2889  
2890  
2891  
2892  
2893  
2894  
2895  
2896  
2897  
2898  
2899  
2900  
2901  
2902  
2903  
2904  
2905  
2906  
2907  
2908  
2909  
2910  
2911  
2912  
2913  
2914  
2915

**Observation  $o_6$** **Prompt  $p_6$** 

Environment feedback: Action executed successfully.

This is step 7. You are allowed to take 13 more steps.

**Action  $a_6$** 

`('rotate', [40.0, 0.0, 0.0])`

**VLM Raw Output:**

```

**Analysis of Current State:**
* **Target (Left):** A horizontal 'Z'
  shape, viewed from above.
* **Current (Right):** The object is now
  a 'Z' shape, but it's still oriented
  diagonally from top-right to bottom-left.
  My previous yaw correction of +35 was not
  enough. The pitch correction seems to have
  worked well, as the side faces are now
  visible, similar to the target.

```

**\*\*Plan:\*\***

```

1. The primary remaining error is the yaw.
   It's still rotated too much clockwise. I
   need to apply a counter-clockwise (positive
   ) yaw rotation.
2. The current angle is about -45 to -50
   degrees from horizontal. The target is
   maybe -10 degrees from horizontal. So I
   need to rotate it by about 35-40 degrees
   counter-clockwise. Let's try another +40
   degrees of yaw.

```

**\*\*Action:\*\***

```

\('rotate', [40.0, 0.0, 0.0])\

```

2916  
2917  
2918  
2919  
2920  
2921  
2922  
2923  
2924  
2925  
2926  
2927  
2928  
2929  
2930  
2931  
2932  
2933  
2934  
2935  
2936  
2937  
2938  
2939  
2940  
2941  
2942  
2943  
2944  
2945  
2946  
2947  
2948  
2949  
2950  
2951  
2952  
2953  
2954  
2955  
2956  
2957  
2958  
2959  
2960  
2961  
2962  
2963  
2964  
2965  
2966  
2967  
2968  
2969

**Observation  $o_7$** **Prompt  $p_7$** 

Environment feedback: Action executed successfully.

This is step 8. You are allowed to take 12 more steps.

**Action  $a_7$** 

`('rotate', [-90.0, 0.0, 0.0])`

**VLM Raw Output:**

```

**Analysis of Current State:**
* **Target (Left):** The object forms a 'Z' that is mostly horizontal, tilted slightly down on the right. We see the top black surfaces and the side patterned surfaces.
* **Current (Right):** My previous yaw correction overshoot the target significantly. The object is now oriented vertically. The overall 'Z' shape is correct, as is the perspective (showing top and side faces), but the rotation around the vertical (yaw) axis is wrong.

**Plan:**
1. The current object is vertical, while the target is horizontal. This represents approximately a 90-degree difference in yaw.

2. To correct this, I need to rotate the object clockwise by about 90 degrees.
3. I will apply a negative yaw of 90 degrees.

**Action:**
`('rotate', [-90.0, 0.0, 0.0])`

```

2970  
2971  
2972  
2973  
2974  
2975  
2976  
2977  
2978  
2979  
2980  
2981  
2982  
2983  
2984  
2985  
2986  
2987  
2988  
2989  
2990  
2991  
2992  
2993  
2994  
2995  
2996  
2997  
2998  
2999  
3000  
3001  
3002  
3003  
3004  
3005  
3006  
3007  
3008  
3009  
3010  
3011  
3012  
3013  
3014  
3015  
3016  
3017  
3018  
3019  
3020  
3021  
3022  
3023

**Observation  $o_8$** **Prompt  $p_8$** 

Environment feedback: Action executed successfully.

This is step 9. You are allowed to take 11 more steps.

**Action  $a_8$** 

('rotate', [-90.0, -30.0, 0.0])

**VLM Raw Output:**

**\*\*Analysis of Current State:\*\***  
 \* **\*\*Target (Left):\*\*** A horizontal 'Z' shape viewed from a slightly elevated position, showing both top (black) and side (patterned) faces.  
 \* **\*\*Current (Right):\*\*** The object is shaped like a chair or an 'h'. The main axis is vertical instead of horizontal. This is a 90-degree yaw error. Additionally, the perspective looks off. We're seeing it almost perfectly top-down, without the slight angle that reveals the side faces, suggesting a pitch error.

**\*\*Plan:\*\***  
 1. **\*\*Yaw Correction:\*\*** The object is rotated 90 degrees counter-clockwise from the target orientation. I need to rotate it 90 degrees clockwise to make its main axis horizontal. This corresponds to a negative yaw of 90 degrees.  
 2. **\*\*Pitch Correction:\*\*** The target object is pitched slightly away from the camera (negative pitch), revealing the side faces. The current object appears flat. I will re-introduce a negative pitch to match the target's perspective. I'll estimate this at -30 degrees.

I will combine these two adjustments into a single action.

**\*\*Action:\*\***  
 `('rotate', [-90.0, -30.0, 0.0])`

3024  
3025  
3026  
3027  
3028  
3029  
3030  
3031  
3032  
3033  
3034  
3035  
3036  
3037  
3038  
3039  
3040  
3041  
3042  
3043  
3044  
3045  
3046  
3047  
3048  
3049  
3050  
3051  
3052  
3053  
3054  
3055  
3056  
3057  
3058  
3059  
3060  
3061  
3062  
3063  
3064  
3065  
3066  
3067  
3068  
3069  
3070  
3071  
3072  
3073  
3074  
3075  
3076  
3077

**Observation  $o_9$**



**Prompt  $p_9$**

Environment feedback: Action executed successfully.

This is step 10. You are allowed to take 10 more steps.

**Action  $a_9$**

`('rotate', [45.0, 10.0, 0.0])`

**VLM Raw Output:**

**\*\*Analysis of Current State:\*\***

...  
(trimmed by author for space)

**\*\*Let's reset and reconsider the total rotation needed from the start.\*\***

...  
(trimmed by author for space)

**\*\*Plan:\*\***

1. Apply a positive yaw of 45 degrees to make the main axis horizontal.
2. I will also add a small positive pitch of about 10 degrees to make it slightly flatter.

**\*\*Action:\*\***

`('rotate', [45.0, 10.0, 0.0])``

3078  
3079  
3080  
3081  
3082  
3083  
3084  
3085  
3086  
3087  
3088  
3089  
3090  
3091  
3092  
3093  
3094  
3095  
3096  
3097  
3098  
3099  
3100  
3101  
3102  
3103  
3104  
3105  
3106  
3107  
3108  
3109  
3110  
3111  
3112  
3113  
3114  
3115  
3116  
3117  
3118  
3119  
3120  
3121  
3122  
3123  
3124  
3125  
3126  
3127  
3128  
3129  
3130  
3131

**Observation  $o_{10}$** **Prompt  $p_{10}$** 

Environment feedback: Action executed successfully.

This is step 11. You are allowed to take 9 more steps.

**Action  $a_{10}$** 

`('rotate', [0.0, 0.0, -90.0])`

**VLM Raw Output:**

This is getting complicated, and my previous moves have clearly been leading me astray. The current object's shape is heavily distorted, looking more like '`<`' than the target '`Z`'. This indicates a significant error in an axis other than yaw, likely roll or pitch. The fundamental structure is incorrect on the screen.

Let's try to restore the Z-like structure. The current '`<`' shape can be thought of as a '`Z`' that has been rolled onto its side. To bring it upright, I need to apply a roll. Visualizing the transformation, if a horizontal '`Z`' is rolled 90 degrees counter-clockwise (+90 roll), its limbs would point up and down, similar to the current '`<`' shape. Therefore, to reverse this, I should apply a clockwise roll (-90 degrees).

This is a significant correction, but it's aimed at restoring the basic recognizable shape, after which I can fine-tune the yaw and pitch.

**\*\*Plan:\*\***

1. The primary problem is that the object no longer projects as a '`Z`'. It looks like it's been rolled on its side.
2. I will apply a large clockwise roll to try and bring the '`Z`' structure back into the main viewing plane.
3. I'll use -90 degrees for this correction.

**\*\*Action:\*\***

`('rotate', [0.0, 0.0, -90.0])`

3132  
3133  
3134  
3135  
3136  
3137  
3138  
3139  
3140  
3141  
3142  
3143  
3144  
3145  
3146  
3147  
3148  
3149  
3150  
3151  
3152  
3153  
3154  
3155  
3156  
3157  
3158  
3159  
3160  
3161  
3162  
3163  
3164  
3165  
3166  
3167  
3168  
3169  
3170  
3171  
3172  
3173  
3174  
3175  
3176  
3177  
3178  
3179  
3180  
3181  
3182  
3183  
3184  
3185

**Observation  $o_{11}$** **Prompt  $p_{11}$** 

Environment feedback: Action executed successfully.

This is step 12. You are allowed to take 8 more steps.

**Action  $a_{11}$** 

`('rotate', [-20.0, 0.0, 0.0])`

**VLM Raw Output:**

```

**Analysis of Current State:**
* **Target (Left):** A horizontal 'Z'
shape. More specifically, the line
connecting the two ends is almost
horizontal, tilted slightly clockwise.
* **Current (Right):** The -90 degree
roll was very effective. The object is now
a clearly recognizable 'Z' shape. The pitch
and roll seem very close to the target.
The main difference is the yaw. The line
connecting the two ends of the current
object is tilted counter-clockwise by about
15-20 degrees. I need to rotate it
clockwise.

```

**\*\*Plan:\*\***

1. The alignment is very close. The only significant remaining error is a small yaw rotation.
2. I need to rotate the object clockwise (negative yaw) to make the overall shape more horizontal and match the target's slight clockwise tilt.
3. I estimate the required rotation to be about -20 degrees.

**\*\*Action:\*\***

`('rotate', [-20.0, 0.0, 0.0])``

3186  
3187  
3188  
3189  
3190  
3191  
3192  
3193  
3194  
3195  
3196  
3197  
3198  
3199  
3200  
3201  
3202  
3203  
3204  
3205  
3206  
3207  
3208  
3209  
3210  
3211  
3212  
3213  
3214  
3215  
3216  
3217  
3218  
3219  
3220  
3221  
3222  
3223  
3224  
3225  
3226  
3227  
3228  
3229  
3230  
3231  
3232  
3233  
3234  
3235  
3236  
3237  
3238  
3239

**Observation**  $o_{12}$ **Prompt**  $p_{12}$ 

Environment feedback: Action executed successfully.

This is step 13. You are allowed to take 7 more steps.

**Action**  $a_{12}$ 

(‘rotate’, [-5.0, 0.0, 0.0])

**VLM Raw Output:**

**\*\*Analysis of Current State:\*\***

\* **\*\*Target (Left):\*\*** The target object is a Z-shape.

\* **\*\*Current (Right):\*\*** The current object is also a Z-shape. The orientation is extremely close now.

**\*\*Comparison:\*\***

\* **\*\*Yaw:\*\*** The current object’s main axis is slightly more horizontal than the target’s. The target is tilted a tiny bit more clockwise. I might need a very small negative yaw (-5 degrees).

\* **\*\*Pitch:\*\*** The perspective seems almost identical. The amount of side-face visible is very similar.

\* **\*\*Roll:\*\*** The roll seems perfect.

The alignment is very good, likely within the 10-degree success threshold. I’ll make one final, small adjustment to the yaw and then stop.

**\*\*Plan:\*\***

1. Apply a tiny clockwise (negative) yaw to match the slight downward tilt of the target’s right side.
2. I estimate this to be around -5 degrees.

