MindTopo: Can Foundation Models Reason in Topological Space?

Yunfei Ge*[1], Anbang Liu*[1], Qineng Wang*[1†], Johnalbert Garnica*[1], Jianwen Lyu[1], Zihan Wang[1], Reuben Tan[2], Jianfeng Gao[2], Ruohan Zhang[1,3], Yining Hong[3], Jiajun Wu[3], Manling Li[1]

[1]**Northwestern University** [2]**Microsoft Research** [3]**Stanford University**

** Equal contribution; † Project lead.*

Website Code Dataset

Spatial reasoning depends not only on metric properties such as distance, angle, and shape, but also on topological relations that remain invariant under continuous deformation. Cognitive science identifies these relations as foundational to spatial understanding, yet foundation-model evaluations largely focus on metric or viewpoint-dependent relations. We introduce **MindTopo**, a benchmark of **topological intuition** across five properties grounded in cognitive science and formal topology, namely continuity, separation, order, enclosure, and knots. MINDTOPO evaluates each property at two cognitive levels. **Reasoning** asks a model to identify topological relations or infer how they change. **Planning** instantiates a foundation model as a closed-loop agent whose policy selects environment actions. MINDTOPO provides a Core Set of 3,900 instances across 13 procedurally generated task types, split into an Easy and a Hard level, and an exact-input-disjoint Extended Set of 8,780 instances. On the Core Set, we benchmark 13 multimodal large language models (MLLMs) and study agent configurations augmented with image and video generation, covering seven video generative models in planning settings. Every MLLM performs better on reasoning than on planning. On the Easy level, where human scores are available, the strongest model trails humans in both reasoning and planning. On Qwen3-VL-2B-Instruct, supervised fine-tuning and reinforcement learning improve reasoning more than planning. Generated observations retain local cues and reach solved endpoints, yet none of the 1,890 evaluated video rollouts passes all process-validity checks.



Figure 1 | **Overview of MindTopo.** Five topological properties (**continuity**, **separation**, **order**, **enclosure**, **knots**) probed through *reasoning* and *planning*. The radar shows selected models on the full Core Set and includes Human as an Easy-only reference. Table 25 gives the matched Easy-level comparison.

1. Introduction

"Representational space has to reconstruct, on its own plane and in the same order of succession, the elementary spatial relationships — first topological, then euclidean and projective."

Jean Piaget & Bärbel Inhelder, *The Child's Conception of Space* (1956)

Spatial reasoning in the physical world involves more than Euclidean or metric properties, such as distance, angle, and shape. Consider bending, stretching, or coiling a closed rope loop without cutting it or passing one segment through another. Its distances, angles, and shape can change greatly while its knot type remains the same. Topology studies spatial relations of this kind, which remain invariant under continuous deformation [1, 2]. Recognizing the invariant is only the beginning. When a foundation model must act, it must predict when an intervention will alter the relation and select actions that produce a desired one. We use **topological intuition** for this ability to perceive, infer, and operate on topological relations.

Cognitive science places these relations at the foundation of spatial understanding. Piaget argued that topological relations precede Euclidean and projective relations in spatial development [3]. Studies in adult vision similarly found that the visual system extracts topological invariants before Euclidean features [4, 5]. Together, these findings motivate the evaluation of topology as a distinct layer of spatial reasoning. Yet spatial benchmarks for foundation models largely test metric or viewpoint-dependent relations such as distance, direction, size, shape, and three-dimensional location [6, 7, 8]. Recent work isolates path connectivity or knot reasoning [9, 10], but does not show whether models can reason across a broader set of topological relations and act on them. We therefore ask *to what extent current foundation models can reason about topological structure and plan actions that transform it*.

To answer this question, we introduce MINDTOPO (Figure 1), a benchmark organized around five properties informed by Piaget's classification and later cognitive work [3, 11, 12]. **Continuity** and **separation** describe how a scene divides into connected parts. **Order** records the sequence of marked elements along a path or boundary. **Enclosure** describes the division between inside and outside, including holes. **Knots** capture entanglement that persists under deformation. Formal topology supplies the invariants and related structural targets used to instantiate these properties [1, 2, 13]. The cognitive taxonomy determines what MINDTOPO tests, while the formal targets determine how its tasks are generated and evaluated.

The same property can pose different cognitive demands depending on what a model must do with it. MINDTOPO therefore evaluates each property at two cognitive levels. **Reasoning** tasks present one or more rendered scenes and ask the model to identify a topological relation or infer how a specified change affects it. **Planning** tasks instantiate a foundation model as a closed-loop agent in an interactive environment. At each step, the agent receives a rendered observation. Its **policy** selects an action that builds, preserves, or alters the corresponding structure. A foundation model may recognize that a loop is knotted yet fail to plan how to untangle it. The two levels thus test whether reasoning about a topological relation transfers to action. Each task type comes from a parametric generator with controllable difficulty and ground truth computed from the scene state. The Core Set contains 3,900 instances across 13 task types, comprising 2,400 reasoning questions and 1,500 planning episodes, split evenly between an Easy and a Hard level. An exact-input-disjoint Extended Set contains 8,780 additional instances from the original generators.

This two-level design lets us compare reasoning with action. On the Core Set, we benchmark 13 multimodal large language models (MLLMs) across the full suite. Every MLLM scores higher on reasoning than on planning. On the Easy level, where human scores are available, the strongest model trails humans in both reasoning and planning. We also examine whether supervised fine-tuning and reinforcement learning can reduce these deficits on Qwen3-VL-2B-Instruct. Supervised fine-tuning followed by reinforcement learning gives the strongest average performance, but gains are much larger for reasoning than for planning. Reinforcement learning across reasoning tasks produces uneven gains on held-out tasks. The remaining gap between reasoning and planning raises a further question, namely whether visual prediction can help a foundation-model agent preserve topology while acting. We study agent configurations

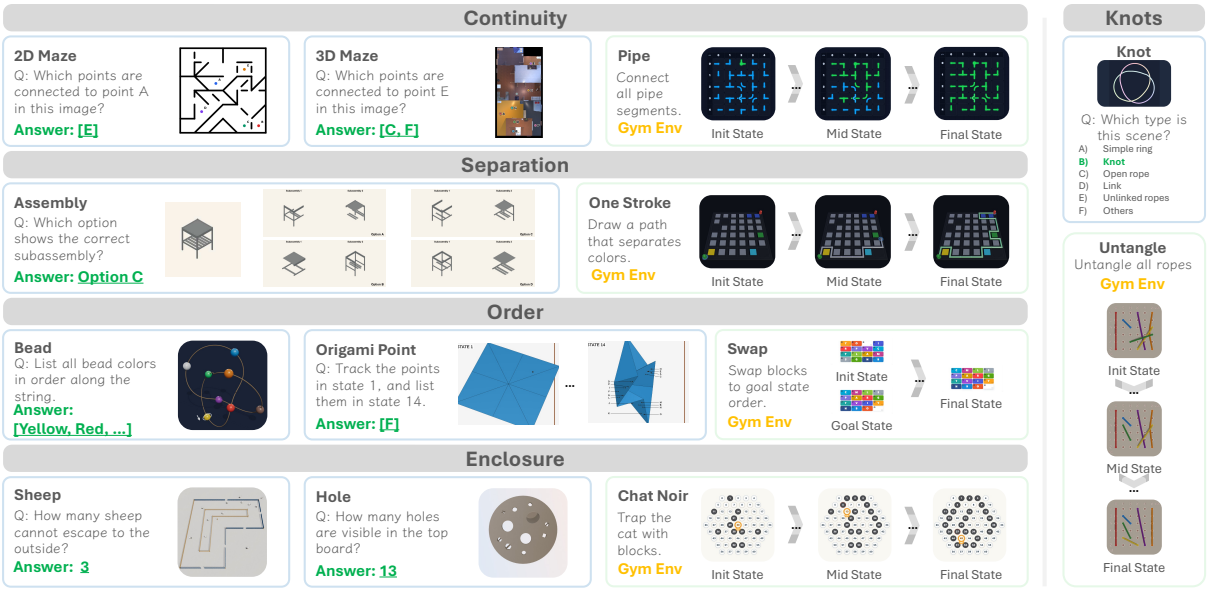


Figure 2 | **MindTopo task overview.** The 13 tasks by topological property and cognitive level. **Gym Env** marks the interactive planning environments, and the remaining tasks pair a rendered scene with a visual question.

augmented with image and video generation, covering seven video generative models in planning environments. Generated observations retain local cues and reach solved endpoints, yet none of the 1,890 evaluated video rollouts passes all process-validity checks. These results expose a gap between reasoning about a topological relation and using it to guide valid actions.

Overall, our contributions are fourfold. First, we formulate topological intuition as an evaluation target for spatial reasoning, grounding five properties in cognitive science and formal topology. Second, we introduce a scalable benchmark with 13 procedurally generated task types that probe these properties through reasoning and planning under controllable difficulty. Third, on the Core Set, we benchmark 13 MLLMs and study agent configurations involving seven video generative models in planning environments, revealing a consistent gap between reasoning about topological structure and acting on it. Fourth, we examine supervised fine-tuning and reinforcement learning on Qwen3-VL-2B-Instruct, showing that training improves reasoning more than planning and produces uneven gains on held-out tasks.

2. MindTopo Benchmark

2.1. Problem Formulation

Following Piaget's account of topological space [3], we study topological intuition as one part of spatial reasoning. It is the ability to recognize and use spatial relations that remain unchanged when a scene is continuously deformed. MINDTOPO evaluates this ability at two cognitive levels [3, 12], *reasoning* and *planning*.

Scenes and properties. Each instance begins with a parametric scene state s. The state determines the spatial structure and the marked elements relevant to the task. A renderer produces one or more visual observations $o = \text{render}(s)$. We write $X(s) \in \mathcal{X}$ for the resulting embedded structure, including its task-relevant marks, and $\mathcal{V}$ for the task-dependent value space of a property $\Phi\colon \mathcal{X} \to \mathcal{V}$.

A **topological property** Φ assigns a value to an embedded structure and remains unchanged under ambient isotopy. In other words,

$$X \simeq X' \quad \Longrightarrow \quad \Phi(X) = \Phi(X'). \tag{1}$$

Here $X \simeq X'$ means that one structure can be continuously deformed into the other without cutting, joining, or passing one part through another [2, Ch. 1]. Some tasks evaluate a closely related target rather than the invariant itself. Appendix A.2 gives the exact definitions and identifies these cases.

Reasoning. A reasoning instance is a triple (o, q, a^*). The question q asks a foundation model to identify a topological relation in the observation or predict how it changes after a stated edit. The model returns

$$\hat{a} = M(o, q). \tag{2}$$

and the prediction is correct when $\hat{a} = a^*$. The reference answer a^* is computed from the scene state. Scalar and categorical answers are scored by exact match, while unordered answers are compared as sets.

Planning. A planning episode starts from a scene state s_0, a task instruction u, and a horizon H. At step t, the environment renders $o_t = \text{render}(s_t)$. Let h_t denote the interaction history made available to the evaluated configuration through step t, including the current observation and any retained previous observations and actions. An evaluated configuration C built around a foundation model induces the policy

$$a_t \sim \pi_C(\cdot \mid u, h_t). \tag{3}$$

The simulator then applies the selected action and updates the scene state according to

$$s_{t+1} \sim P(\cdot \mid s_t, a_t). \tag{4}$$

Together, the configuration and the environment interface form the agent evaluated by MINDTOPO, which acts in a closed loop. The episode succeeds if its goal predicate g is satisfied in some state s_t with $t \leq H$. The policy belongs to the complete configuration, whether it uses an MLLM alone or also uses image or video generation.

Reasoning tests whether a foundation model can identify or predict a topological relation. Planning tests whether that relation can guide a valid sequence of actions.

2.2. Topological Properties

Figure 2 shows representative scenes, questions, and gym environments for every task in the suite. Each property below describes its cognitive grounding, our reasoning task, and the matched planning task. Appendix A.1 expands the cognitive-science grounding of the five properties, and Appendix A gives the full specification of every task, including scene generation, difficulty parameterization, and metrics.

Continuity. Continuity captures whether a path or surface forms an unbroken whole, and is among the earliest spatial concepts a child acquires [3]. It underlies many practical reasoning tasks, such as navigating a maze, deciding whether a wire is severed, or threading a cable through an opening. Our reasoning task asks the model which of several marked points in a rendered maze are reachable from a designated target point, so the answer hinges on topological cuts rather than on metric layout. We additionally include *what-if* sub-questions that probe how reachability changes if a wall is added or removed. The matched planning task places the agent in a grid of rotatable pipe segments. Through 90° rotations, it must connect every pipe segment back to the source, probing whether it can *construct* continuity, not only recognize it.

Separation. Separation is the complement of proximity [12] and undergirds object individuation [3]. Distinguishing adjacent units as distinct is the prerequisite for any reasoning beyond an undifferentiated whole. Our reasoning task, *Assembly*, is a furniture subassembly judgment. Given a fully assembled object and several candidate sub-assemblies, the model selects the

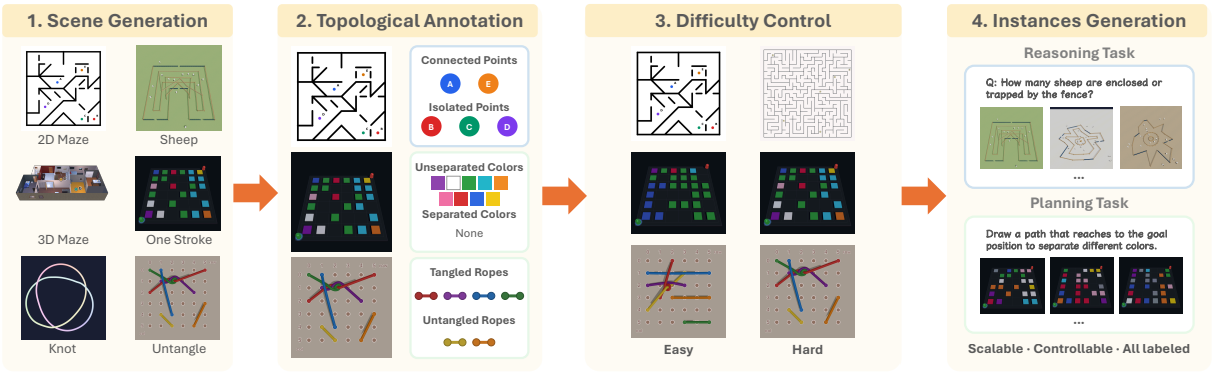


Figure 3 | **Fully automated data collection pipeline for MindTopo.** Acceptance checks specific to each task filter invalid or ambiguous instances before export.

subset that forms a topologically separable component. We add *what-if* sub-questions asking how the partition changes if a connector is removed. The planning counterpart is the One Stroke partitioning environment, in which the agent draws a single corner-to-corner path through a colored grid that keeps same-colored cells in a shared region and separates different colors, operationalizing the act of *creating* separation between previously contiguous regions.

Order. Order captures the sequential arrangement of elements along a path or boundary [3]. It is essential whenever a model must track *which comes before which* under a transformation of the scene. Our reasoning task adapts Piaget's bead-replication paradigm. Given a curved or twisted string of colored beads, the model enumerates the sequence from a designated starting bead, or decides whether two strings preserve the same cyclic order. We further include *what-if* sub-questions that ask how the order changes under a folding or rotation of the string. The accompanying planning environment, Swap, is a swap puzzle in which the agent reaches a target color permutation by repeatedly moving a chosen block into the single empty slot, isolating ordering reasoning from geometric and color confounds.

Enclosure. Enclosure is the inside/outside relation induced by a closed boundary, and is identified by Piaget as the topological origin of three-dimensional "insideness" [3]. We instantiate two reasoning tasks. *Sheep* asks the model which animals lie strictly inside a top-down enclosure boundary, and *Hole* asks it to count the through-holes in a solid object. Both include *what-if* sub-questions, e.g., how the count changes if a piece of material is added or removed. The matched planning task is *Chat Noir*. On a hexagonal grid, the agent places one block per turn so as to encircle a moving cat before it escapes to the boundary, requiring forward reasoning about whether a partial boundary can still be closed.

Knots. Piaget originally subsumed knots under enclosure, but subsequent cognitive evidence supports treating them as a distinct ability. Strohecker characterizes knots as the "mother structure" that coordinates all other topological relations [11]. Croom and Firestone show that humans reason about knots far worse than they perceive them, dissociating knot understanding from domain-general physical reasoning [14]. Our reasoning task presents one or more rendered ropes and loops and asks a battery of topology questions. They ask whether a single loop is truly knotted or just visually tangled, whether two loops are linked, and, in *what-if* variants, which rings become free after a specified ring is cut and removed. The matched planning environment is *Untangle*. Given a board of plug positions joined by ropes, the agent moves plugs across a discrete grid until no two ropes cross.

2.3. Data Collection and Statistics

Figure 3 shows the four-stage pipeline that builds every task in MINDTOPO. All four stages are automated and use fixed seeds. *Scene Generation* samples scene states and renders them using

simulators or rendering scripts specific to each task. *Topological Annotation* refers to programmatic labeling rather than human annotation. Code specific to each task derives the reference answer or success condition directly from the generated state and metadata. Depending on the task, this computation uses graph search, geometric membership tests, known construction metadata, or simulator predicates. *Difficulty Control* varies parameters that change topological complexity, including wall count, grid size, and crossing count, to produce instances at two levels, Easy and Hard, with 150 instances per task each. *Instance Generation* converts each accepted state into either a templated reasoning question or a planning episode with an initial state, goal, and success condition. Reasoning and planning tasks are matched at the topological property level, but their scenes are produced separately by their respective task generators.

Quality control. Before export, each generator applies construction and acceptance criteria specific to its task. Depending on the task, these criteria enforce valid scene structure, reject ambiguous or unsolvable configurations, check visibility when the task depends on visible marks or components, and confirm that the generated answer or success condition agrees with the accepted state. Each exported record retains its generation seed and configuration metadata. Planning records also retain the initial state and action budget. Appendix A describes the scene generation, difficulty settings, output formats, scoring rules, and applicable acceptance criteria for every task.

Potential generation bias. Procedural rendering and templated questions can introduce visual, language, or answer-prior shortcuts. We evaluate matched text-only, answer-prior, appearance-only, and symbolic-input controls in Appendix B.8. Section 5 separately discusses the lack of visual variation from real-world scenes.

Figure 4 summarizes the Core Set. The **Core Set** contains 3,900 instances across the 13 task types and the five topological properties. Each of the eight reasoning tasks (2D Maze, 3D Maze, Assembly, Bead, Origami Point, Sheep, Hole, and Knots) and the five planning tasks (Pipe, One Stroke, Swap, Chat Noir, and Untangle) contributes 300 instances, 150 at each level. The **Extended Set** contains 8,780 additional instances from the same generators. It is exact-input-disjoint from Core after removing 2,250 Core overlaps and five repeated Swap inputs from the 11,035-instance original pool. Table 2 reports its per-task composition and evaluation results; Appendix G defines the deduplication rule and records release traceability. Per-task Core statistics and the evaluation design appear in Appendix A.10 and Appendix A.11.

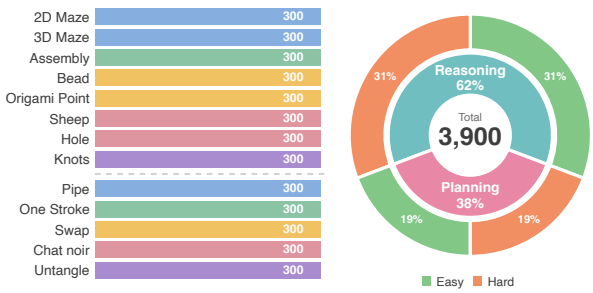


Figure 4 | Core Set statistics for MINDTOPO.

3. Experiments

3.1. Experimental Setups

Models. On the Core Set, we benchmark 13 hosted MLLMs, seven proprietary and six open-weight. The proprietary tier includes GPT-6 Astra [15], GPT-5.6-Sol [16], GPT-5.5 [17], GPT-5.4-mini [18], Gemini-3.1-Flash-Lite [19], Gemini-3.1-Pro [20], and Claude-Haiku-5.5 [21]. The open-weight tier includes Nemotron-Nano-12B-VL-v2 [22], Qwen3.5-397B-A17B [23], Llama-4-Maverick-17B-128E [24], Ministral-3-14B-Instruct-2512 [25], InternVL3.5-241B [26], and Gemma-4-31B-IT [27]. We additionally report a random-chance baseline, the expected accuracy of a uniformly random legal answer for reasoning tasks and the success rate of uniformly random legal actions under the same step budget for planning tasks. Evaluated snapshots, decoding configuration, and prompt templates appear in Appendix B.1.

