Understanding Spatial Reasoning in Vision-Language Models via Sequence-Level Reinforcement Learning

Videet Mehta
MIT EECS
mvideet@mit.edu

Shaurya Agrawal
MIT EECS
shaur10@mit.edu

December 10th

Abstract

Vision-Language Models (VLMs) exhibit strong semantic understanding, similar to Large Language Models, yet remain fundamentally limited in tasks requiring spatial reasoning and geometric awareness. We hypothesize that sequence-level reinforcement learning (RL) algorithms may encourage VLMs to form more coherent spatial representations, enabling one-shot trajectories that solve spatial problems end-to-end. Using maze navigation as a controlled environment for spatial reasoning, we compare single-action policies such as Proximal Policy Optimization (PPO) in the actor–critic setting to sequence-level counterparts such as GRPO, GSPO, and GMPO, that replace value functions with relative rewards. Our early results show that token-level credit assignment remains crucial: PPO achieves significantly higher success rates, while sequence-level algorithms struggle due to coarse trajectory-level credit. These findings suggest that PPO's value function provides essential action-level feedback for visual–spatial reasoning, highlighting the importance of fine-grained credit assignment in training spatially capable VLMs.

1 Introduction

Multi-modal Large language models such as vision language models (VLMs) have significantly advanced visual understanding capabilities since the advent of LLMs. Their language reasoning capabilities make it a strong use-case of vision-question answering, pattern recognition, and object identification [1, 10]. However, agentic tasks such as navigating visual environments, spatial reasoning, and sequential decision-making.

Previous attempts to teach VLMs to navigate complex environment that require visual reasoning include traditional offline fine-tuning given a dataset and running supervised fine-tuning (SFT). This paradigm teaches the model the general distribution of actions to output because of instruction tuning, but does not significantly enhance the reasoning capabilities from explicit Chain-of-Thought reasoning that may benefit a VLMs visual understanding [7]. Additionally, pre-collected off-policy datasets may lack dataset diversity to teach the agent a multitude of scenarios to plan around[5, 4].

In this paper, we investigate reinforcement learning as a paradigm to increase visual reasoning capabilities while trying to solve sequential decision-making tasks. Specifically we post-train a VLM with a multitude of action-level and sequence-level RL algorithms. From previous work, we include the capability for the VLM to output Chain-of-Thought (CoT) tokens, which is designed to provide a good "prior" for the VLM to output more consistent actions with the environment at hand.

We evaluate these RL algorithms by examining the speed of convergence of the reward function and the empirical success rate during training on unseen mazes. Our primary environment that we are training on are mazes spanning from 3x3 to 10x10 in size.

2 Related Work

RL to Train LLMs. RL has had a significant spike in popularity since the rise of large language models to act as an agent. Proximal Policy Optimization (PPO) is a very common actor-critic policy with some modifications. PPO is a clipped policy-gradient method that stabilizes on-policy training by constraining each policy update to remain close to the previous policy. Its actor–critic structure provides dense, per-token credit assignment through a learned value function [13, 3, 8].

In traditional LLM/VLM post-training pipelines, the critic model is often comparable in size to the policy, making actor–critic methods costly during both training and inference. To reduce this overhead, Group Relative Policy Optimization (GRPO) and related approaches replace the value function with a group-based baseline computed from multiple trajectory rollouts [14]. Advantages are then estimated by normalizing rewards across the group, providing a relative advantage signal without a critic. This strategy has yielded notable improvements in mathematical reasoning and the emergence of "aha" moments in chain-of-thought generation. Follow-up methods, including GSPO, DrGRPO, and GMPO, maintain the critic-free design while introducing refinements that improve stability and convergence guarantees [18, 19, 9].

RL for Visual Reasoning. Previous work has shown that using PPO to fine-tune VLMs both increase the CoT reasoning capabilities and improve actions in various games including Blackjack, AlfWorld, and number games [16]. They first fine-tune the VLM to output JSON outputs to encourage proper formatting so that immportance weighting can be calculated. They weight the log probabilities for the CoT trace and the action trace differently to allow the model to focus more on optimizing the action but also still providing focus to the thinking/reasoning trace.

Another concurrent line of work evaluates GRPO on maze-solving tasks and reports strong improvements in sequential navigation accuracy [2]. However, their approach does not leverage visual features directly. Instead of using a VLM, the maze is encoded purely as a tokenized textual grid which limits the model's ability to form coherent spatial representations. Their hypothesis is that GRPO alone, similar to introducing mathematical reasoning in the original GRPO paper, can induce spatial reasoning in an LLM without requiring a vision encoder [14]. While effective to some degree, the reliance on symbolic maze encodings constrains the richness of spatial information available to the model and is a more contrived problem.

