PUZRIAN: SCALABLE PUZZLE REASONING FOR VISION-LANGUAGE-ACTION AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language models (VLMs) have shown remarkable progress as interactive agents. However, systematically evaluating their long-horizon spatial reasoning remains challenging. We introduce PuzRian, a scalable Gymnasium benchmark for interactive visual-spatial reasoning, featuring 3 task families with controllable complexity, action spaces, feedback, and observability. Evaluating 9 frontier and open-source VLMs reveals strong performance on symbolic tasks but substantial degradation as spatial and perceptual complexity increases, with near-zero success on challenging 6-piece puzzles. Controlled analysis further reveals that increasing interaction budgets alone provides limited benefit, while fine-grained feedback enables more effective error correction. We further find that performance is particularly sensitive to required spatial manipulations rather than the size of the action space, and that models become increasingly brittle at recovering from imperfect intermediate states as task complexity grows. PuzRian provides a controlled testbed for diagnosing how vision-language agents plan, use feedback, and recover from errors during long-horizon interaction.

1 INTRODUCTION

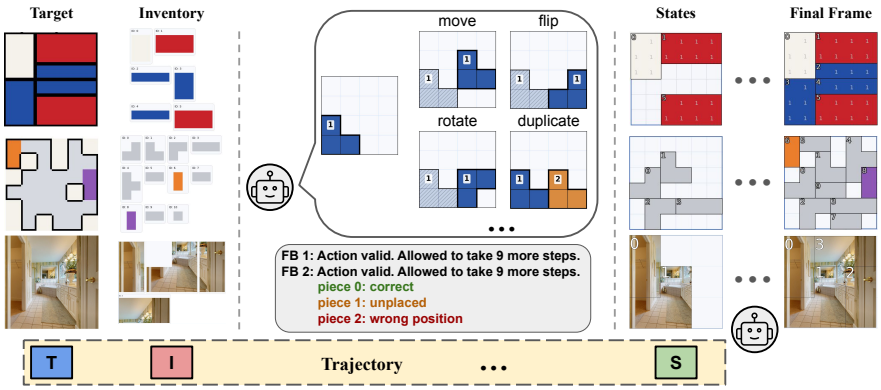


Figure 1: **Overview of PuzRian.** PuzRian consists of 3 scalable puzzle task families, where agents iteratively observe the puzzle state, take actions to reconstruct the target, and receive environment feedback. PuzRian supports a configurable action space and 2 feedback types: global and piece-level, for controlled evaluation of interactive visual-spatial reasoning.

Recent advances in foundation models have transformed vision-language models (VLMs) into interactive agents capable of multi-step reasoning, planning, and tool use (Singh et al., 2026; Comanici et al., 2025; Zeng et al., 2025). Modern VLMs have achieved remarkable success on static multimodal benchmarks (Liu et al., 2024; Lu et al., 2024; Yue et al., 2024), driving their deployment toward interactive settings such as web navigation and embodied manipulation (Zhou et al., 2024; Black et al., 2024). Unlike static visual reasoning, interactive agents must continually update their plans based on evolving states and environmental feedback.

While recent benchmarks have extended multimodal evaluation to interactive settings (Wang et al., 2025; 2026), systematically understanding what drives success and failure in interactive visual rea-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

soning remains challenging. Existing benchmarks often couple multiple sources of difficulty within fixed environments, making it difficult to trace the individual effects of task complexity, available information, and interaction capabilities on model behavior.

To enable such controlled analysis, we introduce **PuzRian**, a scalable and configurable Gymnasium benchmark for interactive visual-spatial reasoning. Agents iteratively reconstruct a target from an inventory of pieces through visual observations, actions, and environment feedback. PuzRian comprises three task families with increasing spatial and perceptual complexity: *Symbolic Assembly* with colored rectangular pieces, *Irregular Assembly* with complex piece geometries, and *Visual Assembly* with natural-image fragments. Its configurable task complexity, observations, feedback, and action spaces isolate factors shaping model behavior, while process-level metrics characterize reasoning beyond final task success.

Across 9 representative proprietary and open-source VLMs, frontier models nearly solve Symbolic Assembly, but performance degrades sharply on Irregular and Visual Assembly. Notably, even GPT-5.4 and Gemini-3.1 achieve near 0% success on Irregular-Hard, with only 6 pieces on a 6×6 grid. Controlled analysis reveals that this gap is not explained by a single factor. Additional interaction provides limited benefit without informative feedback, while required geometric transformations are more challenging than simply expanding the action space. Alternative input representations alleviate some perceptual difficulty but do not resolve failures at higher spatial complexity. Finally, agents struggle to recover from imperfect intermediate states and may prematurely abandon solvable instances. Together, these findings identify spatial manipulation, feedback utilization, and error recovery as key challenges for interactive vision-language agents.

2 Related Work

Driven by rapid advances in foundation models, recent vision-language models (VLMs) have demonstrated strong multimodal perception and reasoning capabilities (Li et al., 2022; Liu et al., 2023; Zhu et al., 2025; Comanici et al., 2025; Team et al., 2025), while vision-language-action (VLA) models extend these capabilities toward interactive decision-making and control (Black et al., 2024; Kim et al., 2025; Black et al., 2025; Fang et al., 2026). These advances increasingly enable vision-language systems to operate as interactive agents that must reason over evolving visual observations and act toward task goals, motivating systematic evaluation beyond static multimodal reasoning.

Vision-language agents increasingly operate in interactive domains such as web navigation, GUI automation, and embodied tasks (Zheng et al., 2024; Xie et al., 2024; Wang et al., 2023). Benchmarks such as WebArena (Zhou et al., 2024) and OSWorld (Xie et al., 2024) evaluate agents in realistic, long-horizon environments. However, these environments often entangle perception, planning, action selection, and state tracking, making controlled analysis difficult. PuzRian complements these benchmarks with a controlled setting for studying interactive visual-spatial reasoning.

Interactive environments have long supported the study of sequential decision-making, and more recently, interactive agents. From general RL platforms such as OpenAI Gym (Brockman et al., 2016; Towers et al., 2025) and Procgen (Cobbe et al., 2020) to embodied environments such as AI2-THOR (Kolve et al., 2017), Habitat (Savva et al., 2019), and ALFWorld (Shridhar et al., 2021), these environments provide controlled and scalable testbeds across diverse tasks. Built on Gymnasium, PuzRian brings this paradigm to configurable visual-spatial reasoning for vision-language agents.

Meanwhile, multimodal benchmarks have evolved from general visual question answering (Goyal et al., 2017; Hudson & Manning, 2019; Singh et al., 2019; Liu et al., 2024) toward increasingly complex visual reasoning tasks, including mathematical and multidisciplinary reasoning (Lu et al., 2024; Yue et al., 2024; Zhang et al., 2024). More targeted benchmarks such as VSP (Wu et al., 2024), Jigsaw-R1 (Wang et al., 2025), and VisPuzzle (Ivanitskiy et al., 2023) evaluate spatial reasoning and visual puzzle solving, while ARC-AGI (Chollet et al., 2024) studies generalization to previously unseen abstract visual reasoning problems. Together, these benchmarks highlight spatial and compositional reasoning as persistent challenges for multimodal models. Recent work has increasingly studied these capabilities in interactive settings, where models reason and act over multiple steps. In particular, VisGym (Wang et al., 2026) provides a diverse, customizable, and scalable suite of long-horizon visual environments for evaluation and training. PuzRian builds on this broader direction,

focusing on interactive visual-spatial assembly with configurable task complexity, observation and feedback settings, and action spaces, together with process-level analysis of model behavior.

3 METHODS

3.1 ENVIRONMENT DESIGN

PuzRian is a configurable Gymnasium benchmark (Towers et al., 2025), where agents iteratively observe the environment, perform actions, and receive feedback (Figure 1). Its configurable task, observation, feedback, and action settings enable controlled analysis of interactive reasoning.

PuzRian Interaction Protocol Each episode proceeds as a multi-turn interaction between the agent and environment. At step t, the agent receives the interaction history $H_t = (P, \{o_\tau, a_\tau, f_\tau\}_{\tau<t})$, where P denotes the task instruction, and o_τ, a_τ, and f_τ are observation, action, and feedback at step τ. Under the default fully observable setting, $o_t = (T, I_t, S_t)$, consisting of the target T, current inventory I_t, and playground state S_t. Observations are primarily visual, with task-specific textual representations provided when applicable. Observation components can be selectively hidden or modified to support partially observable settings. The episode terminates upon successful reconstruction, an explicit `stop()` action, or exhaustion of the interaction budget.

Environment Feedback PuzRian supports two levels of textual feedback. Global feedback reports episode-level information, including the remaining interaction budget and action execution status (e.g., invalid format, out-of-bounds, or execution failure). Piece-level feedback additionally provides status for individual pieces, including position and orientation correctness, and overlap status. Depending on the configuration, overlapping intermediate placements are either rejected while consuming interaction budget or allowed with their overlap status reported.

Action Space PuzRian represents actions as structured function calls with explicit arguments, providing a unified and interpretable interface across task families. Before each episode, the agent receives a natural-language specification of the available actions, argument constraints, and usage examples. The action set is configurable across benchmark settings and ablations. Invalid actions do not terminate the episode but return execution feedback indicating formatting errors, invalid arguments, or execution failures. The complete action space is summarized in Table 1.

Evaluation Metrics Final task success alone cannot fully characterize an agent's interactive spatial reasoning behavior. We therefore evaluate agents at both the episode and process levels. *Success Rate* measures whether the target is fully reconstructed within the interaction budget, while *Piece Accuracy*, *Cell IoU*, *Cell Recall*, and *Edit Distance* capture partial reconstruction quality. To characterize interaction dynamics, we further use *Recovery Rate* and *Preservation Rate*, measuring how often agents correct previously incorrect pieces and maintain previously correct pieces, respectively. We additionally report task-specific metrics for spatial transformations, error timing, and invalid interactions. Detailed definitions are provided in Appendix D.

Table 1: **PuzRian action space.** `place`, `move`, and `stop` form the basic action set shared across task families; `rotate` and `flip` perform spatial transformations; `swap`, `remove`, and `duplicate` provide additional editing operations; `inspect` and `giveup` support task-specific interaction settings.

Action	Effect
`place(id, r, c)`	Place a piece from the inventory, or reposition it to (r, c) as its anchor.
`move(id, u, d, l, r)`	Translate a placed piece by number of grid cells in each direction.
`stop()`	Terminate the episode and submit the current reconstruction.
`rotate(id, steps)`	Rotate a placed piece clockwise in multiples of 90°, preserving its anchor.
`flip(id, axis)`	Reflect a placed piece horizontally or vertically about its anchor.
`swap(id1, id2)`	Exchange the anchor positions of two placed pieces.
`remove(id)`	Remove a placed piece and return it to the inventory.
`duplicate(src, dst)`	Copy the shape of one piece onto another inventory piece.
`giveup()`	Terminate the episode when the puzzle is judged unrecoverable.
`inspect()`	Request an updated workspace observation in partially observable settings.

3.2 Task Families

PuzRian consists of three task families that share a unified interaction interface while progressively increasing perceptual and reasoning difficulty. In all tasks, the agent is given a target image and an inventory of geometric or visual fragments, and must reconstruct the target by executing a sequence of action functions. The current puzzle state is maintained on a larger playground board, allowing agents to temporarily stage, rearrange, and manipulate pieces before their final placement.

Puzzle instances are generated by formulating target decomposition as an exact-cover problem (Knuth, 2000). We first enumerate valid placements of candidate pieces over the target grid, then use a randomized Dancing Links (DLX) search to sample a non-overlapping decomposition with a specified number of pieces. The target and playground are generated independently, allowing their sizes and puzzle complexity to be controlled separately.