Model	Continuity			Separation		Order			Enclosure			Knots		Avg.	
	2D Maze	3D Maze	Pipe	Assembly	One Stroke	Bead	Origami Point	Swap	Sheep	Hole	Chat Noir	Knots	Untangle	Reason	Plan
Proprietary Models															
GPT-6 Astra	70.67	81.33	54.67	71.00	70.67	53.33	69.33	51.33	35.33	33.33	51.67	62.00	20.67	59.54	49.80
GPT-5.6-Sol	33.00	14.00	11.00	36.00	9.33	18.67	9.67	40.33	31.33	26.33	35.33	25.33	3.00	24.29	19.80
GPT-5.5	8.67	7.67	0.00	33.00	0.00	7.33	10.33	19.67	23.67	27.33	12.67	30.67	7.67	18.58	8.00
GPT-5.4-mini	7.67	2.67	0.00	22.67	0.00	5.33	2.33	0.00	5.00	0.33	0.00	12.33	3.33	7.29	0.67
Gemini-3.1-Flash-Lite	6.67	1.33	0.00	24.33	0.00	6.33	2.67	0.00	14.33	1.67	0.33	18.00	3.00	9.42	0.67
Gemini-3.1-Pro	10.67	14.00	0.00	27.00	0.00	6.00	3.00	1.00	30.33	20.67	13.67	41.67	6.33	19.17	4.20
Claude-Haiku-5.5	1.00	13.33	0.00	30.00	0.00	11.00	4.33	0.00	42.33	25.33	22.00	31.67	5.00	19.87	5.40
Hosted Open-Weight Models															
Nemotron-Nano-12B-VL-v2	4.00	1.33	0.00	21.67	0.00	2.67	0.33	0.00	3.33	0.33	0.00	11.00	0.67	5.58	0.13
Qwen3.5-397B-A17B	6.67	4.67	0.00	22.33	0.00	4.67	1.33	0.00	0.33	1.67	0.67	11.00	0.33	6.58	0.20
Llama-4-Maverick-17B-128E	8.00	3.67	0.00	24.00	0.00	4.67	3.00	0.00	2.00	0.00	0.00	10.00	2.67	6.92	0.53
Ministral-3-14B-Instruct-2512	5.00	2.00	0.00	24.00	0.00	7.00	1.67	0.00	1.33	0.00	0.00	8.67	3.00	6.21	0.60
InternVL3.5-241B	2.00	3.33	0.00	20.33	0.00	3.00	3.67	0.00	4.33	0.33	0.00	14.67	0.00	6.46	0.00
Gemma-4-31B-IT	2.33	3.33	0.00	13.00	0.00	0.00	0.00	0.00	7.00	0.00	1.33	8.67	0.67	4.29	0.40
Random Chance	4.30	3.70	0.00	20.00	0.00	7.08	2.18	0.00	5.52	6.67	0.00	13.97	2.67	7.93	0.53

Table 1 | **Performance on the MindTopo Core Set (%).** Each task has 300 instances, 150 per level. Reason and Plan average the reasoning and planning tasks, and shading marks the top three models per column. Human results on the Easy level appear in Figure 1 and Table 25. Reasoning task and Planning task denote task type.

Human evaluation. Five professional annotators evaluated the previous release of the benchmark, whose instances form the Easy level, under the same instructions as the models, with one completed annotation per example. Figure 1 includes Human as an Easy-only reference against full-Core model curves, while Table 25 provides the matched Easy-level comparison. Three annotators also independently evaluated a shared 200-example subset, yielding an inter-annotator agreement score of 0.89. Full details appear in Appendix B.4.1 and Appendix B.4.2.

Protocol. All models are evaluated with deterministic decoding (temperature 0). Reasoning tasks are scored by exact match for scalar and multiple-choice answers and by set equality for unordered list answers. Planning tasks are scored by executing the predicted actions in the corresponding environment and counting an episode as successful only when it reaches the task-defined terminal condition. Primary metrics are task accuracy for reasoning and episode success rate for planning. The primary benchmark evaluation uses the 3,900 Core instances. Table 2 separately reports the exact-input-disjoint Extended Set. Training and generative probing use the separately specified evaluation protocols below.

3.2. Benchmark Results

Planning performance lags behind reasoning. Table 1 reports accuracy or episode success rate on the 300 instances of each of the 13 task types. GPT-6 Astra achieves the highest averages on both levels, 59.54% on reasoning and 49.80% on planning. On the Easy level, where human performance is available, GPT-6 Astra reaches 76.33% on reasoning and 81.87% on planning, against 94.23% and 99.87% for humans. It matches humans on 2D Maze, Pipe, and Swap but stays far behind on Bead, Sheep, Hole, and Untangle, and every other model trails humans on every task. Every model scores lower on planning than on reasoning. GPT-5.6-Sol drops from 24.29% to 19.80%, Claude-Haiku-5.5 from 19.87% to 5.40%, Gemini-3.1-Pro from 19.17% to 4.20%, and Qwen3.5-397B-A17B from 6.58% to 0.20%. Across the evaluated models, identifying topological relations is more successful than acting on them.

The disjoint Extended Set shows the same reasoning–planning gap. Table 2 places the Extended Set composition and results in the main paper. Its 8,780 inputs remain after exact-input deduplication against Core and within the original pool. Among the 11 models with auditable per-instance or overlap-subset results, GPT-5.6-Sol has the highest reasoning and

Task	Core Set	Extended Set
2D Maze	300	850
3D Maze	300	846
Pipe	300	450
Assembly	300	890
One Stroke	300	450
Bead	300	850
Origami Point	300	1,000
Swap	300	445
Sheep	300	700
Hole	300	699
Chat Noir	300	450
Knots	300	850
Untangle	300	300
Total	3,900	8,780

(a) Per-task sizes of the Core and exact-input-disjoint Extended Sets.

Model	Continuity			Separation		Order		Enclosure				Knots		Avg.	
	2D Maze	3D Maze	Pipe	Assembly	One Stroke	Bead	Origami Point	Swap	Sheep	Hole	Chat Noir	Knots	Untangle	Reason	Plan
Proprietary Models															
GPT-5.6-Sol	88.71	70.92	60.00	60.67	41.11	64.71	77.00	86.74	78.29	80.11	70.67	66.71	40.33	73.39	59.77
GPT-5.5	67.29	51.18	2.89	57.42	13.11	39.88	60.00	50.11	67.29	79.54	39.56	64.82	44.33	60.93	30.00
GPT-5.4-mini	17.53	22.34	0.22	36.18	0.00	32.47	27.20	13.71	25.71	23.75	8.44	28.59	44.33	26.72	13.34
Gemini-3.1-Flash-Lite	33.76	36.05	0.00	45.73	0.00	39.76	42.20	11.69	63.14	31.62	8.44	71.53	36.67	45.48	11.36
Gemini-3.1-Pro	36.12	58.27	7.33	53.82	4.44	51.88	41.10	22.47	78.43	76.11	45.11	75.29	52.00	58.88	26.27
Open-Weight Models															
Nemotron-Nano-12B-VL-v2	14.12	11.82	0.00	28.88	0.00	20.47	26.70	1.57	20.43	9.87	0.22	28.35	6.00	20.08	1.56
Qwen3.5-397B-A17B	14.00	38.65	0.00	44.83	0.00	21.88	42.60	12.36	9.43	34.91	11.78	36.59	15.00	30.36	7.83
Llama-4-Maverick-17B-128E	25.65	17.97	0.00	35.73	0.00	30.94	17.10	7.42	15.57	3.72	2.44	35.88	12.67	22.82	4.51
Ministral-3-14B-Instruct-2512	20.35	12.41	0.00	28.76	0.00	16.47	18.40	5.17	16.29	7.30	5.33	47.88	20.67	20.98	6.23
InternVL3.5-241B	17.18	11.82	0.00	12.70	0.00	27.29	19.50	5.62	22.86	22.03	1.33	49.88	7.33	22.91	2.86
Gemma-4-31B-IT	30.35	18.68	0.00	19.10	0.00	9.41	1.60	1.35	9.29	0.00	17.33	6.24	19.67	11.83	7.67

(b) Performance on the MINDTOPO Extended Set (%). Reason and Plan average the reasoning and planning tasks. Shading marks the four highest distinct positive scores per task among the 11 models with auditable deduplicated results.

Table 2 | **Composition and performance of the MindTopo Extended Set.** The Extended Set is exact-input-disjoint from the balanced Core Set in Table 1. Its scores are recomputed after removing Core overlaps and within-set duplicates. Scores from the two sets are reported separately and are not pooled. Reasoning task and Planning task denote task type.

planning averages, at 73.39% and 59.77%. Every reported model again scores lower on planning than on reasoning. Core and Extended scores remain separate because the sets use different difficulty distributions and model lineups.

The action gap widens below the frontier. GPT-6 Astra and GPT-5.6-Sol retain more than 80% of their reasoning average when they plan. GPT-5.5 retains 43%, Claude-Haiku-5.5 27%, and Gemini-3.1-Pro 22%. Claude-Haiku-5.5 outperforms GPT-5.5 on several reasoning tasks, including Sheep (42.33% versus 23.67%), and on the reasoning average (19.87% versus 18.58%), yet its planning average is lower (5.40% versus 8.00%). No hosted open-weight model exceeds 1% average planning success, close to the 0.53% of a random legal policy. Their reasoning averages, between 4.29% and 6.92%, fall below the random-chance average of 7.93%. The gap between reasoning and planning is therefore not a uniform property of the benchmark, and current open-weight models do not close the planning bottleneck.

The runner-up varies across topological properties. GPT-6 Astra leads every property average, with 68.89% on continuity, 70.84% on separation, 58.00% on order, 40.11% on enclosure, and 41.34% on knots. The second place shifts between GPT-5.6-Sol, which follows on continuity, separation, order, and enclosure, and Gemini-3.1-Pro, which follows on knots with 24.00%. Enclosure shows the smallest margin among the top three models (40.11%, 31.00%, and 29.89% for GPT-6 Astra, GPT-5.6-Sol, and Claude-Haiku-5.5), and Claude-Haiku-5.5 achieves the best Sheep score at 42.33%.

Untangle is the hardest planning environment. The best Untangle score is 20.67% by GPT-6 Astra, and no other model exceeds 7.67%. Pipe and One Stroke separate GPT-6 Astra (54.67% and 70.67%) from every other model, which reaches at most 11.00% and 9.33%. Each environment

Training	2D Maze	Knots	Assembly	Bead	Sheep	One Stroke	Untangle	Pipe	Swap
Base	9.20	11.20	32.08	9.31	9.40	0.00	0.80	**0.00**	0.00
SFT	**14.10**	75.50	**57.35**	41.74	51.00	**0.80**	11.81	**0.00**	7.21
RL	10.10	42.70	38.52	34.73	25.30	0.00	15.42	**0.00**	1.40
Leave-one-task-out RL	8.20	15.50	31.32	26.53	8.90	–	–	–	–
SFT + RL	13.50	**79.60**	53.41	**52.95**	**58.20**	0.50	**17.42**	**0.00**	**7.41**

Table 3 | Qwen3-VL-2B-Instruct performance (%) across selected reasoning and planning tasks under the Base, supervised fine-tuning (SFT), reinforcement learning (RL), leave-one-task-out RL, and SFT + RL settings. For leave-one-task-out RL, each reasoning-task result comes from a separate policy trained on the other four reasoning tasks and evaluated on the excluded task. All policies are trained only on the Easy level. Reasoning task and Planning task denote task type.

requires a sequence of legal actions toward a structural goal, namely a connected pipe network, color-consistent regions, or zero projected rope crossings. In particular, Untangle evaluates a viewpoint-dependent crossing criterion, not preservation of a three-dimensional knot class. The low success rates show that these tasks remain challenging under the evaluated protocol.

Swap is the most accessible planning task below the frontier. GPT-6 Astra reaches 51.33% on Swap, GPT-5.6-Sol 40.33%, and GPT-5.5 19.67%, while every other model scores at most 1.00%. Swap targets a discrete permutation, but Pipe, One Stroke, and Untangle also admit discrete task-state descriptions, and the present results do not separate the roles of state representation, task difficulty, and action horizon. Results for the Easy and Hard levels appear in Appendix B.5.

Shortcut controls separate scene evidence from prompt and label priors. We compare the full input with matched text-only, answer-prior, appearance-only, and symbolic-input controls. This analysis also measures dependence on rendering cues and explicit state descriptions. Full results appear in Appendix B.8. Appendix B.7 additionally describes the controlled camera and viewpoint factors, and Appendix B.9 analyzes recurring topological biases behind these scores.

Key Takeaways: Planning Lags Behind Topology Recognition

- Planning exposes failures that reasoning scores do not reveal.
- Planning success concentrates in a few frontier models, and open-weight models stay near random.
- Aggregate scores hide uneven performance across topological properties.

3.3. Training

Methodology. We test whether the deficits exposed by MINDTOPO can be improved through task-specific training. Starting from Qwen3-VL-2B-Instruct [28], we compare the frozen base policy with answer-only supervised fine-tuning (SFT), Group Relative Policy Optimization (GRPO), and SFT followed by GRPO. The comparison covers five reasoning tasks and four planning tasks. For planning, the model predicts a complete action sequence from the initial observation. The sequence is executed in the corresponding environment and receives credit only if it reaches the task-defined success state. We additionally train five leave-one-task-out RL policies, each trained on four reasoning tasks and evaluated on the fifth. All policies are trained only on the Easy level, so the trained policies are not directly comparable with the models in Table 1. Full data, optimization, and evaluation details appear in Appendix B.2.

Training teaches recurring visual patterns. The base policy averages 8.00% across the nine tasks in Table 3. SFT raises this average to 28.83%, GRPO alone to 18.69%, and SFT followed by GRPO to 31.44%. Most of the gain comes from four reasoning tasks. Under SFT followed by GRPO, Knots rises from 11.20% to 79.60%, Sheep from 9.40% to 58.20%, Bead from 9.31% to 52.95%, and Assembly from 32.08% to 53.41%, with 57.35% under SFT alone. In each of these tasks the answer rests on a visual pattern that recurs across instances, such as the crossings of a closed curve, the fence layer around a sheep, or the color order along a single string. Answer

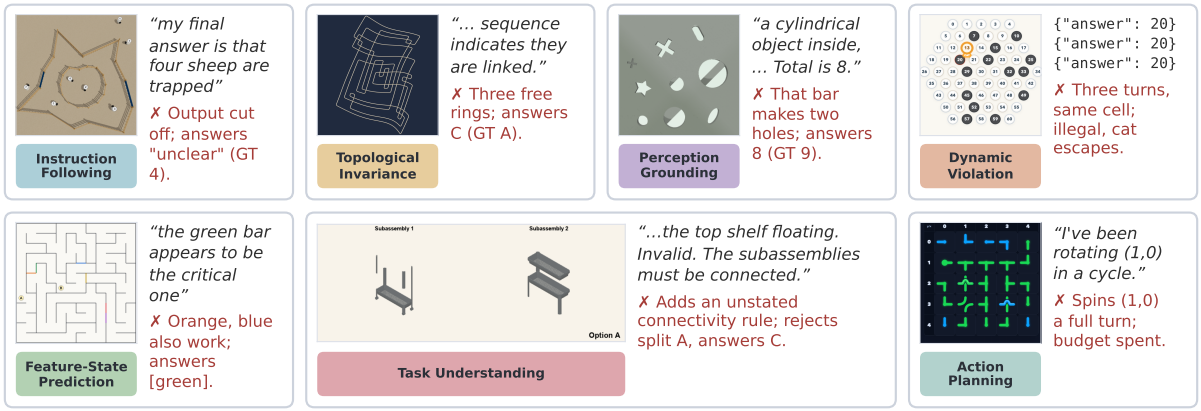


Figure 5 | One labeled failure per error category. Each panel pairs the frame the model saw with a verbatim line from its response.

supervision is enough to teach such patterns, and GRPO after SFT adds further gains on Knots, Bead, Sheep, and Untangle.

Global connectivity and long action sequences remain hard. Three difficulties persist under every training condition. First, tracing connectivity across a whole scene barely improves. 2D Maze reaches at most 14.10% from a base of 9.20%, although it is a reasoning task like the four above. Second, planning stays far behind reasoning. SFT followed by GRPO lifts average reasoning accuracy from 14.24% to 51.53% but average planning success only from 0.20% to 6.33%. The planning gains come from Untangle (17.42%) and Swap (7.41%), while Pipe stays at 0% and One Stroke never exceeds 0.80%. Both of these tasks require every action in a long sequence to respect a constraint over the whole board, which means connecting all pipes in Pipe and separating every color region with one stroke in One Stroke. Third, what a policy learns stays largely within the trained task. Leave-one-task-out policies improve held-out Bead from 9.31% to 26.53% and held-out Knots from 11.20% to 15.50%, but leave 2D Maze, Assembly, and Sheep at or below their base scores, and Appendix B.2 shows the same uneven pattern on three further tasks. These results come from a 2B model trained with answer supervision on one level per task, and its planning policies predict a full action sequence without environment feedback. Whether larger models or closed-loop training would move this boundary remains open.

Key Takeaways: Training Teaches Patterns More Readily Than Global Structure

- SFT and GRPO sharply improve reasoning tasks whose answers rest on recurring visual patterns, namely Knots, Sheep, Bead, and Assembly.
- Connectivity across the whole scene in 2D Maze and constraint-preserving action sequences in Pipe and One Stroke barely improve.
- Gains transfer to a held-out task only in selected cases.

3.4. Error Analysis

Methodology. We analyze Gemini-3.1-Pro and InternVL3.5-241B as representatives of the proprietary and open-weight tiers. For each of the 26 pairs of model and task, we sample 35 failures uniformly from the pair's incorrect predictions on the 300 benchmark instances, yielding 910 labeled failures (560 reasoning and 350 planning, 415 Easy and 495 Hard). Each failure receives one primary category from a seven-class taxonomy that follows the processing pipeline from instruction compliance to action planning. Categories are assigned in fixed causal priority order so an early failure is never credited to a later stage. All percentages are pooled shares of the labeled failures. Figure 5 shows one representative example per category. Full definitions, sampling details, and per-model distributions appear in Appendix C.1.

Planning failures sit further downstream than reasoning failures. Figure 6 shows that reasoning failures concentrate in perception grounding (47.5%) and instruction following (33.8%). In planning, feature-state prediction, dynamic violations, and action planning together account for 50.9% of failures. In reasoning, these three categories account for only 11.4% of failures, all of them feature-state prediction. Perception grounding still accounts for 22.0% of planning failures, but 63 of these 77 come from Pipe. Reasoning is thus gated by reading the scene and returning a usable answer, whereas planning adds failures in choosing and executing valid transitions.

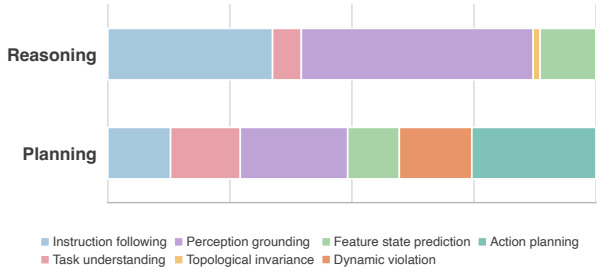


Figure 6 | Category distribution of the labeled failures on reasoning and planning tasks, pooled over Gemini-3.1-Pro and InternVL3.5-241B.

Reasoning failures combine visual grounding and answer delivery. Perception grounding accounts for 58.9% of Gemini-3.1-Pro's reasoning failures, followed by instruction following (21.1%) and feature-state prediction (10.4%). InternVL3.5-241B fails most often on instruction following (46.4%), mostly responses truncated before a committed answer, and then on perception grounding (36.1%). Origami Point stands apart, with 78.6% of its failures assigned to feature-state prediction. Explicit topological-invariance errors are rare in this single-primary-label analysis (1.8% and 1.1%), because earlier perception or task failures take precedence when they already explain the wrong answer.

Planning failures depend on the environment. Gemini-3.1-Pro's largest planning category is action planning (41.1%), followed by perception grounding (24.0%) and feature-state prediction (21.1%), while dynamic violations are almost absent (0.6%). InternVL3.5-241B is led by dynamic violations (29.1%), with instruction following, task understanding, and perception grounding each near 20%. Pipe failures are 90.0% perception grounding, as models misread pipe ends and the source direction. Swap failures are 67.1% task understanding, since on Easy boards both models play an adjacent-only sliding puzzle although any tile may swap with the blank. Chat Noir and One Stroke split by model. Gemini-3.1-Pro fails mainly through action planning on Chat Noir (97.1%) and feature-state prediction on One Stroke (62.9%). InternVL3.5-241B fails mainly through dynamic violations on both (68.6% and 77.1%), and every one of its failures on Untangle is a truncated response. Full distributions appear in Appendix C.3.

Key Takeaways: Reasoning and Planning Fail at Different Stages

- Reasoning failures come mainly from visual grounding and unusable answers.
- About half of planning failures lie in state prediction, dynamics, and action choice.
- The dominant planning failure varies strongly with the environment.

3.5. Probing Experiment

Methodology. Section 3.4 shows that about half of the planning failures on MINDTOPO arise in action choice, dynamics, and state prediction, and that outside Pipe few come from reading the scene. We probe whether explicit visual prediction can supply the topological state that models lose across actions. The success-rate configurations use GPT-5.6-Luna [16] as the planner on six tasks (Knots, Sheep, 2D Maze, Untangle, One Stroke, and Pipe). The **baseline** acts directly from the rendered observation. The **interleaved** variant calls GPT-Image-2 [29] at every step to render the predicted next state. The **video** variants use Wan2.2-I2V-A14B [30], Seedance-2.0-Mini [31], and Veo-3.1-Lite [32] to roll out candidate plans. Figure 7 separately audits GPT-5.4-mini outputs whose exports do not record the exact model snapshot. Each probing condition uses 105 episodes per task, 35 from each of three generator difficulty settings, so the episodes extend beyond the

Planner	Knots	Sheep	2D Maze	Untangle	One Stroke	Pipe
Planner only						
GPT-5.6-Luna	26.00	30.33	22.17	19.05	2.86	29.52
GPT-5.4-mini	25.71	19.05	15.24	23.81	0.00	0.00
InternVL3.5-241B	42.86	17.14	15.24	3.81	0.00	0.00
+ GPT-Image-2 (image)						
GPT-5.6-Luna	57.14	68.57	61.90	24.76	0.00	20.00
GPT-5.4-mini	60.00	57.14	12.38	6.67	0.00	0.00
InternVL3.5-241B	56.19	30.47	13.33	0.00	0.00	0.00

Planner	Untangle	One Stroke	Pipe
+ Wan2.2-I2V-A14B (video)			
GPT-5.6-Luna	24.76	0.00	9.52
InternVL3.5-241B	4.76	0.00	0.00
+ Seedance-2.0-Mini (video)			
GPT-5.6-Luna	25.71	0.00	6.67
InternVL3.5-241B	27.62	0.00	0.00
+ Veo-3.1-Lite (video)			
GPT-5.6-Luna	27.62	0.00	5.71
InternVL3.5-241B	5.71	0.00	0.00

Table 4 | Probing success rate (%). Each gray band names what the planner is paired with, and each row is a planner. The left panel reports planner-only and image-interleaved runs on all six tasks, and the right panel reports video-interleaved runs, which cover the planning tasks only. Lavender marks the best and second-best result per task across both panels. Reasoning task and Planning task denote task type.

