AlphaMaze: Enhancing Spatial Intelligence in Large Language Models

Anonymous ACL submission

Abstract

Although Large Language Models (LLMs) have demonstrated impressive capabilities in language processing, they often struggle with tasks requiring spatial reasoning, particularly for applications like robot navigation where understanding the robot's position relative to its environment is key. We design a text-based reasoning benchmark, MazeBench, consisting of 5x5 mazes rendered as text with varying complexity, to investigate spatial reasoning in text-based reasoning models. On this benchmark, DeepSeek-R1-671B solves 74% of the mazes in a zero-shot manner. However, with Supervised Finetuning (SFT), our model AlphaMaze-SFT, solves 87% of mazes using only 1.5B parameters. Further refinement with Group Relative Policy Optimization (GRPO) allowed AlphaMaze-GRPO to solve 95% of the benchmark. Our results demonstrate that while spatial reasoning can be achieved by a powerful general reasoning model, a smaller specialist model can also achieve significant spatial reasoning capabilities, presenting a viable approach in resource-constrained applications such as robotics.

1 Introduction

Visual spatial reasoning remains a significant challenge for LLMs, despite their impressive performance in natural language processing and code generation (Zhang et al., 2024; Ma et al., 2024). Current Vision-Language Models (VLMs) excel at pattern recognition and object identification but struggle with tasks requiring deeper spatial inference and step-by-step planning in visual domains. While techniques such as Multimodal Visualization-of-Thought (Li et al., 2025) have shown that such reasoning is possible, reliance on VLMs entails large-scale training of vision encoders and expensive multi-modal inference. Spatial reasoning with text-only LLMs (Wang et al., 2024) remain a goal for resource constrained applications such as robotics.

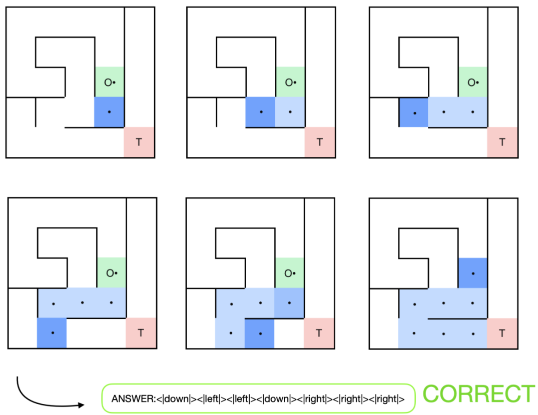

Figure 1: Visualization of AlphaMaze's step-by-step prediction process during maze solving, learned via SFT and refined by GRPO.

This paper tackles the challenge of teaching visual spatial reasoning to a standard LLM through maze navigation. We hypothesize that by providing an LLM with a tokenized *visual* representation of a maze, we can train it to learn step-by-step movement commands from origin to target. Our approach builds on several key research areas: (1) Chain-of-Thought (CoT) prompting (Wei et al., 2022b, 2023; Wang et al., 2023), which encourages LLMs to generate intermediate reasoning steps for multi-step inference tasks and which we extend to visual spatial reasoning; (2) Supervised Fine-Tuning (SFT) (Wei et al., 2022a; Jiang et al., 2024), which adapts pre-trained LLMs to specific tasks through task-specific datasets and serves as our initial training stage; (3) Reinforcement Learning techniques, specifically Group Relative Policy Optimization (GRPO) (Kwon et al., 2023a; Guo et al., 2025; Shao et al., 2024), which offers a computationally efficient alternative to RLHF by estimating advantages based on group scores without a separate critic network, similar to self-play mechanisms like SPIN (Chen et al., 2024); and (4) Visual Reasoning and Maze Solving approaches, traditionally based on graph search algorithms (Lester, 2014-2024; Janamian and Alam, 2023) and recently enhanced by techniques like Microsoft's Multimodal Visualization-of-Thought (Li et al., 2025) and neural-symbolic methods (Mao et al., 2023).

We present a two-stage framework that employs SFT to establish foundational maze navigation skills using tokenized visual representations, then applies GRPO with carefully designed rewards to refine the model's reasoning and decision-making. To evaluate our approach, we introduce MazeBench, a comprehensive benchmark for assessing LLMs' maze-solving capabilities across varying complexity levels.