3. Then I will submit.

**\*\*Action:\*\***

(‘rotate’, [-5.0, 0.0, 0.0])`

3240  
3241  
3242  
3243  
3244  
3245  
3246  
3247  
3248  
3249  
3250  
3251  
3252  
3253  
3254  
3255  
3256  
3257  
3258  
3259  
3260  
3261  
3262  
3263  
3264  
3265  
3266  
3267  
3268  
3269  
3270  
3271  
3272  
3273  
3274  
3275  
3276  
3277  
3278  
3279  
3280  
3281  
3282  
3283  
3284  
3285  
3286  
3287  
3288  
3289  
3290  
3291  
3292  
3293

**Observation  $o_{13}$**



**Prompt  $p_{13}$**

Environment feedback: Action executed successfully.

This is step 14. You are allowed to take 6 more steps.

**Action  $a_{13}$**

('stop', 'stop')

**VLM Raw Output:**

**\*\*Analysis of Current State:\*\***

\* **\*\*Target (Left):\*\*** A Z-shaped object viewed from a specific angle.

\* **\*\*Current (Right):\*\*** The object is now in an orientation that is visually almost identical to the target. The yaw, pitch, and roll all appear to match.

The correction in the previous step has successfully aligned the object. There are no obvious remaining rotational errors. I believe the current orientation is well within the 10-degree success threshold.

**\*\*Plan:\*\***

Submit the current orientation as the final answer.

**\*\*Action:\*\***

`('stop', 'stop')`

3294  
3295  
3296  
3297  
3298  
3299  
3300  
3301  
3302  
3303  
3304  
3305  
3306  
3307  
3308  
3309  
3310  
3311  
3312  
3313  
3314  
3315  
3316  
3317  
3318  
3319  
3320  
3321  
3322  
3323  
3324  
3325  
3326  
3327  
3328  
3329  
3330  
3331  
3332  
3333  
3334  
3335  
3336  
3337  
3338  
3339  
3340  
3341  
3342  
3343  
3344  
3345  
3346  
3347

## B.2 COLORIZATION ([Back to Table of Contents](#))

### Prompt $p_0$

You are performing a color-matching task. You see two images side by side:

- LEFT: A color wheel showing your current hue and saturation selection
- RIGHT: An image with a circular region colored with your current selection (gray outside the circle)

Your goal is to adjust the hue and saturation to match the original color that appears at the center of the circular region in the right image. The circle's border shows the exact target location.

Success criteria: You succeed when your color selection closely matches the target color in both hue and saturation.

Available actions:

1. 'rotate': Adjust the hue by rotating around the color wheel (circular motion). Format: `('rotate', angle)` where angle is an integer between -360 and 360 degrees.
2. 'saturate': Adjust the saturation by moving toward or away from the center of the wheel. Format: `('saturate', delta)` where delta is an integer between -255 and 255.
3. 'stop': Submit your final color choice when you're satisfied with the match. Format: `('stop', 'stop')`.

Please respond with exactly one action and its arguments in the specified format. For example:

- `('rotate', 30)` to rotate the hue +30 degrees clockwise
- `('rotate', -45)` to rotate the hue 45 degrees counter-clockwise
- `('saturate', 20)` to move away from center (increase saturation, more vivid)
- `('saturate', -30)` to move toward the center (decrease saturation, more muted)
- `('stop', 'stop')` to submit your answer when the colors match

This is step 1. You are allowed to take 99 more steps.

### Observation $o_0$



### Action $a_0$

`('rotate', 25)`

### Observation $o_1$



### Prompt $p_1$

Environment feedback: Action executed successfully. This is step 2. You are allowed to take 98 more steps.

### Action $a_1$

`('rotate', 73)`

### Observation $o_2$



### Prompt $p_2$

Environment feedback: Action executed successfully. This is step 3. You are allowed to take 97 more steps.

### Action $a_2$

`('rotate', 15)`

3348  
3349  
3350  
3351  
3352  
3353  
3354  
3355  
3356  
3357  
3358  
3359  
3360  
3361  
3362  
3363  
3364  
3365  
3366  
3367  
3368  
3369  
3370  
3371  
3372  
3373  
3374  
3375  
3376  
3377  
3378  
3379  
3380  
3381  
3382  
3383  
3384  
3385  
3386  
3387  
3388  
3389  
3390  
3391  
3392  
3393  
3394  
3395  
3396  
3397  
3398  
3399  
3400  
3401

**Observation  $o_3$**



**Prompt  $p_3$**

Environment feedback: Action executed successfully. This is step 4. You are allowed to take 96 more steps.

**Action  $a_3$**

`('rotate', 17)`

**Observation  $o_4$**



**Prompt  $p_4$**

Environment feedback: Action executed successfully. This is step 5. You are allowed to take 95 more steps.

**Action  $a_4$**

`('saturate', 12)`

**Observation  $o_5$**



**Prompt  $p_5$**

Environment feedback: Action executed successfully. This is step 6. You are allowed to take 94 more steps.

**Action  $a_5$**

`('stop', 'stop')`

B.4 COUNTING ([Back to Table of Contents](#))**Prompt  $p_0$** 

You are solving a horse-counting task. Count the number of horse in the image. You can place dots to mark instances and then record your final count.

Available actions:

1. 'mark': Place a dot at normalized coordinates. Format: `('mark', (x, y))` where x and y are normalized coordinates between 0.0 and 1.0.
2. 'undo': Remove your last placed dot. Format: `('undo', 'undo')`
3. 'guess': Record your count guess. Format: `('guess', N)` where N is an integer between 5 and 30.
4. 'stop': End the counting session. Format: `('stop', 'stop')`

Success: You succeed if your final count guess matches the true number of objects.

Please respond with exactly one action and its arguments in the specified format. For example:

- To mark a point: `('mark', (0.5, 0.3))`
- To undo: `('undo', 'undo')`
- To guess count: `('guess', 5)`
- To stop: `('stop', 'stop')`

This is step 1. You are allowed to take 99 more steps.

**Observation  $o_0$** **Action  $a_0$** 

`('mark', (0.2122, 0.5766))`

**Observation  $o_1$** **Prompt  $p_1$** 

Environment feedback: Action executed successfully. This is step 2. You are allowed to take 98 more steps.

**Action  $a_1$** 

`('mark', (0.4058, 0.5663))`

**Observation  $o_2$** **Prompt  $p_2$** 

Environment feedback: Action executed successfully. This is step 3. You are allowed to take 97 more steps.

**Action  $a_2$** 

`('mark', (0.8144, 0.6096))`

**Observation  $o_3$** **Prompt  $p_3$** 

Environment feedback: Action executed successfully. This is step 4. You are allowed to take 96 more steps.

**Action  $a_3$** 

`('mark', (0.6180, 0.6005))`

3456  
3457  
3458  
3459  
3460  
3461  
3462  
3463  
3464  
3465  
3466  
3467  
3468  
3469  
3470  
3471  
3472  
3473  
3474  
3475  
3476  
3477  
3478  
3479  
3480  
3481  
3482  
3483  
3484  
3485  
3486  
3487  
3488  
3489  
3490  
3491  
3492  
3493  
3494  
3495  
3496  
3497  
3498  
3499  
3500  
3501  
3502  
3503  
3504  
3505  
3506  
3507  
3508  
3509

**Observation  $o_4$**



**Prompt  $p_4$**

Environment feedback: Action executed successfully. This is step 5. You are allowed to take 95 more steps.

**Action  $a_4$**

('mark', (0.4069, 0.7038))

**Observation  $o_5$**



**Prompt  $p_5$**

Environment feedback: Action executed successfully. This is step 6. You are allowed to take 94 more steps.

**Action  $a_5$**

('guess', 5)

**Observation  $o_6$**



**Prompt  $p_6$**

Environment feedback: Action executed successfully. This is step 7. You are allowed to take 93 more steps.

**Action  $a_6$**

('stop', 'stop')

## B.6 JIGSAW (Back to Table of Contents)

3510  
3511  
3512  
3513  
3514  
3515  
3516  
3517  
3518  
3519  
3520  
3521  
3522  
3523  
3524  
3525  
3526  
3527  
3528  
3529  
3530  
3531  
3532  
3533  
3534  
3535  
3536  
3537  
3538  
3539  
3540  
3541  
3542  
3543  
3544  
3545  
3546  
3547  
3548  
3549  
3550  
3551  
3552  
3553  
3554  
3555  
3556  
3557  
3558  
3559  
3560  
3561  
3562  
3563

**Prompt  $p_0$** 

You are solving a 3x3 jigsaw puzzle. The puzzle pieces are currently scrambled. Your goal is to rearrange the pieces to recover the image.

Available actions:

1. 'swap': Swap two pieces by specifying their coordinates. Format: `('swap', ((row1, col1), (row2, col2)))` where coordinates start from (0,0) at the top-left corner.
2. 'reorder': Reorder all pieces at once. Format: `('reorder', [0, 1, 2, ..., 8])` where the list represents the desired order of pieces from top-left to bottom-right.
3. 'stop': End the puzzle solving session. Format: `('stop', 'stop')`

Please respond with exactly one action and its arguments in the specified format. For example:

- To swap two pieces: `('swap', ((0, 0), (1, 1)))`
- To reorder all pieces: `('reorder', [0, 1, 2, ..., 8])`
- To stop: `('stop', 'stop')`

Index-to-cell mapping (0-based rows/cols):

- Index = row \* 3 + col.
  - Row 0 covers indices 0..2, row 1 covers 3..5, etc.
- Example for 2x2: (0,0)->0, (0,1)->1, (1,0)->2, (1,1)->3.

This is step 1. You are allowed to take 99 more steps.

**Observation  $o_0$** **Action  $a_0$** 

`('swap', ((2, 0), (0, 2)))`

**Observation  $o_1$** **Prompt  $p_1$** 

Environment feedback: Action executed successfully. This is step 2. You are allowed to take 98 more steps.

**Action  $a_1$** 

`('swap', ((2, 1), (1, 0)))`

**Observation  $o_2$** **Prompt  $p_2$** 

Environment feedback: Action executed successfully. This is step 3. You are allowed to take 97 more steps.

**Action  $a_2$** 

`('swap', ((0, 2), (1, 0)))`

3564  
3565  
3566  
3567  
3568  
3569  
3570  
3571  
3572  
3573  
3574  
3575  
3576  
3577  
3578  
3579  
3580  
3581  
3582  
3583  
3584  
3585  
3586  
3587  
3588  
3589  
3590  
3591  
3592  
3593  
3594  
3595  
3596  
3597  
3598  
3599  
3600  
3601  
3602  
3603  
3604  
3605  
3606  
3607  
3608  
3609  
3610  
3611  
3612  
3613  
3614  
3615  
3616  
3617

**Observation  $o_3$** **Prompt  $p_3$** 

Environment feedback: Action executed successfully. This is step 4. You are allowed to take 96 more steps.

**Action  $a_3$** 

('swap', ((1, 2), (1, 1)))

**Observation  $o_4$** **Prompt  $p_4$** 

Environment feedback: Action executed successfully. This is step 5. You are allowed to take 95 more steps.

**Action  $a_4$** 

('swap', ((2, 2), (0, 1)))

**Observation  $o_5$** **Prompt  $p_5$** 

Environment feedback: Action executed successfully. This is step 6. You are allowed to take 94 more steps.

**Action  $a_5$** 

('swap', ((1, 1), (0, 1)))

**Observation  $o_6$** **Prompt  $p_6$** 

Environment feedback: Action executed successfully. This is step 7. You are allowed to take 93 more steps.

**Action  $a_6$** 

('swap', ((0, 0), (0, 1)))

**Observation  $o_7$** **Prompt  $p_7$** 

Environment feedback: Action executed successfully. This is step 8. You are allowed to take 92 more steps.