	Static checks						Dynamic checks			Outcome	
Video generator	Color sep.	Direction valid.	Path cont.	Path valid.	Start/end valid.	**Static (all)**	Grid-cell temp.	Path temp.	**Dynamic (all)**	Final-state success	**Process-valid**
One Stroke											
LTX-2.3	1.0	7.6	2.9	7.6	18.1	0.0	1.0	0.0	0.0	0.0	0.0
LTX-2.5	0.0	0.0	0.0	0.0	0.0	0.0	1.9	0.0	0.0	0.0	0.0
Veo-3.1-Lite	0.0	0.0	0.0	0.0	0.0	0.0	3.8	1.0	0.0	0.0	0.0
Wan2.2-I2V-A14B	0.0	52.4	27.6	52.4	56.2	0.0	18.1	16.2	1.9	0.0	0.0
MiniMax-H3	0.0	0.0	0.0	0.0	0.0	0.0	1.9	0.0	0.0	0.0	0.0
Seedance-2.5	0.0	0.0	0.0	1.0	1.9	0.0	2.9	0.0	0.0	0.0	0.0

	Static checks		Dynamic checks				Outcome	
Video generator	Connectivity–color match	**Static (all)**	Cell occupancy	Pipe-type consist.	Rotation valid.	**Dynamic (all)**	Final-state success	**Process-valid**
Pipe								
LTX-2.3	0.0	0.0	0.0	0.0	0.0	0.0	3.8	0.0
LTX-2.5	0.0	0.0	1.9	1.0	0.0	0.0	35.2	0.0
Veo-3.1-Lite	1.0	1.0	2.9	0.0	0.0	0.0	5.7	0.0
Wan2.2-I2V-A14B	0.0	0.0	5.7	0.0	0.0	0.0	13.3	0.0
MiniMax-H3	0.0	0.0	2.9	0.0	0.0	0.0	19.0	0.0
Seedance-2.5	0.0	0.0	33.3	0.0	0.0	0.0	16.2	0.0

	Static checks			Dynamic checks			Outcome	
Video generator	Endpoint config.	Rope integrity	**Static (all)**	Endpoint tracking	Topology flicker	**Dynamic (all)**	Final-state success	**Process-valid**
Untangle								
LTX-2.3	0.0	1.0	0.0	0.0	66.7	0.0	32.4	0.0
LTX-2.5	1.0	7.6	1.0	1.0	73.3	1.0	6.7	0.0
Veo-3.1-Lite	0.0	5.7	0.0	0.0	32.4	0.0	1.0	0.0
Wan2.2-I2V-A14B	0.0	21.0	0.0	0.0	46.7	0.0	2.9	0.0
MiniMax-H3	4.8	42.9	3.8	2.9	57.1	1.0	5.7	0.0
Seedance-2.5	20.0	48.6	9.5	0.0	70.5	0.0	3.8	0.0

Table 5 | **Strict video-level computer-vision (CV) pass rates (%).** The generators are LTX-2.3 [33], LTX-2.5 [34], Veo-3.1-Lite [32], Wan2.2-I2V-A14B [30], MiniMax-H3 [35], and Seedance-2.5 [36]. Appendix B.6.1 defines every check.

two benchmark levels. The full imagined-rollout protocol and failure-mode analysis appear in Appendix B.6.

Interleaved image prediction remains weakest on trajectory-wide invariants. With GPT-5.6-Luna, the GPT-Image-2 condition reaches 68.57% on Sheep and 61.90% on 2D Maze (Table 4) but 0.00% on One Stroke, because generated frames retain local cues but not invariants that depend on a complete action trajectory.

Endpoint success and process validity are different criteria. A rollout achieves **endpoint success** when its final frame shows a solved task state. It is **process valid** only if, in addition, every sampled frame shows a legal task state and every pair of adjacent frames forms a legal transition. Table 4 reports endpoint success only.

No evaluated video rollout passes all process-validity checks. A manual audit of 119 Wan2.2-I2V-A14B rollouts finds dynamics violations in 116, topological-invariance errors in 93, and consistency errors in 90 (Figure 7). An automatic verifier confirms this pattern at scale on 1,890

rollouts, 105 for each of six video generators on each of One Stroke, Pipe, and Untangle (Table 5). The terminal oracle accepts up to 35.2% of rollouts in a task as solved, yet the process-valid rate is 0.0% for every generator on every task. Single checks often pass, as the Untangle topology readout stays free of flicker in 32.4% to 73.3% of videos, yet no generator passes all static checks in more than 9.5% of the videos of any task.

Verifier validation. The verifier parses 41 evenly spaced frames of each video into task states. It applies static checks to four sampled frames and dynamic checks to every adjacent pair. A video passes only when its final state is readable and every check succeeds. A human audit of 180 LTX-2.3 and Wan2.2-I2V-A14B videos agrees with every verifier decision. Almost all audited videos fail, so the audit validates rejection decisions but does not estimate how often valid rollouts are rejected. We use these checks to screen for invalid rollouts. Appendix B.6.1 details the audit and parser errors.

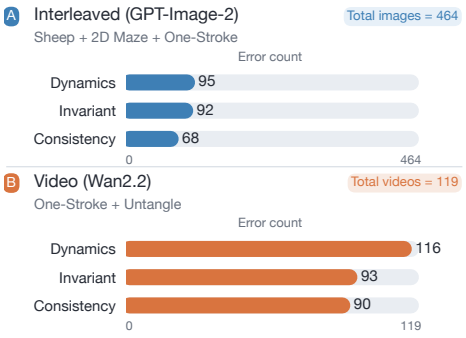


Figure 7 | Per-generation errors in 464 GPT-Image-2 images (top) and 119 Wan2.2-I2V-A14B videos (bottom). Each panel labels its evaluated tasks.

Key Takeaways: Endpoint Success Does Not Certify Topological Rollouts

- Generated images retain local cues but lose invariants that span a trajectory.
- Video generators reach solved endpoints in up to 35.2% of rollouts, yet none of the evaluated videos passes all process-validity checks.
- Rollout evaluation must verify topology at every step, not only at the endpoint.

4. Related Work

Cognitive and Topological Evaluation. Cognitive studies motivate evaluating topological relations as a foundation of spatial understanding [3, 4, 5]. BabyVision tests basic visual abilities through both language and generated visual outputs [37], while related work examines vision models' sensitivity to geometric and topological concepts [38]. Among benchmarks that target topology directly, CurveBench asks models to recover containment trees from images of nested curves [39], whereas TopoBench requires complete solutions to symbolic grid puzzles governed by global spatial constraints [40]. KnotGym brings topology into interactive evaluation through rope manipulation from visual observations [10]. MINDTOPO broadens this coverage by organizing five topological properties within one suite and evaluating each through both reasoning questions and interactive planning tasks.

Spatial Reasoning and Interaction. Spatial benchmarks evaluate relations within scenes and the integration of information across viewpoints [41, 42, 43, 7, 8]. Theory of Space extends this perspective to constructing and revising spatial beliefs through active exploration [44]. Planning with the Views examines whether models can compose individual camera movements into longer plans [45]. Interactive evaluations include sliding puzzles in iVISPAR [46] and tasks across simulation environments in SpatialWorld [47], alongside broader studies of embodied decision interfaces and visual action selection [48, 49]. ESI-Bench grounds active perception and manipulation in Spelke's core knowledge systems [50]. MINDTOPO organizes reasoning and planning around topological properties, providing the complementary coverage summarized in Table 42 of Appendix D.

World Modeling and Agent Training. Predicting the consequences of actions connects spatial evaluation with world modeling. CausalSpatial uses generated visual evidence to answer questions about specified object motions [51], while ENACT evaluates forward and inverse world

modeling through observation and action sequence reordering [52]. For agents acting across multiple turns, RAGEN [53] and RAGEN-2 [54] study training stability, while VAGEN [55] reinforces world-model reasoning. Our rollout diagnostics complement these approaches by checking whether predicted transitions preserve task topology.

5. Conclusion and Limitations

Conclusion. We presented MINDTOPO, a benchmark of topological intuition that covers five Piagetian properties (continuity, separation, order, enclosure, and knots) at two cognitive levels across 13 procedurally generated task types. Every MLLM evaluated on the Core Set scores lower on planning than on reasoning. On the Easy level, where human scores are available, the strongest model trails humans in both reasoning and planning. Training on Qwen3-VL-2B-Instruct improves task performance, yet planning success stays low and gains on held-out tasks are uneven. Generated observations reach solved endpoints, yet none of the 1,890 evaluated video rollouts passes all process-validity checks. MINDTOPO provides a controlled diagnostic for foundation models that aim to internalize the qualitative structure of the physical world.

Limitations. First, all scenes are procedurally rendered, which gives clean ground truth but lacks the visual variability of real photographs. Second, the video evaluation covers three planning environments, so broader coverage is needed to test how general the observed failures are. Third, the five properties do not exhaust topological space, and relations such as orientation or higher-genus surfaces remain for future releases. Fourth, the probing annotations of 464 generated images and 119 videos support coarse comparisons but not subtle effects.

Acknowledgments

This work used Delta at NCSA through ACCESS allocation CIS250698, supported by NSF grants #2138259, #2138286, #2138307, #2137603, and #2138296 [56], and the Quest high performance computing facility at Northwestern University.

References

[1] Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[2] Dale Rolfsen. *Knots and Links*. Publish or Perish, Berkeley, CA, 1976.

[3] Jean Piaget and Bärbel Inhelder. *Child's Conception of Space: Selected Works vol 4*. Routledge, 2013.

[4] Lin Chen. Topological structure in visual perception. *Science*, 218(4573):699–700, 1982.

[5] Lin Chen. The topological approach to perceptual organization. *Visual Cognition*, 12(4):553–637, 2005.

[6] Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14455–14465, 2024.

[7] Jihan Yang, Shusheng Yang, Anjali W Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10632–10643, 2025.

[8] Qineng Wang, Baiqiao Yin, Pingyue Zhang, Jianshu Zhang, Kangrui Wang, Zihan Wang, Jieyu Zhang, Keshigeyan Chandrasegaran, Han Liu, Ranjay Krishna, Saining Xie, Jiajun Wu, Li Fei-Fei, and Manling Li. MindCube: Spatial mental modeling from limited views. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust, editors, *International Conference on Learning Representations*, volume 2026, pages 82559–82622, 2026.

[9] Alan Dao and Dinh Bach Vu. Alphamaze: Enhancing large language models' spatial intelligence via grpo. *arXiv preprint arXiv:2502.14669*, 2025.

[10] Zizhao Chen and Yoav Artzi. Knot so simple: A minimalistic environment for spatial reasoning. *arXiv preprint arXiv:2505.18028*, 2025.

[11] Carol Strohecker. *Why knot?* PhD thesis, Massachusetts Institute of Technology, 1991.

[12] J Larry Martin. An analysis of some of piaget's topological tasks from a mathematical point of view. *Journal for research in mathematics education*, 7(1):8–24, 1976.

[13] Edward V Huntington. A set of independent postulates for cyclic order. *Proceedings of the National Academy of Sciences*, 2(11):630–631, 1916.

[14] Sholei Croom and Chaz Firestone. Tangled physics: knots strain intuitive physical reasoning. *Open Mind*, 8:1170–1190, 2024.

[15] OpenAI. GPT-6 Astra. https://openai.com/index/gpt-6-astra/, 2026.

[16] OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition. https://openai.com/index/gpt-5-6/, 2026.

[17] OpenAI. GPT-5.5 system card. https://openai.com/index/gpt-5-5-system-card/, 2026.

[18] OpenAI. Introducing GPT-5.4 mini and nano. https://openai.com/index/introducing-gpt-5-4-mini-and-nano/, 2026.

[19] The Gemini Team. Gemini 3.1 Flash-Lite: Built for intelligence at scale. https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-flash-lite, 2026.

[20] The Gemini Team. Gemini 3.1 Pro: A smarter model for your most complex tasks. https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-pro/, 2026.

[21] Anthropic. Introducing Claude Haiku 5.5. https://www.anthropic.com/news/claude-haiku-5-5, 2026.

[22] NVIDIA, Amala Sanjay Deshmukh, Kateryna Chumachenko, Tuomas Rintamaki, Matthieu Le, Tyler Poon, Danial Mohseni Taheri, Ilia Karmanov, Guilin Liu, Jarno Seppanen, Guo Chen, et al. NVIDIA Nemotron Nano V2 VL. *arXiv preprint arXiv:2511.03929*, 2025.

[23] Qwen Team. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5, February 2026.

[24] Meta AI. The llama 4 herd: The beginning of a new era of natively multimodal AI innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/, 2025.

[25] Mistral AI. Ministral 3 14b instruct 2512. Model card, https://huggingface.co/mistralai/Ministral-3-14B-Instruct-2512, 2025.

[26] Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. InternVL3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025.

[27] Google DeepMind. Gemma 4 model card. https://deepmind.google/models/gemma/gemma-4/, 2026.

[28] 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.

[29] OpenAI. Introducing ChatGPT Images 2.0. https://openai.com/index/introducing-chatgpt-images-2-0/, April 2026. Accessed: 2026-05-07.

[30] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models, 2025.

[31] ByteDance Seed. Seedance 2.0 Mini. https://seed.bytedance.com/en/seedance, 2026.

[32] Google DeepMind. Build with Veo 3.1 Lite, our most cost-effective video generation model. https://blog.google/innovation-and-ai/technology/ai/veo-3-1-lite/, March 2026. Model: `veo-3.1-lite-generate-preview`. Accessed: 2026-10-17.

[33] Lightricks. LTX-2.3: Open-source video generation model. https://huggingface.co/Lightricks/LTX-2.3, 2026.

[34] Lightricks. LTX-2.5: Open-source video generation model. https://huggingface.co/Lightricks/LTX-2.5, 2026.

[35] MiniMax. MiniMax-H3: First-last-frame-to-video generation model. https://huggingface.co/MiniMaxAI/MiniMax-H3, 2026.

[36] ByteDance Seed. Seedance 2.5. https://seed.bytedance.com/en/seedance, 2026. Model: `doubao-seedance-2-5-260628`.

[37] Liang Chen, Weichu Xie, Yiyan Liang, Hongfeng He, Hans Zhao, Zhibo Yang, Zhiqi Huang, Haoning Wu, Haoyu Lu, Yiping Bao, et al. Babyvision: Visual reasoning beyond language. *arXiv preprint arXiv:2601.06521*, 2026.

[38] Zekun Wang and Sashank Varma. Computer vision models show human-like sensitivity to geometric and topological concepts. *arXiv preprint arXiv:2505.13281*, 2025.

[39] Amirreza Mohseni, Mona Mohammadi, Morteza Saghafian, and Naser Talebizadeh Sardari. CurveBench: A benchmark for exact topological reasoning over nested jordan curves. *arXiv preprint arXiv:2605.14068*, 2026.

[40] Mayug Maniparambil, Nils Hoehing, Janak Kapuriya, Arjun Karuvally, Ellen Rushe, Anthony Ventresque, Noel O'Connor, and Fergal Reid. TopoBench: Benchmarking LLMs on hard topological reasoning. *arXiv preprint arXiv:2603.12133*, 2026. Accepted at the Workshop on Logical Reasoning of Large Language Models at ICLR 2026.

[41] Wufei Ma, Haoyu Chen, Guofeng Zhang, Yu-Cheng Chou, Jieneng Chen, Celso de Melo, and Alan Yuille. 3dsrbench: A comprehensive 3d spatial reasoning benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6924–6934, 2025.

[42] Wenqi Wang, Reuben Tan, Pengyue Zhu, Jianwei Yang, Zhengyuan Yang, Lijuan Wang, Andrey Kolobov, Jianfeng Gao, and Boqing Gong. Site: towards spatial intelligence thorough evaluation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9058–9069, 2025.

[43] Mengdi Jia, Zekun Qi, Shaochen Zhang, Wenyao Zhang, Xinqiang Yu, Jiawei He, He Wang, and Li Yi. Omnispatial: Towards comprehensive spatial reasoning benchmark for vision language models. *arXiv preprint arXiv:2506.03135*, 2025.

[44] Pingyue Zhang, Zihan Huang, Yue Wang, Jieyu Zhang, Letian Xue, Zihan Wang, Qineng Wang, Keshigeyan Chandrasegaran, Ruohan Zhang, Yejin Choi, Ranjay Krishna, Jiajun Wu, Li Fei-Fei, and Manling Li. Theory of space: Can foundation models construct spatial beliefs through active exploration? In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust, editors, *International Conference on Learning Representations*, volume 2026, pages 136983–137020, 2026.

[45] Kangrui Wang, Linjie Li, Zhengyuan Yang, Shiqi Chen, Zihan Wang, Li Fei-Fei, Jiajun Wu, Leonidas Guibas, Lijuan Wang, and Manling Li. Planning with the views. *arXiv preprint arXiv:2605.29563*, 2026.

[46] Julius Mayer, Mohamad Ballout, Serwan Jassim, Farbod Nosrat Nezami, and Elia Bruni. ivispar—an interactive visual-spatial reasoning benchmark for vlms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 26745–26769, 2025.

[47] Hongcheng Gao, Hailong Qu, Jingyi Tang, Jiahao Wang, Zihao Huang, Hengkang Qiao, Shihong Huang, Junming Yang, Yi Li, Hongyixuan Yuan, Wenjie Li, Bohan Zeng, Wenbo Li, Bo Wang, Jianhui Liu, Olive Huang, Haoyang Huang, Wentao Zhang, Guoqing Huang, Nan Duan, and Yinpeng Dong. SpatialWorld: Benchmarking interactive spatial reasoning of multimodal agents in real-world tasks. *arXiv preprint arXiv:2606.09669*, 2026.

[48] Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li Erran Li, Ruohan Zhang, Weiyu Liu, Percy Liang, Li Fei-Fei, Jiayuan Mao, and Jiajun Wu. Embodied agent interface: Benchmarking LLMs for embodied decision making. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 100428–100534. Curran Associates, Inc., 2024.

[49] Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, Heng Ji, Huan Zhang, and Tong Zhang. EmbodiedBench: Comprehensive benchmarking multi-modal large language

models for vision-driven embodied agents. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 70576–70631. PMLR, 13–19 Jul 2025.

[50] Yining Hong, Jiageng Liu, Han Yin, Manling Li, Leonidas Guibas, Li Fei-Fei, Jiajun Wu, and Yejin Choi. ESI-Bench: Towards embodied spatial intelligence that closes the perception-action loop. *arXiv preprint arXiv:2605.18746*, 2026.

[51] Wenxin Ma, Chenlong Wang, Ruisheng Yuan, Hao Chen, Nanru Dai, S. Kevin Zhou, Yijun Yang, Alan Yuille, and Jieneng Chen. CausalSpatial: A benchmark for object-centric causal spatial reasoning. *arXiv preprint arXiv:2601.13304*, 2026.

[52] Qineng Wang, Wenlong Huang, Yu Zhou, Hang Yin, Tianwei Bao, Jianwen Lyu, Weiyu Liu, Ruohan Zhang, Jiajun Wu, Li Fei-Fei, and Manling Li. ENACT: Evaluating embodied cognition with world modeling of egocentric interaction. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust, editors, *International Conference on Learning Representations*, volume 2026, pages 153022–153077, 2026.

[53] Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

[54] Zihan Wang, Chi Gui, Xing Jin, Qineng Wang, Licheng Liu, Kangrui Wang, Shiqi Chen, Linjie Li, Zhengyuan Yang, Pingyue Zhang, Yiping Lu, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN-2: Reasoning collapse in agentic RL. *arXiv preprint arXiv:2604.06268*, 2026.

[55] Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Chi Wan, Yiping Lu, Zhengyuan Yang, Lijuan Wang, Ranjay Krishna, Jiajun Wu, Li Fei-Fei, Yejin Choi, and Manling Li. VAGEN: Reinforcing world model reasoning for multi-turn VLM agents. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, pages 172871–172933. Curran Associates, Inc., 2025.

[56] Timothy J. Boerner, Stephen Deems, Thomas R. Furlani, Shelley L. Knuth, and John Towns. ACCESS: Advancing innovation: NSF's advanced cyberinfrastructure coordination ecosystem: Services & support. In *Practice and Experience in Advanced Research Computing 2023: Computing for the Common Good*, 2023.

[57] Henri Poincaré. *Science and Hypothesis*. Walter Scott Publishing, 1905. English translation of *La Science et l'Hypothèse*, 1902.

[58] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-scale embodied AI using procedural generation. In *Advances in Neural Information Processing Systems*, 2022.

[59] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An interactive 3D environment for visual AI. *arXiv preprint arXiv:1712.05474*, 2017.

[60] Amanda Ghassaei, Erik D. Demaine, and Neil Gershenfeld. Fast, interactive origami simulation using GPU computation. In *Origami^7: Proceedings of the 7th International Meeting on Origami in Science, Mathematics and Education (7OSME)*, pages 1151–1166. Tarquin, 2018.

[61] Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, Guang Shi, and Haoqi Fan. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.

[62] Jiawei Gu, Yunzhuo Hao, Huichen Will Wang, Linjie Li, Michael Qizhe Shieh, Yejin Choi, Ranjay Krishna, and Yu Cheng. ThinkMorph: Emergent properties in multimodal interleaved chain-of-thought reasoning. *arXiv preprint arXiv:2510.27492*, 2025.

[63] Pooyan Rahmanzadehgervi, Logan Bolton, Mohammad Reza Taesiri, and Anh Totti Nguyen. Vision language models are blind. In *Proceedings of the Asian Conference on Computer Vision*, pages 18–34, 2024.

[64] Jingqi Tong, Jixin Tang, Hangcheng Li, Yurong Mou, Ming Zhang, Jun Zhao, Yanbo Wen, Fan Song, Jiahao Zhan, Yuyang Lu, et al. Game-RL: Synthesizing multimodal verifiable game data to boost VLMs' general reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026.