1. **H1:** Sequence-level RL algorithms (e.g., GSPO, GMPO) may better align with long-horizon spatial tasks because they optimize entire trajectories rather than individual tokens.

2. **H2:** Token-level RL (e.g., PPO, GRPO) may still outperform sequence-level methods in environments requiring precise step-level credit assignment.

3 Methodology

3.1 Input Representation and Environment

We randomly generate mazes (via the gym-maze) environment that range in size from 3x3 to 10x10. This is done for curriculum learning as we start by training on smaller mazes and increase maze size as training progresses, with the largest mazes being size 10 by 10. Prior to including it in our dataset, we verify that the mazes are solvable via depth-first search (DFS). Additionally, we have some environment constraints to make actions as realistic as possible. For example, while the agent is traversing the maze, it makes an illegal action that isn't allowed in the current episode (eg: turn into a wall). These properties make mazes highly suitable probes of spatial reasoning. An example maze is shown in Fig. 1. Our pseudo-code for Maze Generation is also provided in the Appendix. Each training episode begins with a maze image and an

instruction to navigate to the goal. The VLM produces a textual action sequence (e.g., "up, up, right, ..."), which is executed by the environment. Depending on GRPO or PPO, we either output a sequence of actions or a single action, respectively.

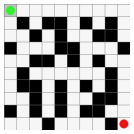


Figure 1: Example 10×10 maze used in our experiments.

3.2 Supervised Fine-Tuning

To enhance instruction following and simplify the action parsing process during RL, we fine-tune the VLM (LlaVA-7b) [6] on sample JSON output to bias the token distribution to similar outputs. We synthetically generate 1000 completions from GPT-4o-mini and run next token prediction on the entire completion, contrary to previous work that did next action token prediction only [2]. We trained for 4 epochs with a learning rate of $5e-6$. Sample prompts from the SFT dataset and the loss curve can be found in the Appendix. Following SFT, we begin our Reinforcement learning formulation of the problem at hand.

4 RL Formulation

Here we will lay down, common terminology for the RL algorithms that we are testing on. Similar to [17], below is the common terminology and variables we will use. Each prompt will be represented as a RGB image t and a corresponding system and instruction prompt x. The output will be represented as $y = \pi_\theta(x)$, where π_θ represents our VLM policy that outputs the actions. Additionally, let G be the number of generations per provided maze such that we have $y_1, y_2, y_3, ..., y_G$.

4.1 GRPO Reward Function

For each completion i, the reward is $R_i = R_i^{\text{fmt}} + R_i^{\text{exec}}$. We hvae a reward for formatting output right, a full large reward for completing the entire maze succesfully, and partial rewards based on the Manhattan distance between the current position and the goal. The detailed equation of the reward function can be found in the Appendix. The training details and hyperparameters can be found in the Appendix as well.

4.2 GRPO

GRPO applies a PPO-style clipped policy-gradient objective to sequence models. They compute token-level likelihood ratios and restricting them via a symmetric clipping function. The advantage is derived directly from normalized reward model outputs, allowing GRPO to replace the value network with a simple, generation-level advantage signal. By averaging these clipped policy-gradient terms across tokens and generations and regularizing with a KL penalty, GRPO achieves stable, high-variance-tolerant updates.

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q,\,\{o_i\}\sim\pi_{\theta_{\text{old}}}} \left[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|o_i|}\sum_{t=1}^{|o_i|} \min\Big(r_{i,t}(\theta)\,\hat{A}_{i,t},\ \text{clip}(r_{i,t}(\theta),\,1-\varepsilon,\,1+\varepsilon)\,\hat{A}_{i,t} \Big) - \beta\, D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}}) \right]. \tag{1}$$

$$\mathbb{D}_{\text{KL}}(\pi_\theta\|\pi_{\text{ref}}) = \frac{\pi_{\text{ref}}}{\pi_\theta} - \log\left(\frac{\pi_{\text{ref}}}{\pi_\theta}\right) - 1 \qquad \hat{A}_{i,t} = \tilde{r}_i = \frac{r_i - \text{mean}(r)}{\text{std}(r)}.$$

4.3 GSPO

Based on this straightforward observation, we propose the **Group Sequence Policy Optimization (GSPO)** algorithm. GSPO employs the following sequence-level optimization objective:

$$\mathcal{J}_{\text{GSPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[\frac{1}{G}\sum_{i=1}^{G}\min\Big(s_i(\theta)\,\hat{A}_i,\ \text{clip}(s_i(\theta),\,1-\varepsilon,\,1+\varepsilon)\,\hat{A}_i\Big)\right], \quad s_i(\theta) = \left(\frac{\pi_\theta(y_i \mid x)}{\pi_{\theta_{\text{old}}}(y_i \mid x)}\right)^{\frac{1}{|y_i|}}. \tag{2}$$

GRPO breaks down because it misuses importance-sampling weights at the **token level**, relying on single next-token samples rather than many behavior-policy samples, which causes high-variance gradients and fails to correctly adjust for distribution mismatch. This flawed design can trigger irreversible model collapse and shows that optimization must happen at the **sequence-level**, matching how rewards are assigned, rather than at individual tokens [19]. Gradient derivation can also be found in the Appendix.