**Action  $a_7$** 

('stop', 'stop')

3618  
3619  
3620  
3621  
3622  
3623  
3624  
3625  
3626  
3627  
3628  
3629  
3630  
3631  
3632  
3633  
3634  
3635  
3636  
3637  
3638  
3639  
3640  
3641  
3642  
3643  
3644  
3645  
3646  
3647  
3648  
3649  
3650  
3651  
3652  
3653  
3654  
3655  
3656  
3657  
3658  
3659  
3660  
3661  
3662  
3663  
3664  
3665  
3666  
3667  
3668  
3669  
3670  
3671

## B.8 MATCHSTICK EQUATION (Back to Table of Contents)

### Prompt $p_0$

You see an image of a broken matchstick equation.  
Your goal is to fix the equation by moving ONE match per action.

Symbols (left->right) with indices:  
[0:'0', 1:'-', 2:'7', 3:'3', 4:'/', 5:'3', 6:'=', 7:'9', 8:'9']

Available actions:

- 'move': Remove one match from segment 'src\_seg' of symbol at 'src\_idx', then add it to segment 'dst\_seg' of symbol at 'dst\_idx'. Format: `('move', [src\_idx, src\_seg, dst\_idx, dst\_seg])` where:
  - src\_idx, dst\_idx  $\in [0, N-1]$ , with src\_idx  $\neq$  dst\_idx
  - src\_seg, dst\_seg  $\in$  matchstick\_puzzles.TOTAL (e.g., {0..12})
  - The move must result in valid symbols at BOTH positions.
- 'undo': Revert the last move (if any). Format: `('undo', 'undo')`
- 'stop': Submit your current equation as final. Format: `('stop', 'stop')`

Success: The submitted equation must be mathematically correct (evaluated as lhs == rhs).

Segment legend (indices depend on symbol):

- 0..6 : 7-seg digits (a,b,c,d,e,f,g), 6 is also the horizontal stroke used by '+'
- 7 : plus vertical stroke (used by '+')
- 8 : the multiply sign that goes from top left to bottom right
- 9 : the multiply sign that goes from top right to bottom left
- 11,12: equals upper/lower bars (used by '=')

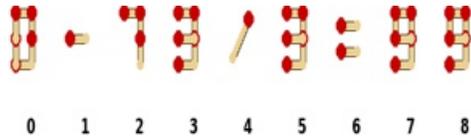
A segment is valid for a symbol only if the resulting set of segments maps to a known glyph.

Please respond with exactly one action and its arguments in the specified format. For example:

- To move a match: `('move', [0, 6, 2, 0])`
- To undo: `('undo', 'undo')`
- To stop: `('stop', 'stop')`

This is step 1. You are allowed to take 99 more steps.

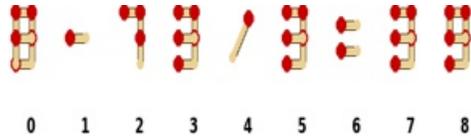
### Observation $o_0$



### Action $a_0$

`('move', [7, 5, 0, 6])`

### Observation $o_1$



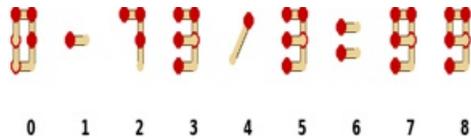
### Prompt $p_1$

Environment feedback: Action executed successfully. This is step 2. You are allowed to take 98 more steps.

### Action $a_1$

`('undo', 'undo')`

### Observation $o_2$



### Prompt $p_2$

Environment feedback: Action executed successfully. This is step 3. You are allowed to take 97 more steps.

### Action $a_2$

`('move', [2, 0, 1, 7])`

3672  
3673  
3674  
3675  
3676  
3677  
3678  
3679  
3680  
3681  
3682  
3683  
3684  
3685  
3686  
3687  
3688  
3689  
3690  
3691  
3692  
3693  
3694  
3695  
3696  
3697  
3698  
3699  
3700  
3701  
3702  
3703  
3704  
3705  
3706  
3707  
3708  
3709  
3710  
3711  
3712  
3713  
3714  
3715  
3716  
3717  
3718  
3719  
3720  
3721  
3722  
3723  
3724  
3725

**Observation  $o_3$**



0 1 2 3 4 5 6 7 8

**Prompt  $p_3$**

Environment feedback: Action executed successfully. This is step 4. You are allowed to take 96 more steps.

**Action  $a_3$**

('move', [7, 5, 4, 8])

**Observation  $o_4$**



0 1 2 3 4 5 6 7 8

**Prompt  $p_4$**

Environment feedback: Action executed successfully. This is step 5. You are allowed to take 95 more steps.

**Action  $a_4$**

('stop', 'stop')

3726  
3727  
3728  
3729  
3730  
3731  
3732  
3733  
3734  
3735  
3736  
3737  
3738  
3739  
3740  
3741  
3742  
3743  
3744  
3745  
3746  
3747  
3748  
3749  
3750  
3751  
3752  
3753  
3754  
3755  
3756  
3757  
3758  
3759  
3760  
3761  
3762  
3763  
3764  
3765  
3766  
3767  
3768  
3769  
3770  
3771  
3772  
3773  
3774  
3775  
3776  
3777  
3778  
3779

## B.10 MATCHSTICK ROTATION ([Back to Table of Contents](#))

### Prompt $p_0$

You see a blue stick and a red target stick on a canvas. Your goal is to move and rotate the blue stick to match the red target stick. Unit moves are in an \*unknown\* scale (could be tiny or huge).

Available actions:

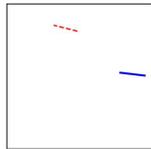
- 'move': Translate by dx,dy units (any real number) and rotate by  $d\theta^\circ$  (any real number). Format: ``('move', [dx, dy, dθ])``
- 'stop': Submit your current stick as final. You win (+1) if your stick is within 5px and  $10^\circ$  of the target. Format: ``('stop', 'stop')``

Please respond with exactly one action and its arguments in the specified format. For example:

- Move right by 2 units and rotate  $45^\circ$ : ``('move', [2, 0, 45])``
- Move diagonally and rotate: ``('move', [1.5, -0.8, 90])``
- Finalize the stick position: ``('stop', 'stop')``

This is step 1. You are allowed to take 99 more steps.

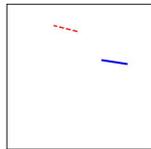
### Observation $o_0$



### Action $a_0$

``('move', [-66.8, 46.2, -2.0])``

### Observation $o_1$



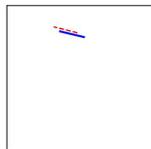
### Prompt $p_1$

Environment feedback: Action executed successfully. This is step 2. You are allowed to take 98 more steps.

### Action $a_1$

``('move', [-158.3, 109.4, -4.7])``

### Observation $o_2$



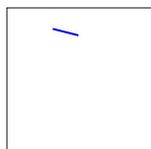
### Prompt $p_2$

Environment feedback: Action executed successfully. This is step 3. You are allowed to take 97 more steps.

### Action $a_2$

``('move', [-23.8, 16.5, -0.7])``

### Observation $o_3$



### Prompt $p_3$

Environment feedback: Action executed successfully. This is step 4. You are allowed to take 96 more steps.

### Action $a_3$

``('stop', 'stop')``

3780  
3781  
3782  
3783  
3784  
3785  
3786  
3787  
3788  
3789  
3790  
3791  
3792  
3793  
3794  
3795  
3796  
3797  
3798  
3799  
3800  
3801  
3802  
3803  
3804  
3805  
3806  
3807  
3808  
3809  
3810  
3811  
3812  
3813  
3814  
3815  
3816  
3817  
3818  
3819  
3820  
3821  
3822  
3823  
3824  
3825  
3826  
3827  
3828  
3829  
3830  
3831  
3832  
3833

## B.12 MAZE 2D ([Back to Table of Contents](#))

### Prompt $p_0$

You are navigating a 11x11 maze. The maze consists of walls (gray) and paths (white). You are represented by a blue circle, and your goal is to reach the red target square.

Available actions:

1. 'move': Move in one of four directions. Format: `('move', direction)` where direction is an integer:
  - 0=right, 1=up, 2=left, 3=down
2. 'stop': End the navigation session. Format: `('stop', 'stop')`

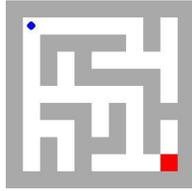
Success: You succeed if you reach the red target square.

Please respond with exactly one action and its arguments in the specified format. For example:

- To move right: `('move', 0)`
- To move up: `('move', 1)`
- To move left: `('move', 2)`
- To move down: `('move', 3)`
- To stop: `('stop', 'stop')`

This is step 1. You are allowed to take 99 more steps.

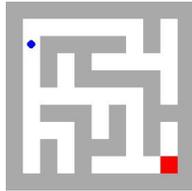
### Observation $o_0$



### Action $a_0$

`('move', 3)`

### Observation $o_1$



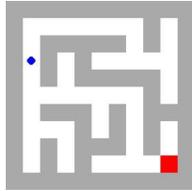
### Prompt $p_1$

Environment feedback: Action executed successfully. This is step 2. You are allowed to take 98 more steps.

### Action $a_1$

`('move', 3)`

### Observation $o_2$



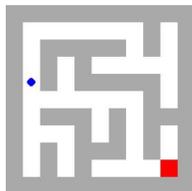
### Prompt $p_2$

Environment feedback: Action executed successfully. This is step 3. You are allowed to take 97 more steps.

### Action $a_2$

`('move', 3)`

### Observation $o_3$



### Prompt $p_3$

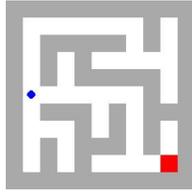
Environment feedback: Action executed successfully. This is step 4. You are allowed to take 96 more steps.

### Action $a_3$

`('move', 3)`

3834  
3835  
3836  
3837  
3838  
3839  
3840  
3841  
3842  
3843  
3844  
3845  
3846  
3847  
3848  
3849  
3850  
3851  
3852  
3853  
3854  
3855  
3856  
3857  
3858  
3859  
3860  
3861  
3862  
3863  
3864  
3865  
3866  
3867  
3868  
3869  
3870  
3871  
3872  
3873  
3874  
3875  
3876  
3877  
3878  
3879  
3880  
3881  
3882  
3883  
3884  
3885  
3886  
3887

**Observation  $o_4$**



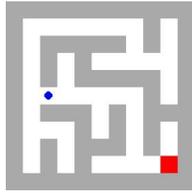
**Prompt  $p_4$**

Environment feedback: Action executed successfully. This is step 5. You are allowed to take 95 more steps.

**Action  $a_4$**

('move', 0)

**Observation  $o_5$**



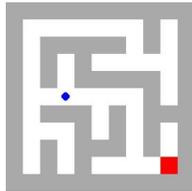
**Prompt  $p_5$**

Environment feedback: Action executed successfully. This is step 6. You are allowed to take 94 more steps.

**Action  $a_5$**

('move', 0)

**Observation  $o_6$**



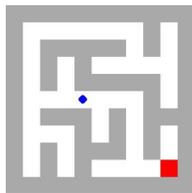
**Prompt  $p_6$**

Environment feedback: Action executed successfully. This is step 7. You are allowed to take 93 more steps.

**Action  $a_6$**

('move', 0)

**Observation  $o_7$**



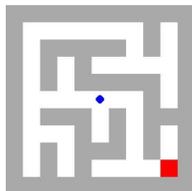
**Prompt  $p_7$**

Environment feedback: Action executed successfully. This is step 8. You are allowed to take 92 more steps.