[65] Yew Ken Chia, Vernon Toh, Deepanway Ghosal, Lidong Bing, and Soujanya Poria. PuzzleVQA: Diagnosing multimodal reasoning challenges of language models with abstract visual patterns. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 16259–16273, 2024.

[66] Ilias Stogiannidis, Steven McDonagh, and Sotirios A Tsaftaris. Mind the gap: Benchmarking spatial reasoning in vision-language models. *arXiv preprint arXiv:2503.19707*, 2025.

[67] Jingping Liu, Ziyan Liu, Zhedong Cen, Yan Zhou, Yinan Zou, Weiyan Zhang, Haiyun Jiang, and Tong Ruan. Can multimodal large language models understand spatial relations? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 620–632, 2025.

[68] Evert Willem Beth and Jean Piaget. *Mathematical epistemology and psychology*. D. Reidel, 1966. Translated by W. Mays.

[69] Seymour Papert. Mindstorms: children, computers, and powerful ideas, 1980.

[70] Roberto Casati and Achille C Varzi. *Holes and other superficialities*. The MIT Press, 1994.

[71] Rolf Nelson and Stephen E Palmer. Of holes and wholes: The perception of surrounded regions. *Perception*, 30(10):1213–1226, 2001.

[72] Marco Bertamini and Camilla J Croucher. The shape of holes. *Cognition*, 87(1):33–54, 2003.

[73] Peter Gärdenfors. *Conceptual Spaces: The Geometry of Thought*. MIT Press, Cambridge, MA, 2000.

[74] Yijiang Li, Qingying Gao, Haoran Sun, Haiyun Lyu, Dezhi Luo, and Hokin Deng. Cogdevelop2k: Reversed cognitive development in multi-modal large language models. *arXiv preprint arXiv:2410.10855*, 2024.

[75] Xinglin Wang, Peiwen Yuan, Shaoxiong Feng, Yiwei Li, Boyuan Pan, Heda Wang, Yao Hu, and Kan Li. Coglm: Tracking cognitive development of large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for*

Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 73–87, 2025.

[76] Dezhi Luo, Haiyun Lyu, Qingying Gao, Haoran Sun, Yijiang Li, and Hokin Deng. Vision language models know law of conservation without understanding more-or-less. *arXiv preprint arXiv:2410.00332*, 2024.

[77] Qingying Gao, Yijiang Li, Haiyun Lyu, Haoran Sun, Dezhi Luo, and Hokin Deng. Vision language models see what you want but not what you see. *arXiv preprint arXiv:2410.00324*, 2024.

[78] Eunice Yiu, Maan Qraitem, Anisa Noor Majhi, Charlie Wong, Yutong Bai, Shiry Ginosar, Alison Gopnik, and Kate Saenko. Kiva: Kid-inspired visual analogies for testing large multimodal models. *arXiv preprint arXiv:2407.17773*, 2024.

[79] Rui Xu, Dakuan Lu, Zicheng Zhao, Xiaoyu Tan, Xintao Wang, Siyu Yuan, Jiangjie Chen, and Yinghui Xu. Origamispace: Benchmarking multimodal llms in multi-step spatial reasoning with mathematical constraints. *arXiv preprint arXiv:2511.18450*, 2025.

[80] Ryan Spencer, Roey Yaari, Ritvik Vemavarapu, Joyce Yang, Steven Ngo, and Utkarsh Sharma. Gamibench: Evaluating spatial reasoning and 2d-to-3d planning capabilities of mllms with origami folding tasks. *arXiv preprint arXiv:2512.22207*, 2025.

[81] Christopher Driggers-Ellis, Gabriel Ayoubi, and Christan Grant. Optical: An abstract positional reasoning benchmark for vision language models. In *2025 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 1437–1441. IEEE, 2025.

[82] Chen Yang, Guanxin Lin, Youquan He, Peiyao Chen, Guanghe Liu, Yufan Mo, Zhouyuan Xu, Linhao Wang, Guohui Zhang, Zihang Zhang, et al. Thinking in structures: Evaluating spatial intelligence through reasoning on constrained manifolds. *arXiv preprint arXiv:2602.07864*, 2026.

[83] Xinmiao Huang, Qisong He, Zhenglin Huang, Boxuan Wang, Zhuoyun Li, Guangliang Cheng, Yi Dong, and Xiaowei Huang. Spatial-dise: A unified benchmark for evaluating spatial reasoning in vision-language models. *arXiv preprint arXiv:2510.13394*, 2025.

[84] 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.

[85] Chan Hee Song, Valts Blukis, Jonathan Tremblay, Stephen Tyree, Yu Su, and Stan Birchfield. Robospatial: Teaching spatial understanding to 2d and 3d vision-language models for robotics. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15768–15780, 2025.

[86] Shiqi Chen, Tongyao Zhu, Ruochen Zhou, Jinghan Zhang, Siyang Gao, Juan Carlos Niebles, Mor Geva, Junxian He, Jiajun Wu, and Manling Li. Why is spatial reasoning hard for VLMs? an attention mechanism perspective on focus areas. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 9910–9932. PMLR, 2025.

[87] Fanqing Meng, Jiaqi Liao, Xinyu Tan, Wenqi Shao, Quanfeng Lu, Kaipeng Zhang, Yu Cheng, Dianqi Li, Yu Qiao, and Ping Luo. Towards world simulator: Crafting physical commonsense-based benchmark for video generation. *arXiv preprint arXiv:2410.05363*, 2024.

[88] Qin Zhang, Peiyu Jing, Hong-Xing Yu, Fangqiang Ding, Fan Nie, Weimin Wang, Yilun Du, James Zou, Jiajun Wu, and Bing Shuai. Physion-eval: Evaluating physical realism in generated video via human reasoning. *arXiv preprint arXiv:2603.19607*, 2026.

[89] Hu Yue, Siyuan Huang, Yue Liao, Shengcong Chen, Pengfei Zhou, Liliang Chen, Maoqing Yao, and Guanghui Ren. Ewmbench: Evaluating scene, motion, and semantic quality in embodied world models. *arXiv preprint arXiv:2505.09694*, 2025.

[90] Yu Shang, Zhuohang Li, Yiding Ma, Weikang Su, Xin Jin, Ziyou Wang, Lei Jin, Xin Zhang, Yinzhou Tang, Haisheng Su, et al. Worldarena: A unified benchmark for evaluating perception and functional utility of embodied world models. *arXiv preprint arXiv:2602.08971*, 2026.

[91] Jiahan Zhang, Muqing Jiang, Nanru Dai, Taiming Lu, Arda Uzunoglu, Shunchi Zhang, Yana Wei, Jiahao Wang, Vishal M Patel, Paul Pu Liang, et al. World-in-world: World models in a closed-loop world. *arXiv preprint arXiv:2510.18135*, 2025.

[92] Chak-Wing Mak, Guanyu Zhu, Boyi Zhang, Hongji Li, Xiaowei Chi, Kevin Zhang, Yichen Wu, Yangfan He, Chun-Kai Fan, Wentao Lu, et al. Physicsmind: Sim and real mechanics benchmarking for physical reasoning and prediction in foundational vlms and world models. *arXiv preprint arXiv:2601.16007*, 2026.

[93] Zefan Cai, Haoyi Qiu, Tianyi Ma, Haozhe Zhao, Gengze Zhou, Kung-Hsiang Huang, Parisa Kordjamshidi, Minjia Zhang, Wen Xiao, Jiuxiang Gu, et al. Mmgr: Multi-modal generative reasoning. *arXiv preprint arXiv:2512.14691*, 2025.

[94] Harold Haodong Chen, Disen Lan, Wen-Jie Shu, Qingyang Liu, Zihan Wang, Sirui Chen, Wenkai Cheng, Kanghao Chen, Hongfei Zhang, Zixin Zhang, et al. Tivibench: Benchmarking think-in-video reasoning for video generative models. *arXiv preprint arXiv:2511.13704*, 2025.

[95] Jingqi Tong, Yurong Mou, Hangcheng Li, Mingzhe Li, Yongzhuo Yang, Ming Zhang, Qiguang Chen, Tianyi Liang, Xiaomeng Hu, Yining Zheng, et al. Thinking with video: Video generation as a promising multimodal reasoning paradigm. *arXiv preprint arXiv:2511.04570*, 2025.

[96] Wei Chow, Jiageng Mao, Boyi Li, Daniel Seita, Vitor Guizilini, and Yue Wang. Physbench: Benchmarking and enhancing vision-language models for physical world understanding. *arXiv preprint arXiv:2501.16411*, 2025.

[97] Xinrun Xu, Pi Bu, Ye Wang, Börje F Karlsson, Ziming Wang, Tengtao Song, Qi Zhu, Jun Song, Zhiming Ding, and Bo Zheng. Deepphy: Benchmarking agentic vlms on physical reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 34160–34168, 2026.

[98] Yuhao Wu, Maojia Song, Yihuai Lan, Lei Wang, Zhiqiang Hu, Yao Xiao, Heng Zhou, Weihua Zheng, Dylan Raharja, Soujanya Poria, et al. From perception to action: An interactive benchmark for vision reasoning. *arXiv preprint arXiv:2602.21015*, 2026.

[99] Chi Wan, Kangrui Wang, Yuan Si, Pingyue Zhang, and Manling Li. WorldAgen: Unified state-action prediction with test-time world model training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 18584–18592, 2026.

Appendix

Table of Contents

A Benchmark Details **23**
- A.1 Cognitive-Science Grounding of the Five Properties . . . 23
- A.2 Mathematical Foundations of the Five Properties . . . 24
- A.3 Task Catalog Overview . . . 26
- A.4 Shared Generation and Quality Control . . . 26
- A.5 Continuity Tasks . . . 27
- A.6 Separation Tasks . . . 31
- A.7 Order Tasks . . . 33
- A.8 Enclosure Tasks . . . 37
- A.9 Knots Tasks . . . 40
- A.10 Dataset Statistics . . . 43
- A.11 Evaluation Design . . . 43

B Experiments and Analysis **44**
- B.1 Models and Setup . . . 44
- B.2 Training . . . 44
- B.3 Illegal Actions . . . 47
- B.4 Human Evaluation . . . 47
- B.5 Full Results by Difficulty . . . 49
- B.6 Generative Models as Topological World Models . . . 51
- B.7 Camera and Viewpoint Controls . . . 53
- B.8 Shortcut and Input Representation Controls . . . 54
- B.9 Do Foundation Models Have Topological Biases? . . . 57

C Error Analysis **58**
- C.1 Annotation Framework . . . 58
- C.2 Error Category Definitions . . . 59
- C.3 Cross-Model and Cross-Task Distributions . . . 63

D Extended Related Work Discussion **63**

E Prompt Templates **67**
- E.1 Property-Level Prompt Templates . . . 67
- E.2 Task-Level Prompt Cards . . . 68

F The Use of Large Language Models **92**

G Extended Set **93**

A. Benchmark Details

A.1. Cognitive-Science Grounding of the Five Properties

Our taxonomy is grounded in the developmental psychology account of topological cognition initiated by Piaget and Inhelder [3], refined by subsequent mathematical and educational analyses [12, 11], and formalized in standard algebraic topology [1]. Piaget's original five classes are proximity, separation, order, enclosure, and continuity. We depart from them in two respects. Proximity, which develops in tandem with separation and is constitutive of it [12], is absorbed into our Separation property, and Knots is promoted to a property of its own for the reasons given below.

Continuity. Continuity, in Piaget's phrasing, is the perception of a line, surface, or spatial field as an unbroken whole [3]. Poincaré captures its minimal form as a relation in which adjacent elements are indistinguishable ($A{=}B$, $B{=}C$) while distant ones are not ($A{\neq}C$) [57]. Piaget himself treats continuity as the synthesis of the other four primitives rather than a primitive on the same level, a view that Martin reinforces by showing the four to be mutually constitutive [12]. The standard mathematical formalization is path-connectedness, the existence of a continuous map from $[0, 1]$ joining two points [1, Ch. 1]. Our Continuity tasks (2D MAZE, 3D MAZE, PIPE) ask whether two locations lie in the same path-component of a region given only its visual depiction.

Separation. Separation is the ability to distinguish neighboring elements. Piaget illustrates the lack of it with the syncretic infant percept of an object leaning against a wall, perceived as a single ill-defined patch until separation analyses it into two units [3]. Martin argues that proximity and separation develop in tandem. Finer separation does not displace proximity but allows the child to perceive different degrees of it over larger fields [12]. The topological correspondent is the decomposition of a space into its connected components, the most basic invariant preserved under homeomorphism [1, Ch. 0]. Our Separation tasks (ASSEMBLY, ONE STROKE) ask the model to decide which substructures are independent units and which belong to a single connected whole.

Order. Order is the relation of spatial succession, by which elements are arranged one after another along a direction. Piaget documents it in the infant's gaze across the rungs of a cot and in the way young children seriate objects along a line [3]. Piaget treats seriation as one of the deep epistemic structures whose combination underwrites mathematical thought, and Martin notes that, among Piaget's primitives, order admits the cleanest mathematical formalization [12]. Its Euclidean refinement gives oriented coordinate systems and similarity transformations that preserve relative position. Our Order tasks test sequence read-out along a one-dimensional substrate (BEAD), point-tracking through ordered folds (ORIGAMI POINT), and recovery of permutations under elementary operations (SWAP).

Enclosure. Enclosure, which Piaget calls *surrounding*, is the relation by which a boundary partitions space into an interior and an exterior [3]. In three dimensions it takes the form of *insideness*, exemplified by an object inside a closed box or by the contrast between a surface with a hole and one without. Algebraic topology formalizes both sides of this intuition. The Jordan Curve Theorem [1, §2.B] shows that any subspace of S^2 homeomorphic to S^1 separates S^2 into two components, and a hole in a higher-dimensional space is detected by a spherical cycle bounding a missing interior [1, Ch. 2]. Our Enclosure tasks cover three faces of this property. SHEEP tests interior versus exterior judgment under fence-induced partition, HOLE

tests the enumeration of bounded missing regions, and CHAT NOIR tests planning under a closing-boundary constraint.

Knots. Knots occupy an ambiguous place in Piaget's own framework. Chapter 4 of *The Child's Conception of Space* uses them to probe surrounding, arguing that distinguishing a true knot from a deceptive one requires grasping how a strand encloses itself [3]. Strohecker later re-positions knot cognition as a comprehensive ability that draws on all of Piaget's substructures at once, together with set- and group-theoretic intuitions that are absent from the original primitives [11]. Recent empirical work supports this dissociation, showing that knot reasoning fails to track other physical-intuition abilities [14]. We therefore give Knots its own property column, since it captures phenomena such as crossing parity, linking number, and isotopy that no single Piagetian primitive accounts for. Our tasks KNOTS and UNTANGLE test the perceptual and interventional sides of this composite ability.

A.2. Mathematical Foundations of the Five Properties

This section defines each property mathematically and explains how the benchmark tasks use the resulting invariants and related structural targets. A scene specification s determines an embedded structure $X(s) = (E, K, \mu)$. Here $E \in \{\mathbb{R}^2, \mathbb{R}^3\}$ is the ambient space, $K \subset E$ is the compact embedded structure, and μ contains the marked elements. When K denotes a free region, we use its closure inside the compact board domain $D \subset E$. This convention makes K compact in every task.

Two marked embedded structures $X = (E, K, \mu)$ and $X' = (E, K', \mu')$ are *ambient-isotopic*, written $X \simeq X'$, if there is a continuous map $h\colon E \times [0,1] \to E$ such that, for every $t \in [0,1]$, the map $h_t(\cdot) = h(\cdot, t)$ is a homeomorphism, $h_0 = \mathrm{id}_E$, $h_1(K) = K'$, and h_1 carries every marked element in μ to its corresponding marked element in μ' [2, Ch. 1].

These definitions draw on four sources. Hatcher provides the algebraic topology used for Continuity, Separation, and Enclosure [1]. Rolfsen provides the knot theory used for Knots [2]. Order follows Huntington's postulates for cyclic order [13]. Martin's mathematical analysis of Piaget's primitives connects the cognitive taxonomy to these formal notions [12].

The structures produced by our generators are locally path-connected. Therefore, connected components and path components coincide. We write $\pi_0(K)$ for the set of components of K. A task's cognitive category does not guarantee that it evaluates the corresponding invariant directly. Some tasks instead use a derived structural target. We identify each such case below.

Definition 1 (Continuity)**.**
Let $p, q \in K$ be marked points. The continuity invariant is $\Phi_{\mathrm{cont}}(X) = \mathbf{1}[\,[p] = [q] \text{ in } \pi_0(K)\,]$, that is, whether some continuous path $\gamma\colon [0,1] \to K$ joins p to q [1, Ch. 1].

2D MAZE and 3D MAZE query Φ_{cont} with K the closed free region of the board. A what-if edit inserts or removes a wall, which changes K and hence possibly $\pi_0(K)$. PIPE uses the invariant as its goal predicate. Writing $K(\theta)$ for the union of pipe segments in rotation state θ, the goal set contains exactly the states in which every segment lies in the component of the source.

Definition 2 (Separation)**.**
The separation invariant $\Phi_{\mathrm{sep}}(X)$ is the partition of the marked elements induced by the components of K, in which two marks are identified exactly when they lie in the same element of $\pi_0(K)$ [1, Ch. 0].

Piaget defines separation as mere disjointness of two sets, which Martin shows to be strictly weaker than the mathematical notion and to be preserved by arbitrary injections [12]. We

therefore formalize the object individuation his tasks target by the component partition instead. ASSEMBLY asks whether the two candidate subassemblies together reconstruct the complete object, without missing, extra, or substituted parts. ONE STROKE evaluates the invariant on the complement. The drawn stroke P partitions the board interior, and an episode succeeds when the components of the board minus P induce exactly the color classes of the marked cells.

Definition 3 (Order)**.**
Let $\gamma \subset E$ be an embedded arc or circle carrying marked points $\mu = (p_1, \ldots, p_n)$ together with a decoration that fixes a start mark and a traversal direction. The order invariant $\Phi_{\text{ord}}(X)$ is the sequence in which the marks are met when γ is traversed from the start mark in the given direction. The underlying linear or cyclic order type is preserved by orientation-preserving homeomorphisms of γ and reversed by orientation-reversing ones [13], and the decoration selects one representative of this orbit under rotations and reversals.

In BEAD, the decoration is rendered explicitly as a white start marker and a tangential direction arrow, so the sequence answer is unique. The relationship queries ignore the decoration and compare the color sequences of two strings inside the orbit, with `IDENTICAL`, `REVERSED`, `CYCLIC_ROTATION`, and `DIFFERENT` naming the orbit relations. Martin verifies that these order relations are topological invariants and that arc, circle, and figure eight are pairwise non-homeomorphic [12]. The other two Order tasks sit next to this invariant rather than inside it. ORIGAMI POINT belongs to the Order category cognitively, but its formal target is a correspondence. It tracks marks through a fold trajectory, a sequence of embeddings of the sheet into $\mathbb{R}^3$, conserving each mark's intrinsic position on the sheet while its Euclidean position varies, and it scores the recovered correspondence between marks and labels by set equality [3]. SWAP's goal predicate matches the full labeled grid arrangement against a target rather than an order type along an embedded curve. It operationalizes ordered rearrangement in the planning setting.

Definition 4 (Enclosure)**.**
For a closed structure K in the plane, the enclosure invariant labels each marked point $p \in E \setminus K$ by whether it lies in a bounded component of $E \setminus K$. For a Jordan curve, the complement has exactly one bounded component, its interior [1, §2.B]. For a solid $K \subset \mathbb{R}^3$, the hole-count invariant is the first Betti number $b_1(K) = \operatorname{rank} H_1(K)$ [1, Ch. 2].

SHEEP queries the planar labeling with K the fence layout, so a sheep is enclosed exactly when its position has no path to the unbounded component of the complement. HOLE queries b_1 of the rendered board. The generator produces solids that are handlebodies with no internal cavities, so $H_1(K)$ is free and b_1 equals the number of through-handles, since each through-hole contributes one independent cycle while pits and shallow depressions contribute none. CHAT NOIR uses enclosure as its goal predicate. An episode succeeds when the cat's cell lies in a component of unblocked cells that contains no boundary cell.

Definition 5 (Knots)**.**
Let K be one or more disjoint embedded circles in $\mathbb{R}^3$. The knot invariant is the ambient isotopy class of K [2, Ch. 1]. The tasks query it coarsely. A single loop is *unknotted* if it is ambient-isotopic to the standard circle, and two loops are *split* if an embedded sphere separates them. With both components oriented, the linking number lk is the sum of crossing signs where one loop passes under the other in a diagram [2, §5.D]. $\text{lk} \neq 0$ certifies a non-split link, while $\text{lk} = 0$ does not certify splitness, as the Whitehead link shows.

KNOTS asks for these classifications from a single rendering, together with component counts and what-if queries about removing one loop. Link relation ground truth is recorded at

generation time in the scene's link graph rather than recovered from an invariant. UNTANGLE operates on the projected diagram rather than on the isotopy class. Its goal set contains the configurations whose overhead projection has zero crossings between rope paths, a projection-level, viewpoint-dependent surrogate for disentanglement rather than an isotopy invariant.

A.3. Task Catalog Overview

MINDTOPO comprises 13 task types organized by the five topological properties and two cognitive levels (Reasoning, Planning). Table 6 lists every task with its property slot, cognitive level, dataset size, and the appendix subsection in which it is described in detail. The remainder of this section (Appendices A.5 to A.9) provides the per-task description.