Our contributions include: (1) a novel training framework for enhanced visual-spatial reasoning in LLMs utilizing SFT and GRPO; (2) empirical demonstration that this approach improves maze navigation accuracy and fosters emergent chain-of-thought reasoning; and (3) We release MazeBench [1], a structured benchmark for visual maze navigation that captures diverse spatial challenges.

2 Methodology

2.1 Stage 1: Supervised Fine-Tuning

First, we fine-tune the model using SFT to establish foundational maze-solving skills. Mazes are represented as a sequence of tokens encoding grid coordinates (<|row-col|>), wall presence relative to the cell (e.g., <|up_wall|>, <|no_wall|>), and special markers for <|origin|> and <|target|>. Empty cells within the representation are marked with <|blank|>. This symbolic tokenization explicitly encodes spatial relationships.

The SFT stage uses a dataset of 500k synthetically generated 5x5 mazes with varied complexity. The training objective is to predict the *next* movement token (<|up|>, <|down|>, <|left|>, or <|right|>) at each step, conditioned on the tokenized maze input and any preceding movement tokens generated by the model. This step-by-step prediction encourages sequential reasoning, as visualized in Figure 1. The SFT dataset also includes 'reset' examples where the model learns to recover from simulated incorrect paths (details deferred to supplementary material).

[1] link to github repository to be shared upon publication

2.2 Stage 2: Group Relative Policy Optimization

Following SFT, we apply GRPO (Guo et al., 2025) to refine the model's policy, enhance robustness, and encourage deeper reasoning. This stage uses a distinct set of 150k mazes. We employ LoRA (Hu et al., 2021) for parameter-efficient fine-tuning, implemented using efficient tooling like Unsloth (Daniel Han and team, 2023) and VLLM (Kwon et al., 2023b) for inference during RL.

Reward Structure: GRPO estimates advantages based on relative scores within sampled batches (groups) of trajectories, updating the policy to maximize expected reward without requiring a separate critic network, thus refining the initial SFT policy towards more accurate and robust maze navigation.

Our GRPO implementation uses three complementary rewards: (1) a *Correctness Reward* (+0.2 per correct step) scaled by the ground-truth path length when the target is reached, incentivizing efficient solutions; (2) an *Integrity Reward* (+0.5) awarded for outputs containing only valid movement tokens and optional <think> tags; and (3) a *Thinking Reward* (+0.25) for correctly using the <think> tag before movement tokens, encouraging emergent reasoning.

2.3 MazeBench

To evaluate spatial reasoning and planning capabilities, we introduce MazeBench, a benchmark of 100 maze-solving challenges generated using the same approach as the training and development set. Unlike existing benchmarks such as (Rein et al., 2024) that focus on logical reasoning or commonsense knowledge, MazeBench specifically targets spatial understanding, multi-step planning, and sequential action execution—capabilities crucial for robotics, navigation, and virtual agent control. A similar benchmark, SpatialEval (Wang et al., 2024) contains a maze-solving subset which has both image and text representations. However, unlike MazeBench, SpatialEval is designed as a question answering (QA) benchmark. Other spatial reasoning benchmarks such as StepGame (Shi et al., 2022) and GSR-Bench (Rajabi and Kosecka, 2024) are also largely QA-based and tackle reasoning of spatial relationships between objects.

MazeBench is structured into three difficulty levels based on the path length required to reach the goal, as detailed in Table 1. The Easy category (50 mazes, 1-4 steps) establishes a baseline for funda-

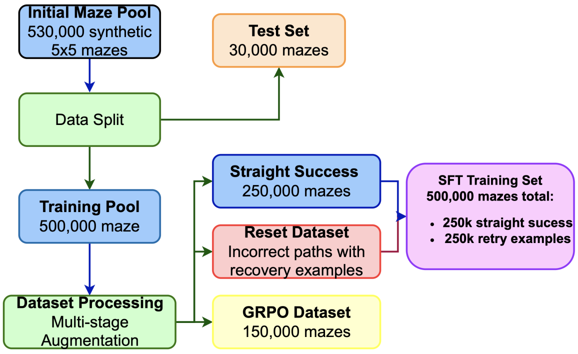

Figure 2: Dataset creation process showing the generation and partitioning of maze data into test, straight success, reset, and GRPO training datasets.

mental navigation skills. The Medium category (40 mazes, 5-8 steps) requires more advanced planning. The Hard category (10 mazes, 9-13 steps) tests the model's capacity to handle complex spatial structures and extended solution paths.