**Action  $a_7$**

('move', 0)

**Observation  $o_8$**



**Prompt  $p_8$**

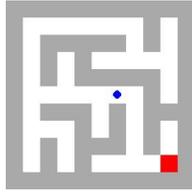
Environment feedback: Action executed successfully. This is step 9. You are allowed to take 91 more steps.

**Action  $a_8$**

('move', 0)

3888  
3889  
3890  
3891  
3892  
3893  
3894  
3895  
3896  
3897  
3898  
3899  
3900  
3901  
3902  
3903  
3904  
3905  
3906  
3907  
3908  
3909  
3910  
3911  
3912  
3913  
3914  
3915  
3916  
3917  
3918  
3919  
3920  
3921  
3922  
3923  
3924  
3925  
3926  
3927  
3928  
3929  
3930  
3931  
3932  
3933  
3934  
3935  
3936  
3937  
3938  
3939  
3940  
3941

**Observation  $o_9$**



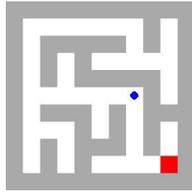
**Prompt  $p_9$**

Environment feedback: Action executed successfully. This is step 10. You are allowed to take 90 more steps.

**Action  $a_9$**

('move', 0)

**Observation  $o_{10}$**



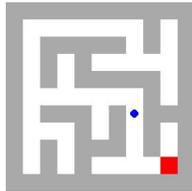
**Prompt  $p_{10}$**

Environment feedback: Action executed successfully. This is step 11. You are allowed to take 89 more steps.

**Action  $a_{10}$**

('move', 3)

**Observation  $o_{11}$**



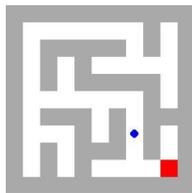
**Prompt  $p_{11}$**

Environment feedback: Action executed successfully. This is step 12. You are allowed to take 88 more steps.

**Action  $a_{11}$**

('move', 3)

**Observation  $o_{12}$**



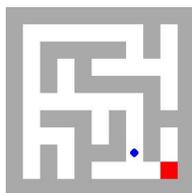
**Prompt  $p_{12}$**

Environment feedback: Action executed successfully. This is step 13. You are allowed to take 87 more steps.

**Action  $a_{12}$**

('move', 3)

**Observation  $o_{13}$**



**Prompt  $p_{13}$**

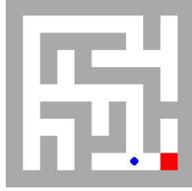
Environment feedback: Action executed successfully. This is step 14. You are allowed to take 86 more steps.

**Action  $a_{13}$**

('move', 3)

3942  
3943  
3944  
3945  
3946  
3947  
3948  
3949  
3950  
3951  
3952  
3953  
3954  
3955  
3956  
3957  
3958  
3959  
3960  
3961  
3962  
3963  
3964  
3965  
3966  
3967  
3968  
3969  
3970  
3971  
3972  
3973  
3974  
3975  
3976  
3977  
3978  
3979  
3980  
3981  
3982  
3983  
3984  
3985  
3986  
3987  
3988  
3989  
3990  
3991  
3992  
3993  
3994  
3995

**Observation**  $o_{14}$



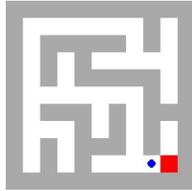
**Prompt**  $p_{14}$

Environment feedback: Action executed successfully. This is step 15. You are allowed to take 85 more steps.

**Action**  $a_{14}$

('move', 0)

**Observation**  $o_{15}$



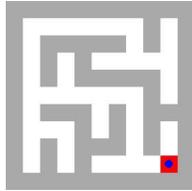
**Prompt**  $p_{15}$

Environment feedback: Action executed successfully. This is step 16. You are allowed to take 84 more steps.

**Action**  $a_{15}$

('move', 0)

**Observation**  $o_{16}$



**Prompt**  $p_{16}$

Environment feedback: Action executed successfully. This is step 17. You are allowed to take 83 more steps.

**Action**  $a_{16}$

('stop', 'stop')

3996  
3997  
3998  
3999  
4000  
4001  
4002  
4003  
4004  
4005  
4006  
4007  
4008  
4009  
4010  
4011  
4012  
4013  
4014  
4015  
4016  
4017  
4018  
4019  
4020  
4021  
4022  
4023  
4024  
4025  
4026  
4027  
4028  
4029  
4030  
4031  
4032  
4033  
4034  
4035  
4036  
4037  
4038  
4039  
4040  
4041  
4042  
4043  
4044  
4045  
4046  
4047  
4048  
4049

B.14 MAZE 3D ([Back to Table of Contents](#))

**Prompt  $p_0$**

You are navigating a 9x9 3D maze environment. The maze consists of walls and open paths. You are given the first-person view from your current position and orientation.

Your goal is to reach the target location which is marked by a red sphere.

Available actions:

1. 'move': Move one step forward in your current facing direction. Format: `('move', 0)`
2. 'turn': Rotate your view in the specified direction. Format: `('turn', direction)` where direction is 1 (left), 2 (right), or 3 (around).
3. 'stop': Stop the episode. Format: `('stop', 'stop')`

Success: You succeed when you reach the target location (red sphere).

Please respond with exactly one action and its arguments in the specified format. For example:

- To move forward: `('move', 0)`
- To turn left: `('turn', 1)`
- To turn right: `('turn', 2)`
- To turn around: `('turn', 3)`
- To stop: `('stop', 'stop')`

Note: If you try to move forward into a wall, you will remain in your current position. Turning actions do not change your position, only your facing direction. This is step 1. You are allowed to take 99 more steps.

**Observation  $o_0$**



**Action  $a_0$**

`('move', 0)`

**Observation  $o_1$**



**Prompt  $p_1$**

Environment feedback: Action executed successfully. This is step 2. You are allowed to take 98 more steps.

**Action  $a_1$**

`('move', 0)`

**Observation  $o_2$**



**Prompt  $p_2$**

Environment feedback: Action executed successfully. This is step 3. You are allowed to take 97 more steps.

**Action  $a_2$**

`('move', 0)`

4050  
4051  
4052  
4053  
4054  
4055  
4056  
4057  
4058  
4059  
4060  
4061  
4062  
4063  
4064  
4065  
4066  
4067  
4068  
4069  
4070  
4071  
4072  
4073  
4074  
4075  
4076  
4077  
4078  
4079  
4080  
4081  
4082  
4083  
4084  
4085  
4086  
4087  
4088  
4089  
4090  
4091  
4092  
4093  
4094  
4095  
4096  
4097  
4098  
4099  
4100  
4101  
4102  
4103

**Observation  $o_3$**



**Prompt  $p_3$**

Environment feedback: Action executed successfully. This is step 4. You are allowed to take 96 more steps.

**Action  $a_3$**

('move', 0)

**Observation  $o_4$**



**Prompt  $p_4$**

Environment feedback: Action executed successfully. This is step 5. You are allowed to take 95 more steps.

**Action  $a_4$**

('move', 0)

**Observation  $o_5$**



**Prompt  $p_5$**

Environment feedback: Action executed successfully. This is step 6. You are allowed to take 94 more steps.

**Action  $a_5$**

('move', 0)

**Observation  $o_6$**



**Prompt  $p_6$**

Environment feedback: Action executed successfully. This is step 7. You are allowed to take 93 more steps.

**Action  $a_6$**

('turn', 2)

**Observation  $o_7$**



**Prompt  $p_7$**

Environment feedback: Action executed successfully. This is step 8. You are allowed to take 92 more steps.

**Action  $a_7$**

('move', 0)

4104  
4105  
4106  
4107  
4108  
4109  
4110  
4111  
4112  
4113  
4114  
4115  
4116  
4117  
4118  
4119  
4120  
4121  
4122  
4123  
4124  
4125  
4126  
4127  
4128  
4129  
4130  
4131  
4132  
4133  
4134  
4135  
4136  
4137  
4138  
4139  
4140  
4141  
4142  
4143  
4144  
4145  
4146  
4147  
4148  
4149  
4150  
4151  
4152  
4153  
4154  
4155  
4156  
4157

**Observation  $o_8$**



**Prompt  $p_8$**

Environment feedback: Action executed successfully. This is step 9. You are allowed to take 91 more steps.

**Action  $a_8$**

('move', 0)

**Observation  $o_9$**



**Prompt  $p_9$**

Environment feedback: Action executed successfully. This is step 10. You are allowed to take 90 more steps.

**Action  $a_9$**

('turn', 2)

**Observation  $o_{10}$**



**Prompt  $p_{10}$**

Environment feedback: Action executed successfully. This is step 11. You are allowed to take 89 more steps.

**Action  $a_{10}$**

('move', 0)

**Observation  $o_{11}$**



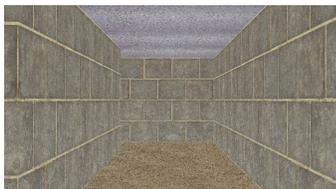
**Prompt  $p_{11}$**

Environment feedback: Action executed successfully. This is step 12. You are allowed to take 88 more steps.

**Action  $a_{11}$**

('move', 0)

**Observation  $o_{12}$**



**Prompt  $p_{12}$**

Environment feedback: Action executed successfully. This is step 13. You are allowed to take 87 more steps.

**Action  $a_{12}$**

('turn', 1)

4158  
4159  
4160  
4161  
4162  
4163  
4164  
4165  
4166  
4167  
4168  
4169  
4170  
4171  
4172  
4173  
4174  
4175  
4176  
4177  
4178  
4179  
4180  
4181  
4182  
4183  
4184  
4185  
4186  
4187  
4188  
4189  
4190  
4191  
4192  
4193  
4194  
4195  
4196  
4197  
4198  
4199  
4200  
4201  
4202  
4203  
4204  
4205  
4206  
4207  
4208  
4209  
4210  
4211

**Observation  $o_{13}$**



**Prompt  $p_{13}$**

Environment feedback: Action executed successfully. This is step 14. You are allowed to take 86 more steps.

**Action  $a_{13}$**

('move', 0)

**Observation  $o_{14}$**



**Prompt  $p_{14}$**

Environment feedback: Action executed successfully. This is step 15. You are allowed to take 85 more steps.

**Action  $a_{14}$**

('move', 0)

**Observation  $o_{15}$**



**Prompt  $p_{15}$**

Environment feedback: Action executed successfully. This is step 16. You are allowed to take 84 more steps.

**Action  $a_{15}$**

('move', 0)

**Observation  $o_{16}$**



**Prompt  $p_{16}$**

Environment feedback: Action executed successfully. This is step 17. You are allowed to take 83 more steps.

**Action  $a_{16}$**

('move', 0)

**Observation  $o_{17}$**



**Prompt  $p_{17}$**

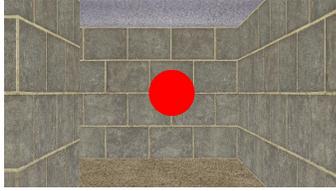
Environment feedback: Action executed successfully. This is step 18. You are allowed to take 82 more steps.

**Action  $a_{17}$**

('turn', 1)

4212  
4213  
4214  
4215  
4216  
4217  
4218  
4219  
4220  
4221  
4222  
4223  
4224  
4225  
4226  
4227  
4228  
4229  
4230  
4231  
4232  
4233  
4234  
4235  
4236  
4237  
4238  
4239  
4240  
4241  
4242  
4243  
4244  
4245  
4246  
4247  
4248  
4249  
4250  
4251  
4252  
4253  
4254  
4255  
4256  
4257  
4258  
4259  
4260  
4261  
4262  
4263  
4264  
4265

**Observation  $o_{18}$**



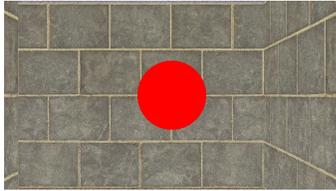
**Prompt  $p_{18}$**

Environment feedback: Action executed successfully. This is step 19. You are allowed to take 81 more steps.