Task	Property	Cognitive level	Easy	Hard	Total	Section
2D Maze	Continuity	Reasoning	150	150	300	§A.5.1
3D Maze	Continuity	Reasoning	150	150	300	§A.5.2
Pipe	Continuity	Planning	150	150	300	§A.5.3
Assembly	Separation	Reasoning	150	150	300	§A.6.1
One Stroke	Separation	Planning	150	150	300	§A.6.2
Bead	Order	Reasoning	150	150	300	§A.7.1
Origami Point	Order	Reasoning	150	150	300	§A.7.2
Swap	Order	Planning	150	150	300	§A.7.3
Hole	Enclosure	Reasoning	150	150	300	§A.8.1
Sheep	Enclosure	Reasoning	150	150	300	§A.8.2
Chat Noir	Enclosure	Planning	150	150	300	§A.8.3
Knots	Knots	Reasoning	150	150	300	§A.9.1
Untangle	Knots	Planning	150	150	300	§A.9.2
Reasoning total			1,200	1,200	2,400	
Planning total			750	750	1,500	
Total			**1,950**	**1,950**	**3,900**	

Table 6 | MINDTOPO task catalog. Reasoning instances are single-shot questions and planning instances are multi-step episodes. Section points to the subsection that describes each task's scene generator, difficulty levels, and scoring rule.

A.4. Shared Generation and Quality Control

All benchmark instances are produced automatically by generators specific to each task that use fixed seeds. Each generator samples a state, renders the model-visible observation, and derives the reference answer or terminal success condition from the generated state and metadata, as described in Section 2.3. Human annotators are not used to create benchmark labels.

Every exported record stores a generation seed and configuration metadata. Reasoning records contain the rendered inputs, question, reference answer, and difficulty. Planning records contain the initial state, difficulty, and action budget, while the success condition is implemented by the environment.

Quality control differs by task because the validity criteria differ. Depending on the task, the generator enforces structural constraints, rejects ambiguous or unsolvable states, verifies visibility when visibility is required, or checks the requested difficulty constraints. An instance is exported only after its applicable conditions hold. Planning environments validate submitted actions and evaluate success from the resulting simulator state.

The following task sections describe the generation settings, difficulty controls, output formats, scoring rules, and applicable acceptance criteria for every task.

A.5. Continuity Tasks

A.5.1. 2D Maze (Continuity, Reasoning)

Targeted ability. The 2D Maze environment isolates a model's ability to reason about *global connectivity* purely from a top-down rendering, with no symbolic graph input.

Task formulation. Two question types are posed over rendered mazes. The reachability question (`reachability_set`) marks several labeled points and asks which of them are connected to the target point A, with the answer returned as a bracketed name list such as `[B, D]`. The bar removal question (`bar_removal`) marks two points A and B that are disconnected in the shown maze and paints a subset of the walls as colored bars. The model must list every bar whose removal on its own reconnects A and B, for example `[purple, red]`. Both types share `answer_type = name_list` and are scored by set equality. The benchmark contains 300 questions, 150 at each difficulty level. The Easy level has 61 reachability and 89 bar removal questions, and the Hard level has 75 of each type.

Scene generation at the Easy level. Easy scenes are small mazes with mixed wall styles, produced in three deterministic, seed-driven stages with ground truth supplied by a connectivity oracle.

Skeleton. Recursive-backtracking depth-first search (DFS) on the $N \times N$ grid with $N = 6$ yields a spanning tree, after which walls are closed or opened until $W = \text{round}(\rho M) = 36$ of the $M = 2N(N-1) = 60$ internal wall slots are occupied, which corresponds to a wall density of $\rho = 0.6$.

Wall taxonomy. A fraction of the closed walls is converted into full diagonal walls (Mode-dA), and a further fraction is demoted to half-length variants (Mode-B/C for horizontal and vertical edges, Mode-dB/dC for diagonals). Across the Easy scenes, diagonal walls make up 39% of all walls in reachability scenes and 25% in bar removal scenes, and partial walls make up 31% in both. The renderer admits five wall types in three families (Table 7). Full walls (Mode-A on a shared cell edge, Mode-dA across a cell diagonal) block passage, while every partial wall (Mode-B/C/dB/dC) leaves an unobstructed sub-segment of its host edge and therefore does not separate the adjacent regions in the underlying maze graph. The task prompt states this rule explicitly. Mode-dA walls partition their host cell into two halves through the cell center, so labeled points at cell centers are never placed in diagonal-bearing cells.

Annotation. Reachability scenes place five labeled points (A and four candidates) in diagonal-free cells. Bar removal scenes place A and B and paint four bars (purple, red, green, and blue) on blocking walls, which are either full horizontal or vertical edges (Mode-A) or full diagonals (Mode-dA). Across the 89 bar removal questions, 260 bars lie on axis-aligned walls and 96 on diagonal walls. A scene is rejected unless A and B are disconnected, at least two bars reconnect them when removed individually, and each of the two components containing A and B covers at least one third of the board area. The last two constraints rule out answers that can be guessed from a single bar and lopsided components in which the bars around the smaller side dominate the correct set.

Connectivity oracle. Ground truth is computed by a breadth-first search (BFS) over a *4-triangle decomposition* of each cell (T_N, T_E, T_S, T_W sharing the cell center). Each horizontal or vertical edge is owned by one triangle and each cell-internal diagonal separates exactly two of them, so a single BFS uniformly handles axial walls and diagonal walls. Because labeled points sit at cell centers (the shared vertex of all four triangles), the start cell's four triangles are all seeded at hop 0. The same routine drives both the answer key and the ORACLE sanity baseline, which by construction achieves acc. = 1.000.

Wall type	Has passage?	Geometry (cell edge length = 1)
Mode-A	×	full horizontal/vertical edge, length 1
Mode-B	✓	centered half-edge, length 1/2
Mode-C	✓	endpoint-anchored half-edge, length 1/2
Mode-dA	×	full cell diagonal, length $\sqrt{2}$
Mode-dB / dC	✓	partial cell diagonal, length $\sqrt{2}/2$

Table 7 | Wall taxonomy of the 2D Maze renderer, assuming unit cell edge length. "Has passage?" indicates whether the two regions adjacent to the wall remain connected in the underlying maze graph. Easy scenes use all five types, while Hard scenes use only Mode-A walls.

Scene generation at the Hard level. Hard scenes are large mazes built from full edge walls only (Mode-A), so the difficulty comes from long, winding routes rather than from wall styles. The level contains 50 questions on each of the 12×12, 18×18, and 24×24 grids, split evenly between the two question types. Each scene starts from a random DFS spanning tree of the grid, in which every pair of cells is joined by exactly one corridor route. Two tree edges are then cut, which splits the maze into three connected regions. A scene is kept only if every region covers at least $\max(10, 0.1N^2)$ cells and some pair of regions shares at least N boundary walls, so the regions are large and irregular rather than small rooms. After the cuts, between 46.6% and 48.1% of the internal wall slots are closed, depending on the grid size.

Every answer is tied to a long route. For a route through the maze we measure its length in cell moves, its number of turns, and its number of branch entrances, that is, side corridors that open off the route at its interior cells. On the 12×12, 18×18, and 24×24 grids, a qualifying route must have at least 25, 60, and 110 moves, at least 8, 18, and 30 turns, and at least 4, 8, and 12 branch entrances. In reachability scenes, A lies in the largest region, every reachable candidate is joined to A by a qualifying route, and the remaining candidates lie in the other two regions. Each scene shows six labeled points (A and five candidates), and the number of reachable candidates is balanced over 0 to 5 (13, 13, 13, 12, 12, and 12 questions). In bar removal scenes, A and B lie in two neighboring regions and six bars are painted in purple, red, green, blue, yellow, and orange. A correct bar sits on the boundary between the regions of A and B, and the route it opens from A to B must qualify, with at least a quarter of the minimum length on each side of the bar. Distractor bars sit on boundaries between other pairs of regions, at a Manhattan distance of at least $\lfloor N/4 \rfloor$ cells from the marked points. The number of correct bars is 1, 2, or 3 in 25 questions each, and every color is correct in exactly 25 questions. Because every wall is a full edge, ground truth is a plain BFS over the cell graph. All Hard answers were rechecked with an independent union-find solver, and each scene admits a single-wall edit that changes its answer, so the answer depends on the specific wall layout.

Difficulty levels. Difficulty is determined directly by the generation configuration rather than by a post-hoc score. The Easy level tests connectivity in a compact maze whose wall styles must be told apart, and the Hard level tests whether connectivity can be tracked along long routes in much larger mazes (Table 8).

Output format and scoring. Both question types share the `name_list` answer contract. The prompt instructs the model to return a single JSON object whose `answer` field holds a list of names, with `[]` reserved for the empty case. The legal vocabulary is sample specific, namely the scene's labeled point names for reachability questions and its bar colors for bar removal questions. Predictions and ground truth are coerced to uppercased, whitespace-stripped frozensets and compared by *set equality*, so order, case, and duplicates do not affect the score. Parsing and aggregation follow the shared pipeline of Appendix A.11. The complete task prompt and qualitative examples appear in Figures 19 and 20 (Appendix E.2).

Parameter	Easy	Hard
Grid size	6×6	12×12, 18×18, 24×24
Wall styles	full, partial, diagonal	full edge only
Closed internal wall slots	36 of 60 (ρ = 0.6)	46.6% to 48.1%
Minimum route length (moves)	not constrained	25, 60, 110
Minimum route turns	not constrained	8, 18, 30
Minimum branch entrances	not constrained	4, 8, 12
Reachability labeled points	5	6
Reachability correct set size	0 to 4	0 to 5
Bar removal bars	4	6
Bar removal correct set size	2 to 4	1 to 3
Reachability questions	61	75
Bar removal questions	89	75

Table 8 | Difficulty levels of the 2D Maze task. Hard entries with three values refer to the 12×12, 18×18, and 24×24 grids, which contribute 50 questions each. Route constraints apply to every route that defines a Hard answer. Easy bar removal scenes additionally require at least two correct bars and components covering at least one third of the board.

A.5.2. 3D Maze (Continuity, Reasoning)

Targeted ability. This task evaluates whether a model can infer global connectivity in a metric 3D environment from visual evidence alone. Unlike the 2D Maze task, the scene contains furnished rooms, occlusions, and door states, so the model must integrate multiple views before deciding whether two marked locations belong to the same connected component.

Task formulation. Each instance provides five rendered views of the same ProcTHOR [58] house at 1920×1080 resolution, namely a top-down view and four oblique views in the front, right, rear, and left directions relative to the sampled house orientation. Labeled navigation points are placed on valid floor locations. We use two reasoning question types. In the target-point connectivity question, the model is given a source point and must list every other labeled point reachable from it under the shown door states. In the door-opening question, two points A and B are disconnected, every currently closed controllable door is marked with a distinct color, and the model must list the unique minimum set of closed doors that must be opened to connect the two points. The benchmark contains 300 questions, each with its own rendered house, 150 at each difficulty level. The Easy level has 79 connectivity and 71 door-opening questions, and the Hard level has 75 of each.

Scene generation. We instantiate furnished indoor layouts with the ProcTHOR/AI2-THOR renderer [58, 59] and treat each house as a 3D maze over valid navigation locations. For each sampled scene, the generator fixes the requested number of rooms, controllable doors, and labeled points, then samples door states and point locations with a deterministic seed. The five views are rendered from the same state and stored with the question row. Ground truth is computed by a navigation connectivity oracle that respects walls, closed doors, furniture, and other solid obstacles, while open doors and open floor or corridor spaces remain passable.

At the Easy level, houses have 6 rooms with 5 controllable doors or 7 rooms with 6 controllable doors, and every scene shows five labeled points. Connectivity questions use random door states, with between 0 and 6 doors open, and the correct set contains 0 to 4 of the four candidate points. Door-opening questions start with all five doors closed and colored, and the unique minimum set contains 1 to 4 doors.

At the Hard level, every house has 8 rooms, and the generator adds constraints that require tracking connectivity across many rooms. In connectivity questions the house has 8 controllable doors, of which 5 to 7 are open, and seven labeled points are shown (the source and six

candidates). The source's component spans 6 or 7 of the 8 rooms, and the correct set contains 2 to 5 candidates. At least two of the connected candidates are reachable only along long routes through several rooms, and at least one disconnected candidate lies close to the source's component, typically separated from it by a single closed door. One disconnected candidate also serves as a deliberate distractor, reached by a route whose first doors are open but which is blocked by a closed door near its end (37 questions) or by several such detours that are each blocked (38 questions). In door-opening questions the house has 7 or 8 controllable doors, of which 2 or 3 are already open, and only the 4 to 6 closed doors are colored. The prompt therefore asks the model to count only currently closed doors and to minimize the number of doors opened rather than walking distance. The unique minimum set contains 3 or 4 doors. In 37 questions a single route reaches B, and in the other 38 two candidate routes compete, so the model must compare how many closed doors each route crosses. The answer route is the one with more doors in total in 19 of these questions and the one with fewer doors in the other 19, so route length alone does not reveal the answer. Every Hard answer is verified against the navigation oracle.

Difficulty levels. Difficulty is defined by the room, door, and point setup together with the door-state constraints above (Table 9).

Level	Question type	Rooms	Doors	Points	Open doors	Answer size	Questions
Easy	Connectivity	6 or 7	5 or 6	5	0 to 6	0 to 4	79
Easy	Door opening	6	5	5	0	1 to 4	71
Hard	Connectivity	8	8	7	5 to 7	2 to 5	75
Hard	Door opening	8	7 or 8	5	2 or 3	3 or 4	75

Table 9 | Difficulty levels of the 3D Maze task. Doors counts the controllable doors in the house, and answer size counts the points or doors in the correct set.

Output format and scoring. For target-point connectivity questions, the expected answer is a JSON object whose `answer` field lists point names in alphabetical order, such as `["B","D"]`, and the empty set is written as `[]`. Door-opening questions use the same schema with door color names. Both are scored by exact set equality after normalization. The complete task prompt and qualitative examples appear in Figures 21 and 22 (Appendix E.2).

*A.5.3. **Pipe*** (Continuity, Planning)

Targeted ability. Pipe evaluates sequential continuity reasoning under local rotations. A successful model must identify disconnected pipe components, anticipate how 90-degree rotations change openings, and plan a sequence that connects every pipe segment to the source.

Task formulation. The model observes a 5×5 pipe board at each step. The green source marks the root of the network, connected pipes are rendered in green, and disconnected pipes in blue. At every turn, the model chooses one non-empty cell and the environment rotates that pipe clockwise by 90°. An episode succeeds when all non-empty pipe cells are connected to the source through matching openings before the action budget is exhausted. The benchmark contains 300 episodes, 150 at each difficulty level.

Scene generation. Each puzzle is generated from a connected tree-shaped pipe network whose solved state is stored with the episode. The generator samples the source cell, the active pipe count, and the number of three-way junctions, rejects loops, and then scrambles the pipes by seeded random rotations. This construction preserves a known solved state while requiring the

model to recover it through local rotations from the rendered board. The action budget is $\lceil 1.3\,R \rceil$, where R is the number of clockwise rotations in the reference solution.

At the Easy level, 17 to 23 of the 25 cells hold a pipe and the network has 5 to 7 three-way junctions. Four-way junctions are excluded, the reference solution needs 23 to 27 rotations, and between 10 and 20 pipes receive a nonzero scramble rotation.

At the Hard level, every one of the 25 cells holds a pipe and the network again has 5 to 7 three-way junctions, which leaves 7 to 9 single-port endpoints. The board also contains two kinds of four-port pieces, each carrying two separate channels rather than a four-way junction. A raised overpass connects north to south and east to west without joining the two channels, and its connections do not change under rotation. A double bend shows two separate quarter-circle curves that pair north with east and south with west, or north with west and south with east, and one clockwise quarter-turn switches between the two pairings. Every Hard board contains 1 to 3 overpasses and 1 to 3 double bends, 2 to 4 four-port pieces in total, all placed in interior cells, and the per-turn prompt explains these rules. The solved state is a tree over channels, each channel turns green independently, and both channels of every four-port piece must reach the source. Hard boards are also constrained to be hard to read off locally. Only 1 to 3 channels are connected to the source at the start, counting the source itself, at least 20 pipes start in a wrong orientation, and at least one double bend starts with the wrong pairing. The generator further proves that the solved orientation is unique. After propagating the constraints imposed by the board boundary and by neighboring pipes, at least 14 pipes must still admit several orientations, and at least 6 must remain ambiguous after every single-pipe trial assignment has been tested and its consequences propagated. Because the solution is unique, its 40 to 48 rotations are the minimum number of actions needed.

Difficulty levels. Difficulty is controlled by network density, the presence of two-channel pieces, the scrambling constraints, and the reference solution length (Table 10).

Level	Grid	Active pipes	Junctions	Four-port pieces	Reference rotations	Budget	Episodes
Easy	5×5	17 to 23	5 to 7	0	23 to 27	30 to 36	150
Hard	5×5	25	5 to 7	2 to 4	40 to 48	52 to 63	150

Table 10 | Difficulty levels of the Pipe task. Junctions counts three-way junctions, and four-port pieces are the overpasses and double bends of the Hard level. The budget equals $\lceil 1.3\,R \rceil$ for a reference solution of R rotations.

Output format and scoring. Each action is a JSON object whose `answer` field holds the integer coordinates `x` (column) and `y` (row) of the cell to rotate, following the visual labels on the board. We evaluate the full trajectory rather than individual moves. An episode is correct if the environment reaches a state in which every non-empty pipe, and at the Hard level every channel of every pipe, belongs to the source-connected component within the prescribed step budget. The complete task prompt and qualitative examples appear in Figure 23 (Appendix E.2).

A.6. Separation Tasks

A.6.1. Assembly (Separation, Reasoning)

Targeted ability. Assembly tests whether a model can recover object decomposition from a rendered 3D assembly. The task targets separation reasoning. The complete object must be mentally partitioned into valid subassemblies, while distractors preserve plausible shape and category cues.

Task formulation. Each instance contains one image of a complete object and five candidate decomposition options. The complete-object image combines two oblique views, while each option shows a proposed split into two subassemblies. The model must select the option that exactly matches a valid decomposition of the original object.

Scene generation. We build the task from a catalog of assembly-style objects with known primitive parts and object categories. An object enters the pool only if every primitive part is visible in the two fixed complete-object views, with at least $\min(5\,\text{px}, 5\%)$ of the part's solo rendered area in view. For each instance, the generator selects a balanced two-part split as the correct option, in which the two subassemblies differ in size by at most one part, and constructs four distractors. Each distractor is built on a different two-part partition of the same object, so no option can be rejected from its partition alone. The five options are shuffled deterministically. After rendering, a scene is kept only if every primitive part shown in every option keeps at least $\min(13\,\text{px}, 13\%)$ of its solo rendered area visible in that option view. Otherwise it is regenerated from the next scene seed. Each question therefore consists of one complete-object rendering and five option renderings.

Difficulty levels. The benchmark contains 300 reasoning questions, 150 per level, built from 17 catalog objects in six categories. Every object in both levels has at least 11 primitive parts, and the two levels differ mainly in how the distractors are constructed (Table 11). The Easy level uses 16 objects with 11 to 19 parts from all six categories. Its four distractors are structurally heterogeneous. One adds an extra same-category component, two replace one target component with a similar component from another object of the same category, and one removes a target component. Part counting or a single mismatched component can therefore expose some of these distractors. The Hard level uses 12 objects with 11 to 15 parts from four categories. All four of its distractors are similar-part replacements. Each one swaps exactly one target part for a part taken from a different object of the same category, scaled and rotated to the bounding box of the removed part (per-axis size ratios between 0.61 and 1.59). Every Hard option thus has the same number of parts as the complete object and differs from it only in the shape of one component. A per-option visibility check additionally requires the substituted part to stay visible in the option rendering, so the deciding evidence is always in view. Each object is sampled with several seeds so that option ordering and distractor selection vary while the underlying decomposition remains exact.

Level	Part count	Objects	Object categories	Distractors	Questions
Easy	11 to 19	16	Bench, Chair, Desk, Misc, Shelf, Table	1 extra, 1 missing, 2 replaced	150
Hard	11 to 15	12	Bench, Chair, Misc, Table	4 aligned similar-part replacements	150

Table 11 | Difficulty levels for the Assembly task.

Output format and scoring. The canonical response is a multiple-choice JSON answer such as `{"answer":"E"}`. Predictions are normalized to the option letter and scored by exact match against the shuffled correct option. The complete task prompt and qualitative examples appear in Figures 24 to 27 (Appendix E.2).

A.6.2. One Stroke (Separation, Planning)

Targeted ability. One Stroke evaluates separation-aware path planning. The model must draw a single continuous stroke that separates differently colored regions while keeping cells of the same color connected on the same side of the stroke.

Task formulation. The environment presents a colored grid with a cursor starting at the bottom-left corner and a target at the top-right corner. At every turn, the model issues one of four moves, `U`, `D`, `L`, or `R`. The stroke cannot leave the board, reuse an edge, or create a closed loop. Moving back over the most recent edge is legal and undoes that edge. The episode succeeds only when the completed path reaches the target while satisfying the color-region separation constraint.

Scene generation. Every puzzle is played on a 6×6 cell board, which corresponds to a 7×7 grid of stroke vertices. The generator first samples a legal construction path, a simple lattice path from the bottom-left to the top-right vertex, and computes the regions into which this path cuts the board. It then gives each of several regions its own color and fills part of that region's cells with it, leaving the remaining cells uncolored, so that the construction path satisfies the grouping and separation constraints by design. Candidates are rejected when the path has too few turns, cuts the board into too few regions, or does not realize the required number of colors. The generator then searches for a shorter valid stroke and stores the shortest one it finds as the reference solution, which is therefore never longer than the construction path. The per-instance action budget is 1.2 times the construction-path length, rounded up.

Difficulty levels. Both levels share the board size, and difficulty is controlled by the number of colors, the number of uncolored cells, and the length of the stroke the model must plan (Table 12). Easy puzzles use 5 colors (6 in a single instance) from a seven-color palette, leave 0 to 13 of the 36 cells uncolored, and are accepted only if an exhaustive breadth-first search finds a shortest valid solution of exactly 20 moves. Their construction paths have 20 to 28 edges. Hard puzzles use 9 or 10 colors from a ten-color palette and leave 11 to 14 cells uncolored. Their reference solutions have 40 to 44 moves with 25 to 36 turns and visit 41 to 45 of the 49 vertices, so the stroke has to weave through nearly the entire board while every turn must respect the color constraints. The benchmark contains 300 planning episodes, 150 per level.