Symbolic Shape Assembly: Symbolic Shape Assembly uses synthetic colored rectangular pieces, minimizing perceptual ambiguity and emphasizing geometric spatial reasoning. The agent must determine the position of each piece from the target configuration and reconstruct it on the playground.

Irregular Puzzle Assembly: Irregular Puzzle Assembly replaces rectangular blocks with irregular polyomino pieces, introducing more complex piece geometry and spatial relationships. The agent must reason about piece shapes and their spatial arrangement to reconstruct the target configuration.

Visual Fragment Assembly: Visual Fragment Assembly replaces symbolic pieces with polyomino-shaped image fragments extracted from COCO dataset (Lin et al., 2014), while retaining the same underlying protocol. This setting combines spatial manipulation with visual correspondence, requiring agents to infer both where fragments belong and how they should be arranged.

4 Experiments

We evaluate 9 representative vision-language models spanning both open-source and proprietary systems, including GPT-5.4 (Singh et al., 2026), Gemini-3.1-Pro-Preview (Google DeepMind, 2026), Gemini-2.5-Pro (Comanici et al., 2025), Claude-Sonnet-4.6 (Anthropic, 2026), Grok-4.3 (xAI, 2025), Qwen3-VL-235B-A22B (Bai et al., 2025), Gemma-3-27B-PT (Team et al., 2025), GLM-4.6V (Zeng et al., 2025), and Inkling (Thinking Machines Lab, 2026). Unless otherwise specified, all models are evaluated using the same prompting template and generation settings, with comparable reasoning budgets where supported.

We evaluate models on standardized configurations across the three PuzRian task families and multiple difficulty levels. Difficulty scales along two complementary dimensions: puzzle complexity, through factors such as board size and number of pieces, and interaction complexity, through the available manipulation actions. Each configuration is evaluated under both global and piece-level feedback. Detailed benchmark configurations and evaluation settings are provided in Appendix B.1.

4.1 Main Results

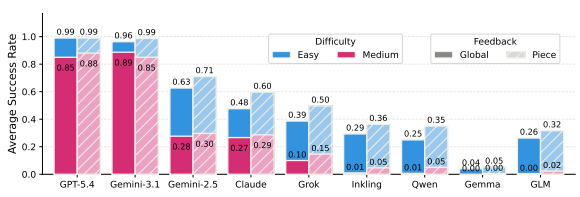


Figure 2: Average success rate across the 3 tasks under the Easy and Medium difficulty settings.

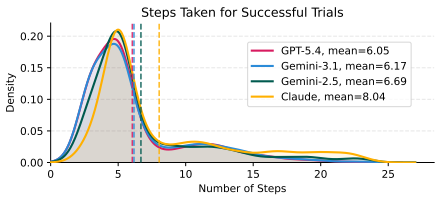


Figure 3: **Density curve of steps for successful trajectories.** Colored dashed line marks each model's mean number of steps.

Table 2 reports success rates across the three task families and difficulty levels. Figures 2 and 3 summarize average performance and trajectory lengths under global and piece-level feedback, respectively. Detailed results and metrics are provided in Appendix B.2 and D.

Table 2: **Main results across task families and difficulty levels. GF** and **PF** denote global and piece-level feedback respectively.

Models	Symbolic				Irregular						Visual					
	Easy		Medium		Easy		Medium		Hard		Easy		Medium		Hard	
	GF	PF	GF	PF	GF	PF	GF	PF	GF	PF	GF	PF	GF	PF	GF	PF
GPT-5.4	1.00	1.00	1.00	1.00	1.00	0.98	0.74	0.80	0.00	0.00	0.97	0.99	0.81	0.84	0.00	0.16
Gemini-3.1	1.00	1.00	1.00	0.96	0.92	0.96	0.81	0.61	0.00	0.00	0.97	1.00	0.85	0.98	0.00	0.26
Gemini-2.5	0.98	1.00	0.61	0.42	0.29	0.29	0.03	0.04	0.00	0.00	0.61	0.84	0.19	0.43	0.00	0.00
Claude	0.97	1.00	0.72	0.51	0.19	0.22	0.02	0.01	0.00	0.00	0.27	0.57	0.06	0.34	0.00	0.00
Grok-4.3	0.89	0.98	0.27	0.36	0.14	0.18	0.01	0.01	0.00	0.00	0.13	0.33	0.02	0.07	0.00	0.00
Inkling	0.80	0.96	0.03	0.12	0.03	0.08	0.00	0.01	0.00	0.00	0.05	0.05	0.00	0.01	0.00	0.00
Qwen	0.61	0.71	0.02	0.13	0.02	0.04	0.00	0.00	0.00	0.00	0.12	0.30	0.00	0.02	0.00	0.00
Gemma	0.08	0.12	0.00	0.00	0.00	0.01	0.0	0.0	0.00	0.00	0.04	0.03	0.00	0.01	0.00	0.00
GLM	0.67	0.79	0.01	0.07	0.02	0.02	0.00	0.00	0.00	0.00	0.10	0.14	0.00	0.00	0.00	0.00

Strong symbolic performance does not transfer to more complex spatial tasks. Overall, current frontier models perform strongly on Symbolic Shape Assembly, with GPT and Gemini-3.1 achieving near-perfect success rates. In contrast, performance drops substantially on Irregular Puzzle Assembly and Visual Fragment Assembly across models. These results reveal a substantial gap between structured symbolic spatial reasoning and reasoning over irregular geometry or visual fragments. As a small-scale sanity check, we additionally evaluate two human participants on selected configurations. Humans achieve success rates of 1.00 on *Symbolic-Medium*, 0.56 on *Irregular-Hard*, and 0.96 on *Visual-Hard*, suggesting that Irregular Assembly remains challenging even for humans, while a large gap persists on the harder Irregular and Visual settings (Appendix C.9).

Spatial transformations introduce a sharp increase in difficulty. Performance drops substantially when moving to the Hard settings, which introduce more complex spatial manipulation through actions such as `rotate`. This effect is particularly pronounced for Irregular and Visual Assembly: models that perform strongly on the Medium settings often achieve nearly 0 success once such transformations are required. The sharp degradation suggests that current VLMs struggle not only with increasing puzzle complexity, but particularly with reasoning and planning.

Fine-grained feedback improves interactive performance. Across most models and settings, piece-level feedback improves success over global feedback, with particularly clear gains on the more challenging Irregular and Visual tasks. This suggests that localizing errors to individual pieces helps agents identify incorrect intermediate states and revise their solutions more effectively. Nevertheless, the hardest configurations remain challenging even with piece-level feedback, with only a small number of successful episodes observed.

Taken together, these results raise a broader question: what limits current VLMs as interactive spatial reasoning becomes more challenging, and which aspects of interaction help overcome these limitations? We investigate these questions through a series of controlled analyses in the next section.

5 Controlled Analysis of Interactive Spatial Reasoning

5.1 Understanding Performance Degradation

The main results reveal substantial performance degradation as task difficulty increases, particularly when more complex spatial manipulation is required. Several factors may contribute to this behavior, including increased puzzle complexity, longer interaction horizons, and more demanding action spaces. We next disentangle the factors underlying this degradation through controlled analysis.

5.1.1 Scaling Puzzle Complexity and Interaction Budget.

Recent work suggests that increasing reasoning budgets can improve the performance of vision-language agents (Comanici et al., 2025; Xie et al., 2024). Here, we vary puzzle complexity and interaction budget to test whether additional interaction offsets increasing task difficulty. Experimental settings are described in Appendix C.1 and Table 5.

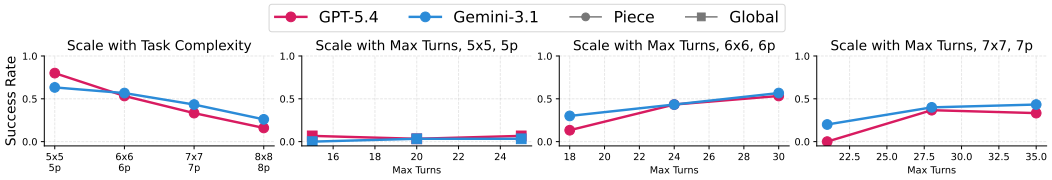


Figure 4: **Effect of Scaling Puzzle Complexity and Interaction Budget.** Circles and squares denote global and piece feedback. Labels such as "5×5, 5p" indicate a 5×5 board with 5 puzzle pieces.

Figure 4 shows that performance decreases consistently as puzzle complexity increases, with frontier models remaining reliable on simpler settings. We further study whether increasing the interaction budget can compensate for this degradation. Under global feedback, additional interaction steps provide little to no improvement. In contrast, piece-level feedback enables both models to benefit more from additional interaction steps, consistent with more effective iterative correction. However, performance remains limited as puzzle complexity increases, even with larger interaction budgets.

5.1.2 Spatial Manipulation and Action-Space Complexity.

Beyond puzzle complexity and interaction budget, difficulty may also arise from the action space itself, as real-world agents often operate with more actions that impose different reasoning demands. We therefore separately examine the effects of required actions and action-space size.

We first evaluate the effect of different manipulation primitives. In addition to the default basic actions, we individually introduce spatial transformation actions (`flip`, `rotate`) and object management actions (`duplicate`, `remove`). Detailed settings are provided in Appendix C.2. Figure 5 shows that spatial transformations, particularly `flip` and `rotate`, substantially reduce success, whereas object management actions such as `duplicate` have a smaller effect. This suggests that explicit geometric transformations are particularly challenging for current VLM agents.

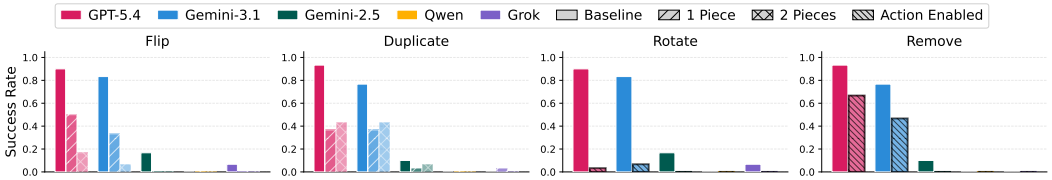


Figure 5: **Effect of Additional Actions.** Bars compare the baseline with action-augmented settings. For `flip` and `duplicate`, 1-piece and 2-piece variants indicate the number of pieces that require the corresponding action. For `rotate` and `remove`, only the action-enabled setting is evaluated.

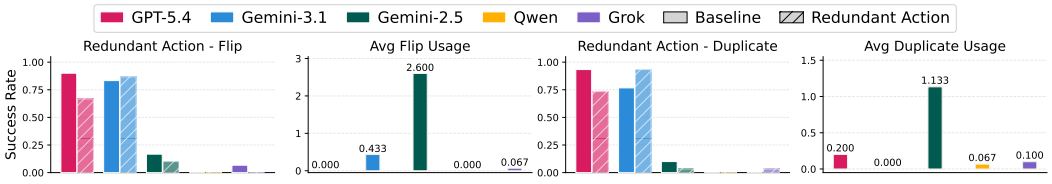


Figure 6: **Effect of Redundant Actions.** The introduced `flip` and `duplicate` actions are never required. We compare model performance with and without these redundant actions, and report the average usage of the redundant actions.

We next ask whether this degradation can instead be explained simply by more available actions. We augment the default action space with additional manipulation actions that are unnecessary for solving the puzzle, increasing the number of choices without changing the solution (Appendix C.2). Figure 6 shows that adding redundant actions has only a modest effect on final performance across models, although some models, such as Gemini-2.5, invoke unnecessary actions relatively frequently.

Together, these results suggest that performance is more sensitive to the reasoning required by specific spatial manipulations than to the size of the available action space alone.

5.2 Representation and Information

The preceding analysis shows that performance is sensitive to increasing spatial complexity and specific manipulation requirements. However, these tasks also place substantial demands on how spatial information is presented and accessed during interaction. We therefore examine how observation format, input representation, and information availability affect model performance.