4.4 GMPO

GMPO uses the geometric mean of token-level ratios to reduce variance. For one, this lowers the value range of the objective function, trivially reducing the probability of a policy going off-the-tracks and having stable policy updates. In return, GMPO becomes less sensitive to outliers because of the properties of the geometric mean compared to the arithmetic mean [18].

$$\mathcal{J}^{*}_{\text{GMPO}}(\pi_\theta) = \mathbb{E}_{q\sim\mathcal{Q},\,\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)}\left[\frac{1}{G}\sum_{i=1}^{G}\left(\prod_{t=1}^{|o_i|}\rho_{i,t}(\theta)\,\hat{A}_i\right)^{\frac{1}{|o_i|}}\cdot\text{sgn}(\hat{A}_i)\right], \tag{3}$$

Additionally, GMPO is more stable than GRPO because it remains robust to large importance sampling ratios.

4.5 PPO

PPO adds on top of Trust Region Policy Optimization (TRPO) and introduces dense, step-wise value estimates and generalized advantage estimates (GAE), yielding fine-grained credit assignment [11, 12]. Generalized Advantage estimation is a bias-variance controller estimator of the advantage function by combining multi-step TD residuals. This property aligns closely with the local reward structure inherent to maze navigation. We attach a value-head (MLP) to the VLM to represent our value function $V_\psi(.)$. As we have talked about this in class, the objective function is discussed in the appendix.

5 Results

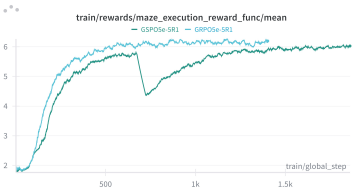
5.1 GRPO

(a) Mean Reward: GRPO

(b) Standard Deviation of Reward: GRPO

Overall, we see an improvement in the average rewards and convergence in the rewards as they stabilize at around a value of 6. We also see high variance relative to the average rewards, reflecting sensitivity to group baselines. An interesting observation is that standard deviation increases as the training steps increase, which is a signal of further exploration, depicting that the model is accurately exploring more paths as the training progresses.

5.2 GSPO

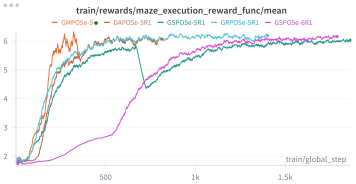


(c) Mean Reward: GSPO vs. GRPO

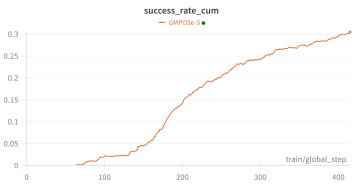
(d) Standard Deviation of Reward: GSPO

We see that the average GSPO reward converges to a lower value than for GRPO while variance for the two are in similar ranges. This is quite surprising as we hypothesize that this maze task is a sequence-level task and so a sequence-level RL algorithm should perform well in identifying accurate trajectories. However, this is the first signal that sequence-level algorithms may not be the best for this task. As a side, we aren't quite sure what causes the large dip in reward mid-training, however the value seems to converge back to its original value. This then motivates us to go back to token-level algorithms, leading us to trying GMPO, which is a variant of GRPO as discussed earlier.

5.3 GMPO



(a) Mean Reward: GMPO vs. GSPO vs. GRPO



(b) Success Rate: GMPO

Overall we see that GMPO performs the best in terms of mean reward, yielding the highest values as compared to the rest of the algorithms. Note that GMPO returns to the token-level view of GRPO.

However, we see that even while GMPO seems to have the best performance, it yields surprisingly low success, confirming that coarse, trajectory-level credit assignment encourages almost-correct paths and dilutes signal for accurate step-level updates to increase success-rate.

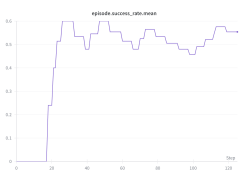
This motivates us to try PPO which provides dense step-level (TD(O)-type) rewards and uses value functions and GAE estimates to assign credit to individual actions as opposed to singular cumulative rewards per trajectory used to calculate advantages for updates. PPO directly maximizes expected return, which is much closer to optimizing for success rate.