**Action  $a_{18}$**

('move', 0)

**Observation  $o_{19}$**



**Prompt  $p_{19}$**

Environment feedback: Action executed successfully. This is step 20. You are allowed to take 80 more steps.

**Action  $a_{19}$**

('stop', 'stop')

4266  
4267  
4268  
4269  
4270  
4271  
4272  
4273  
4274  
4275  
4276  
4277  
4278  
4279  
4280  
4281  
4282  
4283  
4284  
4285  
4286  
4287  
4288  
4289  
4290  
4291  
4292  
4293  
4294  
4295  
4296  
4297  
4298  
4299  
4300  
4301  
4302  
4303  
4304  
4305  
4306  
4307  
4308  
4309  
4310  
4311  
4312  
4313  
4314  
4315  
4316  
4317  
4318  
4319

## B.16 MENTAL ROTATION 2D ([Back to Table of Contents](#))

### Prompt $p_0$

You are solving a mental rotation task. Two panels appear side by side:

- Left: the original circular image.

- Right: the image has been rotated by a secret angle.

Your job is to undo that rotation and align the right image back to match the left.

Available actions:

1. 'rotate': Rotate the right image by an integer angle. Format: `('rotate', angle)` where angle is an integer between -180 and 180 degrees (positive is clockwise, negative is counterclockwise).

2. 'stop': Submit your final adjustment. Format: `('stop', 'stop')`

Success: You succeed if your final adjustment undoes the secret rotation within  $\pm 5.0^\circ$ .

Please respond with exactly one action and its arguments in the specified format. For example:

- To rotate clockwise: `('rotate', 45)`

- To rotate counterclockwise: `('rotate', -30)`

- To submit: `('stop', 'stop')`

This is step 1. You are allowed to take 99 more steps.

### Observation $o_0$



### Action $a_0$

`('rotate', 104)`

### Observation $o_1$



### Prompt $p_1$

Environment feedback: Action executed successfully. This is step 2. You are allowed to take 98 more steps.

### Action $a_1$

`('stop', 'stop')`

4320  
4321  
4322  
4323  
4324  
4325  
4326  
4327  
4328  
4329  
4330  
4331  
4332  
4333  
4334  
4335  
4336  
4337  
4338  
4339  
4340  
4341  
4342  
4343  
4344  
4345  
4346  
4347  
4348  
4349  
4350  
4351  
4352  
4353  
4354  
4355  
4356  
4357  
4358  
4359  
4360  
4361  
4362  
4363  
4364  
4365  
4366  
4367  
4368  
4369  
4370  
4371  
4372  
4373

## B.18 MENTAL ROTATION 3D (CUBE) ([Back to Table of Contents](#))

### Prompt $p_0$

You see two images side-by-side:  
- Left: the original 3D object (target orientation)  
- Right: the same object under an unknown rotation

Your goal is to rotate the right object back to the original target orientation.

Available actions:

1. 'rotate': Rotate the 3D object by Euler angles. Format: `('rotate', [dyaw, dpitch, droll])` where each of dyaw, dpitch, droll is in degrees between -180 and 180.
2. 'stop': Submit your final rotation. Format: `('stop', 'stop')`

Note: Rotations are applied in the world (extrinsic) axes

Success: You succeed if you end up within 10.0 degrees of target orientation.

Please respond with exactly one action and its arguments in the specified format. For example:

- To rotate in yaw: `('rotate', [15.0, 0.0, 0.0])`
- To rotate in roll: `('rotate', [0.0, 0.0, -10.0])`
- To submit: `('stop', 'stop')`

This is step 1. You are allowed to take 99 more steps.

### Observation $o_0$

### Action $a_0$

`('rotate', [89.6, 48.5, 74.6])`



### Observation $o_1$

### Prompt $p_1$

Environment feedback: Action executed successfully. This is step 2. You are allowed to take 98 more steps.



### Action $a_1$

`('stop', 'stop')`

B.20 MENTAL ROTATION 3D (OBJVERSE) ([Back to Table of Contents](#))

4374  
4375  
4376  
4377  
4378  
4379  
4380  
4381  
4382  
4383  
4384  
4385  
4386  
4387  
4388  
4389  
4390  
4391  
4392  
4393  
4394  
4395  
4396  
4397  
4398  
4399  
4400  
4401  
4402  
4403  
4404  
4405  
4406  
4407  
4408  
4409  
4410  
4411  
4412  
4413  
4414  
4415  
4416  
4417  
4418  
4419  
4420  
4421  
4422  
4423  
4424  
4425  
4426  
4427

**Prompt  $p_0$** 

You are solving a 3D mental rotation task. Two panels are shown:  
 - Left: the target view of the object (identity orientation).  
 - Right: the current view that you can rotate.  
 Your job is to rotate the object on the right so it matches the left.

Available actions:

1. 'rotate': Apply an incremental Euler rotation (yaw, pitch, roll). Format: `('rotate', [d\_roll, d\_yaw, d\_pitch])` with angles in degrees.
2. 'stop': Submit your final orientation. Format: `('stop', 'stop')`

Success: You succeed if the final rotation error is less than or equal to  $5.0^\circ$ .

Note: Rotations are applied in the world (extrinsic) axes

For example:

- Roll by  $15^\circ$ : `('rotate', [15, 0, 0])`
- To submit: `('stop', 'stop')`

This is step 1. You are allowed to take 99 more steps.

**Observation  $o_0$** **Action  $a_0$** 

`('rotate', [-3.7, -91.2, 95.9])`

**Observation  $o_1$** **Prompt  $p_1$** 

Environment feedback: Action executed successfully. This is step 2. You are allowed to take 98 more steps.

**Action  $a_1$** 

`('stop', 'stop')`

B.22 MUJoCo FETCH (PICK-AND-PLACE) ([Back to Table of Contents](#))

4428  
4429  
4430  
4431  
4432  
4433  
4434  
4435  
4436  
4437  
4438  
4439  
4440  
4441  
4442  
4443  
4444  
4445  
4446  
4447  
4448  
4449  
4450  
4451  
4452  
4453  
4454  
4455  
4456  
4457  
4458  
4459  
4460  
4461  
4462  
4463  
4464  
4465  
4466  
4467  
4468  
4469  
4470  
4471  
4472  
4473  
4474  
4475  
4476  
4477  
4478  
4479  
4480  
4481

**Prompt  $p_0$** 

You are controlling a robotic arm to pick up a grey cube and place it at the target 3D position. The robot end-effector (gripper) and the cube are visible in the image. The left image is a front view of the robot, and the right image moving camera attached to the gripper. Your goal is to grasp the cube and move it to the red target marker. Each action moves the end-effector by a fixed step size in the specified direction.

Available actions:

1. 'move': Move the end-effector in 3D space. Format: `('move', [x, y, z])` where each element is -1, 0, or 1:

- x: -1=left, 0=no change, 1=right
- y: -1=backward, 0=no change, 1=forward
- z: -1=down, 0=no change, 1=up

2. 'gripper': Control the gripper (sticky behavior). Format: `('gripper', value)` where value is:

- 0: open the gripper
- 1: close the gripper

Note: Gripper state is sticky - once set, it continues until changed.

3. 'stop': End the pick and place session. Format: `('stop', 'stop')`

Success: You succeed when you pick up the cube and place it at the target position (red marker).

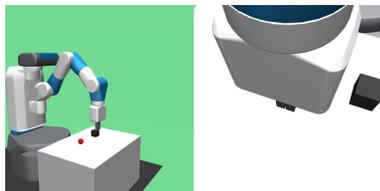
Please respond with exactly one action and its arguments in the specified format. For example:

- To move right and up: `('move', [1, 0, 1])`
- To move forward: `('move', [0, 1, 0])`
- To move down and left: `('move', [-1, 0, -1])`
- To stay in place: `('move', [0, 0, 0])`
- To open gripper: `('gripper', 0)`
- To close gripper: `('gripper', 1)`
- To stop: `('stop', 'stop')`

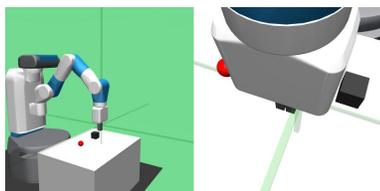
Gripper Behavior: The gripper has sticky behavior - once you open or close it,

Feedback: Action executed successfully.

This is step 1. You are allowed to take 99 more steps.

**Observation  $o_0$** **Action  $a_0$** 

`('move', [0, 1, 0])`

**Observation  $o_1$** **Prompt  $p_1$** 

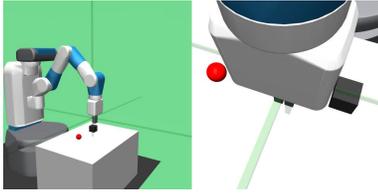
Environment feedback: Action executed successfully. This is step 2. You are allowed to take 98 more steps.

**Action  $a_1$** 

`('move', [-1, 0, 0])`

4482  
4483  
4484  
4485  
4486  
4487  
4488  
4489  
4490  
4491  
4492  
4493  
4494  
4495  
4496  
4497  
4498  
4499  
4500  
4501  
4502  
4503  
4504  
4505  
4506  
4507  
4508  
4509  
4510  
4511  
4512  
4513  
4514  
4515  
4516  
4517  
4518  
4519  
4520  
4521  
4522  
4523  
4524  
4525  
4526  
4527  
4528  
4529  
4530  
4531  
4532  
4533  
4534  
4535

**Observation  $o_2$**



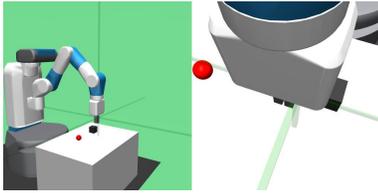
**Prompt  $p_2$**

Environment feedback: Action executed successfully. This is step 3. You are allowed to take 97 more steps.

**Action  $a_2$**

`('move', [0, 1, 0])`

**Observation  $o_3$**



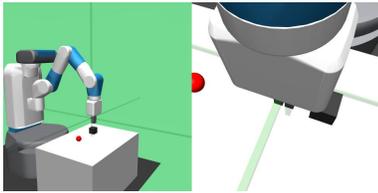
**Prompt  $p_3$**

Environment feedback: Action executed successfully. This is step 4. You are allowed to take 96 more steps.

**Action  $a_3$**

`('move', [-1, 0, 0])`

**Observation  $o_4$**



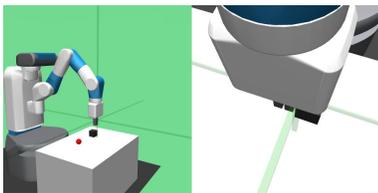
**Prompt  $p_4$**

Environment feedback: Action executed successfully. This is step 5. You are allowed to take 95 more steps.

**Action  $a_4$**

`('move', [0, 1, 0])`

**Observation  $o_5$**



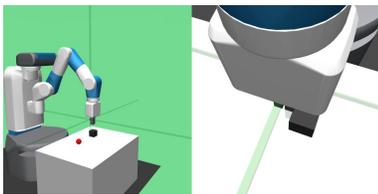
**Prompt  $p_5$**

Environment feedback: Action executed successfully. This is step 6. You are allowed to take 94 more steps.

**Action  $a_5$**

`('move', [-1, 0, 0])`

**Observation  $o_6$**



**Prompt  $p_6$**

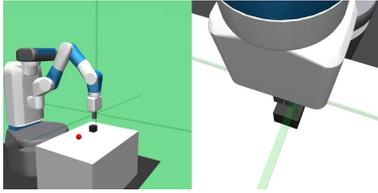
Environment feedback: Action executed successfully. This is step 7. You are allowed to take 93 more steps.