5.2.1 Effect of Multi-Image Observation

PuzRian's default observation contains multiple visual inputs, requiring agents to integrate information across the target, piece inventory, and current workspace. Previous work has shown that multi-image reasoning can itself be challenging for VLMs (Zhao et al., 2024; Das et al., 2026). We therefore examine whether the multi-image observation format contributes substantially to the difficulty of the benchmark. We compare the default three-image observation with two merged-image variants while keeping all other evaluation settings unchanged. The three observation formats are summarized in Appendix C.3.

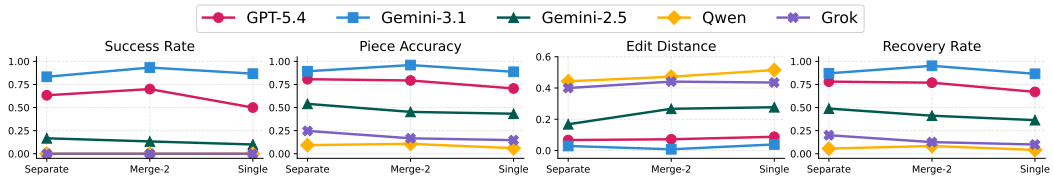


Figure 7: **Effect of Multi-Image Observation.** Three observation layouts are evaluated: **Separate**, **Merge-2** (target + inventory), and **Single** (all observations).

Figure 7 shows that observation format has only limited effect on performance. Frontier models achieve broadly comparable success rates across the three settings: Gemini-3.1 improves slightly under the two-image configuration, while merging all observations into a single image produces a small performance decrease across models. Overall, these results suggest that the use of multiple input images is not a primary explanation for the observed performance degradation.

5.2.2 Alternative Task Representations

We further examine whether making spatial information more explicit alleviates the difficulty of Irregular Assembly. We consider two variants: replacing visual observations with equivalent text representations, and augmenting visual observations with an explicit directional constraint between two colored pieces. Results are shown in Figure 8 and 9, and detailed settings are provided in Appendix C.4.

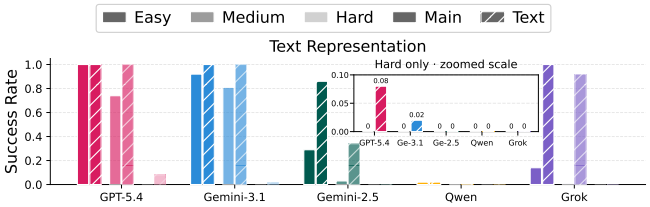


Figure 8: **Effect of text representation on Irregular Assembly.** Visual vs. text representations across difficulties, with *Irregular-Hard* enlarged.

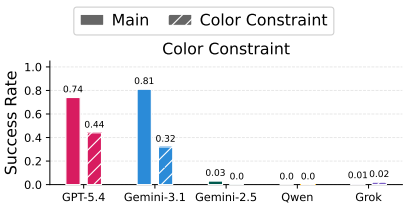


Figure 9: **Effect of color constraints.** *Irregular-Medium* with and without an additional color constraint.

Text representations improve performance on *Irregular-Medium*, particularly for Gemini-2.5 and Grok, but provide limited gains on *Irregular-Hard*. It suggests that visual interpretation contributes to the difficulty, while spatial planning remains the major bottleneck as task complexity increases.

In contrast, adding explicit spatial constraints substantially degrades performance, suggesting that models struggle to effectively integrate additional relational information into their planning.

5.2.3 Effect of Partial Observability

We next ask how much task-relevant information must remain available throughout interaction. While the default setting provides access to the target and environment feedback, real-world agents may need to act under incomplete observations and actively acquire missing information. We therefore progressively restrict access to the target and feedback, with an `inspect()` action allowing limited active information acquisition. Detailed configurations are provided in Appendix C.5.

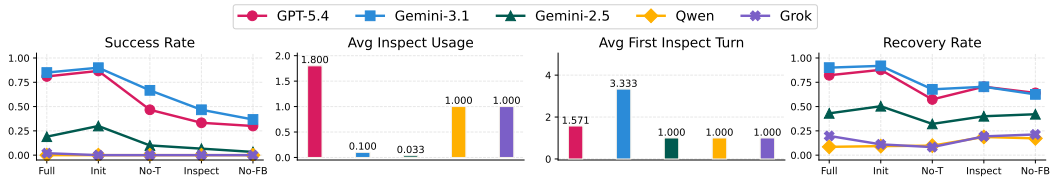


Figure 10: **Effect of Partial Observability.** We compare five observation settings: **Full** (fully observable), **Init** (target shown only at initialization), **No-T** (no target image after initialization), **Inspect** (no feedback with up to two `inspect` actions), and **No-FB** (no feedback or `inspect`).

As shown in Figure 10, performance generally degrades as less information is available. Showing the target only initially causes a modest drop, while removing target access produces a larger degradation. Restricting environment feedback leads to substantially larger performance losses, even with `inspect` actions. Together with the benefits of piece-level feedback in the main benchmark, these results suggest that informative feedback is particularly important for interactive spatial reasoning, allowing agents to adapt their actions based on the evolving environment state.

5.3 Interactive Reasoning Behavior

Beyond final task performance, we examine how models reason throughout multi-turn interaction. We study the benefit of iterative interaction, the ability to reason from partially completed states, and the recognition of task infeasibility.

5.3.1 Interactive vs. Non-agentic Reasoning

To examine the benefit of iterative interaction, we compare the standard interactive setting with a non-agentic variant, where models predict the complete action sequence from the initial observation without receiving intermediate states or feedback. We evaluate both variants on Easy and Medium settings of Irregular and Visual Assembly. Interaction provides only modest gains on Irregular Assembly, while substantially improves performance on Visual Assembly. This contrast suggests that interaction is particularly useful when models can exploit updated visual states and feedback to resolve the task incrementally, whereas it provides less benefit when the primary bottleneck is spatial planning itself. This is also consistent with our fine-grained results showing that models often struggle to recover from local errors with feedback. Detailed settings and results are provided in Appendix C.6.

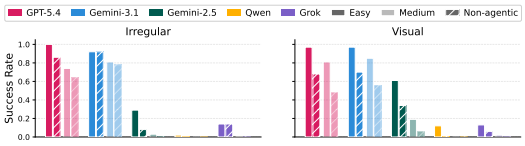


Figure 11: **Interactive vs. non-agentic reasoning.** Success rates on Easy and Medium Irregular and Visual settings.

5.3.2 Reasoning from Partial States

Interactive agents often need to continue from partially completed states rather than solve a task from scratch. We therefore initialize puzzles with different numbers of pieces already correctly placed and evaluate whether reducing the remaining work makes challenging instances substantially easier. Detailed settings are provided in Appendix C.7.

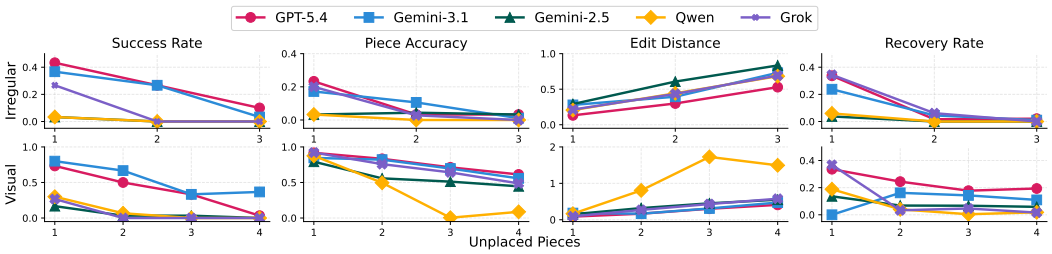


Figure 12: **Effect of Partial States.** The first row reports results on the *Irregular-Medium* setting, while the second row reports results on *Visual-Hard*. The x-axis denotes the number of unplaced pieces at the beginning.

As shown in Figure 12, performance improves as more pieces are correctly fixed in advance, but challenging instances remain difficult even when only a few pieces are left to place. More importantly, for the same number of pieces that remain to be placed, solving from a partial state of a more challenging puzzle is consistently harder than solving a puzzle with that number of pieces from scratch. This indicates that difficulty is not determined solely by the amount of remaining work: agents must reason about the existing partial configuration and integrate it with the remaining pieces to complete the global target.

5.3.3 Recognizing Task Infeasibility

Interactive agents are supposed to distinguish tasks that are truly infeasible from solvable tasks. We therefore introduce a `giveup` action and evaluate whether agents invoke it on unsolvable puzzles while avoiding it on solvable instances. Detailed settings are provided in Appendix C.8.

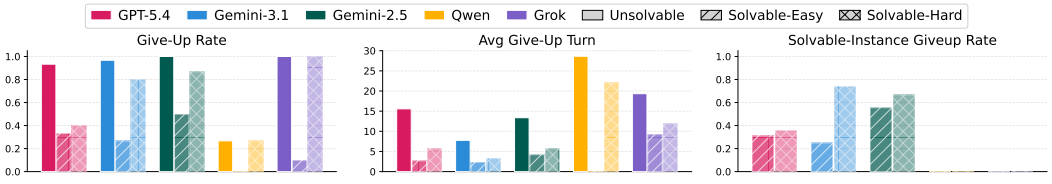


Figure 13: **Ability to Recognize Unsolvable Tasks.** The solvable-instance give-up rate: among instances solved by the same model under the baseline setting, the fraction on which it gives up after the `giveup` action is introduced.

The results are shown in Figure 13. On unsolvable puzzles, frontier models generally identify task infeasibility and invoke `giveup` correctly. However, when the same action is available on solvable puzzles, performance decreases noticeably. We observe that many models invoke `giveup` prematurely after making early mistakes or entering locally difficult states, despite the existence of a valid solution. Compared with the main result, they successfully complete these instances in most cases. This behavior is particularly evident on more challenging puzzles, where models appear to confuse temporary planning failures with true task infeasibility.

6 Conclusion

We introduced PuzRian, a scalable Gymnasium benchmark for evaluating long-horizon interactive visual-spatial reasoning in vision-language agents. Through configurable puzzle complexity, interaction settings, and process-level metrics, PuzRian enables controlled analysis of agent reasoning and adaptation. Our evaluations reveal that performance degrades substantially as spatial and perceptual complexity increases, and that more interaction or a larger action space alone does not resolve these failures. Instead, informative feedback, spatial manipulation, and recovery from imperfect intermediate states emerge as important factors in successful interaction. We hope PuzRian provides a useful testbed for understanding and improving these capabilities in future vision-language agents.

AI USE STATEMENT

In this work, we used generative AI tools for providing feedback on research methodology or experiments and implementing methods. We have not used generative AI tools for generating synthetic data sets, helping develop theoretical models or conceptual frameworks, proposing or refining hypotheses, cleaning and reformatting dataset, supporting qualitative and thematic data analysis, and interpreting results, and formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, assisting with translation are not applicable to this work. Additionally, we used generative AI tools for suggesting experimental parameters, creating or editing software code, drafting parts of a research paper, summarizing or analyzing existing literature, sourcing/searching for information and editing a research paper to improve readability. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by two authors. We also checked and modified all the LLM-generated drafts. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To support the reproducibility of our work, we provide the complete source code for PuzRian, including task and environment implementation, task-data generation, evaluation, and the default scripts used for the main experiments and ablation studies reported in this paper. The code is available at `https://anonymous.4open.science/r/puzrian_env-BCE7/README.md`. The README provides detailed, step-by-step instructions for setting up the environment, generating task instances with different configurations, and running the experiments for each task family and ablation setting. We also document the corresponding experimental configurations and evaluation procedures to facilitate reproduction of the reported results and further evaluation of vision-language agents under different PuzRian settings. We additionally provide an interactive playground for human evaluation on selected PuzRian tasks at `https://playground.puzrian.com/`.