5.4 PPO

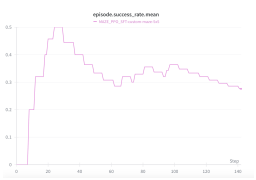
Graphs (c) and (d) are the success rates for our two best PPO implementations.

The first utilizes a dense reward where the reward function is the Manhattan distance of the current state minus the Manhattan distance of the next state. Since these cells are placed at integer positions, the maximum reward an action can achieve with this reward formulation is 1 and the minimum is -1. The second utilizes regular Euclidean distance as the reward where the goal is to minimize the distance from the end goal state.

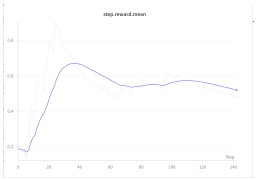
So, it is evident that the PPO trained on dense reward performs the best in terms of success rate, providing improvements past the results of GMPO. Graphs (e) and (f) are the average reward and standard deviation curves for this PPO trained on dense rewards.



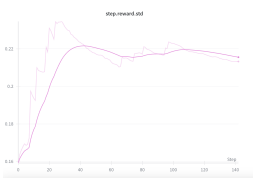
(c) Success Rate: PPO (Dense Reward)



(d) Success Rate: PPO (Normalized Euclidean)



(e) Mean Reward: PPO (Dense Reward)



(f) Standard Deviation of Reward: PPO (Dense Reward)

6 Discussion

Our findings contradict the initial expectation (H1). Sequence-level RL does *not* offer an advantage for this spatial task. Instead, token-level algorithms, specifically step-level update/value function-based algorithms, succeed because:

- Sequence-level averaging and updates dilute gradients across all actions in trajectory, adding noise and weakening the learning signal
- Fine-grained differences in late-stage steps are essential to success, which sequence-level does not necessarily optimize for.
- This leads us to conclude that this maze solving task is a locally-rewarded, MDP problem more than a long-horizon sequential task

First, let's focus on the issues with value-free algorithms both sequence-level and token-level, specifically coarse credit assignment. So, all three of these algorithms are similar in how they calculate the rewards. They all calculate one singular reward per trajectory which are then converted to advantages and used (once the appropriate importance sampling is performed) to weight the log-probs of the tokens of that trajectory. As a result, the output singular reward/signal gets distributed across every action in the sequence to update the tokens. So even if we have early steps that had minimal positive impact on the trajectory relative to the final few actions/middle actions and the overall reward was positive/the trajectory was successful, the early steps will still get the same positive reinforcement from the gradients due to this distribution of this singular reward. In the case the trajectories are mildly noisy with suboptimal moves but eventual success, algorithms like GMPO will still positively reinforce those suboptimal actions since they were a part of the trajectroy rather than focusing on individual actions that made the most difference.

In the case of a maze, the last few actions matter a huge amount and due to the coarse credit assignment of algorithms like GMPO, the model might fail to optimize for these last few steps giving us the results we see with GMPO where there is mean rewards increase and converge but success rates remain low. Even though, there is a goal state bonus to help with rewarding optimal actions in the last few states, the rarity of successes and the high variance in these sequence-level gradients can lead to the goal state bonus not having much of an impact. Furthermore, group normalization used in GMPO further compresses these goal state bonuses, giving us relative, re-scaled reward causing the relative difference in advantage between almost-complete paths and complete-paths to not be large and resulting in the final state bonuses to become insignificant.

In contrast, PPO offers local, dense rewards that rewards each step. Using the critic value function and GAE estimates, the model gets strong signals (even though the value loss for the critic does not necessarily converge) to maximize the sum of discounted returns and assign credit to invidual actions which is much closer to the true objective of the task and the success rate metric since it is sensitive to step-level rewards/updates. With PPO, we also mitigate the issue of high-variance returns from rare goal state bonuses as each time a complete path and a goal state bonus is achieved we see a cascade of informative TD updates across multiple timesteps. The value function and GAE estimates ensure each action update at the step-level is associated with that states future return which is exactly the information needed for such as task. So, it is quite evident that GRPO/GSPO/GMPO are not the correct algorithms for this task as they are not maximzing the objective necessary for this task and as a result are not solving the task directly in comparison to PPO which focuses on dense local structure. This leads us to conclude that this task is *not*

a sequential, long-horizon task and is closer to a locally-rewarded, step-level, MDP problem as we have seen many times in class. This would explain why PPO's step-level updates that resemble TD learning perform better as opposed to broader and coarser sequence-level optimizations. Thus, we confirm H2: token-level credit assignment remains critical for structured spatial reasoning in VLMs.