Level	Board	Vertex grid	Colors	Uncolored cells	Reference solution	Budget	Episodes
Easy	6×6	7×7	5 to 6	0 to 13	20	24 to 34	150
Hard	6×6	7×7	9 to 10	11 to 14	40 to 44	48 to 53	150

Table 12 | Difficulty levels for the One Stroke planning task.

Output format and scoring. The action contract is `{"answer":"U"}` with the answer drawn from `{U,D,L,R}`. We score the full trajectory. An episode is successful if the model reaches the target within the action budget and the final stroke satisfies the same-color grouping and different-color separation constraints. The complete task prompt and qualitative examples appear in Figure 28 (Appendix E.2).

A.7. Order Tasks

*A.7.1. **Bead*** (Order, Reasoning)

Targeted ability. Bead tests whether a model can recover color order along a string embedded as a complex 3D curve, including curves that close into a loop. Each bead is identified by its color. A white marker designates the first bead and a tangential arrow specifies the traversal direction, so the required structure is the ordered color sequence encountered from that marked starting point.

Task formulation. The benchmark contains two question types. The sequence task (`T_BS01`) presents one image and asks the model to list every bead color in traversal order. At the Easy level the prompt asks the model to trace the string from one end to the other, and the image marks the start bead and the direction. At the Hard level the prompt also tells the model to start at the marked bead and follow the indicated direction. The colors are separated by commas, as in `"RED, BLUE, GREEN"`. The relationship task (`T_BS02`) presents two images that each show a bead string and asks whether their full color sequences are `IDENTICAL`, `REVERSED`, a `CYCLIC_ROTATION` of one another, or `DIFFERENT`.

Scene generation. Scenes are rendered with a Vite and Playwright pipeline over a Three.js renderer, and bead colors come from a fixed eight-color palette. Each level has its own scene generator. The Easy generator samples a tangled torus-knot loop and places beads at equal arc-length intervals along it. Its color sequences include palindromic and periodic patterns, and scenes are rendered at 1024×1024, mostly from isometric-front or oblique viewpoints. The Hard generator has three settings. The first renders tangled open and closed curves with few crossings. The other two build a single rope from a random braid in which 7 or 8 strands exchange positions over 38 to 46 crossing rows, or 9 or 10 strands over 58 to 68 rows. Exterior connectors join the strand ends into one component, which is cut once when an open string is required. Beads sit on the connectors and, for some strands, inside the braid passes, which leaves long bead-free stretches of rope that must be traced through many crossings. Hard sequence questions use exactly six colors, never repeat a color on adjacent beads, and have no rotational or reflective symmetry. For relationship questions, the two strings share the same color multiset and the same first and last colors (the first two and last two in braid scenes), so the relation cannot be read off from color counts or endpoints. Hard scenes are rendered at 2560×2560 without ground grid or cast shadows, and every image passes geometric clearance checks, a minimum crossing angle of 12°, and a raster check that every bead is fully visible. Their ground truth is additionally checked against the bead encounter order along the rendered rope. In both levels, ground truth is the generated color sequence and is exactly reproducible from the deterministic scene seed.

Difficulty levels. Difficulty is controlled jointly by bead count, curve topology, and the number of projected crossings between consecutive beads (Table 13). The benchmark contains 300 reasoning questions, 150 per level, with 215 sequence questions and 85 relationship questions in total. The Easy level has 137 sequence and 13 relationship questions on torus-knot loops with 6 to 8 beads. Of its relationship questions, 12 are `IDENTICAL` and one is `REVERSED`. The Hard level has 78 sequence and 72 relationship questions, with 18 questions for each relation label. It combines three generator settings of 50 questions each (26 sequence and 24 relationship questions), namely tangled curves with 11 to 14 beads, 7 or 8 strand braids with 14 to 16 beads, and 9 or 10 strand braids with 18 to 20 beads. Half of the Hard sequence images show closed loops and half show open strings, and every relation label receives the same mix of open and closed image pairs.

Level	Curve family	Bead count	Projected crossings	Resolution	Questions (seq./rel.)
Easy	torus-knot loops	6 to 8	n/a	1024×1024	150 (137/13)
Hard	tangled open and closed curves	11 to 14	3 to 10	2560×2560	50 (26/24)
Hard	7 or 8 strand braids	14 to 16	54 to 72	2560×2560	50 (26/24)
Hard	9 or 10 strand braids	18 to 20	93 to 120	2560×2560	50 (26/24)

Table 13 | Difficulty levels for the Bead task. Projected crossings count rope crossings in the rendered image and are recorded for the Hard scenes only.

Output format and scoring. `T_BS01` returns a JSON string such as `{"answer":"RED, BLUE, GREEN"}`. The parser converts this string into an ordered list of color names from the eight color palette. A prediction is correct only when both order and multiplicity match the ground truth exactly. `T_BS02` returns one of `IDENTICAL`, `REVERSED`, `CYCLIC_ROTATION`, or `DIFFERENT` and is scored by exact match after case normalization. The complete task prompt and qualitative examples appear in Figures 29 and 30 (Appendix E.2).

A.7.2. Origami Point (Order, Reasoning)

Targeted ability. Under Piaget's framework of conservation under transformation [3], this task tests whether current multimodal large language models are capable of *distinguishing the invariant* (the identity of marked points on a piece of paper) from *the variant* (their Euclidean location in 3D space) as the paper is folded.

Task formulation. We modify the Origami Simulator [60] to predefine points on the paper and to play a preset folding and rotation trajectory in front of a fixed camera. The task contains 300 instances built on six origami bases (bird, boat, open-sink, pinwheel, square, and waterbomb), with 150 instances at each of the two difficulty levels. Each instance is a sequence of rendered images in which one base folds from a flat sheet to 70% of its folded state while the paper rotates between images. The first image shows the flat sheet with one to three unmarked dots. The intermediate images show the paper without any dots, and the final image shows the folded paper with every dot labeled by an upper-case letter. The model must report the letters in the final image that belong to the dots shown on the flat sheet. Dots in the final image are drawn at randomized sizes, and the prompt states that size carries no identity information, so the dots can only be matched by following the paper through the fold sequence.

Scene generation. Each dot lies at a fixed barycentric coordinate on a face of the crease pattern, so its position in every frame follows from the simulated mesh. Besides the tracked dots, the generator places decoy dots in clusters around them, and every tracked dot has a twin decoy on a different face, placed under one of the base's rotational symmetries (90°, 180°, or 270°), so that local appearance alone does not identify the target. For each preset, the generator searches over sequences of (fold amount, model rotation) pairs and accepts a trajectory only if every tracked dot remains geometrically visible at every step and the labeled dots in the final image are separated by a minimum distance in pixels. Visibility is computed in closed form by casting a ray from each dot to the camera and testing it against the full mesh, since folded panels block sight from either side. Every label is a geometric property of the simulated mesh rather than a human annotation, so ground truth is exact and reproducible from a fixed seed.

Difficulty levels. The two levels differ in sequence length, rotation, and distractor design (Table 14). Easy instances show 10 images, carry two or three tracked dots among 20 lettered dots, and place every dot on the front side of the paper. The paper turns gradually and mostly in one direction, by about 13° per image in the median step, accumulating between 26° and 308° of rotation (median 118°). Hard instances show 14 images and carry one or two tracked dots among 16 to 22 lettered dots, of which up to eight lie on the underside of the paper and become visible only after the paper turns over. Rotation in Hard is both larger and non-monotone, as the paper swings back and forth by about 30° per image and accumulates between 288° and 419° of rotation (median 380°), although its final pose is at most 179° from the initial one. Hard also uses a smaller minimum separation between lettered dots, so the final image is more crowded.

Parameter	Easy	Hard
Origami bases (instances)	bird 30, boat 29, pinwheel 29, waterbomb 29, open-sink 23, square 10	pinwheel 73, bird 53, waterbomb 23, boat 1
Images per instance	10	14
Fold amount	0% to 70%	0% to 70%
Unmarked dots on the flat sheet	2 (103), 3 (47)	1 (121), 2 (29)
Lettered dots in the final image	20	16 to 22
Dots on the back side	0	0 to 8
Accumulated rotation	26° to 308°	288° to 419°
Largest rotation between consecutive images	3° to 34°	23° to 37°
Minimum dot separation (px)	50	39
Final dot size scale	0.6 to 1.4	0.75 to 1.25
Questions	150	150

Table 14 | Easy and Hard levels of the Origami Point task. Counts in parentheses give the number of instances.

Output format and scoring. The expected response is a JSON list of upper-case letters drawn from the instance-specific label set, e.g. `{"answer":["A","C"]}`. The ground truth always contains between one and three letters. Predictions and ground truth are normalized to sets and compared by set equality, so the answer is order-insensitive but identity-sensitive. Every emitted letter must belong to the label set of the final image, and a missing or spurious letter makes the answer incorrect. The complete task prompt and qualitative examples appear in Figures 31 to 34 (Appendix E.2).

A.7.3. Swap (Order, Planning)

Targeted ability. Swap evaluates whether a model can reason about order through a sequence of constrained permutations. The model must transform an initial grid into a target grid by using the blank cell as the only exchange medium.

Task formulation. Each episode provides two images at every turn, the current arrangement and the goal arrangement. The grid contains one blank cell, and every other cell holds a colored block marked with a unique letter. At each step, the model selects one non-empty cell, and the chosen block is swapped with the blank. At the Easy level any non-empty cell may be selected. At the Hard level the selected cell must share a row or a column with the blank. It need not be adjacent to the blank, and the blocks between the two cells do not move. An illegal action leaves the grid unchanged but still consumes a step. The task is solved when the current arrangement exactly matches the goal arrangement. The benchmark contains 300 episodes, 150 at each level.

Scene generation. Easy episodes use a 4×4 grid with 15 blocks. Under a deterministic seed, the generator samples both the initial and the goal arrangement and computes the exact shortest-path distance in the state graph where any non-empty cell may swap with the blank. The step budget is this theoretical minimum multiplied by 1.2 and rounded up. Hard episodes use a 5×5 grid with 24 blocks, and the goal places the blocks in alphabetical order with the blank in the bottom right corner. The seed selects a target distance between 40 and 44. The blank starts at its goal position and all 24 blocks start misplaced, each with a wrong row, a wrong column, or both, so that the total number of wrong coordinates equals the target distance. A bounded beam search then looks for a legal solution with exactly that many moves, and boards without such a solution are rejected. Because a same-row or same-column swap changes only one coordinate of the moved block, each move corrects at most one wrong coordinate, so the number of wrong coordinates is a lower bound on the solution length and the solution found is provably shortest. Every Hard episode receives a fixed budget of 75 actions.

Difficulty levels. The two levels differ in grid size, move rule, and solution length (Table 15). Hard episodes require roughly three times as many moves as Easy ones, and the row and column restriction forces the model to route the blank through intermediate cells instead of fetching any block directly.

Parameter	Easy	Hard
Grid shape	4×4	5×5
Colored blocks	15	24
Legal cells	any non-empty cell	same row or column as the blank
Goal arrangement	sampled	alphabetical, blank in the corner
Misplaced blocks at start	9 to 15	24
Blank starts at its goal position	12 of 150	150 of 150
Shortest solution (steps)	11 to 18	40 to 44
Step budget	14 to 22	75
Episodes	150	150

Table 15 | Easy and Hard levels of the Swap task.

Output format and scoring. Actions specify a grid position, e.g., `{"answer":{"row":1, "col":2}}`. We evaluate the executed episode, and the prediction is correct only if the model reaches the goal arrangement within the step budget. The complete task prompt and qualitative examples appear in Figure 35 (Appendix E.2).

A.8. Enclosure Tasks

*A.8.1. **Hole*** (Enclosure, Reasoning)

Targeted ability. Hole evaluates enclosure reasoning over solid 3D objects. The model must count only through-holes that connect the top surface to open space, while ignoring pits, shadows, and shallow depressions that do not pass through the board.

Task formulation. Each instance shows a single top-down rendering of a procedurally generated board. The prompt asks for the number of visible holes on the top board. If a lower board is visible in the scene, the prompt instructs the model to ignore it and evaluate only the top board.

Scene generation. The renderer builds a 10×10 board whose outline is drawn from rectangles, circles, and polygons, and views it from almost directly above. It then cuts four kinds of openings. A plain hole passes through the board and counts once. A pit is a blind depression that does not pass through the board and does not count. A partially floored hole passes through the board but has a shelf at half the board thickness covering 25% to 50% of its extent along one edge, so part of it looks like a pit, and it counts once. A split hole has a band at half the board thickness across its middle that divides it into two through openings, so it counts twice. For half of the seeds, a second board is placed partly beneath the top board with a small offset, so that open space under some holes shows another surface. Ground truth is generated directly from the procedural layout as the number of through openings that remain visible on the top board.

Difficulty levels. The generator has three settings that widen both the answer range and the set of opening types. The first targets 5 to 10 through openings with plain holes and one to three pits, the second targets 7 to 13 and adds one to three partially floored holes, and the third targets 9 to 17 and uses all four opening types with at least one of each. The Easy level contains 50 instances from each setting, and the Hard level contains 150 instances from the third setting. Each instance has one rendered image, and the lower board appears in 74 instances at each level. Table 16 reports the composition and the ground-truth range of both levels.

Parameter	Easy	Hard
Generator settings	first, second, third (50 each)	third (150)
Board shapes	circle 56, rect 50, polygon 44	circle 57, polygon 50, rect 43
Lower board present	74	74
Ground-truth hole count	4 to 18	7 to 17
Mean ground-truth count	10.2	12.3
Questions	150	150

Table 16 | Easy and Hard levels of the Hole reasoning task. Within the Easy level, ground-truth counts range from 4 to 10, 7 to 13, and 9 to 18 for the first, second, and third generator settings.

Output format and scoring. The expected answer is a scalar integer, e.g., `{"answer":6}`. Predictions are scored by exact integer match after parsing, so off-by-one counts and counts that include non-through pits are marked incorrect. The complete task prompt and qualitative examples appear in Figure 36 (Appendix E.2).

A.8.2. Sheep (Enclosure, Reasoning)

Targeted ability. Sheep probes inside and outside judgments and bounded-region reasoning in nested and partitioned fence layouts. The model must recover each labeled sheep's enclosing layer or partition cell and determine whether an opening in the outer fence leaves a route to the outside. These judgments operationalize the enclosure relation induced by closed planar boundaries [1, §2.B].

Task formulation. Each instance shows an elevated 3D rendering of a pasture in which numbered sheep are scattered across a fence layout. The benchmark contains four question types. Q1, asked for both scene families, requires counting the sheep that cannot reach the outside without crossing a fence. For nested-fence scenes, Q2 returns the sheep IDs that can escape through outer-fence gaps, and Q4 asks how many such gaps must be repaired to close the outermost fence. For partitioned scenes, Q3 asks which labeled cell contains the most sheep.

Scene generation. Scenes are produced by a Vite and Playwright renderer that supports two layout families. *Nested-fence* scenes contain two to four nested convex, concave, or irregular boundaries, with zero or more gaps placed only on the outermost fence. *Partitioned* scenes divide one closed enclosure into labeled cells using one of five layouts, namely grid, hex-cross, nested polygon, polygon star, or radial. For each scene, the generator samples its shape or partition parameters, the camera tilt away from the overhead view, and the sheep coordinates from a deterministic seed, then derives ground truth from geometric layer and cell membership. In a nested-fence scene, a sheep in the outermost layer can escape whenever the outer fence has at least one gap, whereas sheep in deeper layers cannot. In a partitioned scene, every sheep inside the closed outer boundary is counted as unable to escape. Scenes are resampled until all requested sheep satisfy spacing constraints, and partitioned scenes must have a unique most-populated cell containing at least two sheep.

Difficulty levels. Sheep contains 300 questions, 150 at each of the Easy and Hard levels (Table 17). The Easy level uses the generator's densest setting throughout. Its nested-fence scenes have three or four concave or irregular layers, 13 to 18 sheep, and one to five gaps in the outer fence. Its partitioned scenes cover all five layouts and place 12 to 16 sheep in 4 to 16 cells. Every Easy scene is viewed with a camera tilt of 30° or 45°. The level contains 56 Q1 questions (24 nested and 32 partitioned), 45 Q2, 4 Q3, and 45 Q4 questions. The Hard level spans three generator settings with 50 questions each. They range from two or three layers with 7 to 10 sheep, through two to four layers with 10 to 14 sheep, to three or four layers with 13 to 18 sheep,

and together cover zero to five outer gaps and camera tilts from 15° to 45°. Only 3 Hard scenes are partitioned, each with 11 sheep in 6 to 8 cells. The Hard level is weighted toward escape reasoning, with 117 Q2, 21 Q4, and 12 Q1 questions (9 nested and 3 partitioned) and no Q3 question. Twelve of its Q2 questions have the answer `"NONE"`, and eleven of its Q4 questions require no repair. All 300 questions are distinct. They are posed on 214 rendered scenes, and the 30 scenes that appear at both levels are asked with a different question type at each level.

Level	Setting	Nested layers	Outer gaps	Sheep (nested/partitioned)	Partition cells	Camera tilt	Q1/Q2/Q3/Q4
Easy	single	3 to 4	1 to 5	13 to 18 / 12 to 16	4 to 16	30°, 45°	56/45/4/45
Hard	a	2 to 3	0 to 2	7 to 10 / 11	6 to 8	15°, 30°	5/33/0/12
	b	2 to 4	0 to 3	10 to 14 / n/a	n/a	15° to 45°	5/40/0/5
	c	3 to 4	1 to 5	13 to 18 / n/a	n/a	30°, 45°	2/44/0/4

Table 17 | Composition of the Sheep task. Each level has 150 questions, and each Hard setting contributes 50. Ranges are taken over the scenes used at each level.

Output format and scoring. Q1 and Q4 return JSON integers. Q2 returns a list of sheep IDs separated by commas or `"NONE"`. Q3 returns a region label. Integer answers and region labels are scored by exact match after parsing. Q2 answers are compared as sets of IDs, so their order does not affect correctness. The complete task prompt and qualitative examples appear in Figures 37 to 40 (Appendix E.2).

A.8.3. Chat Noir (Enclosure, Planning)

Targeted ability. Chat Noir evaluates adversarial enclosure planning. The model must progressively block a hex-grid board so that the moving cat loses every path to the boundary before it can escape.

Task formulation. At each turn, the model observes the current board and selects one open non-cat cell to block. The cat then moves according to the level's policy. The episode succeeds if the cat has no remaining path from its current cell to any boundary cell, and fails if the cat reaches the boundary or the action budget is exhausted. The budget equals the number of open non-cat cells at the start of the episode.

Scene generation. The generator samples the board radius and the initial blockers from a deterministic seed and places the cat on the center cell, so its initial distance to the boundary equals the board radius. Initial blockers are drawn uniformly from non-cat cells. A candidate layout is rejected if the cat is already trapped, has no legal move, or lacks a path to the boundary. A search filter then verifies, against the deterministic policy of the cat used at that level, that the player has a winning opening. The two levels apply different versions of this filter, described below, so that every episode is solvable but still requires multi-step enclosure planning.

Difficulty levels. Difficulty is set mainly by the cat policy, together with the opening filter (Table 18). Chat Noir contains 300 episodes, 150 per level, and every episode starts from a distinct layout. The Easy level uses a connectivity-aware greedy cat that avoids immediate one-move traps when possible and otherwise prefers a shorter boundary distance while keeping more escape routes open. Boards have radius 3 (62 episodes, 12 to 14 initial blockers) or radius 4 (88 episodes, 16 to 19 initial blockers), which gives budgets of 22 to 24 and 41 to 44 actions. Each Easy layout admits at least five winning first blocks within a bounded search of depth six or seven that expands at most six candidate blocks per turn, and at least three of these blocks

are adjacent to the cat. The Hard level uses an expert cat that, before each move, checks every legal move for a forced capture within three complete player turns. It first takes an immediate escape when one exists, then avoids moves that allow capture within one, two, or three turns, and finally prefers shorter escape paths and more shortest escape routes. All Hard boards have radius 4 with 10 to 13 initial blockers, which gives budgets of 47 to 50 actions. Each Hard layout must have at least one certified winning first block and provably no winning first block adjacent to the cat. The certification searches against the expert policy over the full action budget, under a limit of 200,000 search nodes and 30 seconds, and an unfinished search never counts as a proof. The winning lines found by this search require 6 to 24 player blocks. Blocking a cell next to the cat therefore never starts a win, and the model must open with a block away from the cat whose value only appears several moves later.

Level	Cat policy	Radius	Initial blockers	Budget	Episodes
Easy	connectivity-aware greedy	3 / 4	12 to 14 / 16 to 19	22 to 24 / 41 to 44	62 / 88
Hard	expert three-turn lookahead	4	10 to 13	47 to 50	150

Table 18 | Difficulty levels for the Chat Noir planning task. Easy entries separated by a slash refer to radius 3 and radius 4 boards. Easy layouts admit at least five winning first blocks, at least three of them adjacent to the cat, whereas Hard layouts admit no adjacent winning first block.

Output format and scoring. The canonical action is a blocked cell index. An example response is `{"answer":12}`. We score the resulting interaction, and an episode is correct only if the model traps the cat before escape and within the maximum number of allowed actions. The complete task prompt and qualitative examples appear in Figure 41 (Appendix E.2).

A.9. Knots Tasks

*A.9.1. **Knots*** (Knots, Reasoning)

Targeted ability. The Knots task evaluates recognition of global knot and link structure from a single rendered image. It tests whether a model can distinguish knots from visually deceptive unknots and open ropes, count continuous rope components, recover link relations among multiple components, and reason counterfactually about removing one component.