REFERENCES

Anthropic. Introducing claude sonnet 4.6. `https://www.anthropic.com/news/claude-sonnet-4-6`, 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025. URL `https://proceedings.mlr.press/v305/black25a.html`.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Francois Chollet, Mike Knoop, Gregory Kamradt, and Bryan Landers. Arc prize 2024: Technical report. *arXiv preprint arXiv:2412.04604*, 2024.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pp. 2048–2056. PMLR, 2020.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Anurag Das, Adrian Bulat, Alberto Baldrati, Ioannis Maniadis Metaxas, Bernt Schiele, Georgios Tzimiropoulos, and Brais Martinez. More images, more problems? a controlled analysis of vlm failure modes. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 7423–7442, 2026.

Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, et al. Molmoact2: Action reasoning models for real-world deployment. *arXiv preprint arXiv:2605.02881*, 2026.

Google DeepMind. Gemini 3.1 pro. `https://deepmind.google/models/model-cards/gemini-3-1-pro/`, 2026.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Drew A. Hudson and Christopher D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

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

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025. URL `https://proceedings.mlr.press/v270/kim25c.html`.

Donald E Knuth. Dancing links. *arXiv preprint cs/0011047*, 2000.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pp. 12888–12900. PMLR, 2022.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pp. 216–233. Springer, 2024.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations*, volume 2024, pp. 23439–23554, 2024.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9338–9346. IEEE, 2019.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2010.03768.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2026.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8317–8326, 2019.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Thinking Machines Lab. Inkling: Our open-weights model. https://thinkingmachines.ai/news/introducing-inkling/, 2026.

Mark Towers, Ariel Kwiatkowski, John Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Kallinteris Andreas, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, J Terry, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025. doi: 10.52202/085713-4916. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/d7ff1795e8527f6443371c3933bdb52b-Paper-Datasets_and_Benchmarks_Track.pdf.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Zifu Wang, Junyi Zhu, Bo Tang, Zhiyu Li, Feiyu Xiong, Jiaqian Yu, and Matthew B Blaschko. Jigsaw-r1: A study of rule-based visual reinforcement learning with jigsaw puzzles. *arXiv preprint arXiv:2505.23590*, 2025.

Zirui Wang, Junyi Zhang, Jiaxin Ge, Long Lian, Letian Fu, Lisa Dunlap, Ken Goldberg, XuDong Wang, Ion Stoica, David M Chan, et al. Visgym: Diverse, customizable, scalable environments for multimodal agents. *arXiv preprint arXiv:2601.16973*, 2026.

Qiucheng Wu, Handong Zhao, Michael Saxon, Trung Bui, William Yang Wang, Yang Zhang, and Shiyu Chang. Vsp: Assessing the dual challenges of perception and reasoning in spatial planning tasks for vlms. *arXiv preprint arXiv:2407.01863*, 2024.

xAI. Grok 4 model card. https://data.x.ai/2025-08-20-grok-4-model-card.pdf, 2025.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9556–9567, 2024.

Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, et al. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint arXiv:2508.06471*, 2025.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Yu Qiao, et al. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? In *European Conference on Computer Vision*, pp. 169–186. Springer, 2024.

Bingchen Zhao, Yongshuo Zong, Letian Zhang, and Timothy Hospedales. Benchmarking multi-image understanding in vision and language models: Perception, knowledge, reasoning, and multi-hop reasoning. *arXiv preprint arXiv:2406.12742*, 2024.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v (ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*, 2024.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

A Evaluation Details

All models are evaluated using the same task instructions and interaction protocol. We query GPT models through the OpenAI API, Gemini models through the Gemini API, selected models through OpenRouter, and Inkling through the Thinking Machines API. We report the native reasoning configuration used for each model because reasoning controls differ across providers and are not directly comparable.

Table 3: Model and inference configurations used throughout the evaluation. A dash indicates that the corresponding parameter is not exposed or not applicable. All reasoning configurations and maximum output-token limits are specified per interaction turn (i.e., per model call).

Model	Provider	Temp.	Top-p	Reasoning	Max Tokens
GPT-5.4	OpenAI	–	–	Medium	20000
Gemini-3.1	Gemini	0.3	0.95	Budget: 4096	20000
Gemini-2.5	Gemini	0.3	0.95	Budget: 4096	20000
Claude	Amazon Bedrock	–	–	Budget: 4096	20000
Grok-4.3	OpenRouter	0.3	0.95	Budget: 4096	20000
Inkling	Thinking Machines	0.3	0.95	Medium	20000
Qwen	Amazon Bedrock	0.3	0.95	–	20000
Gemma	OpenRouter	0.3	0.95	–	20000
GLM	OpenRouter	0.3	0.95	–	20000

B Main Experiments

B.1 Main Benchmark Configurations

Table 4 summarizes the default benchmark configurations used throughout the paper.

The default benchmark is designed to focus on spatial reasoning rather than action engineering. Unless otherwise specified, the action space consists of `place`, `move`, and `stop`, with `rotate` additionally enabled only in the highest difficulty setting. More expressive action spaces are evaluated separately in the action-space ablations.

All benchmark configurations are evaluated under both global and piecewise feedback. Harder settings use larger action budgets to provide sufficient opportunities for correction and refinement. Unless otherwise specified, each Easy and Medium configuration is evaluated on 100 independently generated puzzle instances, while each Hard configuration is evaluated on 50 instances. All evaluated models are tested on the same set of puzzle instances for each benchmark configuration to ensure a fair comparison.

Table 4: Benchmark configurations used for the main results in Table 2.

Task	Difficulty	Grid Size	Pieces	Turns	Available Actions
Symbolic	Easy	5×5	5	15	`place`, `move`, `stop`
	Medium	6×6	8	24	`place`, `move`, `rotate`, `stop`
Irregular	Easy	4×4	3	9	`place`, `move`, `stop`
	Medium	4×5	5	15	`place`, `move`, `stop`
	Hard	6×6	6	30	`place`, `move`, `rotate`, `stop`
Visual	Easy	4×4	3	9	`place`, `move`, `stop`
	Medium	5×5	5	15	`place`, `move`, `stop`
	Hard	8×8	8	40	`place`, `move`, `rotate`, `stop`

B.2 Additional Main Benchmark Results

Beyond final success rates, we report fine-grained metrics that capture intermediate reconstruction quality, recovery dynamics, and task-specific interaction behavior. Figure 15, Figure 16, and Fig-

ure 17 report these results for the Symbolic, Irregular, and Visual task families, respectively. Formal definitions of all evaluation metrics are provided in Appendix D.

Fine-Grained Performance Patterns. The fine-grained metrics provide a more nuanced view of model behavior beyond episode-level success. Piece accuracy provides a more continuous measure of local reconstruction quality, revealing partial progress even when the full puzzle is not solved. This is particularly evident for Visual Assembly under piece-level feedback, whereas piece accuracy remains near zero on the most challenging Irregular settings, indicating substantial difficulty in planning valid piece placements. Cell IoU and Cell Recall provide smoother measures of overall reconstruction quality, while Edit Distance shows that predictions in the hardest settings can remain substantially different from the target even after interaction.

The interaction-level metrics further characterize how these failures arise. Models generally place most pieces onto the playground rather than leaving them unplaced, suggesting that they tend to commit to placements and repeatedly adjust them even when feedback indicates remaining errors. Out-of-bound actions remain relatively uncommon, although their frequency increases somewhat with task difficulty. Recovery and Preservation Rates reveal an additional asymmetry: models are generally better at preserving already-correct placements than recovering from incorrect ones. Piece-level feedback improves these behaviors, particularly on more difficult settings, suggesting that successful iterative correction relies substantially on localized environmental feedback rather than accurate planning alone.

C Detailed Experiment Settings

C.1 Scaling Puzzle Complexity and Action Budget

We evaluate GPT and Gemini-3.1, the two strongest models in the main benchmark, on the Visual task. For the complexity-scaling experiment, we use piece-level feedback and a fixed action space consisting of `place`, `move`, `rotate`, and `stop`. We focus on the Visual task because Irregular puzzles may admit multiple valid solutions, making piece-level feedback less reliable for evaluating intermediate states. Detailed configurations are provided in Table 5.

To scale puzzle complexity, we increase the target grid size from 5×5 to 8×8, together with the number and maximum size of puzzle pieces. We set the maximum interaction budget to five times the number of pieces, providing additional opportunities for correction as puzzle complexity increases.

For the interaction-budget study, we vary the maximum number of turns while holding puzzle complexity fixed. Under global feedback, we evaluate the 5×5 setting with budgets of 15, 20, and 25 turns. We then repeat the study under piece-level feedback on increasingly challenging puzzle settings, increasing the interaction budget until additional turns yield only marginal improvements. We evaluate up to the 7×7 setting, where performance begins to saturate despite the increased interaction budget.

For each setting, all evaluated models are tested on the same set of 30 puzzle instances to ensure a fair comparison.

Table 5: Experimental settings for scaling puzzle complexity and interaction budget. All experiments use the Visual task with `place`, `move`, `rotate`, and `stop`.

Setting	Feedback	Grid Size	Pieces	Piece Size	Max Turns
Complexity	Piece	5×5	5	3–8	25
	Piece	6×6	6	3–8	30
	Piece	7×7	7	3–10	35
	Piece	8×8	8	5–10	40
Interaction Budget	Global	5×5	5	3–8	15, 20, 25
	Piece	6×6	6	3–8	18, 24, 30
	Piece	7×7	7	3–10	21, 28, 35

C.2 Spatial Manipulation and Action-Space Complexity

Required Manipulation Actions. To evaluate different manipulation actions, we introduce one additional required action while keeping the remaining settings unchanged. The `flip` and `rotate` experiments are based on *Visual-Medium*, whereas `duplicate` and `remove` are based on *Irregular-Medium*, as these actions require additional pieces. The maximum interaction budget is set to $5\times$ the number of pieces.

For `flip` and `duplicate`, we control the number of pieces requiring the corresponding action and evaluate both one- and two-piece variants. The `rotate` setting follows the *Visual-Hard* task design while fixing the puzzle size to a 5×5 grid with five pieces. For `remove`, all pieces are initially scattered on the playground outside the target region together with one redundant piece. The agent must identify and remove the redundant piece before reconstructing the target.

Redundant Actions. To isolate the effect of action-space size from the difficulty of executing a required operation, we additionally expose `flip` or `duplicate` while ensuring that neither action is required by the optimal solution. All other settings remain identical to the corresponding baseline.

Unless otherwise specified, all experiments use global feedback, and each model is evaluated on the same set of 30 puzzle instances for each setting.

Results and Discussion. The results further distinguish the difficulty of executing spatial transformations from that of simply operating over a larger action space. When additional actions such as `rotate`, `flip`, `duplicate`, or `remove` are required to reach the solution, performance decreases substantially. In contrast, exposing additional actions that are not required by the solution causes a much smaller degradation. Thus, the performance drop cannot be explained primarily by the increased number of available actions.

Instead, the results suggest that models struggle when solving the task requires reasoning about how an action changes the spatial configuration. Such actions introduce additional dependencies between identifying the appropriate piece, selecting the correct transformation, and subsequently placing it consistently with the global target. This provides a more challenging form of spatial reasoning than choosing among additional but unnecessary actions.

C.3 Multi-Image Observation

We conduct this analysis on the *Visual-Medium* setting, which contains the richest visual information and therefore presents the greatest challenge for multi-image reasoning. To isolate the effect of observation format, we keep the task, interaction protocol, prompts, and evaluation settings unchanged, varying only how visual observations are presented to the model.

By default, PuzRian provides three separate images corresponding to the target, inventory, and playground. We compare this *Separate* setting with two merged-image variants. In *Merge-2*, the target and inventory are combined while the playground remains separate. In *Single*, all three observations are combined into one image. All input images are resized to 512×512, controlling for image resolution across settings.

For all settings, each model is evaluated on the same set of 30 *Visual-Medium* puzzles.