7 Conclusion

We present an early empirical study comparing sequence-level and token-level RL methods for improving spatial reasoning in VLMs. Our results show that sequence-level approaches underperform in structured, geometry-dependent tasks, while PPO, a token-level method with dense credit assignment, achieves the strongest outcomes.

These results suggest that spatial reasoning in VLMs, especially for tasks like mazes even if they are large and complex tasks, may require RL algorithms explicitly aligned with step-level structure rather than trajectory-level smoothing. Future work will explore hierarchical RL objectives, state-space modeling, and learned world models to further enhance VLM spatial competence. We also hope to make the problem more realistic to move from 2D deterministic agent navigation to 3D stochastic and noisy robot exploration policies. Overall, our results provide early analyses as to how these algorithms might scale in further solving and advancing the larger issue of optimizing vision models for real world interactions and reasoning.

References

[1] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models, 2022.

[2] Alan Dao and Dinh Bach Vu. Alphamaze: Enhancing large language models' spatial intelligence via grpo, 2025.

[3] Tiantian Fan, Lingjun Liu, Yu Yue, Jiaze Chen, Chengyi Wang, Qiying Yu, Chi Zhang, Zhiqi Lin, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Bole Ma, Mofan Zhang, Gaohong Liu, Ru Zhang, Haotian Zhou, Cong Xie, Ruidong Zhu, Zhi Zhang, Xin Liu, Mingxuan Wang, Lin Yan, and Yonghui Wu. Truncated proximal policy optimization, 2025.

[4] Jiaxing Huang, Jingyi Zhang, Kai Jiang, Han Qiu, and Shijian Lu. Visual instruction tuning towards general-purpose multimodal model: A survey, 2023.

[5] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning, 2024.

[6] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024.

[7] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023.

[8] Jiashun Liu, Johan Obando-Ceron, Han Lu, Yancheng He, Weixun Wang, Wenbo Su, Bo Zheng, Pablo Samuel Castro, Aaron Courville, and Ling Pan. Asymmetric proximal policy optimization: mini-critics boost llm reasoning, 2025.

[9] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective, 2025.

[10] Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision-language-action models for embodied ai, 2025.

[11] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2017.

[12] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018.

[13] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[14] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.

[15] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, 1998.

[16] Yuexiang Zhai, Hao Bai, Zipeng Lin, Jiayi Pan, Shengbang Tong, Yifei Zhou, Alane Suhr, Saining Xie, Yann LeCun, Yi Ma, and Sergey Levine. Fine-tuning large vision-language models as decision-making agents via reinforcement learning.

[17] Yuexiang Zhai, Hao Bai, Zipeng Lin, Jiayi Pan, Shengbang Tong, Yifei Zhou, Alane Suhr, Saining Xie, Yann LeCun, Yi Ma, and Sergey Levine. Fine-tuning large vision-language models as decision-making agents via reinforcement learning, 2024.

[18] Yuzhong Zhao, Yue Liu, Junpeng Liu, Jingye Chen, Xun Wu, Yaru Hao, Tengchao Lv, Shaohan Huang, Lei Cui, Qixiang Ye, Fang Wan, and Furu Wei. Geometric-mean policy optimization, 2025.

[19] Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025.

8 Appendix

8.1 SFT Dataset

We run next-token prediction task on following synthetic data format.

```
{
  "id": "fcf8c553-08c1-4cce-b6b7-d22e9cf10f10",
  "conversations": [
    {
      "from": "human",
      "value": "Solve this maze... {\"thoughts\":\"...\",\"actions\":[...]}"
    },
    {
      "from": "gpt",
      "value": "{\"thoughts\":\"I need to find my way out...\",\"actions\":[\"
          up\",\"left\",...]}",
    }
  ]
}
```

8.2 GRPO+ System Prompt

```
REASONING_START = "<REASONING>"
REASONING_END = "</REASONING>"
ACTIONS_START = "<ACTIONS>"
ACTIONS_END = "</ACTIONS>"

SYSTEM_PROMPT = """You are a maze-solving AI. You will be shown an image of a
    maze and must navigate from the green circle (start) to the red circle (
    goal).

The maze uses a grid system where:
- Black cells are walls (impassable)
- White cells are paths (walkable)
- Green circle = your current position (agent)
- Red circle = goal position
```

```
Available actions: UP, DOWN, LEFT, RIGHT

Output format:
1. First, provide your reasoning about the maze layout and optimal path inside
     {reasoning_start} and {reasoning_end} tags.
2. Then, provide your action sequence as a comma-separated list inside {
    actions_start} and {actions_end} tags.

Example output:
{reasoning_start}
I can see a 5x5 maze. I'm at the top-left corner and need to reach the bottom-
    right.
There's a wall blocking the direct path, so I need to go around...
{reasoning_end}
{actions_start}
DOWN, DOWN, RIGHT, RIGHT, DOWN, RIGHT
{actions_end}
""".format(
   reasoning_start=REASONING_START,
   reasoning_end=REASONING_END,
   actions_start=ACTIONS_START,
   actions_end=ACTIONS_END
)
```