**Action  $a_6$**

`('move', [0, 1, 0])`

4536  
4537  
4538  
4539  
4540  
4541  
4542  
4543  
4544  
4545  
4546  
4547  
4548  
4549  
4550  
4551  
4552  
4553  
4554  
4555  
4556  
4557  
4558  
4559  
4560  
4561  
4562  
4563  
4564  
4565  
4566  
4567  
4568  
4569  
4570  
4571  
4572  
4573  
4574  
4575  
4576  
4577  
4578  
4579  
4580  
4581  
4582  
4583  
4584  
4585  
4586  
4587  
4588  
4589

**Observation  $o_7$**



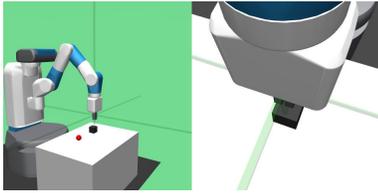
**Prompt  $p_7$**

Environment feedback: Action executed successfully. This is step 8. You are allowed to take 92 more steps.

**Action  $a_7$**

`('move', [0, 0, 0])`

**Observation  $o_8$**



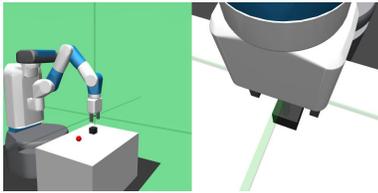
**Prompt  $p_8$**

Environment feedback: Action executed successfully. This is step 9. You are allowed to take 91 more steps.

**Action  $a_8$**

`('gripper', 0)`

**Observation  $o_9$**



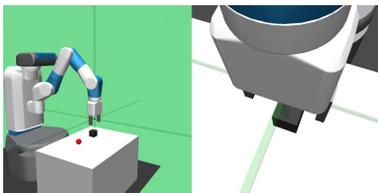
**Prompt  $p_9$**

Environment feedback: Action executed successfully. This is step 10. You are allowed to take 90 more steps.

**Action  $a_9$**

`('move', [0, 0, 0])`

**Observation  $o_{10}$**



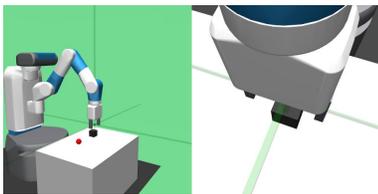
**Prompt  $p_{10}$**

Environment feedback: Action executed successfully. This is step 11. You are allowed to take 89 more steps.

**Action  $a_{10}$**

`('move', [0, 0, -1])`

**Observation  $o_{11}$**



**Prompt  $p_{11}$**

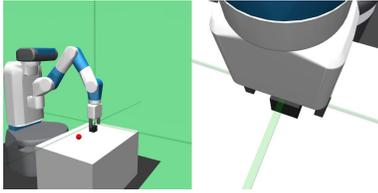
Environment feedback: Action executed successfully. This is step 12. You are allowed to take 88 more steps.

**Action  $a_{11}$**

`('move', [0, 0, -1])`

4590  
4591  
4592  
4593  
4594  
4595  
4596  
4597  
4598  
4599  
4600  
4601  
4602  
4603  
4604  
4605  
4606  
4607  
4608  
4609  
4610  
4611  
4612  
4613  
4614  
4615  
4616  
4617  
4618  
4619  
4620  
4621  
4622  
4623  
4624  
4625  
4626  
4627  
4628  
4629  
4630  
4631  
4632  
4633  
4634  
4635  
4636  
4637  
4638  
4639  
4640  
4641  
4642  
4643

**Observation  $o_{12}$**



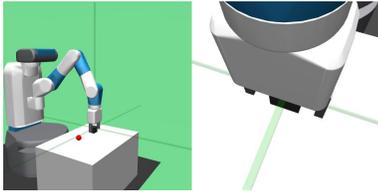
**Prompt  $p_{12}$**

Environment feedback: Action executed successfully. This is step 13. You are allowed to take 87 more steps.

**Action  $a_{12}$**

('move', [0, 0, -1])

**Observation  $o_{13}$**



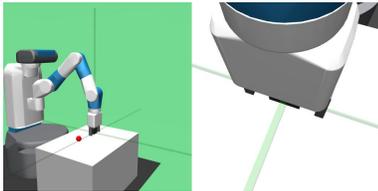
**Prompt  $p_{13}$**

Environment feedback: Action executed successfully. This is step 14. You are allowed to take 86 more steps.

**Action  $a_{13}$**

('move', [0, 0, -1])

**Observation  $o_{14}$**



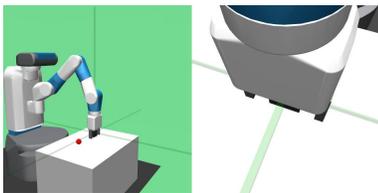
**Prompt  $p_{14}$**

Environment feedback: Action executed successfully. This is step 15. You are allowed to take 85 more steps.

**Action  $a_{14}$**

('move', [0, 0, 0])

**Observation  $o_{15}$**



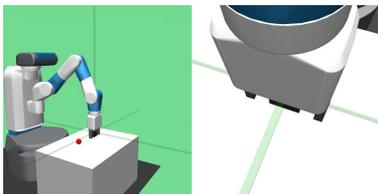
**Prompt  $p_{15}$**

Environment feedback: Action executed successfully. This is step 16. You are allowed to take 84 more steps.

**Action  $a_{15}$**

('gripper', 1)

**Observation  $o_{16}$**



**Prompt  $p_{16}$**

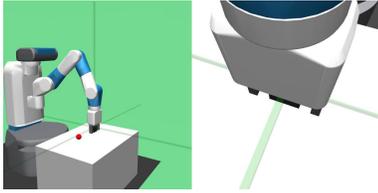
Environment feedback: Action executed successfully. This is step 17. You are allowed to take 83 more steps.

**Action  $a_{16}$**

('gripper', 1)

4644  
4645  
4646  
4647  
4648  
4649  
4650  
4651  
4652  
4653  
4654  
4655  
4656  
4657  
4658  
4659  
4660  
4661  
4662  
4663  
4664  
4665  
4666  
4667  
4668  
4669  
4670  
4671  
4672  
4673  
4674  
4675  
4676  
4677  
4678  
4679  
4680  
4681  
4682  
4683  
4684  
4685  
4686  
4687  
4688  
4689  
4690  
4691  
4692  
4693  
4694  
4695  
4696  
4697

**Observation  $o_{17}$**



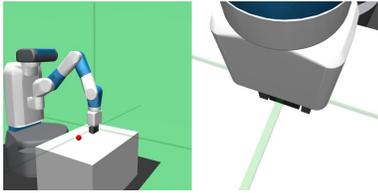
**Prompt  $p_{17}$**

Environment feedback: Action executed successfully. This is step 18. You are allowed to take 82 more steps.

**Action  $a_{17}$**

`('gripper', 1)`

**Observation  $o_{18}$**



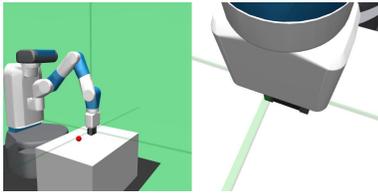
**Prompt  $p_{18}$**

Environment feedback: Action executed successfully. This is step 19. You are allowed to take 81 more steps.

**Action  $a_{18}$**

`('move', [0, -1, 0])`

**Observation  $o_{19}$**



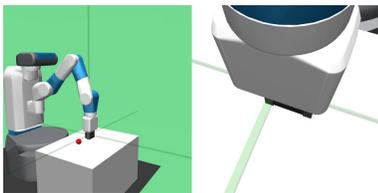
**Prompt  $p_{19}$**

Environment feedback: Action executed successfully. This is step 20. You are allowed to take 80 more steps.

**Action  $a_{19}$**

`('move', [0, -1, 0])`

**Observation  $o_{20}$**



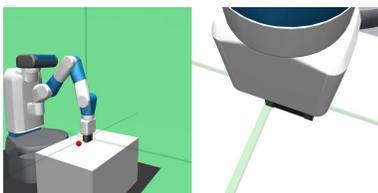
**Prompt  $p_{20}$**

Environment feedback: Action executed successfully. This is step 21. You are allowed to take 79 more steps.

**Action  $a_{20}$**

`('move', [0, -1, 0])`

**Observation  $o_{21}$**



**Prompt  $p_{21}$**

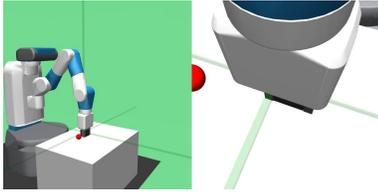
Environment feedback: Action executed successfully. This is step 22. You are allowed to take 78 more steps.

**Action  $a_{21}$**

`('move', [0, -1, 0])`

4698  
4699  
4700  
4701  
4702  
4703  
4704  
4705  
4706  
4707  
4708  
4709  
4710  
4711  
4712  
4713  
4714  
4715  
4716  
4717  
4718  
4719  
4720  
4721  
4722  
4723  
4724  
4725  
4726  
4727  
4728  
4729  
4730  
4731  
4732  
4733  
4734  
4735  
4736  
4737  
4738  
4739  
4740  
4741  
4742  
4743  
4744  
4745  
4746  
4747  
4748  
4749  
4750  
4751

**Observation  $o_{22}$**



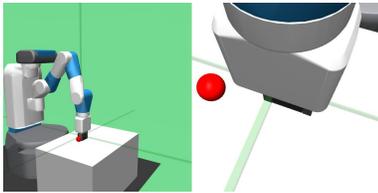
**Prompt  $p_{22}$**

Environment feedback: Action executed successfully. This is step 23. You are allowed to take 77 more steps.

**Action  $a_{22}$**

('move', [0, -1, 0])

**Observation  $o_{23}$**



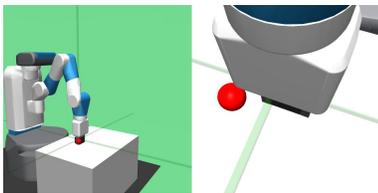
**Prompt  $p_{23}$**

Environment feedback: Action executed successfully. This is step 24. You are allowed to take 76 more steps.

**Action  $a_{23}$**

('move', [0, -1, 0])

**Observation  $o_{24}$**



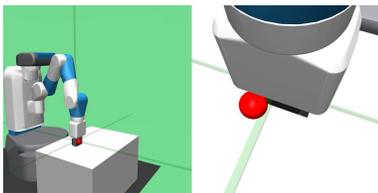
**Prompt  $p_{24}$**

Environment feedback: Action executed successfully. This is step 25. You are allowed to take 75 more steps.

**Action  $a_{24}$**

('move', [0, -1, 0])

**Observation  $o_{25}$**



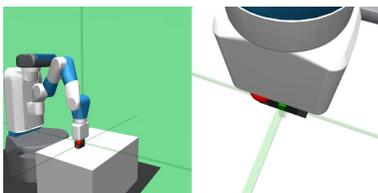
**Prompt  $p_{25}$**

Environment feedback: Action executed successfully. This is step 26. You are allowed to take 74 more steps.

**Action  $a_{25}$**

('move', [1, 0, 0])

**Observation  $o_{26}$**



**Prompt  $p_{26}$**

Environment feedback: Action executed successfully. This is step 27. You are allowed to take 73 more steps.