Table 1: Maze Configuration by Difficulty Level

Category	Number of Mazes	Steps
Easy	50	1 – 4
Medium	40	5 – 8
Hard	10	9 – 13
Total	**100**	**1 – 13**

Mazes are presented to LLMs in the tokenized format described in Appendix A. During evaluation, we extract movement tokens from the model's output, with the correct sequence being crucial. A solution is considered correct only if the extracted sequence of movement tokens leads to the target without invalid moves. Our primary evaluation metric is the success rate: the percentage of mazes solved correctly.

2.4 Generating Training Data

Figure 2 illustrates our dataset construction process. We generated 530,000 synthetic 5×5 mazes using randomized depth-first search via the **maze-dataset** framework (Ivanitskiy et al., 2023), ensuring each maze has a guaranteed solution path. We reserved 30,000 mazes as our test set and used the remaining 500,000 for training. From this pool, we created three components: (1) a straight success dataset of 250,000 mazes with direct solution paths, (2) a reset dataset of 250,000 mazes containing algorithmically generated incorrect paths followed by reset messages and correct solutions, and (3) a 150,000-maze dataset for GRPO training. The final SFT dataset combines equal parts success and retry examples (250,000 each), balancing direct navigation with error recovery capabilities. Further algorithmic details are reported in in Appendix C.

2.4.1 Error Types

To evaluate the performance of the maze-solving model, we conducted a detailed analysis of its incorrect solution attempts. We categorized the errors encountered into three primary types:

E1 - Invalid Solution: where the model's output deviates significantly from the expected format of a sequence of directional tokens. This error can also arise due truncation from maximum token limits during generation.

E2 - Path Blocked: where the model's proposed move sequence results in moving into a wall within the maze environment.

E3 - Incomplete Solution: where the model generated a sequence of syntactically valid moves not resulting in wall collisions but fails to reach the target location.

2.5 Hyperparameters

We fine-tuned the DeepSeek-R1-Distill-Qwen-1.5B model (Guo et al., 2025) using a two-stage process. The initial Supervised Fine-Tuning (SFT) stage ran for 1 hour on 8 NVIDIA H200 GPUs, employing a learning rate of 1.0e-5 with a warm-up proportion of 0.1. The GRPO stage was conducted for 5-6 hours on a single H200 GPU. For LoRA, we set the rank (r) to 128, alpha (α) to 128, and the learning rate to 1.0e-6. The number of generations per prompt to sample was set to 4.

For model inference and output generation, we experimented with various prompts. We set the maximum number of new tokens to be generated to 30,000. The temperature parameter was set to 0.6 across all experiments reported here.

3 Results

3.1 Model Performance on MazeBench

As shown in Table 2, the initial model, trained for direct path prediction without explicit reasoning, achieved 1% accuracy on MazeBench. The SFT-only model reached a baseline of 87%, demonstrating the effectiveness of supervised fine-tuning for learning step-by-step maze navigation. Further enhancement with GRPO led to significant improvement, reaching 95% after 1600 steps of GRPO training.

Table 2: Performance Comparison of Reasoning Models. Category scores are report as counts. The total for each of the categories are Easy (50), Medium (40) and Hard (10). Error types are as follows: E1-Invalid Solution, E2-Path Blocked, E3-Incomplete Solution.