9 PPO System Prompt

```
"You are an extremely smart maze solver. You see a top-down view of the maze.
The green marker is your CURRENT position. The red marker is the GOAL.
At this time step you must choose EXACTLY ONE next move for the agent,
NOT the entire path. Do NOT plan or output a full trajectory.
You are doing step-by-step decision making: on each call you output only the
   NEXT move.

You can choose between four directions: ['N', 'S', 'E', 'W'],
where N = move up, S = move down, E = move right, W = move left.

Your response MUST be a valid JSON object with EXACTLY ONE action, in the
   following format:
{
  "thoughts": "briefly think about the BEST next move only (do NOT describe
     multiple steps)",
  "action": "N" or "S" or "E" or "W"
}

NEVER output a list of actions. NEVER output more than one action.
If you are unsure, still choose exactly one of ['N', 'S', 'E', 'W']."
```

9.1 PPO Reward Function

We try various reward functions, the two that seem the most promising with the best results are:

Normalized Euclidean Reward Function:

Let the agent state at time step t be $s_t = (x_t, y_t)$, and let the goal state be $g = (x_g, y_g)$.
The Euclidean distance to the goal is

$$d_t^{\text{Euc}} = \|s_t - g\|_2 = \sqrt{(x_t - x_g)^2 + (y_t - y_g)^2}.$$

Let $d_{\max}^{\text{Euc}} > 0$ denote the maximum possible Euclidean distance between any valid state and the goal in this environment. Define the normalized Euclidean distance

$$\bar{d}_t^{\text{Euc}} = \frac{d_t^{\text{Euc}}}{d_{\max}^{\text{Euc}}}.$$

With a time penalty $\lambda > 0$ and a terminal goal bonus $R_{\text{goal}} > 0$, a purely state-based Euclidean reward can be written as

$$r_t^{\text{Euc}} = -\bar{d}_t^{\text{Euc}} \; - \; \lambda \; + \; R_{\text{goal}} \, \mathbf{1}[s_t = g] \, .$$

Manhattan Distance Difference Reward (Dense Reward):

The Manhattan distance to the goal is

$$d_t^{\text{Man}} = |x_t - x_g| + |y_t - y_g|.$$

A dense shaping reward based on Manhattan distance change, with time penalty $\lambda > 0$ and terminal goal bonus $R_{\text{goal}} > 0$, can be defined as

$$r_t^{\text{Man}} = \begin{cases} -\, d_0^{\text{Man}} \; - \; \lambda, & t = 0, \\ \left(d_{t-1}^{\text{Man}} - d_t^{\text{Man}}\right) \; - \; \lambda \; + \; R_{\text{goal}} \, \mathbf{1}[s_t = g] \, , & t \geq 1. \end{cases}$$

We found that the PPO trained on the Dense Reward function had the best performance, so we further did a simple (non-exhaustive) hyperparameter search on this and found a goal state bonus of 5 respectively with a time penalty of 0.3 seemed to converge well and perform best.

9.2 PPO Pseudo-Code

Algorithm 1 Iterative Maze-Solving Policy Optimization (Maze-PPO)

Require: Initial policy–value model $(\pi_{\theta_{\text{init}}}, V_{\psi_{\text{init}}})$; dense step reward function r_{dense}; maze dataset $\mathcal{D}$; hyperparameters $\gamma, \lambda, \epsilon, K$

Ensure: Trained maze-solving policy π_θ

$\pi_\theta \leftarrow \pi_{\theta_{\text{init}}},\ V_\psi \leftarrow V_{\psi_{\text{init}}}$
for iteration $= 1, \ldots, I$ **do**
  $\pi_{\text{old}} \leftarrow \pi_\theta$
  Sample a batch of mazes $\mathcal{D}_b \subset \mathcal{D}$
  Roll out trajectories in each maze $m \in \mathcal{D}_b$ using π_{old}:
  collect states s_t, actions a_t, and dense rewards
  $r_t = r_{\text{dense}}(s_t, a_t, s_{t+1})$ at every step
  Compute returns and advantages $\hat{A}_t$ for all steps using the dense rewards (e.g., GAE with γ, λ)
  for PPO epoch $= 1, \ldots, K$ **do**
    Update (π_θ, V_ψ) by maximizing the clipped PPO objective
  on minibatches of the collected maze rollouts
  end for
end for
return π_θ