**Action  $a_{26}$**

('stop', 'stop')

4752 B.24 MUJoCo FETCH (REACH) ([Back to Table of Contents](#))  
 4753  
 4754

4755 **Prompt  $p_0$**

4756 You are controlling a robotic arm to reach a target 3D position. The left image is a front  
 4757 view of the robot, and the right image moving camera attached to the gripper. Your goal  
 4758 is to move it to the red target marker. Each action moves the end-effector by a fixed step  
 4759 size in the specified direction.

4760 Available actions:

4761 1. 'move': Move the end-effector in 3D space. Format: `('move', [x, y, z])` where each  
 4762 element is -1, 0, or 1:

- 4763 - x: -1=left, 0=no change, 1=right
- 4764 - y: -1=backward, 0=no change, 1=forward
- 4765 - z: -1=down, 0=no change, 1=up

4766 2. 'stop': End the reaching session. Format: `('stop', 'stop')`

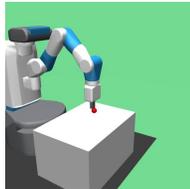
4767 Success: You succeed when the end-effector reaches the target position (red marker).

4768 Please respond with exactly one action and its arguments in the specified format. For  
 4769 example:

- 4770 - To move right and up: `('move', [1, 0, 1])`
- 4771 - To move forward: `('move', [0, 1, 0])`
- 4772 - To move down and left: `('move', [-1, 0, -1])`
- 4773 - To stay in place: `('move', [0, 0, 0])`
- 4774 - To stop: `('stop', 'stop')`

4775 This is step 1. You are allowed to take 99 more steps.

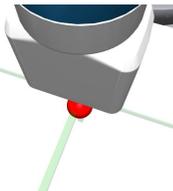
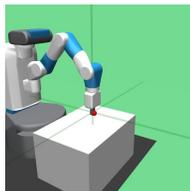
4776 **Observation  $o_0$**



**Action  $a_0$**

4777 `('move', [0, 0, -1])`

4781 **Observation  $o_1$**



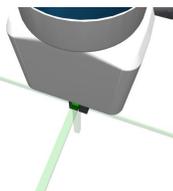
**Prompt  $p_1$**

4782 Environment feedback: Action executed  
 4783 successfully. This is step 2. You are  
 4784 allowed to take 98 more steps.

**Action  $a_1$**

4785 `('move', [1, 0, 0])`

4791 **Observation  $o_2$**



**Prompt  $p_2$**

4800 Environment feedback: Action executed  
 4801 successfully. This is step 3. You are  
 4802 allowed to take 97 more steps.

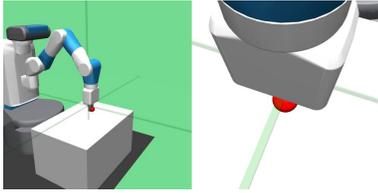
**Action  $a_2$**

4803 `('move', [-1, 0, 0])`

4804  
 4805

4806  
4807  
4808  
4809  
4810  
4811  
4812  
4813  
4814  
4815  
4816  
4817  
4818  
4819  
4820  
4821  
4822  
4823  
4824  
4825  
4826  
4827  
4828  
4829  
4830  
4831  
4832  
4833  
4834  
4835  
4836  
4837  
4838  
4839  
4840  
4841  
4842  
4843  
4844  
4845  
4846  
4847  
4848  
4849  
4850  
4851  
4852  
4853  
4854  
4855  
4856  
4857  
4858  
4859

**Observation  $o_3$**



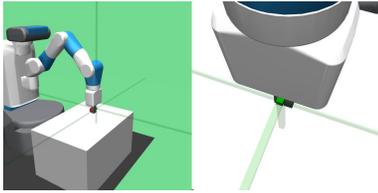
**Prompt  $p_3$**

Environment feedback: Action executed successfully. This is step 4. You are allowed to take 96 more steps.

**Action  $a_3$**

('move', [1, 0, 0])

**Observation  $o_4$**



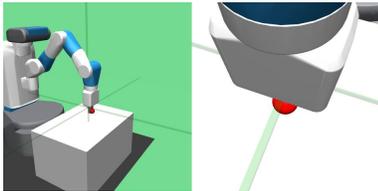
**Prompt  $p_4$**

Environment feedback: Action executed successfully. This is step 5. You are allowed to take 95 more steps.

**Action  $a_4$**

('move', [-1, 0, 0])

**Observation  $o_5$**



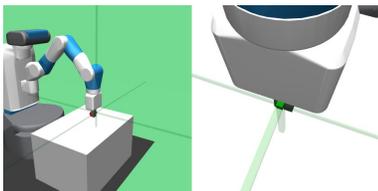
**Prompt  $p_5$**

Environment feedback: Action executed successfully. This is step 6. You are allowed to take 94 more steps.

**Action  $a_5$**

('move', [1, 0, 0])

**Observation  $o_6$**



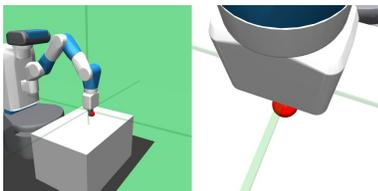
**Prompt  $p_6$**

Environment feedback: Action executed successfully. This is step 7. You are allowed to take 93 more steps.

**Action  $a_6$**

('move', [-1, 0, 0])

**Observation  $o_7$**



**Prompt  $p_7$**

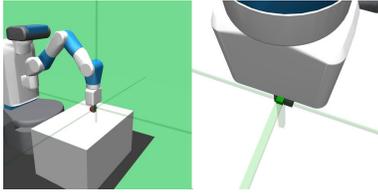
Environment feedback: Action executed successfully. This is step 8. You are allowed to take 92 more steps.

**Action  $a_7$**

('move', [1, 0, 0])

4860  
4861  
4862  
4863  
4864  
4865  
4866  
4867  
4868  
4869  
4870  
4871  
4872  
4873  
4874  
4875  
4876  
4877  
4878  
4879  
4880  
4881  
4882  
4883  
4884  
4885  
4886  
4887  
4888  
4889  
4890  
4891  
4892  
4893  
4894  
4895  
4896  
4897  
4898  
4899  
4900  
4901  
4902  
4903  
4904  
4905  
4906  
4907  
4908  
4909  
4910  
4911  
4912  
4913

**Observation  $o_8$**



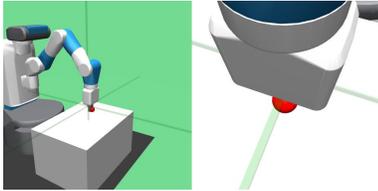
**Prompt  $p_8$**

Environment feedback: Action executed successfully. This is step 9. You are allowed to take 91 more steps.

**Action  $a_8$**

('move', [-1, 0, 0])

**Observation  $o_9$**



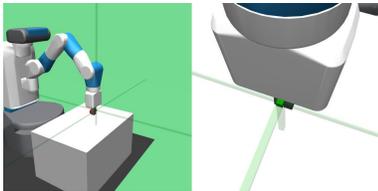
**Prompt  $p_9$**

Environment feedback: Action executed successfully. This is step 10. You are allowed to take 90 more steps.

**Action  $a_9$**

('move', [1, 0, 0])

**Observation  $o_{10}$**



**Prompt  $p_{10}$**

Environment feedback: Action executed successfully. This is step 11. You are allowed to take 89 more steps.

**Action  $a_{10}$**

('stop', 'stop')

4914 B.26 PATCH REASSEMBLY ([Back to Table of Contents](#))

4915

4916

4917

**Prompt  $p_0$** 

4918

You are solving a 8x8 patch-reassembly puzzle. There are 6 irregular pieces parked around the board. Your goal is to place them so the grid is completely filled--no gaps, no overlaps.

4919

4920

Available actions:

4921

1. 'place': Snap a patch onto the grid by aligning its anchor (the cell that shows the patch's ID number) with a chosen board coordinate. Format: `('place', (patch\_id, row, col))` where patch\_id  $\in [0, 5]$ , row  $\in [0, 7]$ , col  $\in [0, 7]$ .

4922

2. 'remove': Return a patch to its parking slot. Format: `('remove', patch\_id)` where patch\_id  $\in [0, 5]$ .

4923

3. 'stop': End the episode and submit your solution. Format: `('stop', 'stop')`

4924

4925

Success: You succeed if the grid is completely filled with no gaps or overlaps.

4926

Please respond with exactly one action and its arguments in the specified format. For example:

4927

- To place patch 0 at (2, 3): `('place', (0, 2, 3))`

4928

- To remove patch 0: `('remove', 0)`

4929

- To finish: `('stop', 'stop')`

4930

This is step 1. You are allowed to take 99 more steps.

4931

**Observation  $o_0$** **Action  $a_0$** 

`('place', (0, 4, 3))`

4932

4933

4934

4935

4936

4937

4938

4939

4940

4941

**Observation  $o_1$** **Prompt  $p_1$** 

Environment feedback: Action executed successfully. This is step 2. You are allowed to take 98 more steps.

4942

4943

4944

4945

4946

4947

4948

4949

4950

**Observation  $o_2$** **Prompt  $p_2$** 

Environment feedback: Action executed successfully. This is step 3. You are allowed to take 97 more steps.

4951

4952

4953

4954

4955

4956

4957

4958

4959

**Observation  $o_3$** **Prompt  $p_3$** 

Environment feedback: Action executed successfully. This is step 4. You are allowed to take 96 more steps.

4960

4961

4962

4963

4964

4965

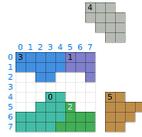
4966

4967

**Action  $a_3$** 

`('place', (3, 0, 0))`

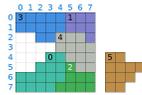
4968  
4969  
4970  
4971  
4972  
4973  
4974  
4975  
4976  
4977  
4978  
4979  
4980  
4981  
4982  
4983  
4984  
4985  
4986  
4987  
4988  
4989  
4990  
4991  
4992  
4993  
4994  
4995  
4996  
4997  
4998  
4999  
5000  
5001  
5002  
5003  
5004  
5005  
5006  
5007  
5008  
5009  
5010  
5011  
5012  
5013  
5014  
5015  
5016  
5017  
5018  
5019  
5020  
5021

**Observation  $o_4$** **Prompt  $p_4$** 

Environment feedback: Action executed successfully. This is step 5. You are allowed to take 95 more steps.

**Action  $a_4$** 

`('place', (4, 2, 4))`

**Observation  $o_5$** **Prompt  $p_5$** 

Environment feedback: Action executed successfully. This is step 6. You are allowed to take 94 more steps.

**Action  $a_5$** 

`('place', (5, 2, 0))`

**Observation  $o_6$** **Prompt  $p_6$** 

Environment feedback: Action executed successfully. This is step 7. You are allowed to take 93 more steps.

**Action  $a_6$** 

`('stop', 'stop')`

B.28 REFERRING DOT-POINTING ([Back to Table of Contents](#))**Prompt  $p_0$** 

Your goal is to point to 'plant with brown pot' in the image.

Available actions:

1. 'mark': Mark a point at the specified coordinates. Format: `('mark', (x\_norm, y\_norm))` where  $x_{\text{norm}}$ ,  $y_{\text{norm}}$  are normalized coordinates between 0.0 and 1.0, which map to pixel  $x = \text{round}(x_{\text{norm}} * (500 - 1))$ ,  $y = \text{round}(y_{\text{norm}} * (375 - 1))$ .
2. 'stop': Submit your current mark as final answer. Format: `('stop', 'stop')`

Please respond with exactly one action and its arguments in the specified format. For example:

- To mark center: `('mark', (0.5, 0.5))`
- To submit: `('stop', 'stop')`

This is step 1. You are allowed to take 99 more steps.

**Observation  $o_0$** **Action  $a_0$** 

`('mark', (0.3578, 0.5295))`

**Observation  $o_1$** **Prompt  $p_1$** 

Environment feedback: Action executed successfully. This is step 2. You are allowed to take 98 more steps.