Model	**Overall Acc (%)**	**Score by Category**			**Error Count by Type**		
		Easy	**Medium**	**Hard**	**E1**	**E2**	**E3**
DeepSeek-R1-Distill-Qwen-1.5B	1	1	0	0	43	36	20
DeepSeek-R1-Distill-Qwen-7B	5	4	1	0	16	37	42
DeepSeek-R1-671B	74	43	28	3	0	22	4
AlphaMaze-1.5B-SFT (Ours)	87	46	34	7	0	11	2
AlphaMaze-1.5B-GRPO (Ours)	**95**	**49**	**38**	**8**	0	5	0

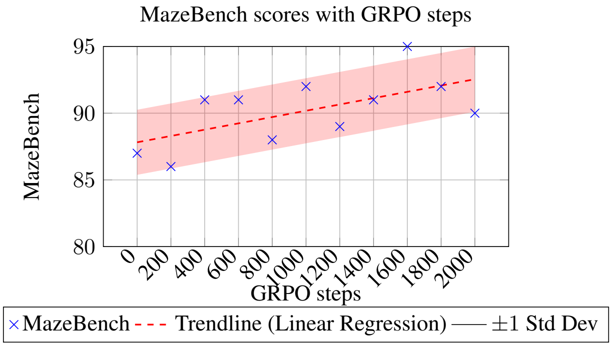

Figure 3: MazeBench scores over GRPO steps with a linear regression trendline and its ± 1 standard deviation bounds.

The progression of performance across our models highlights the impact of each training stage. While the base 1.5B parameter model struggles in the zero-shot setting, task-specific SFT dramatically improved its performance. The subsequent application of GRPO further refined the learned policy, leading to even higher accuracy and suggesting that reinforcement learning techniques can effectively optimize the model's decision-making process for complex spatial tasks. This demonstrates a viable pathway for enhancing specific reasoning skills in more resource-constrained models.

We also evaluated two additional models under a zero-shot setting, to test the baseline capabilities of pre-trained reasoning models. DeepSeek-R1-Distill-Qwen-7B performs better than the 1.5B model with a 5% completion rate. Meanwhile the much larger 671B DeepSeek-R1 model, accessed through DeepSeek's API, achieves 75% accuracy.

The strong zero-shot performance of the much larger DeepSeek-R1-67B model indicates that scaling model size can inherently improve spatial reasoning capabilities, but our results demonstrate that targeted training, even on a smaller model, can achieve competitive performance with significantly fewer parameters.

3.1.1 Model Evolution During GRPO

Figure 3 displays the MazeBench scores (blue crosses) over GRPO steps along with a linear regression trendline (red dashed line) and its ± 1 standard deviation bounds. The steady increase in the trendline indicates that the reinforcement learning with GRPO is able to guide the model towards improved maze-solving capabilities. This process is nevertheless noisy, an additional GRPO steps could still potentially further improve performance.

While the absolute accuracy gain (87% to 95%) over a strong SFT baseline is moderate, we observe notable qualitative improvements during this process. The GRPO-refined model exhibits more robust, self-correcting reasoning patterns resembling chain-of-thought, suggesting GRPO encourages deeper sequential deliberation beyond simple path prediction, guided by our tailored reward function. We present our full observations in Appendix B.

4 Conclusion

This paper presents AlphaMaze, a two-stage training framework that combines SFT and GRPO to enhance spatial reasoning capabilities in LLMs for maze navigation tasks. Our approach utilizes a tokenized representation of mazes, with SFT establishing foundational navigation skills that are refined through GRPO. Experimental results on our MazeBench benchmark demonstrate that this approach improves performance from 87% to 95% accuracy using only a 1.5B parameter LLM. This research demonstrates the effectiveness of applying RL techniques developed for language tasks to the domain of visual-spatial reasoning, particularly for parameter-efficient models. These findings suggest promising applications in domains requiring integrated spatial understanding and sequential decision-making.

Limitations

While our results are promising, several limitations warrant consideration. The performance improvement from GRPO implementation (7% absolute gain) is modest, suggesting potential for further optimization of the reward function and training parameters. Our evaluation methodology primarily focuses on solution accuracy rather than incorporating more nuanced metrics for path efficiency or reasoning quality. Although qualitative observations suggest improved reasoning patterns after GRPO training, more rigorous interpretability studies are needed to substantiate these findings. Additionally, our experiments are limited to synthetic 5×5 mazes; future work should investigate generalizability to larger, more complex environments and real-world spatial reasoning tasks. Finally, while our tokenized representation proves effective, it represents a simplified abstraction compared to raw visual input processing.

References

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. *Preprint*, arXiv:2401.01335.

Michael Han Daniel Han and Unsloth team. 2023. Unsloth.