C.4 Alternative Task Representations

Text Representation. To examine whether performance degradation arises from the visual representation of puzzle states, we construct a text-only variant of Irregular Puzzle Assembly. We evaluate all difficulty settings using the same puzzle configurations, action space, interaction budget, and global feedback settings as in the main benchmark, but replace the target, inventory, and playground images with structured textual representations.

The target and playground are represented as grids, where `#` and `.` denote occupied and empty cells, respectively. Each inventory piece is represented by its bounding-box dimensions, current orientation, and binary occupancy pattern. The current location of each piece is additionally provided through its piece pose. After each action, the textual state is updated using the same environment

transition and feedback mechanisms as in the visual setting. Thus, the underlying puzzle and interaction protocol remain unchanged, while visual perception is removed from the observation. Detailed text representation settings are shown in Figure 14.

Results and Discussion. The results for text representation are shown in Figure 8. Replacing visual observations with structured text generally improves performance on *Irregular-Medium*, with particularly large gains for Gemini-2.5 and Grok. This suggests that explicitly exposing the underlying spatial structure facilitates reasoning, indicating that the visual modality itself introduces non-trivial difficulty for VLMs despite Irregular Assembly task is very straightforward in terms of perception for human.

However, the benefit of text representation becomes considerably smaller on *Irregular-Hard*, where performance remains low even when the target, pieces, and current state are explicitly represented in text. This suggests that removing much of the visual interpretation burden is insufficient as task complexity increases, and that the difficulty of harder instances arises increasingly from the underlying spatial planning problem rather than from visual representation alone.

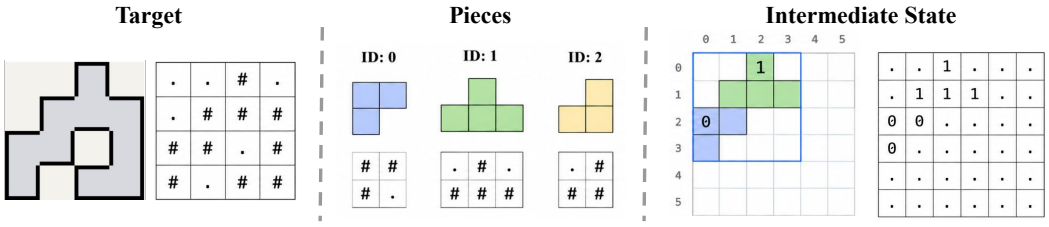


Figure 14: **Example of the text representation.** The visual target, inventory pieces, and intermediate playground state are converted into equivalent grid-based text representations. The underlying puzzle state and interaction protocol remain unchanged.

Color-Constraint Representation. We further investigate whether providing an explicit spatial relation between pieces facilitates planning. This analysis is conducted on *Irregular-Medium*, with all other benchmark settings unchanged.

For each puzzle, we select one directional relation from $\{$*left of*, *right of*, *above*, *below*$\}$. We enumerate all ordered pairs of distinct pieces whose ground-truth bounding boxes strictly satisfy the selected relation. For example, piece A is *left of* piece B only if the rightmost column of A lies strictly to the left of the leftmost column of B. Among all valid pairs, we select the pair with the largest separation along the corresponding spatial axis. The two selected pieces are assigned distinct colors (orange and cyan) in the inventory, while all remaining pieces retain the default appearance. The target image remains unchanged and does not reveal the piece colors.

The task instruction additionally specifies the corresponding directional relation between the two colored pieces (e.g., "the orange piece should be to the left of the cyan piece"). Thus, the underlying puzzle remains unchanged, while the agent receives one additional relational cue about the ground-truth arrangement.

Results and Discussion. As shown in Figure 9, introducing the additional color-based spatial constraint substantially reduces performance, despite the constraint providing valid information about the ground-truth arrangement. In principle, the directional relation should reduce the space of possible placements by specifying the relative position of two pieces. Empirically, however, models fail to effectively exploit this additional information.

This result suggests that providing more task-relevant information does not necessarily simplify spatial planning. Unlike the original setting, models must additionally identify the two colored pieces, maintain their specified relation, and integrate this constraint with the remaining geometric constraints when constructing the global arrangement. The additional reasoning burden appears to cover the informational benefit of the spatial hint, leading to worse overall performance. This behavior further highlights a limitation in integrating explicit relational constraints into multi-step spatial planning.

C.5 Partial Observability

We conduct this analysis on the *Visual-Medium* setting. To isolate the effect of partial observability, we keep the puzzle instances, prompts, interaction protocol, and evaluation settings unchanged while varying only the information available to the agent.

Starting from the default fully observable (*Full*) setting, we evaluate four increasingly restricted observation settings. *Init* provides the target image only at the beginning of the episode, requiring the agent to retain the target information throughout interaction. *No-T* removes access to the target after initialization. *Inspect* hides the playground and environmental feedback by default, but allows the agent to invoke `inspect()` up to two times during an episode to obtain the current playground and feedback. Finally, *No-FB* removes access to environmental feedback and the current playground throughout the episode, leaving the agent to act from the initial observation and interaction history alone.

For all settings, each model is evaluated on the same set of 30 puzzle instances.

Results and Discussion. The results confirm that restricting access to the target and environment feedback progressively makes interaction more difficult, consistent with the findings discussed in the main text. Beyond final performance, we observe notable differences in how models use the `inspect` action when it is available.

In particular, models do not appear to use inspection wisely. Gemini models often avoid `inspect` altogether, whereas several other models tend to use both available inspections near the beginning of the episode. Although early inspection provides additional state information, using the limited inspection budget immediately leaves no opportunity to query the environment later, when observations could be more informative for correcting intermediate decisions. These behaviors suggest that current models not only struggle under partial observability, but also have difficulty deciding when to actively acquire information.

C.6 Interactive vs. Non-Agentic Reasoning

We compare the standard interactive setting with a non-agentic variant on *Irregular-Easy*, *Irregular-Medium*, *Visual-Easy* and *Visual-Medium* with global feedback. In the interactive setting, the model executes one action at each step and receives the updated observation and environment feedback before selecting its next action.

In the non-agentic setting, the model receives only the initial puzzle observation and is asked to generate the complete sequence of actions in a single response. The predicted actions are then executed sequentially in the environment without providing intermediate observations or feedback to the model. We otherwise keep the puzzle instances, action space, and evaluation settings identical to the interactive setting.

For each setting, each model is evaluated on the same set of 50 puzzle instances. We increase the max tokens per output for each model to be 65535 in non-agentic settings. The non-agentic setting uses an enlarged output-token limit to reduce truncation from generating the complete action sequence in a single response.

Results and Discussion. Figure 11 reveals a clear difference between Irregular and Visual Assembly. Removing interaction causes a relatively modest performance drop on Irregular Assembly, whereas the degradation is much larger on Visual Assembly. This suggests that the value of interaction depends on the underlying source of task difficulty.

For Irregular Assembly, the primary challenge appears to lie in planning a globally consistent arrangement of pieces. Although interaction provides updated states and feedback after each action, these signals do not necessarily resolve the underlying planning difficulty. This observation is consistent with our fine-grained analysis: models often spend multiple turns adjusting an incorrect piece after receiving negative feedback, but still fail to recover a correct configuration. Consequently, access to additional interaction alone provides only small improvement when the model cannot effectively translate feedback into corrective spatial reasoning.

The larger gap on Visual Assembly suggests a different role for interaction. Planning a complete solution directly from the initial observation requires models to simultaneously identify visually complex fragments, reason about their spatial relationships, and determine the full sequence of placements. Interactive execution instead decomposes this process across multiple steps: after each placement, models can observe the updated state and use environment feedback to guide subsequent decisions. Thus, interaction can reduce the burden of jointly resolving perception and planning over all pieces at once. Together, these results suggest that interaction is most beneficial when intermediate observations and feedback provide actionable information that can simplify subsequent decisions, rather than merely additional opportunities to revise an otherwise difficult spatial plan.

C.7 Reasoning from Partial States

To evaluate whether models can reason from imperfect intermediate states, we initialize each episode from a partially completed puzzle rather than from the default empty playground. A subset of pieces is pre-placed on the playground, including both correctly and incorrectly positioned pieces, while the remaining pieces stay in the inventory. The model receives the resulting state as its initial observation and must complete the puzzle using the same interaction protocol and action space as in the main benchmark.

We construct partial states with different proportions of correctly placed pieces to control the quality of the initial state. Correctly placed pieces match their ground-truth locations, whereas incorrectly placed pieces are placed at valid but non-ground-truth locations. All other benchmark settings are kept unchanged.

For each setting, each model is evaluated on the same set of 30 puzzle instances.

C.8 Recognizing Task Infeasibility

We evaluate whether agents can distinguish genuinely infeasible tasks from solvable tasks in which their current intermediate state may be difficult to recover from. We augment the action space with a `giveup` action, which terminates the episode and indicates that the agent believes the puzzle cannot be completed.

We evaluate two complementary settings. In the *Unsolvable* setting, we construct puzzle instances for which no valid reconstruction of the target exists. A correct response therefore requires the agent to recognize the infeasibility and invoke `giveup`. In the *Solvable* setting, we use standard feasible puzzle instances while exposing the same `giveup` action. Here, the agent should solve the puzzle without prematurely invoking `giveup`.

All other interaction settings remain unchanged. For each setting, each model is evaluated on the same set of 30 puzzle instances.

C.9 HUMAN EVALUATION

We conduct a small-scale human evaluation to provide a sanity check on the solvability and practical difficulty of PuzRian. Two participants independently solve instances from *Symbolic-Medium*, *Irregular-Hard*, and *Visual-Hard* using the same puzzle environment. Both participants were members of the research team and voluntarily participated in the evaluation. We recorded only task performance and interaction trajectories; no personally identifiable or sensitive information was collected. This small-scale evaluation is intended only to verify that the tasks are practically solvable by humans and to understand model performance, rather than to establish a rigorous human baseline.

As shown in Table 6, the 3 task families exhibit different levels of difficulty for human participants. *Symbolic-Medium* is solved perfectly and with short trajectories, consistent with its explicit colored target providing strong information about the desired placement of each piece. In contrast, *Irregular-Hard* remains challenging even for humans: participants spend the most time on this setting while achieving only a 0.56 success rate. This is consistent with the greater planning burden introduced by the larger grid, increased number of irregular pieces, and required spatial transformations (rotate action). Nevertheless, human performance remains substantially above that of the evaluated models, which achieve near-zero success on this setting.

Interestingly, humans achieve a 0.96 success rate on *Visual-Hard* despite its larger grid and number of pieces. The visual content of each fragment provides local correspondence with the target image, allowing participants to identify plausible piece locations without solving the entire spatial arrangement jointly. The remaining failures primarily arise from ambiguous small fragments with limited distinctive visual content. This contrasts sharply with current models and suggests that effectively exploiting local visual correspondence remains an important challenge for visual-spatial reasoning agents.

Table 6: **Human evaluation.**

Task	Succ.	Time (s)	Turns
Symbolic-M	1.00	19.7	11.7
Irregular-H	0.56	76.2	16.1
Visual-H	0.96	64.1	18.3

D EVALUATION METRICS

This section formally defines the evaluation metrics used throughout our experiments.

D.1 NOTATION

Let $\mathcal{E} = \{E_1, \ldots, E_N\}$ denote the set of evaluation episodes, where $N = |\mathcal{E}|$.

For each episode E_i:

- $R_i \in \{0, 1\}$ denotes whether the episode is successfully completed.
- $\mathcal{P}_i = \{P_1^i, \ldots, P_{n_i}^i\}$ denotes the set of required puzzle pieces.
- $F_k^i \in \{0, 1\}$ denotes whether the first placement of piece P_k^i is correct.
- $L_k^i \in \{0, 1\}$ denotes whether the final placement of piece P_k^i is correct.
- $\mathcal{T}_i = \{T_1^i, \ldots, T_{m_i}^i\}$ denotes the set of interaction turns.
- A_j^i denotes the action executed at turn j.
- $\mathrm{Token}(T_j^i)$ denotes the number of output tokens generated at turn j.