Task formulation. Each question presents a single rendered image of one or more ropes. `T01` assigns one of six categories to the whole scene, namely A (simple closed ring), B (knot), C (open rope with no knot), D (link), E (unlinked multiple ropes), and F (other or mixed). `T02` counts the number of distinct ropes. `T03` classifies a closed ring scene as A (not linked), B (chain, including a two-ring paired link), C (all interlocked), or D (mixed). `T04` asks which remaining rings become free after a specified colored ring is cut and removed. This removal question is generated only for supported configurations with three to six distinctly colored rings and metadata that records the link graph. Finally, `T05` counts how many ropes are linked to at least one other rope.

Scene generation. Both levels draw their structures from a catalog of 24 scene types. Its 16 types with one rope cover closed and open unknots, visually deceptive unknots, trefoil and figure eight knots, torus knots with more crossings, and loose or occluded variants. The other 8 types cover Hopf links, unlinked rings, chains, Borromean rings, multiple link groups, and mixed scenes containing both linked and free rings. The two levels present these structures in different ways. Easy scenes are rendered as 3D ropes seen from an oblique camera, with rope color, curve slackness, and deformation controlled by a deterministic seed. Loose knots,

near-miss unlinks, and mixed connectivity provide explicit visual traps. Hard scenes start from a planar diagram of the same known knot or link, built with the Spherogram library, and expand it through recorded Reidemeister II and III moves until the projection reaches a target number of crossings. These moves preserve the isotopy class [2], so the ground truth is that of the original structure, while the drawing no longer resembles its canonical silhouette. Every move is checked for legality, and the rendered centerlines are checked against the complete signed crossing order of the expanded diagram. Each Hard image is a 2560×2560 rendering of rounded tubes with explicit over and under bridges, a rope diameter of at least 11 pixels, and crossing angles of at least 31.9°. Open ropes end at visible fixed posts. All ropes in a Hard image share one color except in `T04`, where each ring has its own color, and six Hard scenes used for `T02` and `T05` adopt a more strongly wrapped projection.

Difficulty levels. Knots contains 300 questions, 150 per level, and every question uses its own image (Table 19). The Easy level comes from the generator's densest scene setting, which favors dense or mixed link structures and component counts above six. It contains 47 `T01`, 49 `T02`, 3 `T03`, 6 `T04`, and 45 `T05` questions, rendered from 80 distinct scenes. Its `T01` scenes are either single knotted ropes, such as the trefoil, a loose cinquefoil, and torus knots up to $T(3,5)$ and $T(2,9)$, or multi-component scenes with 2 to 10 ropes, and their answers are 20 B, 21 D, 4 E, and 2 F. The `T02` and `T05` scenes contain 8 to 10 ropes, with 8 to 10 ropes to count in `T02` and 7 to 10 linked ropes in `T05`. The `T04` scenes contain 4 to 6 rings, and three or four rings become free after each removal. The Hard level spans three settings of projected crossings, 36 to 50, 66 to 90, and 112 to 160, with 50 questions each and 10 per task in every setting. Its scenes contain 1 to 8 ropes, and all 24 scene types appear. The answers are balanced by construction. Each setting contains 2 A, 4 B, and one each of C, D, E, and F among its `T01` questions. `T02` answers range from 3 to 8. `T03` has 8, 8, 7, and 7 answers for A to D. `T04` has 10 questions each in which no ring, one ring, or several rings become free. `T05` always shows six ropes, and its answers 0, 2, 3, 4, 5, and 6 each occur five times. Borromean rings are excluded from `T05`, which asks about pairwise linkage.

Level	Setting	Presentation	Ropes	Projected crossings	T01/T02/T03/T04/T05
Easy	single	oblique 3D view	1 to 10	not controlled	47/49/3/6/45
Hard	a	expanded planar diagram	1 to 8	36 to 50	10/10/10/10/10
	b	expanded planar diagram	1 to 8	66 to 90	10/10/10/10/10
	c	expanded planar diagram	1 to 8	112 to 160	10/10/10/10/10

Table 19 | Composition of the Knots task. Each level has 150 questions, and each Hard setting contributes 50. Projected crossings count the crossings of the expanded diagram that is rendered.

Output format and scoring. `T01` returns one option letter from A through F. `T03` returns one option letter from A through D. `T02` and `T05` return zero or positive integers. `T04` returns a JSON list of the colors of all rings that become free, or `["none"]` when none do. Primary correctness requires an exact match after parsing. The `T04` color list is compared as a set, so list order does not affect correctness. The complete task prompt and qualitative examples appear in Figures 42 to 45 (Appendix E.2).

A.9.2. Untangle (Knots, Planning)

Targeted ability. The Untangle task evaluates sequential spatial planning for eliminating crossings among simulated ropes on a pegboard. The model must track rope endpoints, anticipate how relocating a lifted endpoint changes the projected rope layout, and select a sequence of legal moves that removes all crossings between rope paths in the projection from above. The

task therefore probes planning over projected entanglement rather than determining the isotopy class of a knot in three dimensions.

Task formulation. The environment presents an overhead view of a $G \times G$ grid of holes through which R colored ropes are threaded, with each rope's two endpoints occupying distinct holes. Following the renderer's coordinate convention, row indices are shown along the top edge and increase from left to right, while column indices are shown along the left edge and increase from top to bottom. At every turn, the model observes the rendered grid and selects an occupied endpoint and an unoccupied target hole. The environment lifts the endpoint above the other ropes, moves it to the target hole, lowers it, and advances the rope physics for 60 frames. After the ropes settle, the crossing count is recomputed as the number of rope pairs whose simulated centerline paths intersect in the projection from above. An episode succeeds when this count reaches zero before the action budget is exhausted. Invalid or illegal actions consume one step without changing the state.

Scene generation. For each generator setting, the generator starts from a bank of noncrossing endpoint templates formed by parallel rows or columns, then performs multiple seeded scramble searches using legal endpoint relocations. Immediate reversals are excluded. Intermediate moves are weighted by the target crossing range, the number and connectivity of involved ropes, crossing separation, and visual readability. The generator samples a candidate that completes the setting's scramble depth and passes its final structural constraints. In the six-rope setting, the generator also seeds a compact central tangle before physics relaxation, and candidates must contain a crossing component of three or four ropes. At least three ropes must each cross more than one other rope, and the crossing graph must contain at least one cycle. Evaluation also computes auxiliary metadata with a symbolic endpoint solver. This solver represents each rope by the straight segment joining its two holes. It runs breadth first search to remove all segment overlaps defined using the rope width, subject to a time limit of 10 seconds and an expansion limit of 200,000 states. The resulting plan length is reported as a proxy for the theoretical minimum and also drives the local oracle policy. The solver does not compute an exact shortest plan for the simulated rope physics, and it plays no role in scoring.

Difficulty levels. The generator settings scale the grid dimension, the rope count, and the scramble depth, and they raise the required crossing participation and graph structure (Table 20). Untangle contains 300 episodes, 150 per level, and all of them share a budget of 15 actions. The Easy level uses the six-rope setting throughout, with 6×6 grids, scrambles of 7 to 9 moves, and 5 to 7 initial crossings. The Hard level spans three settings with 50 episodes each, namely four ropes on a 5×5 grid, five ropes on a 6×6 grid, and six ropes on a 6×6 grid. The two levels share no episode. The table reports the initial crossing counts under the same physics metric that decides success.

Level	Grid	Ropes	Scramble	Crossings	Budget	Episodes
Easy	6×6	6	7 to 9	5 to 7	15	150
Hard	5×5	4	2 to 3	2 to 3	15	50
	6×6	5	5 to 6	4 to 6	15	50
	6×6	6	7 to 9	5 to 7	15	50

Table 20 | Difficulty levels for the Untangle task. The Hard level contains one row per generator setting.

Output format and scoring. Each turn returns a nested JSON action with four integer fields named `src_row`, `src_col`, `tgt_row`, and `tgt_col`. These fields identify the occupied source

hole and the empty target hole. An episode is correct only when the executed actions reduce the physics crossing count defined above to zero within the action budget. The complete task prompt and qualitative examples appear in Figure 46 (Appendix E.2).

A.10. Dataset Statistics

Table 21 breaks the benchmark down by topological property, cognitive level, and difficulty level.

Property	Reason. #Q	Plan. inst.	Easy	Hard	Total
Continuity	600	300	450	450	900
Separation	300	300	300	300	600
Order	600	300	450	450	900
Enclosure	600	300	450	450	900
Knots	300	300	300	300	600
Total	**2,400**	**1,500**	**1,950**	**1,950**	**3,900**

Table 21 | MINDTOPO dataset composition across the five topological properties, the Reasoning and Planning levels, and the Easy and Hard difficulty levels, with 150 instances per task at each level.

A.11. Evaluation Design

Reasoning tasks. Reasoning tasks share a single response contract. The model receives one rendered scene or a fixed bundle of views, emits one JSON response, and is scored without environment interaction. Across the eight reasoning tasks, the answers fall into four families. *Set* answers occur in the 2D MAZE reachability and bar-removal questions, 3D MAZE connectivity, ORIGAMI POINT, SHEEP escape IDs, and KNOTS T04 free ring colors. *Sequence* answers occur in the BEAD sequence description task. *Scalar integer* answers occur in HOLE, SHEEP Q1/Q4, and KNOTS T02/T05. *Categorical label* answers occur in ASSEMBLY, KNOTS T01/T03, the BEAD sequence relationship task, and SHEEP Q3. All four families are parsed by the same layered extractor in `topobench_eval.answer_parser`. It first attempts a strict `{"answer": ...}` JSON extraction. If that fails, it tries final answer phrase matching, scans the text for legal values, and examines the response tail before declaring a parse failure. Set answers are scored by frozenset equality, sequences by ordered list equality, scalars by exact integer match, and labels by exact match against the legal vocabulary for each instance.

Planning tasks. Planning tasks expose an action interface and are scored on the executed trajectory rather than on the textual plan. At every turn the model receives the current rendered state and emits one JSON action. The environment validates legality, applies the action or charges the budget for an illegal one, and renders the next state. When a model returns a thought summary before its answer, only the text after the summary is parsed, so a move mentioned while thinking is never executed in place of the final answer. Each task ships with a per-instance step budget. PIPE allows 1.3 times the reference number of rotations, rounded up. SWAP allows 1.2 times the shortest solution, rounded up, at the Easy level and a fixed 75 actions at the Hard level. ONE STROKE allows 1.2 times the construction path length, rounded up (Table 12). UNTANGLE allows a fixed 15 actions, and CHAT NOIR allows one action per open cell other than the cat's cell. There is no retry, so budget exhaustion or violation of the success predicate ends the episode and counts as a failure. Each planning task additionally ships with two reference policies, an ORACLE solver and a RANDOM policy that samples legal actions uniformly under the same budget. The reference is exact where an environment exposes a precomputed optimal plan. For UNTANGLE, it is the auxiliary symbolic endpoint solver described above rather than an exact oracle for the simulated rope physics.

Metrics. The primary per-task metric is *accuracy*, the per-instance 0/1 correctness averaged over the evaluation split. For planning tasks, this metric is the *episode success rate*, the fraction of episodes whose executed trajectory satisfies the success predicate within the step budget. For diagnosis, the pipeline also computes a *JSON parse rate* (fraction recovered by the strict-JSON layer) and an *answer-format hit rate* (fraction whose strict JSON additionally passes the per-instance shape and legal-value validator), isolating format-following failures from reasoning failures, and, for planning tasks, a *mean over-optimality*, the average gap between the executed trajectory length and the task's reference plan length on solved episodes. The tables in this paper report accuracy. The Untangle reference length is the symbolic endpoint proxy defined above. We aggregate accuracy in two ways. The *per cognitive level mean* reports separate averages for Reasoning and Planning. The *macro property mean* averages the five property accuracies, weighting each topological invariant equally regardless of its question count.

Summary. The unified pipeline therefore handles three sources of answer non-uniqueness in a consistent way. *Set-valued* answers are coerced to frozensets so that order, case, and duplicates do not inflate or deflate the score. *Multi-valid* planning trajectories are accepted as long as the final-state predicate holds within the budget, so any plan that removes all projected crossings or traps the cat is correct. Finally, *length-mismatched* sequences (e.g., a bead-color list of the wrong length) fall through the strict equality check and are marked incorrect, rather than being partially credited.

B. Experiments and Analysis

B.1. Models and Setup

Models. Table 22 lists every foundation model and generative backend assessed in this study, together with the evaluated snapshot and the pipeline each one enters.

Evaluation protocol. For reasoning tasks, each model answers a single-turn visual question. Decoding uses `temperature=0` and `do_sample=false`, with at most 2048 newly generated tokens. Each prompt asks for exactly one JSON object, and the evaluator applies the shared fallback parser only after the model has finished. Planning tasks use the same deterministic decoding settings and run through the Hydra+Playwright environment runner. Each turn includes the current rendered observation, the task rules, and the legal answer schema. We do not give symbolic state graphs, oracle connectivity, or hidden generator metadata to any model. Failed API calls are retried according to the per-provider configuration, after which the episode or sample is marked invalid.

All static images are supplied at the renderer's native resolution and preserved as PNG inputs. Planning screenshots are captured from the browser frontend. Environments that need a fixed viewport use the configured 1440×960 or 1440×1080 viewport before scene-only cropping.

Prompts. We use one prompt template per task type, kept fixed across all models, with no task-specific prompt tuning after seeing model outputs. Representative property-level prompt templates appear at the end of the paper in Appendix E.1, Figures 14 to 18.

B.2. Training

Training tasks and data splits. All training experiments start from `Qwen/Qwen3-VL-2B-Instruct`. The reasoning suite contains 2D Maze, Assembly, Bead, Sheep, and Knots, while the planning suite contains Pipe, One Stroke, Swap, and Untangle. For each single-task experiment, we split the Easy level data into training, validation, and test sets in proportions of 80%, 10%,

Organization	Model Name	Snapshot	Full Name	Evaluation Pipeline
Proprietary Models				
OpenAI	GPT-6 Astra [15]	OpenAI API	`gpt-6-astra`	Reasoning + planning
OpenAI	GPT-5.6-Sol [16]	OpenAI API	`gpt-5.6-sol`	Reasoning + planning
OpenAI	GPT-5.5 [17]	OpenAI API	`gpt-5.5`	Reasoning + planning
OpenAI	GPT-5.4-mini [18]	OpenAI API	`gpt-5.4-mini-2026-03-17`	Reasoning + planning
OpenAI	GPT-5.6-Luna [16]	OpenAI API	`gpt-5.6-luna`	Probing planner
OpenAI	GPT-5.4-mini [18]	OpenAI API	`gpt-5.4-mini`	Legacy probing audit
Google	Gemini-3.1-Pro [20]	Google AI Studio	`gemini-3.1-pro-preview`	Reasoning + planning
Google	Gemini-3.1-Flash-Lite [19]	Google AI Studio	`gemini-3.1-flash-lite-preview`	Reasoning + planning
Anthropic	Claude-Haiku-5.5 [21]	Anthropic API	`claude-haiku-5-5`	Reasoning + planning
Open-Weight Models				
InternLM	InternVL3.5-241B [26]	Intern-AI API	`OpenGVLab/InternVL3_5-241B-A28B-Instruct`	Reasoning + planning
NVIDIA	Nemotron-Nano-12B-VL-v2 [22]	DigitalOcean	`nvidia/NVIDIA-Nemotron-Nano-12B-v2-VL-BF16`	Reasoning + planning
Google	Gemma-4-31B-IT [27]	Google AI Studio	`google/gemma-4-31B-it`	Reasoning + planning
Alibaba	Qwen3.5-397B-A17B [23]	Alibaba Cloud	`Qwen/Qwen3.5-397B-A17B`	Reasoning + planning
Meta	Llama-4-Maverick-17B-128E [24]	DigitalOcean	`meta-llama/Llama-4-Maverick-17B-128E-Instruct`	Reasoning + planning
Mistral AI	Ministral-3-14B-Instruct-2512 [25]	Mistral API	`mistralai/Ministral-3-14B-Instruct-2512`	Reasoning + planning
Image Generative Models				
OpenAI	GPT-Image-2 [29]	OpenAI API	`gpt-image-2`	Image-edited imagined rollouts
ByteDance	BAGEL-7B [61]	self-hosted	`ByteDance-Seed/BAGEL-7B-MoT`	Image imagined rollouts
ThinkMorph	ThinkMorph-7B [62]	self-hosted	`ThinkMorph/ThinkMorph-7B`	Image imagined rollouts
Video Generative Models				
Lightricks	LTX-2.3 [33]	self-hosted	`Lightricks/LTX-2.3`	CV-audited image-to-video rollouts
Lightricks	LTX-2.5 [34]	self-hosted	`Lightricks/LTX-2.5`	CV-audited image-to-video rollouts
Wan-AI	Wan2.2-I2V-A14B [30]	self-hosted	`Wan-AI/Wan2.2-I2V-A14B`	Probing and CV-audited rollouts
ByteDance	Seedance-2.0-Mini [31]	API	`seedance-2.0-mini`	Probing video rollouts
Google	Veo-3.1-Lite [32]	API	`veo-3.1-lite-generate-preview`	Probing and CV-audited rollouts
MiniMax	MiniMax-H3 [35]	self-hosted	`MiniMaxAI/MiniMax-H3`	CV-audited image-to-video rollouts
ByteDance	Seedance-2.5 [36]	API	`doubao-seedance-2-5-260628`	CV-audited image-to-video rollouts

Table 22 | Details of foundation models and generative backends assessed in this study. "Snapshot" records the evaluated API/checkpoint snapshot when no stable public release date is available.

and 10%, using the fixed seed `20260705`. Reasoning examples are stratified by question type. The held-out experiments train on all Easy level examples from four reasoning tasks and assign one fifth of the fifth task to validation and the remaining four fifths to test. For Bead, we use a simplified variant with fewer beads than in the standard benchmark.

Supervised fine-tuning. SFT uses answer-only supervision. Reasoning targets contain only the JSON answer required by the benchmark, with no chain-of-thought or explanatory text. For planning, the target is one complete JSON action sequence from the initial observation. One Stroke uses exact shortest-path search, Pipe uses the minimum required clockwise rotations, Swap uses the environment's shortest action sequence, and Untangle uses exact breadth-first

search under the same width-aware overlap criterion as the environment. Every planning target is replayed and must solve the corresponding environment before it enters the training set.

We train completion-only LoRA adapters for one epoch with a global batch size of 5. The prompt and assistant-prefix tokens are masked from the loss. LoRA is applied to all linear modules with rank 32, alpha 64, and dropout 0.05. We use AdamW with learning rate 10^{-4}, cosine decay, a 0.03 warmup ratio, weight decay 0.01, gradient clipping at 1.0, BF16, and gradient checkpointing. The maximum completion length is 128 tokens for reasoning and 2048 tokens for planning.

Reinforcement learning. RL uses GRPO through the VAGEN/VERL training stack [55]. Each update contains five prompts. We sample eight completions per reasoning prompt and 32 per planning prompt, giving 40 and 160 sampled completions per update, respectively. Sampled completions use temperature 0.8 and top-p 0.95. We train for one epoch with learning rate 10^{-6} and KL coefficient 0.001. A correct answer or successful action sequence receives reward 1.0, a parseable but incorrect answer receives 0.0, and an invalid output receives -0.1. Planning reward is computed by executing the predicted sequence in the actual environment. No credit is assigned for intermediate actions that do not reach the success state. The RL runs do not enable LoRA and update the model from either the base checkpoint or the merged SFT checkpoint.

SFT followed by RL and leave-one-task-out RL. For SFT + RL, we merge the task-specific LoRA adapter into the base checkpoint and initialize GRPO from the merged model. The subsequent data split, reward, sampling, and optimization settings are identical to the corresponding RL-only run. For leave-one-task-out RL, we train five separate policies. Each policy sees four reasoning environments during training and is evaluated on the excluded environment, so every value in the Leave-one-task-out RL row of Table 3 comes from a different held-out policy.

Evaluation protocol. Decoding is deterministic with one greedy completion per example. Reasoning predictions use the benchmark's answer parser and task-specific scorer. Planning predictions are parsed as complete action sequences and executed from a fresh environment reset. A prediction is correct only if execution reaches the task-defined success state. The single-task comparisons keep the input format, scorer, and test examples fixed across Base, SFT, RL, and SFT + RL, using the same test set. Leave-one-task-out RL uses the same input format and scorer and is evaluated on the test portion of the excluded task.

Training	3D Maze	Origami Point	Hole
Base	18.37	19.60	0.10
Held-out 2D Maze	–	0.90	4.80
Held-out Sheep	18.78	0.40	–
Held-out Knots	18.57	0.90	8.41
Held-out Bead	20.28	–	4.20
Held-out Assembly	18.78	6.50	0.00

Table 23 | Transfer performance (%) for the Base policy and RL policies trained while holding out one reasoning task.

Held-out transfer. Table 23 evaluates the five leave-one-task-out RL policies on three additional reasoning tasks. Performance on 3D Maze remains close to the base policy, ranging from 18.57% to 20.28% across the applicable held-out policies. All policies fall below the 19.60% base result on Origami Point, while Hole ranges from 0.00% to 8.41% against a base of 0.10% and remains low in absolute terms. Together with the held-out results in the main table, these results show

that cross-task transfer is uneven and depends on the target relation rather than following automatically from multi-environment training.

B.3. Illegal Actions

Legality rules. An action is illegal when it falls outside the environment-specific legal-action set. The precise rejection conditions differ by environment and are summarized in Table 24. These checks are applied by the environment after parsing the model output and before any state transition is executed.

Environment	Illegal-action condition
`knots_untangle`	The source is not a movable occupied endpoint, the target is not an empty in-bounds hole, or the pair of source and target is not a permitted endpoint move.
`continuity_pipe`	The selected coordinate is out of bounds or points to an empty cell.
`separation_one_stroke`	The direction is invalid, leaves the board, reuses a prohibited edge, or creates a closed loop. Immediate backtracking over the previous edge remains legal.
`order_swap_2d_puzzle`	The selected coordinate is out of bounds or does not identify a movable non-empty cell. At the Hard level, selecting a cell that shares neither a row nor a column with the blank is also illegal.
`enclosure_chat_noir`	The selected index is out of bounds, is already blocked, or identifies the cat's current cell.