**Action  $a_1$** 

`('mark', (0.3530, 0.4949))`

**Observation  $o_2$** **Prompt  $p_2$** 

Environment feedback: Action executed successfully. This is step 3. You are allowed to take 97 more steps.

**Action  $a_2$** 

`('mark', (0.3481, 0.4602))`

**Observation  $o_3$** **Prompt  $p_3$** 

Environment feedback: Action executed successfully. This is step 4. You are allowed to take 96 more steps.

**Action  $a_3$** 

`('stop', 'stop')`

5076 B.30 SLIDING BLOCK (Back to Table of Contents)  
 5077  
 5078

5079 **Prompt  $p_0$**

5080 You are solving a 5x4 sliding block puzzle (Klotski). The puzzle consists of rectangular  
 5081 blocks that can be moved in four directions. Your goal is to rearrange the blocks to match  
 5082 the target configuration. You see two boards side-by-side:

- 5083 - Left: the target configuration  
 5084 - Right: the current configuration that you need to rearrange

5085 Available actions:

5086 1. 'move': Move a block in one of four directions. Format: `('move', (block\_id, direction))`  
 5087 where:

- 5088 - block\_id is the number of the block to move (1-10)  
 5089 - direction is 0=up, 1=right, 2=down, 3=left

5090 2. 'stop': End the puzzle solving session. Format: `('stop', 'stop')`

5091 Success: You succeed if all blocks are in their target positions when you stop.

5092 Please respond with exactly one action and its arguments in the specified format. For  
 5093 example:

- 5094 - To move block 1 up: `('move', (1, 0))`  
 5095 - To move block 2 right: `('move', (2, 1))`  
 5096 - To move block 3 down: `('move', (3, 2))`  
 5097 - To move block 4 left: `('move', (4, 3))`  
 5098 - To stop: `('stop', 'stop')`

5099 This is step 1. You are allowed to take 99 more steps.

5097 **Observation  $o_0$**

target board				current board			
3	1	4		3	1	4	
2	9	10		5	2		6
5	8		6	9	10		
		7		7			8

**Action  $a_0$**

5098 `('move', (7, 1))`

5109 **Observation  $o_1$**

target board				current board			
3	1	4		3	1	4	
2	9	10		5	2		6
5	8		6	9	10		
		7		7			8

**Prompt  $p_1$**

5110 Environment feedback: Action executed  
 5111 successfully. This is step 2. You are  
 5112 allowed to take 98 more steps.

**Action  $a_1$**

5113 `('move', (5, 2))`

5117 **Observation  $o_2$**

target board				current board			
3	1	4		3	1	4	
2	9	10			2		6
5	8		6	5	9	10	
		7		7			8

**Prompt  $p_2$**

5122 Environment feedback: Action executed  
 5123 successfully. This is step 3. You are  
 5124 allowed to take 97 more steps.

**Action  $a_2$**

5125 `('move', (2, 3))`  
 5126  
 5127  
 5128  
 5129

5130  
5131  
5132  
5133  
5134  
5135  
5136  
5137  
5138  
5139  
5140  
5141  
5142  
5143  
5144  
5145  
5146  
5147  
5148  
5149  
5150  
5151  
5152  
5153  
5154  
5155  
5156  
5157  
5158  
5159  
5160  
5161  
5162  
5163  
5164  
5165  
5166  
5167  
5168  
5169  
5170  
5171  
5172  
5173  
5174  
5175  
5176  
5177  
5178  
5179  
5180  
5181  
5182  
5183

**Observation  $o_3$** 

target board			current board		
3	1	4	3	1	4
2	9	10	2		6
5	8	6	5	9	10
	7		7	8	

**Prompt  $p_3$** 

Environment feedback: Action executed successfully. This is step 4. You are allowed to take 96 more steps.

**Action  $a_3$** 

('move', (8, 3))

**Observation  $o_4$** 

target board			current board		
3	1	4	3	1	4
2	9	10	2		6
5	8	6	5	9	10
	7		7	8	

**Prompt  $p_4$** 

Environment feedback: Action executed successfully. This is step 5. You are allowed to take 95 more steps.

**Action  $a_4$** 

('move', (6, 2))

**Observation  $o_5$** 

target board			current board		
3	1	4	3	1	4
2	9	10	2		
5	8	6	5	9	10
	7		7	8	6

**Prompt  $p_5$** 

Environment feedback: Action executed successfully. This is step 6. You are allowed to take 94 more steps.

**Action  $a_5$** 

('move', (10, 0))

**Observation  $o_6$** 

target board			current board		
3	1	4	3	1	4
2	9	10	2	10	
5	8	6	5	9	6
	7		7	8	

**Prompt  $p_6$** 

Environment feedback: Action executed successfully. This is step 7. You are allowed to take 93 more steps.

**Action  $a_6$** 

('move', (9, 1))

**Observation  $o_7$** 

target board			current board		
3	1	4	3	1	4
2	9	10	2	10	
5	8	6	5	9	6
	7		7	8	

**Prompt  $p_7$** 

Environment feedback: Action executed successfully. This is step 8. You are allowed to take 92 more steps.

**Action  $a_7$** 

('move', (10, 1))

5184  
5185  
5186  
5187  
5188  
5189  
5190  
5191  
5192  
5193  
5194  
5195  
5196  
5197  
5198  
5199  
5200  
5201  
5202  
5203  
5204  
5205  
5206  
5207  
5208  
5209  
5210  
5211  
5212  
5213  
5214  
5215  
5216  
5217  
5218  
5219  
5220  
5221  
5222  
5223  
5224  
5225  
5226  
5227  
5228  
5229  
5230  
5231  
5232  
5233  
5234  
5235  
5236  
5237

**Observation  $o_8$** 

target board			current board		
3	1	4	3	1	4
2	9	10	2		10
5	8		5	9	6
		7		7	8

**Prompt  $p_8$** 

Environment feedback: Action executed successfully. This is step 9. You are allowed to take 91 more steps.

**Action  $a_8$** 

('move', (9, 0))

**Observation  $o_9$** 

target board			current board		
3	1	4	3	1	4
2	9	10	2	9	10
5	8		5		6
		7		7	8

**Prompt  $p_9$** 

Environment feedback: Action executed successfully. This is step 10. You are allowed to take 90 more steps.

**Action  $a_9$** 

('move', (8, 0))

**Observation  $o_{10}$** 

target board			current board		
3	1	4	3	1	4
2	9	10	2	9	10
5	8		5		8
		7		7	6

**Prompt  $p_{10}$** 

Environment feedback: Action executed successfully. This is step 11. You are allowed to take 89 more steps.

**Action  $a_{10}$** 

('move', (7, 1))

**Observation  $o_{11}$** 

target board			current board		
3	1	4	3	1	4
2	9	10	2	9	10
5	8		5		8
		7		7	6

**Prompt  $p_{11}$** 

Environment feedback: Action executed successfully. This is step 12. You are allowed to take 88 more steps.

**Action  $a_{11}$** 

('move', (8, 3))

**Observation  $o_{12}$** 

target board			current board		
3	1	4	3	1	4
2	9	10	2	9	10
5	8		5	8	6
		7		7	6

**Prompt  $p_{12}$** 

Environment feedback: Action executed successfully. This is step 13. You are allowed to take 87 more steps.

**Action  $a_{12}$** 

('stop', 'stop')

5238  
5239  
5240  
5241  
5242  
5243  
5244  
5245  
5246  
5247  
5248  
5249  
5250  
5251  
5252  
5253  
5254  
5255  
5256  
5257  
5258  
5259  
5260  
5261  
5262  
5263  
5264  
5265  
5266  
5267  
5268  
5269  
5270  
5271  
5272  
5273  
5274  
5275  
5276  
5277  
5278  
5279  
5280  
5281  
5282  
5283  
5284  
5285  
5286  
5287  
5288  
5289  
5290  
5291

### B.32 VIDEO UNSHUFFLE ([Back to Table of Contents](#))

#### Prompt $p_0$

You are given 5 video frames extracted from a short clip, laid out left->right in a shuffled order and labeled 1 through 5. The action being performed in the video is: 'pushing a chair with a hand'.

Your goal is to rearrange the frames so they appear in their original chronological order from left to right. Pay attention to the temporal progression of the action described above.

Available actions:

1. 'swap': Swap two frames by specifying their 1-based positions. Format: `('swap', (i, j))`
2. 'reorder': Provide the complete ordering in one shot. Format: `('reorder', [1, 2, ..., 5])` where the list gives the desired left-to-right arrangement.
3. 'stop': Finish and lock in the current ordering. Format: `('stop', 'stop')`

Please respond with exactly one action and its arguments in the specified format. For example:

- Swap the first and third frames: `('swap', (1, 3))`
- Reorder all frames: `('reorder', [2, 1, 4, 3, ..., 5])`
- Finalize the ordering: `('stop', 'stop')`

This is step 1. You are allowed to take 99 more steps.

#### Observation $o_0$



#### Action $a_0$

`('swap', (1, 4))`

#### Observation $o_1$



#### Prompt $p_1$

Environment feedback: Action executed successfully. This is step 2. You are allowed to take 98 more steps.

#### Action $a_1$

`('swap', (3, 5))`

#### Observation $o_2$



#### Prompt $p_2$

Environment feedback: Action executed successfully. This is step 3. You are allowed to take 97 more steps.

#### Action $a_2$

`('stop', 'stop')`

5292  
5293  
5294  
5295  
5296  
5297  
5298  
5299  
5300  
5301  
5302  
5303  
5304  
5305  
5306  
5307  
5308  
5309  
5310  
5311  
5312  
5313  
5314  
5315  
5316  
5317  
5318  
5319  
5320  
5321  
5322  
5323  
5324  
5325  
5326  
5327  
5328  
5329  
5330  
5331  
5332  
5333  
5334  
5335  
5336  
5337  
5338  
5339  
5340  
5341  
5342  
5343  
5344  
5345

### B.34 ZOOM-IN PUZZLE ([Back to Table of Contents](#))

#### Prompt $p_0$

You are given an original image and 5 zoomed-in views laid out left→right. Your goal is to rearrange them so they are ordered from \*least\* to \*most\* zoomed.

Available actions:

1. 'swap': Swap two zoom-tiles by specifying their 1-based positions. Format: `('swap', (i, j))` where i and j are 1-based positions.
2. 'reorder': Provide the complete ordering in one shot. Format: `('reorder', [1, 2, ..., 5])` where the list gives the desired left-to-right arrangement.
3. 'stop': Finish and lock in the current ordering. Format: `('stop', 'stop')`

Success: Arrange the views from least to most zoomed (ascending zoom level order).

Please respond with exactly one action and its arguments in the specified format. For example:

- To swap views: `('swap', (1, 3))`
- To reorder all: `('reorder', [1, 3, 2, 4])`
- To finalize: `('stop', 'stop')`

This is step 1. You are allowed to take 99 more steps.

#### Observation $o_0$



#### Action $a_0$

`('swap', (1, 4))`

#### Observation $o_1$



#### Prompt $p_1$

Environment feedback: Action executed successfully. This is step 2. You are allowed to take 98 more steps.

#### Action $a_1$

`('swap', (2, 3))`

#### Observation $o_2$



#### Prompt $p_2$

Environment feedback: Action executed successfully. This is step 3. You are allowed to take 97 more steps.

#### Action $a_2$

`('swap', (3, 4))`

#### Observation $o_3$



#### Prompt $p_3$

Environment feedback: Action executed successfully. This is step 4. You are allowed to take 96 more steps.

#### Action $a_3$

`('stop', 'stop')`