Across the evaluation set, let

$$|EP| = \sum_{i=1}^{N} |\mathcal{P}_i|, \qquad |ET| = \sum_{i=1}^{N} |\mathcal{T}_i|$$

denote the total number of required pieces and interaction turns, respectively.

For geometric metrics, Pos_k^i and Deg_k^i denote the final position and normalized rotation of piece P_k^i, while $GT(Pos_k^i)$ and $GT(Deg_k^i)$ denote the corresponding ground-truth values.

Throughout this section, $\mathbf{1}[\cdot]$ denotes the indicator function.

D.2 METRICS

Episode Success Rate. Episode Success Rate measures the fraction of episodes that are successfully completed:

$$\texttt{success_rate} = \frac{1}{N}\sum_{i=1}^{N} R_i.$$

Piece Accuracy. Piece Accuracy measures the average fraction of required pieces that are correctly placed in the final state:

$$\texttt{piece_accuracy} = \frac{1}{N}\sum_{i=1}^{N} \frac{\sum_{k=1}^{|\mathcal{P}_i|} L_k^i}{|\mathcal{P}_i|}.$$

Cell IoU. Cell IoU measures the overlap between the occupied cells in the final prediction and the target assembly. Let C_i denote the set of occupied grid cells in the final prediction and G_i denote the set of occupied cells in the target. Then

$$\texttt{cell_iou} = \frac{1}{N}\sum_{i=1}^{N} \frac{|C_i \cap G_i|}{|C_i \cup G_i|}.$$

Cell Recall. Cell Recall measures the fraction of target cells covered by the final prediction:

$$\texttt{cell_recall} = \frac{1}{N}\sum_{i=1}^{N} \frac{|C_i \cap G_i|}{|G_i|}.$$

Unlike Cell IoU, Cell Recall does not penalize additional occupied cells outside the target configuration.

Edit Distance. Edit Distance measures the discrepancy between the predicted and ground-truth piece configurations. For each placed piece P_k^i, the rotation error is defined as

$$D_r^i(k) = \mathbf{1}\left[Deg_k^i \neq GT(Deg_k^i)\right].$$

After aligning the predicted piece to its ground-truth orientation, let Δx_k^i and Δy_k^i denote its horizontal and vertical displacement from the ground-truth position. The normalized position error is

$$D_p^i(k) = \frac{|\Delta x_k^i|}{W_i} + \frac{|\Delta y_k^i|}{H_i},$$

where W_i and H_i denote the target width and height of episode i, respectively.

If a required piece is never placed, its total edit distance is defined as

$$D_r^i(k) + D_p^i(k) = 2.$$

The overall metric is

$$\texttt{edit_distance} = \frac{\sum_{i=1}^{N}\sum_{k=1}^{|\mathcal{P}_i|}\left(D_r^i(k) + D_p^i(k)\right)}{|EP|}.$$

Grid Match Episode Rate. Grid Match Episode Rate measures the fraction of episodes in which the occupied cells in the final state exactly match the target occupancy pattern, while ignoring piece identity and appearance:

$$\texttt{grid_match_episode_rate} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[C_i = G_i].$$

Content Match Episode Rate. Content Match Episode Rate measures the fraction of episodes in which the final reconstruction exactly matches the target according to the environment's complete target-matching criterion:

$$\texttt{content_match_episode_rate} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[\text{final assembly exactly matches the target}].$$

Grid Overlap Episode Rate. Grid Overlap Episode Rate measures the fraction of episodes whose final state contains at least one grid cell occupied by multiple pieces. Let o_i denote the number of overlapping cells in the final state. Then

$$\texttt{grid_overlap_episode_rate} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[o_i > 0].$$

This metric measures collisions between pieces and does not compare the predicted assembly against the target configuration.

Outcome Transitions. To characterize how piece-level correctness changes throughout interaction, we compare the correctness of each piece at its first placement with its correctness in the final state. For $a, b \in \{0, 1\}$, define

$$N_{a\to b} = \sum_{i=1}^{N}\sum_{k=1}^{|\mathcal{P}_i|} \mathbf{1}\left[F_k^i = a \wedge L_k^i = b\right].$$

The corresponding transition rate is

$$\texttt{transition}(a, b) = \frac{N_{a\to b}}{|EP|}.$$

The four possible transitions are

$$0\to 0, \qquad 0\to 1, \qquad 1\to 0, \qquad 1\to 1,$$

corresponding to persistent failure, successful recovery, degradation of a correct placement, and preservation of a correct placement, respectively.

Recovery Rate. Recovery Rate measures the fraction of initially incorrect piece placements that become correct by the end of the episode:

$$\texttt{recovery_rate} = \frac{N_{0\to 1}}{N_{0\to 0} + N_{0\to 1}}.$$

Thus, Recovery Rate is conditioned on pieces whose first placement is incorrect and measures the agent's ability to recover from placement errors. When there are no initially incorrect pieces, we define the Recovery Rate as zero.

Preservation Rate. Preservation Rate measures the fraction of initially correct piece placements that remain correct in the final state:

$$\texttt{preservation_rate} = \frac{N_{1\to 1}}{N_{1\to 0} + N_{1\to 1}}.$$

Thus, Preservation Rate is conditioned on pieces whose first placement is correct and measures the agent's ability to preserve correct intermediate progress. When there are no initially correct pieces, we define the Preservation Rate as zero.

Boundary Action Rate. Boundary Action Rate measures the fraction of interaction turns in which the executed action attempts to place or move a piece beyond the valid playground boundary:

$$\texttt{boundary_action_rate} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{|\mathcal{T}_i|} \mathbf{1}[A_j^i \text{ exceeds the playground boundary}]}{|ET|}.$$

Average Number of Unplaced Pieces. Let u_i denote the number of required pieces that remain unplaced at the end of episode i. The average number of unplaced pieces is

$$\texttt{unplaced_piece_per_episode} = \frac{1}{N}\sum_{i=1}^{N} u_i.$$

Maximum Token Error Rate. Maximum Token Error Rate measures the fraction of interaction turns for which the model-generated response exceeds the maximum output-token budget:

$$\texttt{max_token_error_rate} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{|\mathcal{T}_i|}\mathbf{1}\left[\mathrm{Token}(T_j^i) > \texttt{max_token}\right]}{|ET|}.$$

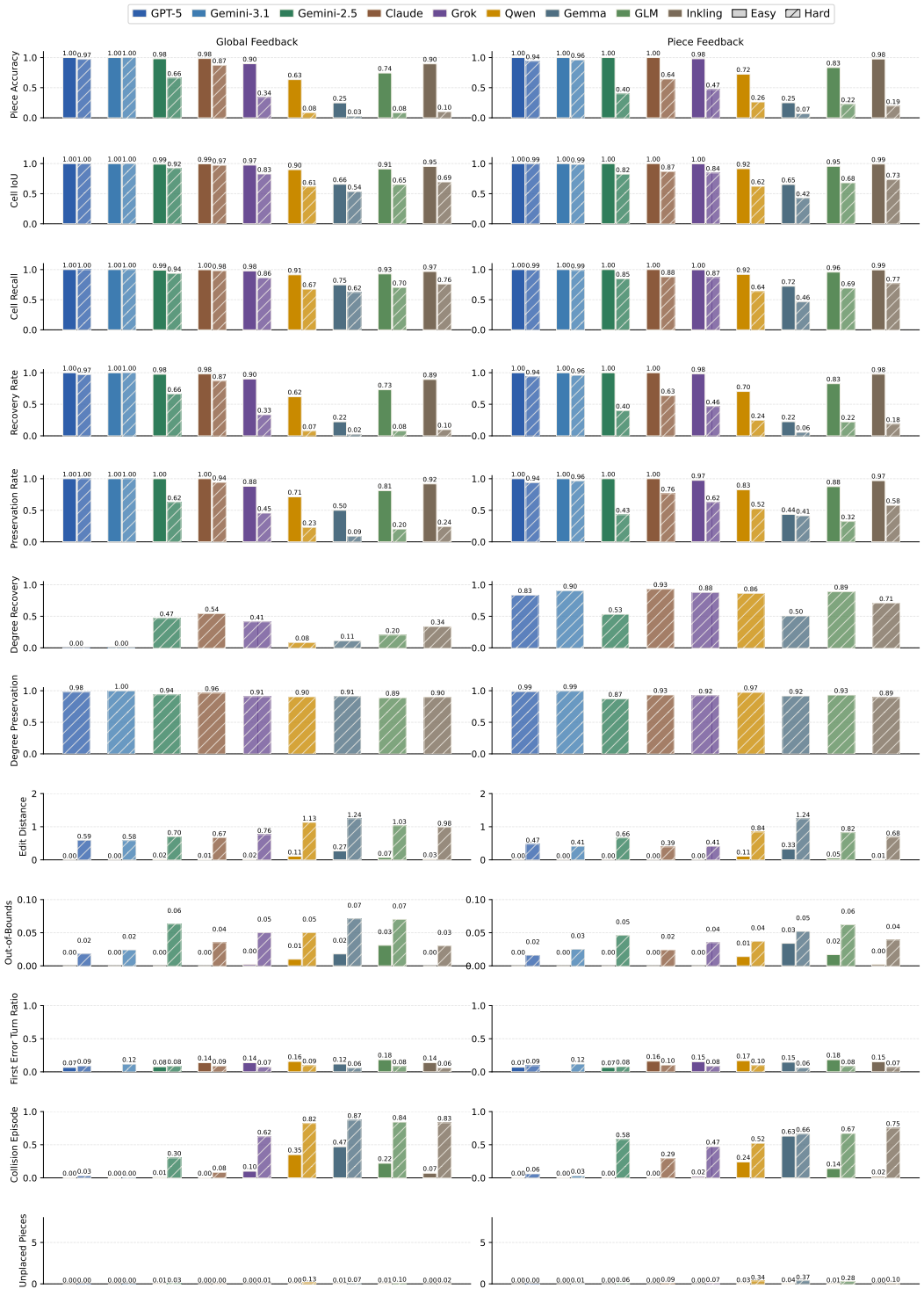


Figure 15: Fine-grained results on **Symbolic Shape Assembly** across difficulty settings and feedback types. Results complement the success rates by characterizing partial reconstruction quality and interaction dynamics.