9.3 GRPO Reward Function

Formatting. Let 1_i^{reason} and 1_i^{actions} indicate whether the output contains exactly one reasoning and actions block. With weight λ_{fmt}:

$$R_i^{\text{fmt}} = \lambda_{\text{fmt}} \left(1_i^{\text{reason}} + 1_i^{\text{actions}}\right).$$

Execution. If actions cannot be parsed, $R_i^{\text{exec}} = -1$. Otherwise, let d_0 and d_i be initial and final Manhattan distances to the goal and T_i the number of steps. Solved mazes receive

$$R_i^{\text{exec}} = \lambda_{\text{solve}} + \lambda_{\text{eff}} \max(0,\, 2d_0 - T_i),$$

and unsolved mazes receive

$$R_i^{\text{exec}} = \lambda_{\text{partial}} \frac{d_0 - d_i}{\max(d_0, 1)}.$$

9.4 GRPO Psuedo-Code

Algorithm 2 Iterative Maze-Solving Policy Optimization (Maze-GRPO)

Require: Initial policy model $\pi_{\theta_{\text{init}}}$; reward model r_φ; maze dataset $\mathcal{D}$; hyperparameters ε, β, μ
Ensure: Trained maze-solving policy π_θ

$\pi_\theta \leftarrow \pi_{\theta_{\text{init}}}$
for iteration $= 1, \ldots, I$ **do**
  $\pi_{\text{ref}} \leftarrow \pi_\theta$
  for step $= 1, \ldots, M$ **do**
    Sample a batch of mazes $\mathcal{D}_b \subset \mathcal{D}$
    $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$
    Sample G action sequences for each maze $m \in \mathcal{D}_b$:
      $o_i \sim \pi_{\theta_{\text{old}}}(\cdot \mid m)$ for $i = 1, \ldots, G$
    Compute rewards r_i for each sequence o_i by running the **parsed** actions through the maze:
    Compute advantages $\hat{A}_{i,t}$ for each action t in o_i using group-relative estimation
    for GRPO iteration $= 1, \ldots, \mu$ **do**
      Update policy π_θ by maximizing the GRPO objective
    end for
    Update reward model r_φ using replayed maze-solving rollouts
  end for
end for
return π_θ

9.5 GRPO Gradient Derivation

Let $q \sim p_{\text{sf}}$ be a prompt and let $\{o_l\}_{l=1}^{G}$ denote G sampled output sequences from the behavior policy π_{old}. For $o_l = (o_{l,1}, \ldots, o_{l,|o_l|})$, denote the tokenwise advantage by $\hat{A}_{l,i}$. Define the importance ratio

$$r_{l,i}(\theta) := \frac{\pi_\theta(o_{l,i} \mid q, o_{l,<i})}{\pi_{\text{old}}(o_{l,i} \mid q, o_{l,<i})},$$

and the reference ratio

$$x_{l,i}(\theta) := \frac{\pi_{\text{ref}}(o_{l,i} \mid q, o_{l,<i})}{\pi_\theta(o_{l,i} \mid q, o_{l,<i})}.$$

The GRPO objective (Eq. 19) is

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q, o_{1:G} \sim \pi_{\text{old}}} \left[\frac{1}{G} \sum_{l=1}^{G} \frac{1}{|o_l|} \sum_{i=1}^{|o_l|} \Big(r_{l,i}(\theta)\, \hat{A}_{l,i} - \beta \big(x_{l,i}(\theta) - \log x_{l,i}(\theta) - 1 \big) \Big) \right]. \tag{4}$$

Gradient of the RL term. Since π_{old} is fixed,

$$\nabla_\theta \Big(r_{l,i}(\theta) \hat{A}_{l,i} \Big) = \hat{A}_{l,i}\, \nabla_\theta r_{l,i}(\theta) = \hat{A}_{l,i}\, r_{l,i}(\theta)\, \nabla_\theta \log \pi_\theta(o_{l,i}). \tag{5}$$

Gradient of the reference penalty. Let $f(x) = x - \log x - 1$, so $f'(x) = 1 - \frac{1}{x}$. Because $x_{l,i} = \pi_{\text{ref}}/\pi_\theta$,

$$\nabla_\theta x_{l,i} = -x_{l,i}\, \nabla_\theta \log \pi_\theta(o_{l,i}),$$

and therefore

$$\nabla_\theta [-\beta f(x_{l,i})] = \beta\, (x_{l,i} - 1)\, \nabla_\theta \log \pi_\theta(o_{l,i}). \tag{6}$$