Duyu Guo, Guoxin Xu, Yuchen Chen, Chen Tang, and Others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Michael Igorevich Ivanitskiy, Rusheb Shah, Alex F. Spies, Tilman Räuker, Dan Valentine, Can Rager, Lucia Quirke, Chris Mathwin, Guillaume Corlouer, Cecilia Diniz Behn, and Samy Wu Fung. 2023. A configurable library for generating and manipulating maze datasets. *Preprint*, arXiv:2309.10498.

Saba Janamian and MD Sahabul Alam. 2023. Maze solver robot using a* algorithm. ScholarWorks@CSUN.

Xiaohu Jiang, Yixiao Ge, Yuying Ge, Dachuan Shi, Chun Yuan, and Ying Shan. 2024. Supervised fine-tuning in turn improves visual foundation models. *Preprint*, arXiv:2401.10222.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. 2023a. Reward design with language models. *Preprint*, arXiv:2303.00001.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023b. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Patrick Lester. 2014-2024. Pathfinding algorithms. Red Blob Games.

Chengzu Li, Wenshan Wu, Huanyu Zhang, Yan Xia, Shaoguang Mao, Li Dong, Ivan Vulić, and Furu Wei. 2025. Imagine while reasoning in space: Multimodal visualization-of-thought. *Preprint*, arXiv:2501.07542.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. 2024. A survey on vision-language-action models for embodied ai. *Preprint*, arXiv:2405.14093.

Jiajun Mao, Chuang Gan, Fan Zhang, and Others. 2023. Neural-symbolic visual reasoning: A survey.

Navid Rajabi and Jana Kosecka. 2024. Gsr-bench: A benchmark for grounded spatial reasoning evaluation via multimodal llms. *arXiv preprint arXiv:2406.13246*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Zhengxiang Shi, Qiang Zhang, and Aldo Lipani. 2022. Stepgame: A new benchmark for robust multi-hop spatial reasoning in texts. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 11321–11329.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2717–2739, Toronto, Canada. Association for Computational Linguistics.

Jiayu Wang, Yifei Ming, Zhenmei Shi, Vibhav Vineet, Xin Wang, Yixuan Li, and Neel Joshi. 2024. Is a picture worth a thousand words? delving into spatial reasoning for vision language models. In *The Thirty-Eighth Annual Conference on Neural Information Processing Systems*.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022a. Finetuned language models are zero-shot learners. *Preprint*, arXiv:2109.01652.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Jason Wei, Denny Zhou, Quoc Le, Denny Zhou, Quoc Le, and Others. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. 2024. Vision-language models for vision tasks: A survey. *Preprint*, arXiv:2304.00685.

A Maze Tokenization Example

This section provides a concrete example of the tokenization scheme used to represent mazes for the LLM input. The scheme encodes the grid structure, walls relative to each cell, and the origin/target locations.

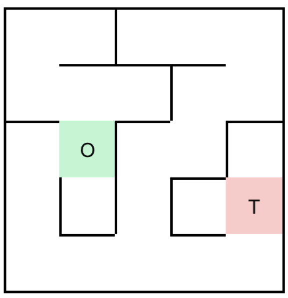

Figure 4: Visual representation of the 5x5 maze.

The full tokenized input sequence for the maze depicted in Figure 4 is presented below. Each line represents a row of the maze in the token sequence.

```
<|0-0|><|up_left_wall|><|blank
    |><|0-1|><|up_down_right_wall|><|
    blank|><|0-2|><|up_down_left_wall
    |><|blank|><|0-3|><|up_down_wall|><|
    blank|><|0-4|><|up_right_wall|><|
    blank|>
<|1-0|><|down_left_wall|><|blank
    |><|1-1|><|up_wall|><|blank
    |><|1-2|><|up_down_right_wall|><|
    blank|><|1-3|><|up_left_wall|><|
    blank|><|1-4|><|down_right_wall|><|
    blank|>
<|2-0|><|up_left_wall|><|blank
    |><|2-1|><|right_wall|><|origin
    |><|2-2|><|up_left_wall|><|blank
    |><|2-3|><|down_right_wall|><|blank
    |><|2-4|><|up_left_right_wall|><|
    blank|>
<|3-0|><|left_right_wall|><|blank
    |><|3-1|><|down_left_right_wall|><|
    blank|><|3-2|><|left_right_wall|><|
    blank|><|3-3|><|up_down_left_wall
    |><|blank|><|3-4|><|right_wall|><|
    target|>
<|4-0|><|down_left_wall|><|blank
    |><|4-1|><|up_down_wall|><|blank
    |><|4-2|><|down_wall|><|blank
    |><|4-3|><|up_down_wall|><|blank
    |><|4-4|><|down_right_wall|><|blank
    |>
```