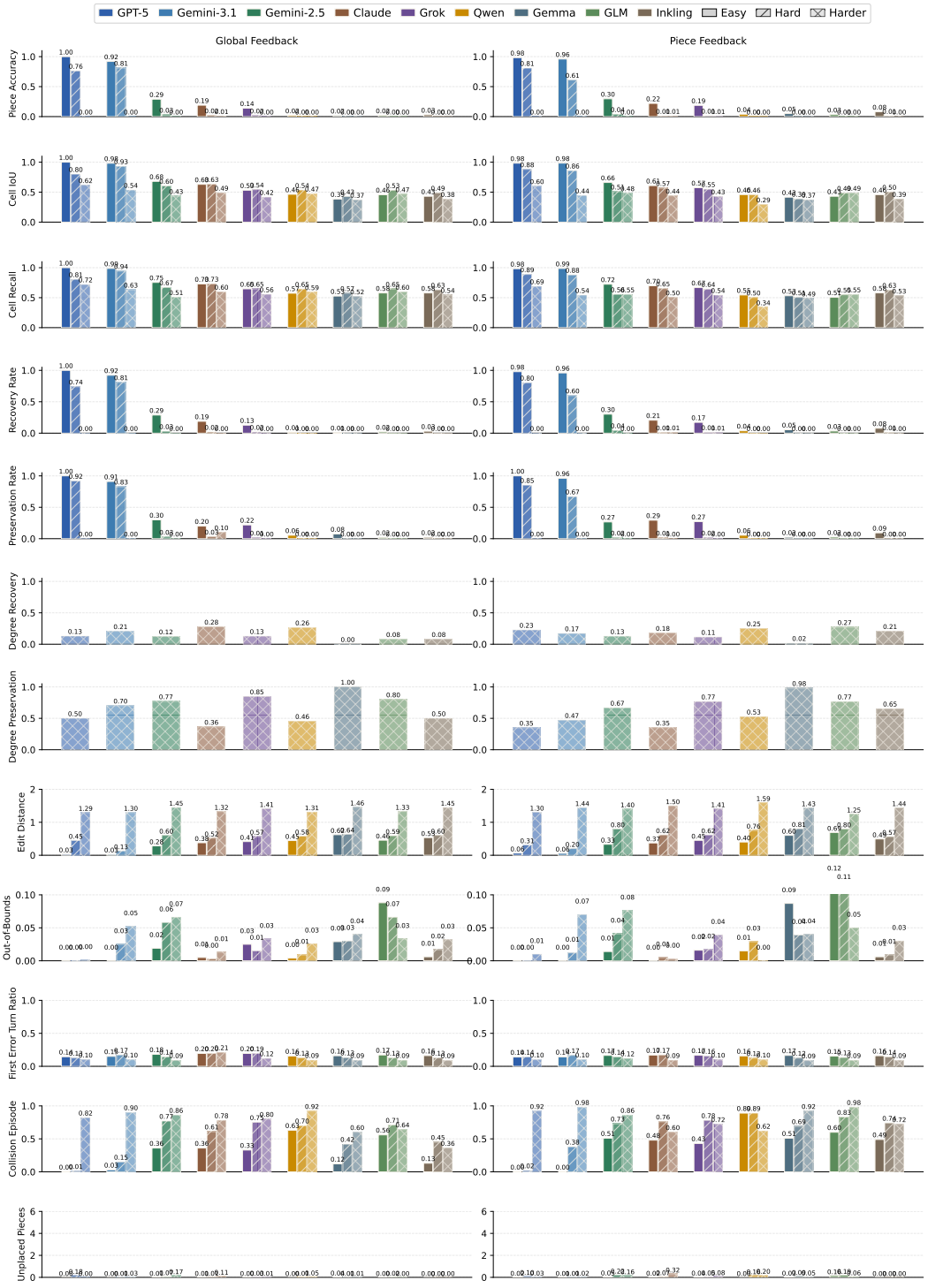


Figure 16: Fine-grained results on **Irregular Shape Assembly** across difficulty settings and feedback types. Results complement the success rates by characterizing partial reconstruction quality and interaction dynamics.

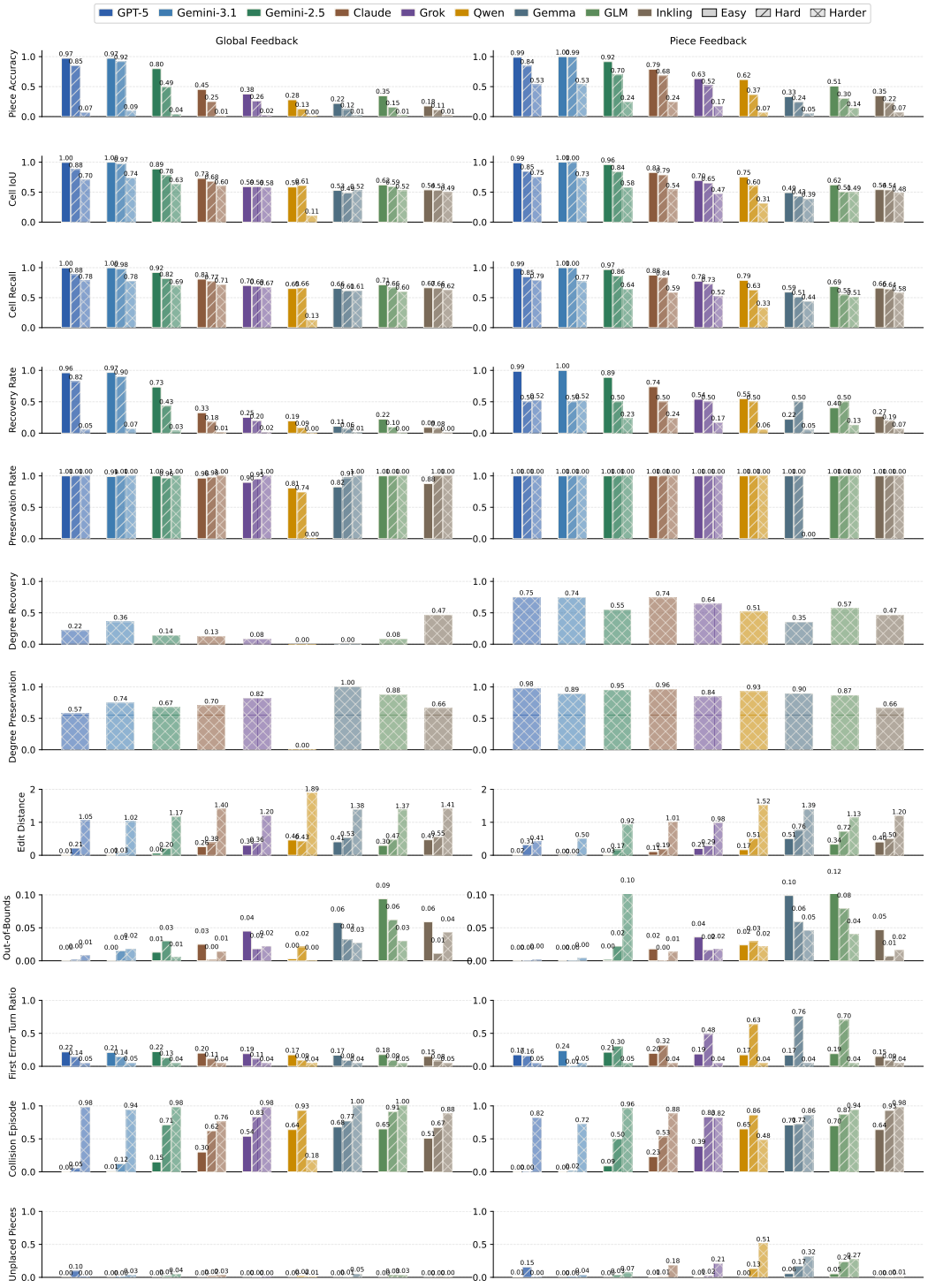


Figure 17: Fine-grained results on **Visual Shape Assembly** across difficulty settings and feedback types. Results complement the success rates by characterizing partial reconstruction quality and interaction dynamics.

E TASK INSTRUCTIONS

Main Table: Symbolic Task System Prompt

You are a Puzzle-solving planning agent.

Task: reconstruct the target color pattern using the available solid-color block pieces. Every inventory piece must be used exactly once to reconstruct the target. The target defines both shape and color. The final placement must exactly match the target pattern, with no missing cells, wrong colors, extra occupied cells, or overlaps. Output ONLY one JSON object. No markdown. No extra text.

Target region:
- The target is a {target_grid_h} × {target_grid_w} grid.
- The target grid corresponds to the blue boxed region in the upper-left of the playground.
- The final placement must exactly match the target inside this region.

Coordinate system:
- The playground origin is the upper-left cell: row 0, col 0. Rows and columns are zero-indexed.
- Row indices increase downward; column indices increase to the right.
- Use row/col grid coordinates, not pixel coordinates.
- piece_id is the integer label printed on an inventory or playground piece.
- Each piece is a solid rectangular block.
- The piece anchor is its top-left cell.

Rotation:
- rotate rotates an already placed piece clockwise by 90-degree steps.
- The anchor cell (row, col) does not change during rotation.
- After rotation, that same (row, col) becomes the top-left cell of the rotated piece.
Example:
- A 2×4 piece anchored at (1,2) becomes a 4×2 piece after a 90° rotation.
- The rotated piece is still anchored at (1,2).

Available actions:
Note: move and rotate require the piece to already be on the playground; use place first. 1) place(piece_id, row, col]): place or reposition a piece so that its top-left anchor is at (row, col).
2) move(piece_id, up_steps, down_steps, left_steps, right_steps): move an already placed piece by integer cell steps.
3) rotate(piece_id, clockwise_steps): rotate an already placed piece. clockwise_steps specifies the number of 90° clockwise rotations.
4) stop(): call stop only when you believe the puzzle is solved and no further actions are needed.

Examples:
- To place a piece: {"action":"place","piece_id":0,"row":2,"col":3}
- To move a piece:
{"action":"move","piece_id":0,"up_steps":0,"down_steps":1,"left_steps":0,"right_steps":0}
- To rotate a piece: {"action":"rotate","piece_id":0,"clockwise_steps":1}
- To stop: {"action":"stop"}

Main Table: Irregular Task System Prompt

You are a Puzzle-solving planning agent.

Task: fill exactly the target shape using the available pieces in inventory. Every inventory piece must be used exactly once to fill the target shape. The target defines shape only. The final placement must exactly match the target shape, with no missing cells, extra occupied cells, or overlaps. Output ONLY one JSON object. No markdown. No extra text.

Target region:
- The target is a {target_grid_h} × {target_grid_w} grid.
- The target grid corresponds to the blue boxed region in the upper-left of the playground. - The final placement must exactly match the target inside this region.

Coordinate system:
- The playground origin is the upper-left cell: row 0, col 0. Rows and columns are zero-indexed.
- Row indices increase downward; column indices increase to the right.
- piece_id is the integer label printed on an inventory or playground piece.

Piece coordinates:
- Each piece is represented by its minimal bounding box.
- The piece anchor is the top-left cell of this bounding box.
- Empty cells inside the bounding box are preserved and still count for coordinates.
- The anchor is defined by the bounding box, not by the first occupied cell.

Example piece mask:
011
110
The piece occupies a 2×3 bounding box. Its anchor is the upper-left box cell (the first 0), not the first occupied cell.

Available actions:
1) place(piece_id, row, col): place or reposition a piece so that its bounding-box anchor is at (row, col).
2) move(piece_id, up_steps, down_steps, left_steps, right_steps): move an already placed piece by integer cell steps.
3) stop(): call stop only when you believe the puzzle is solved and no further actions are needed.

Examples:
- To place a piece:{"action":"place","piece_id":0,"row":2,"col":3}
- To move a piece:
{"action":"move","piece_id":0,"up_steps":0,"down_steps":1,"left_steps":0,"right_steps":0}
- To stop: {"action":"stop"}

Main Table: Visual Task System Prompt

You are a Puzzle-solving planning agent.

Task: reconstruct the target image using the available image fragments. Every inventory piece must be used exactly once to reconstruct the target image. The target defines both shape and image content. The final placement must exactly match the target image, with no missing cells, mismatched image content, extra occupied cells, or overlaps. Output ONLY one JSON object. No markdown. No extra text.

Target region:
- The target is a {target_grid_h}×{target_grid_w} grid.
- The target grid corresponds to the blue boxed region in the upper-left of the playground.
- The final placement must exactly match the target inside this region.

Coordinate system:
- The playground origin is the upper-left cell: row 0, col 0. Rows and columns are zero-indexed.
- Row indices increase downward; column indices increase to the right.
- piece_id is the integer label printed on an inventory or playground piece.
- Use row/col grid coordinates, not pixel coordinates.

Piece coordinates:
- Each piece is represented by its minimal bounding box.
- The piece anchor is the top-left cell of this bounding box.
- Empty cells inside the bounding box are preserved and still count for coordinates.
- The anchor is defined by the bounding box, not by the first occupied cell.