Full GRPO gradient Combining the two components yields

$$\nabla_\theta \mathcal{J}_{\text{GRPO}} = \mathbb{E}\left[\frac{1}{G}\sum_{l=1}^{G}\frac{1}{|o_l|}\sum_{i=1}^{|o_l|}\Big(r_{l,i}(\theta)\,\hat{A}_{l,i} + \beta\,(x_{l,i}(\theta)-1)\Big)\,\nabla_\theta \log \pi_\theta(o_{l,i})\right]. \tag{7}$$

Substituting the ratio definitions,

$$\nabla_\theta \mathcal{J}_{\text{GRPO}} = \mathbb{E}\left[\frac{1}{G}\sum_{l=1}^{G}\frac{1}{|o_l|}\sum_{i=1}^{|o_l|}\left(\frac{\pi_\theta(o_{l,i})}{\pi_{\text{old}}(o_{l,i})}\hat{A}_{l,i} + \beta\left(\frac{\pi_{\text{ref}}(o_{l,i})}{\pi_\theta(o_{l,i})}-1\right)\right)\nabla_\theta \log \pi_\theta(o_{l,i})\right]. \tag{8}$$

9.6 GSPO Gradient Derivation

Let $x \sim \mathcal{D}$ be a prompt and $\{y_i\}_{i=1}^{G} \sim \pi_{\text{old}}(\cdot \mid x)$ be G sampled sequences from the behavior policy. For sequence $y_i = (y_{i,1}, \ldots, y_{i,|y_i|})$, let $\hat{A}_i$ denote its advantage. Define the likelihood ratio

$$s_i(\theta) := \frac{\pi_\theta(y_i \mid x)}{\pi_{\text{old}}(y_i \mid x)}.$$

The GSPO objective is

$$\mathcal{J}_{\text{GSPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y_{1:G}\sim\pi_{\text{old}}}\left[\frac{1}{G}\sum_{i=1}^{G} s_i(\theta)\,\hat{A}_i\right]. \tag{9}$$

Since neither $\mathcal{D}$ nor π_{old} depends on θ, the gradient moves inside the expectation:

$$\nabla_\theta \mathcal{J}_{\text{GSPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\hat{A}_i\,\nabla_\theta s_i(\theta)\right]. \tag{10}$$

Using $\nabla_\theta s_i(\theta) = s_i(\theta)\,\nabla_\theta \log s_i(\theta)$, we obtain

$$\nabla_\theta \mathcal{J}_{\text{GSPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G} s_i(\theta)\,\hat{A}_i\,\nabla_\theta \log s_i(\theta)\right]. \tag{11}$$

Since

$$\log s_i(\theta) = \log \pi_\theta(y_i \mid x) - \log \pi_{\text{old}}(y_i \mid x), \qquad \nabla_\theta \log \pi_{\text{old}} = 0,$$

we have

$$\nabla_\theta \log s_i(\theta) = \nabla_\theta \log \pi_\theta(y_i \mid x) = \sum_{t=1}^{|y_i|}\nabla_\theta \log \pi_\theta(y_{i,t} \mid x, y_{i,<t}).$$

$$\nabla_\theta \mathcal{J}_{\text{GSPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\left(\frac{\pi_\theta(y_i \mid x)}{\pi_{\text{old}}(y_i \mid x)}\hat{A}_i\right)\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\nabla_\theta \log \pi_\theta(y_{i,t} \mid x, y_{i,<t})\right]. \tag{12}$$

9.7 PPO Objective

We begin with the one-step TD residual [15]:

$$\delta_t = r_t + \gamma V_\psi(s_{t+1}) - V_\psi(s_t),$$

GAE takes an exponentially weighted average over all k where $\lambda \in [0, 1]$ controls the bias–variance tradeoff:

$$\hat{A}_t^{\text{GAE}} = \sum_{l=0}^{\infty}(\gamma\lambda)^l\,\delta_{t+l}, \tag{13}$$

The PPO surrogate relies on the per-timestep likelihood ratio:

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}.$$

$$J_{\text{PPO}}(\theta) = \mathbb{E}_{t\sim\pi_{\theta_{\text{old}}}}\Big[\min\Big(r_t(\theta)\,\hat{A}_t,\ \text{clip}(r_t(\theta),\,1-\varepsilon,\,1+\varepsilon)\,\hat{A}_t\Big)\ -\ \beta\, D_{\text{KL}}(\pi_\theta \parallel \pi_{\theta_{\text{old}}})\Big]. \tag{14}$$