B Qualitative Results

Qualitative analysis of model outputs revealed notable differences in reasoning behavior. The baseline model often produced nonsensical or incomplete movement sequences, frequently failing to reach the target and exhibiting "hallucinations" by predicting movements invalid within the maze structure. The **AlphaMaze-SFT** model demonstrated improved coherence and step-by-step progression, but still struggled with longer or more complex mazes, sometimes becoming trapped in loops or making incorrect turns in later stages of the solution path.

In contrast, the **AlphaMaze-SFT+GRPO** model exhibited the most sophisticated reasoning. In many instances, emergent chain-of-thought patterns were observed, with AlphaMaze (two-stage) appearing to explicitly consider wall constraints and spatial relationships at each step before predicting the next movement. Furthermore, outputs occasionally displayed instances reminiscent of the "aha moments" reported in prior work on DeepSeek-R1. For example, in some complex mazes, AlphaMaze (two-stage) would initially begin along one path, then appear to "re-evaluate" its trajectory mid-sequence, correcting its course to find a more efficient or correct solution. Error analysis indicated that AlphaMaze (two-stage) made fewer invalid moves and was more robust to long-context reasoning challenges compared to the AlphaMaze-SFT model. However, limitations remained, particularly in mazes requiring backtracking or complex spatial planning beyond the immediate next step.

C Algorithm

This appendix details the algorithm used to generate the maze reasoning dataset with reset demonstrations. The algorithm processes a base dataset of maze navigation problems and augments it with demonstration of incorrect attempts followed by resets and correct solutions.

Algorithm 1 Maze Reasoning Reset Data Generation - Main Process

Require: Base dataset D containing maze problems with:
- Adjacency list representation of 5×5 maze grid
- Origin and target coordinates
- Correct solution path

Ensure: Augmented dataset with reset demonstrations

Initialize empty datasets D_1 and D_2
for all example $e \in D$ **do**
  Extract adjacency list A, origin O, target T, and path P from e
  Count walls W around origin O
  if $W = 1$ **then**
    Add e to D_1
    Call ProcessOrder1(e) {See Algorithm 2}
  else if $W = 2$ **then**
    Add e to D_2
    Call ProcessOrder2(e) {See Algorithm 3}
  end if
end for
Combine processed examples from D_1 and D_2 into the final dataset

Algorithm 2 Order-1 Processing (1 wall at origin)

Procedure ProcessOrder1(example)
$WP \leftarrow \emptyset$ {Initialize wrong paths set}
for all adjacent node N to origin O **do**
  if $N \notin$ correct path P **then**
    for n_steps from max_n_steps down to 1 **do**
      Attempt to extend path from N until a dead end or n_steps are reached.
      if path length = n_steps or a dead end is reached **then**
        $WP \leftarrow WP \cup \{\text{path}\}$
        break
      end if
    end for
  end if
end for
for all path $p \in WP$ **do**
  Generate chain-of-thought steps for path p.
  Add “Heading in wrong direction” message.

  Add RESET marker.
end for
Append original correct solution (path P).
Format as conversation pairs.
End Procedure

Algorithm 3 Order-2 Processing (2 walls at origin)

Procedure ProcessOrder2(example)
for n_steps from max_n_steps down to 1 **do**
  Generate wrong path WP of length n_steps starting from O.
  if a valid path WP is found **then**
    Generate chain-of-thought for WP.
    if WP ends at a dead end (3 walls) **then**
      Add “Hit a dead end” message.
    else
      Add “Heading in wrong direction” message.
    end if
    Add RESET marker.
    break
  end if
end for
Append original correct solution (path P).
Format as conversation pairs.
End Procedure