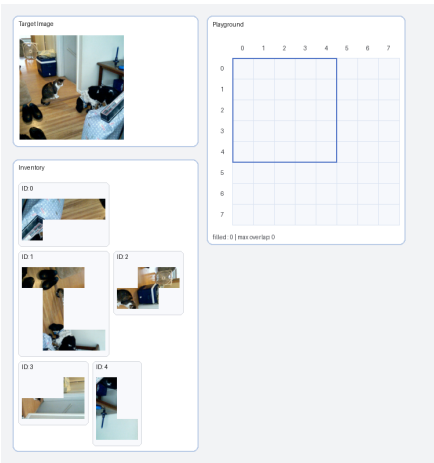
Example piece mask:
011
110
The piece occupies a 2×3 bounding box. Its anchor is the upper-left box cell (the first 0), not the first occupied cell.

Available actions:
1) place(piece_id, row, col): place or reposition a piece so that its bounding-box anchor is at (row, col).
2) move(piece_id, up_steps, down_steps, left_steps, right_steps): move an already placed piece by integer cell steps.
3) stop(): call stop only when you believe the puzzle is solved and no further actions are needed.

Examples:
- To place a piece: {"action":"place","piece_id":0,"row":2,"col":3}
- To move a piece:
{"action":"move","piece_id":0,"up_steps":0,"down_steps":1,"left_steps":0,"right_steps":0}
- To stop: {"action":"stop"}

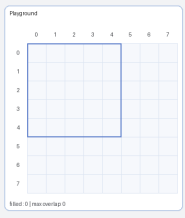
F SAMPLE TRAJECTORY FOR VISUAL TASK (GEMINI-3.1)

Initial Observation



You are a puzzle-solving planning agent.
Task: Reconstruct the target image using the available image fragments.
Every inventory piece must be used exactly once. The completed placement must exactly match the target image, with no missing cells, mismatched image content, extra occupied cells, or overlaps. Output only one JSON object. Do not use Markdown or include extra text.
Target region:
- The target is a 5 × 5 grid.
- The target grid corresponds to the blue boxed region in the upper-left of the playground.
- The final placement must exactly match the target inside this region.
Coordinate system:
- The playground origin is the upper-left cell at row 0, col 0.
- Rows and columns are zero-indexed.
- Row indices increase downward.
- Column indices increase to the right.
- piece_id is the integer label printed on an inventory or playground piece.
- Use row and column grid coordinates, not pixel coordinates.
Piece coordinates:
- Each piece is represented by its minimal bounding box.
- The piece anchor is the top-left cell of this bounding box.
- Empty cells inside the bounding box are preserved and still count when determining coordinates.
- The anchor is defined by the bounding box, not by the first occupied cell.
Example piece mask:
011
110
This piece occupies a 2 × 3 bounding box. Its anchor is the upper-left cell of the box, which is the first 0, not the first occupied cell.
Rotation:
- rotate rotates an already placed piece clockwise in 90-degree steps.
- The anchor, defined as the top-left of the bounding box, does not change during rotation.
- After rotation, the same (row, col) remains the top-left of the new bounding box.
Example:
- A piece with a 2 × 3 bounding box anchored at (1, 2) becomes a 3 × 2 bounding box after a 90-degree rotation.
- The rotated piece remains anchored at (1, 2).
Available actions:
Note: move and rotate require the piece to already be on the playground. Use place first.
1. place(piece_id, row, col): Place or reposition a piece so that its bounding-box anchor is at (row, col).
2. move(piece_id, up_steps, down_steps, left_steps, right_steps): Move an already placed piece by an integer number of cells.
3. rotate(piece_id, clockwise_steps): Rotate an already placed piece. clockwise_steps specifies the number of 90-degree clockwise rotations.
4. stop(): Call stop only when you believe the puzzle is solved and no further actions are needed.
Examples:
{"action":"place","piece_id":0,"row":2,"col":3}
{"action":"move","piece_id":0,"up_steps":0,"down_steps":1,"left_steps":0,"right_steps":0}
{"action":"rotate","piece_id":0,"clockwise_steps":1}
{"action":"stop"}

Playground State s_0



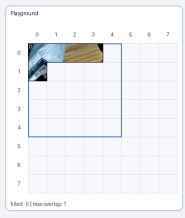
Feedback f_0

Piece feedback:
- piece 0: unplaced
- piece 1: unplaced
- piece 2: unplaced
- piece 3: unplaced
- piece 4: unplaced
This is step 1. You are allowed to take 24 more steps.
Return one next action JSON.

Action a_1

place(piece_id=0, row=0, col=0)

Playground State s_1



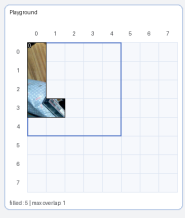
Feedback f_1

Last action result: valid
Piece feedback:
- piece 0: wrong position and wrong rotation
- piece 1: unplaced
- piece 2: unplaced
- piece 3: unplaced
- piece 4: unplaced
This is step 2. You are allowed to take 23 more steps.
Return one next action JSON.

Action a_2

rotate(piece_id=0, clockwise_steps=1)

Playground State s_2



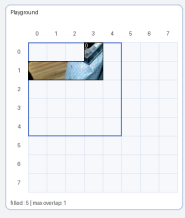
Feedback f_2

Last action result: valid
Piece feedback:
- piece 0: wrong position and wrong rotation
- piece 1: unplaced
- piece 2: unplaced
- piece 3: unplaced
- piece 4: unplaced
This is step 3. You are allowed to take 22 more steps.
Return one next action JSON.

Action a_3

rotate(piece_id=0, clockwise_steps=1)

Playground State s_3



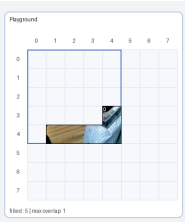
Feedback f_3

Last action result: valid
Piece feedback:
- piece 0: wrong position
- piece 1: unplaced
- piece 2: unplaced
- piece 3: unplaced
- piece 4: unplaced
This is step 4. You are allowed to take 21 more steps.
Return one next action JSON.

Action a_4

move(piece_id=0, up_steps=0, down_steps=3, left_steps=0, right_steps=1)

Playground State s_4



Feedback f_4

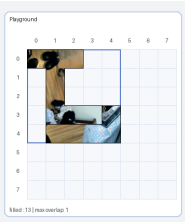
Last action result: valid
Piece feedback:
- piece 0: correct
- piece 1: unplaced
- piece 2: unplaced
- piece 3: unplaced
- piece 4: unplaced

This is step 5. You are allowed to take 20 more steps.
Return one next action JSON.

Action a_5

place(piece_id=1, row=0, col=0)

Playground State s_5



Feedback f_5

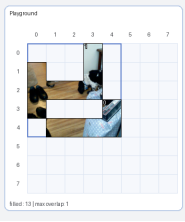
Last action result: valid
Piece feedback:
- piece 0: correct
- piece 1: wrong position and wrong rotation
- piece 2: unplaced
- piece 3: unplaced
- piece 4: unplaced

This is step 6. You are allowed to take 19 more steps.
Return one next action JSON.

Action a_6

rotate(piece_id=0, clockwise_steps=1)

Playground State s_6



Feedback f_6

Last action result: valid
Piece feedback:
- piece 0: correct
- piece 1: wrong position
- piece 2: unplaced
- piece 3: unplaced
- piece 4: unplaced

This is step 7. You are allowed to take 18 more steps.
Return one next action JSON.

Action a_7

move(piece_id=1, up_steps=0, down_steps=1, left_steps=0, right_steps=0)

Playground State s_7



Feedback f_7

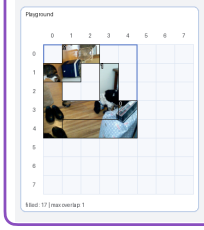
Last action result: valid
Piece feedback:
- piece 0: correct
- piece 1: correct
- piece 2: unplaced
- piece 3: unplaced
- piece 4: unplaced

This is step 8. You are allowed to take 17 more steps.
Return one next action JSON.

Action a_8

place(piece_id=2, row=0, col=0)

Playground State s_8



Feedback f_8

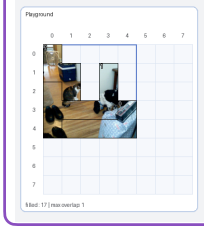
Last action result: valid
Piece feedback:
- piece 0: correct
- piece 1: correct
- piece 2: wrong rotation
- piece 3: unplaced
- piece 4: unplaced

This is step 9. You are allowed to take 16 more steps.
Return one next action JSON.

Action a_9

rotate(piece_id=2, clockwise_steps=1)

Playground State s_9



Feedback f_9

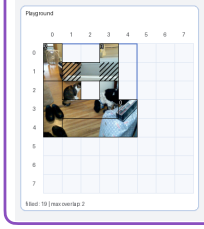
Last action result: valid
Piece feedback:
- piece 0: correct
- piece 1: correct
- piece 2: correct
- piece 3: unplaced
- piece 4: unplaced

This is step 10. You are allowed to take 15 more steps.
Return one next action JSON.

Action a_{10}

place(piece_id=3, row=0, col=1)

Playground State s_{10}



Feedback f_{10}

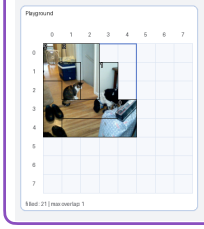
Last action result: valid
Overlap: 2 cells total — piece 1 has 1 overlap cells, piece 2 has 1 overlap cells, piece 3 has 2 overlap cells. Overlapped cells are shown with black diagonal hatching on the playground.
Piece feedback:
- piece 0: correct
- piece 1: correct
- piece 2: correct
- piece 3: wrong rotation
- piece 4: unplaced

This is step 11. You are allowed to take 14 more steps.
Return one next action JSON.

Action a_{11}

rotate(piece_id=3, clockwise_steps=1)

Playground State s_{11}



Feedback f_{11}

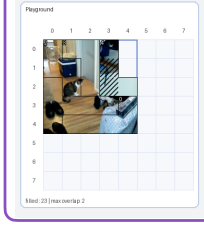
Last action result: valid
Piece feedback:
- piece 0: correct
- piece 1: correct
- piece 2: correct
- piece 3: correct
- piece 4: unplaced

This is step 12. You are allowed to take 13 more steps.
Return one next action JSON.

Action a_{12}

place(piece_id=4, row=0, col=3)

Playground State s_{12}



Feedback f_{12}

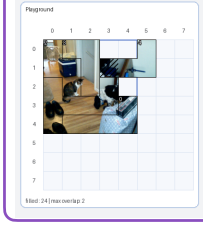
Last action result: valid
Overlap: 2 cells total — piece 1 has 2 overlap cells, piece 4 has 2 overlap cells. Overlapped cells are shown with black diagonal hatching on the playground.
Piece feedback:
- piece 0: correct
- piece 1: correct
- piece 2: correct
- piece 3: correct
- piece 4: wrong rotation

This is step 13. You are allowed to take 12 more steps.
Return one next action JSON.

Action a_{13}

rotate(piece_id=4, clockwise_steps=1)

Playground State s_{13}



Feedback f_{13}

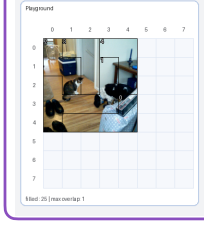
Last action result: valid
Overlap: 1 cells total — piece 1 has 1 overlap cells, piece 4 has 1 overlap cells. Overlapped cells are shown with black diagonal hatching on the playground.
Piece feedback:
- piece 0: correct
- piece 1: correct
- piece 2: correct
- piece 3: correct
- piece 4: wrong rotation

This is step 14. You are allowed to take 11 more steps.
Return one next action JSON.

Action a_{14}

rotate(piece_id=4, clockwise_steps=1)

Playground State s_{14}



Feedback f_{14}

Last action result: valid
Piece feedback:
- piece 0: correct
- piece 1: correct
- piece 2: correct
- piece 3: correct
- piece 4: correct

This is step 15. You are allowed to take 10 more steps.
Return one next action JSON.

Success!