Table 24 | Environment-specific conditions under which a parsed action is rejected as illegal.

Invalid responses and API failures. Malformed model output is recorded as an invalid response and converted to an invalid-action sentinel before environment execution. API failures are tracked separately and do not reach the environment, so they are not counted as environment-level illegal actions.

Effect of rejection. An illegal action leaves the environment state unchanged and consumes one interaction step. The harness records both the rejection flag and its associated reason, but under the default evaluation protocol the model receives only the resulting unchanged observation rather than an explicit rejection message. The harness can also list the legal actions in the prompt through the option `run.include_legal_moves_in_prompt`, which is disabled by default.

B.4. Human Evaluation

B.4.1. Annotation Interface and Human Performance Evaluation

Human annotators use the same visual inputs and answer schemas as model evaluations. For reasoning tasks, the interface loads each JSONL sample, renders the image panel(s), exposes a task-specific answer widget (single choice, integer field, set/list entry, or sequence entry), and saves an optional comment field for ambiguous cases. For planning tasks, annotators play the same browser-based environment as the model runner. The system resets the episode from the JSONL `reset_config`, records every selected action, and scores success from the environment terminal state rather than from self-reported answers. Figures 8 and 9 show the two interfaces.

Five annotators were recruited through a professional annotation company and had substantial experience in visual data annotation. Before production, each annotator received written instructions and video training, completed 300 trial examples, and passed a qualification round. Two additional auditors monitored annotation quality throughout the evaluation. Annotators were paid more than 1.5 times the applicable local minimum hourly wage and agreed that their answers and performance results could be used for research. The evaluation used only synthetic puzzle data and collected no personal or sensitive information. It did not involve institutional review.

The five annotators completed one annotation per example on the previous release of the benchmark, whose instances form the Easy level. The Hard level was generated afterwards and has no human annotations. We therefore compare humans with models on the Easy level only. Annotations were kept as per-tier accuracies, so Table 25 takes for each task the human accuracy on the previous-release tier from which its Easy instances are drawn. Humans average 94.23% on the reasoning tasks and 99.87% on the planning tasks, against 76.33% and 81.87% for GPT-6 Astra, the strongest model. Human accuracies are computed with the same parser and scorer used for models. Set-valued answers are canonicalized by sorting and de-duplicating labels, scalar answers are normalized through the shared answer parser, and planning episodes are marked correct only when the environment terminates successfully. Because ground-truth labels are derived by the generators, these accuracies serve as an empirical upper-bound check rather than as a source of labels.

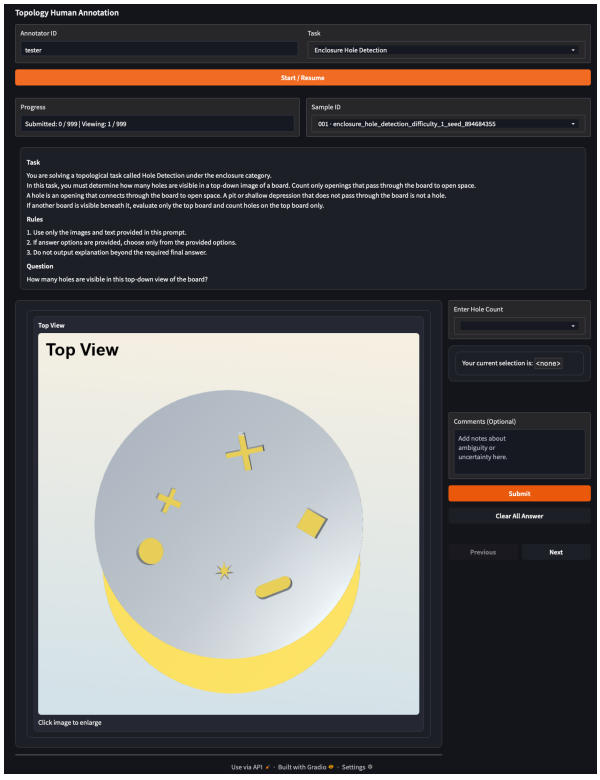


Figure 8 | Human annotation interface for reasoning tasks. Annotators are shown the same visual inputs and task instructions used in model evaluation, together with a task-specific answer widget and an optional comment field for ambiguous cases.

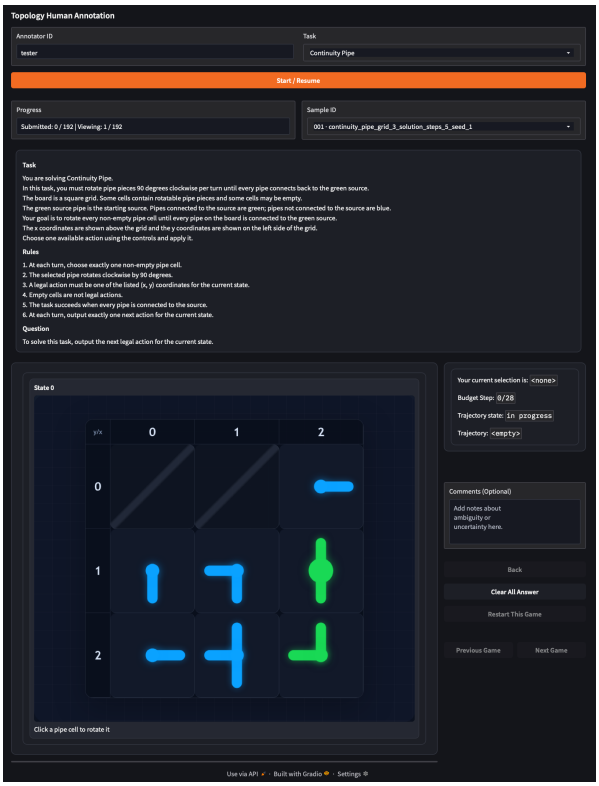


Figure 9 | Human annotation interface for planning tasks. Annotators interact with the same browser-based environment used by the model runner, and success is determined from the environment terminal state.

B.4.2. Inter-Annotator Agreement

To measure inter-annotator agreement (IAA), three annotators independently evaluated a shared subset of 200 examples across both reasoning and planning tasks under the same instructions as the models. After applying the same answer canonicalization used by the benchmark scorer, the IAA score is 0.89, indicating high consistency among annotators.

For reasoning tasks, agreement is computed by exact match after answer canonicalization. For planning tasks, agreement is computed under the task's deterministic evaluation protocol,

Model	Continuity			Separation		Order			Enclosure			Knots		Avg.	
	2D Maze	3D Maze	Pipe	Assembly	One Stroke	Bead	Origami Point	Swap	Sheep	Hole	Chat Noir	Knots	Untangle	Reason	Plan
GPT-6 Astra	99.33	86.67	99.33	74.67	95.33	52.00	83.33	100.00	59.33	66.67	96.67	88.67	18.00	76.33	81.87
GPT-5.6-Sol	60.00	18.67	22.00	34.67	18.67	8.00	18.67	80.67	55.33	52.67	70.67	30.00	4.00	34.75	39.20
GPT-5.5	5.33	13.33	0.00	34.67	0.00	0.00	9.33	39.33	39.33	54.67	25.33	36.67	9.33	24.17	14.80
GPT-5.4-mini	2.00	4.00	0.00	20.67	0.00	0.00	0.00	0.00	10.00	0.67	0.00	7.33	3.33	5.58	0.67
Gemini-3.1-Flash-Lite	0.67	2.67	0.00	25.33	0.00	0.00	1.33	0.00	26.67	3.33	0.67	18.00	0.67	9.75	0.27
Gemini-3.1-Pro	6.67	27.33	0.00	30.00	0.00	0.00	1.33	2.00	56.67	41.33	27.33	64.00	6.00	28.42	7.07
Claude-Haiku-5.5	0.67	24.67	0.00	29.33	0.00	0.67	3.33	0.00	56.00	48.67	44.00	38.67	2.67	25.25	9.33
Nemotron-Nano-12B-VL-v2	4.00	2.67	0.00	26.00	0.00	0.00	0.00	0.00	6.67	0.67	0.00	3.33	0.00	5.42	0.00
Qwen3.5-397B-A17B	2.00	8.00	0.00	20.67	0.00	0.00	0.00	0.00	0.00	3.33	1.33	5.33	0.67	4.92	0.40
Llama-4-Maverick-17B-128E	4.67	2.67	0.00	25.33	0.00	0.00	2.00	0.00	4.00	0.00	0.00	1.33	2.00	5.00	0.40
Ministral-3-14B-Instruct-2512	5.33	4.00	0.00	23.33	0.00	0.00	0.00	0.00	2.67	0.00	0.00	3.33	2.67	4.83	0.53
InternVL3.5-241B	2.67	6.67	0.00	17.33	0.00	0.00	0.00	0.00	8.67	0.67	0.00	7.33	0.00	5.42	0.00
Gemma-4-31B-IT	4.00	6.00	0.00	17.33	0.00	0.00	0.00	0.00	2.67	0.00	2.67	2.00	0.67	4.00	0.67
Human	98.33	95.33	100.00	83.33	100.00	93.00	96.00†	100.00	96.33	99.56	99.33	92.00	100.00	94.23	99.87

Table 25 | **Easy-level performance of humans and models (%).** Each task contributes its 150 Easy instances. Human accuracy comes from the human study of the previous release, whose instances form the Easy level, taking for each task the accuracy on the difficulty tier that the Easy instances are drawn from. Origami Point (†) was regenerated after the study and uses the previous release's overall human accuracy on this task. Shading marks the best, second, and third model score in each column, excluding Human.

so two annotations agree only when they reach the same evaluated outcome. Disagreements are not resolved by majority vote for the purpose of IAA calculation. Instead, they are reviewed separately to identify potential ambiguity, interface issues, or annotation errors.

B.5. Full Results by Difficulty

Table 1 reports each task on 300 instances. Tables 26 to 30 split every score into its two levels of 150 instances, grouped by topological category. Each score in Table 1 is the mean of its two columns.

Model	2D Maze		3D Maze		Pipe	
	Easy	Hard	Easy	Hard	Easy	Hard
Proprietary Models						
GPT-6 Astra	99.33	42.00	86.67	76.00	99.33	10.00
GPT-5.6-Sol	60.00	6.00	18.67	9.33	22.00	0.00
GPT-5.5	5.33	12.00	13.33	2.00	0.00	0.00
GPT-5.4-mini	2.00	13.33	4.00	1.33	0.00	0.00
Gemini-3.1-Flash-Lite	0.67	12.67	2.67	0.00	0.00	0.00
Gemini-3.1-Pro	6.67	14.67	27.33	0.67	0.00	0.00
Claude-Haiku-5.5	0.67	1.33	24.67	2.00	0.00	0.00
Hosted Open-Weight Models						
Nemotron-Nano-12B-VL-v2	4.00	4.00	2.67	0.00	0.00	0.00
Qwen3.5-397B-A17B	2.00	11.33	8.00	1.33	0.00	0.00
Llama-4-Maverick-17B-128E	4.67	11.33	2.67	4.67	0.00	0.00
Ministral-3-14B-Instruct-2512	5.33	4.67	4.00	0.00	0.00	0.00
InternVL3.5-241B	2.67	1.33	6.67	0.00	0.00	0.00
Gemma-4-31B-IT	4.00	0.67	6.00	0.67	0.00	0.00

Table 26 | **Continuity results by difficulty level.** Accuracy (%) for reasoning tasks and episode success rate (%) for planning tasks on the Easy and Hard levels, 150 instances each. Each Table 1 score is the mean of the two columns. Dark and light lavender mark the highest and second highest positive score per column.

Model	Assembly		One Stroke	
	Easy	Hard	Easy	Hard
Proprietary Models				
GPT-6 Astra	74.67	67.33	95.33	46.00
GPT-5.6-Sol	34.67	37.33	18.67	0.00
GPT-5.5	34.67	31.33	0.00	0.00
GPT-5.4-mini	20.67	24.67	0.00	0.00
Gemini-3.1-Flash-Lite	25.33	23.33	0.00	0.00
Gemini-3.1-Pro	30.00	24.00	0.00	0.00
Claude-Haiku-5.5	29.33	30.67	0.00	0.00
Hosted Open-Weight Models				
Nemotron-Nano-12B-VL-v2	26.00	17.33	0.00	0.00
Qwen3.5-397B-A17B	20.67	24.00	0.00	0.00
Llama-4-Maverick-17B-128E	25.33	22.67	0.00	0.00
Ministral-3-14B-Instruct-2512	23.33	24.67	0.00	0.00
InternVL3.5-241B	17.33	23.33	0.00	0.00
Gemma-4-31B-IT	17.33	8.67	0.00	0.00

Table 27 | **Separation results by difficulty level.** Same layout as Table 26.

Model	Bead		Origami Point		Swap	
	Easy	Hard	Easy	Hard	Easy	Hard
Proprietary Models						
GPT-6 Astra	52.00	54.67	83.33	55.33	100.00	2.67
GPT-5.6-Sol	8.00	29.33	18.67	0.67	80.67	0.00
GPT-5.5	0.00	14.67	9.33	11.33	39.33	0.00
GPT-5.4-mini	0.00	10.67	0.00	4.67	0.00	0.00
Gemini-3.1-Flash-Lite	0.00	12.67	1.33	4.00	0.00	0.00
Gemini-3.1-Pro	0.00	12.00	1.33	4.67	2.00	0.00
Claude-Haiku-5.5	0.67	21.33	3.33	5.33	0.00	0.00
Hosted Open-Weight Models						
Nemotron-Nano-12B-VL-v2	0.00	5.33	0.00	0.67	0.00	0.00
Qwen3.5-397B-A17B	0.00	9.33	0.00	2.67	0.00	0.00
Llama-4-Maverick-17B-128E	0.00	9.33	2.00	4.00	0.00	0.00
Ministral-3-14B-Instruct-2512	0.00	14.00	0.00	3.33	0.00	0.00
InternVL3.5-241B	0.00	6.00	0.00	7.33	0.00	0.00
Gemma-4-31B-IT	0.00	0.00	0.00	0.00	0.00	0.00

Table 28 | **Order results by difficulty level.** Same layout as Table 26.

Model	Sheep		Hole		Chat Noir	
	Easy	Hard	Easy	Hard	Easy	Hard
Proprietary Models						
GPT-6 Astra	59.33	11.33	66.67	0.00	96.67	6.67
GPT-5.6-Sol	55.33	7.33	52.67	0.00	70.67	0.00
GPT-5.5	39.33	8.00	54.67	0.00	25.33	0.00
GPT-5.4-mini	10.00	0.00	0.67	0.00	0.00	0.00
Gemini-3.1-Flash-Lite	26.67	2.00	3.33	0.00	0.67	0.00
Gemini-3.1-Pro	56.67	4.00	41.33	0.00	27.33	0.00
Claude-Haiku-5.5	56.00	28.67	48.67	2.00	44.00	0.00
Hosted Open-Weight Models						
Nemotron-Nano-12B-VL-v2	6.67	0.00	0.67	0.00	0.00	0.00
Qwen3.5-397B-A17B	0.00	0.67	3.33	0.00	1.33	0.00
Llama-4-Maverick-17B-128E	4.00	0.00	0.00	0.00	0.00	0.00
Ministral-3-14B-Instruct-2512	2.67	0.00	0.00	0.00	0.00	0.00
InternVL3.5-241B	8.67	0.00	0.67	0.00	0.00	0.00
Gemma-4-31B-IT	2.67	11.33	0.00	0.00	2.67	0.00

Table 29 | **Enclosure results by difficulty level.** Same layout as Table 26.

Model	Knots		Untangle	
	Easy	Hard	Easy	Hard
Proprietary Models				
GPT-6 Astra	88.67	35.33	18.00	23.33
GPT-5.6-Sol	30.00	20.67	4.00	2.00
GPT-5.5	36.67	24.67	9.33	6.00
GPT-5.4-mini	7.33	17.33	3.33	3.33
Gemini-3.1-Flash-Lite	18.00	18.00	0.67	5.33
Gemini-3.1-Pro	64.00	19.33	6.00	6.67
Claude-Haiku-5.5	38.67	24.67	2.67	7.33
Hosted Open-Weight Models				
Nemotron-Nano-12B-VL-v2	3.33	18.67	0.00	1.33
Qwen3.5-397B-A17B	5.33	16.67	0.67	0.00
Llama-4-Maverick-17B-128E	1.33	18.67	2.00	3.33
Ministral-3-14B-Instruct-2512	3.33	14.00	2.67	3.33
InternVL3.5-241B	7.33	22.00	0.00	0.00
Gemma-4-31B-IT	2.00	15.33	0.67	0.67

Table 30 | **Knots results by difficulty level.** Same layout as Table 26.

B.6. Generative Models as Topological World Models

Setup. We evaluate generative models as auxiliary world models rather than as direct answerers. At each step, a planner first writes an image or video prompt describing a plausible topological rollout from the current observation. The generator then produces an imagined future observation, and the same planner commits to an answer or action from the original observation plus the generated artifact. Image experiments use edit-mode generation anchored to the current PNG observation, with `gpt-image-2`, BAGEL, or ThinkMorph as the image backend. Video experiments use image-to-video rollouts with Wan2.2-I2V-A14B, Seedance-2.0-Mini, or Veo-3.1-Lite as the replay backend. Controls keep the planner, initial observation, task prompt, and action budget fixed across no-imagination and imagination variants.

Imagined-rollout protocol. For continuity tasks, the imagined artifact is asked to show a path opening, pipe flow, or maze traversal after the candidate manipulation. For separation, the rollout visualizes whether a stroke or object remains one connected piece. For order, it shows the post-fold or post-swap ordering. For enclosure, it shows escape paths, boundary closure, or hole visibility after a viewpoint or state change. For knots, it asks the generator to simulate an endpoint move or strand deformation while preserving over/under crossings. The planner may call the image generator at every step, capped at ten calls per episode. The end-to-end video setting uses one video per episode. Per-task success rates for the interleaved-image and video variants appear in Table 4. Generated conditions and the GPT-5.6-Luna planning baselines use 105 episodes per task, 35 from each of three generator difficulty settings, so they include instances outside the Easy and Hard benchmark levels. The three probing analyses draw on different samples. The error annotation in Figure 7 uses a separate set of Wan2.2-I2V-A14B rollouts, with 60 videos targeted for One Stroke and 60 for Untangle. One Untangle video was unavailable, leaving 60 scored One Stroke videos and 59 scored Untangle videos (119 total). The CV diagnostic audit in Table 5 uses a third sample of 105 videos for each pairing of a task with one of the six generators, 1,890 videos in total.

Failure modes. Imagined rollouts often look locally plausible while violating the exact topological invariant that the task tests. Rather than a ranked taxonomy of visual subtypes, our annotations track three diagnostics, namely dynamics, invariance, and cross-frame consistency. Across the 119 annotated Wan2.2-I2V-A14B video rollouts, 116 contain a dynamics error, 93 an invariance error, and 90 a consistency error. Qualitative manifestations include identity drift,

silent crossings, hallucinated openings, and visually smooth but impossible transitions. We therefore treat generated rollouts as an auxiliary diagnostic rather than a replacement for oracle state transitions.

B.6.1. Computer-Vision Video Diagnostics

Verifier and aggregation. We apply task-specific computer-vision verifiers to One Stroke, Pipe, and Untangle rollouts from LTX-2.3, LTX-2.5, Veo-3.1-Lite, Wan2.2-I2V-A14B, MiniMax-H3, and Seedance-2.5. Each task and generator cell contains 105 videos, for 1,890 videos in total, and an episode without a generated video counts as failing every check. Static checks test whether each selected parsed frame preserves the task state. Dynamic checks test whether consecutive frames form a valid transition. The reported CV audit uses the default `sample_41` mode, which samples 41 evenly spaced frames from each video, including the first and last frames. A video passes a check only when every applicable frame or transition passes. Unscorable videos and videos with no applicable instance of a check count as failures, so every rate in Table 5 (Figure 10) is a count out of 105.

Terminal-state parser status and scorability. Parser labels describe detector completeness, not whether the task itself is solved. They are task-specific and are converted to the common binary *terminal state scorable* gate only after parsing. For the three tasks in Table 5, `ok` and `degraded` are readable, whereas `failed` and `grid_not_found` are unreadable. Thus, a degraded frame can remain scorable when it contains enough state for the task metrics, while a failed frame is counted as a failure in outcome, static, dynamic, and overall video-level aggregates. Table 31 states the exact triggers and observed terminal-frame counts.

Operational metric definitions. Tables 32 and 33 define every task-specific metric in Table 5. Static metrics are evaluated on four evenly spaced static frames on which they apply, while terminal outcome metrics use the final sampled frame. Dynamic metrics compare adjacent sampled frames. Only rows explicitly marked `not_applicable` are omitted from a check's denominator. Low-confidence rows remain applicable and their pass/fail decision is retained. At video level, *static validity* and *dynamic validity* are strict logical conjunctions over their respective applicable rows. *Final-state task success* applies the task-specific terminal oracle, and *process-valid task success* requires terminal scorability, task success, static validity, and dynamic validity simultaneously.

Qualitative examples. Figure 11 connects sampled generated-video frames to the task-specific CV overlays used for dynamic verification. Its three columns cover Pipe, Untangle, and One Stroke. The overlays illustrate the cross-frame correspondences defined in Tables 32 and 33.

Human validation. Manual review audits the CV evaluator on LTX-2.3 and Wan2.2-I2V-A14B videos. The evaluator validation set is a balanced sample of 180 videos, with 30 videos in every task and generator cell. For each video, a single reviewer inspects the video and detection overlays, then records whether the visible task state is scorable, whether the CV parse matches it exactly or at least partially, and whether the resulting automatic metric decision agrees with manual review.

Panel A of Table 34 shows that every sampled parse is usable (partial or exact) and that all 180 video-level metric decisions agree with the manual audit. These results support using the CV verifier to compute the diagnostic pass rates in Table 5. Exact state reconstruction is less reliable, especially on Untangle. Panel B provides an error analysis of a separate component,

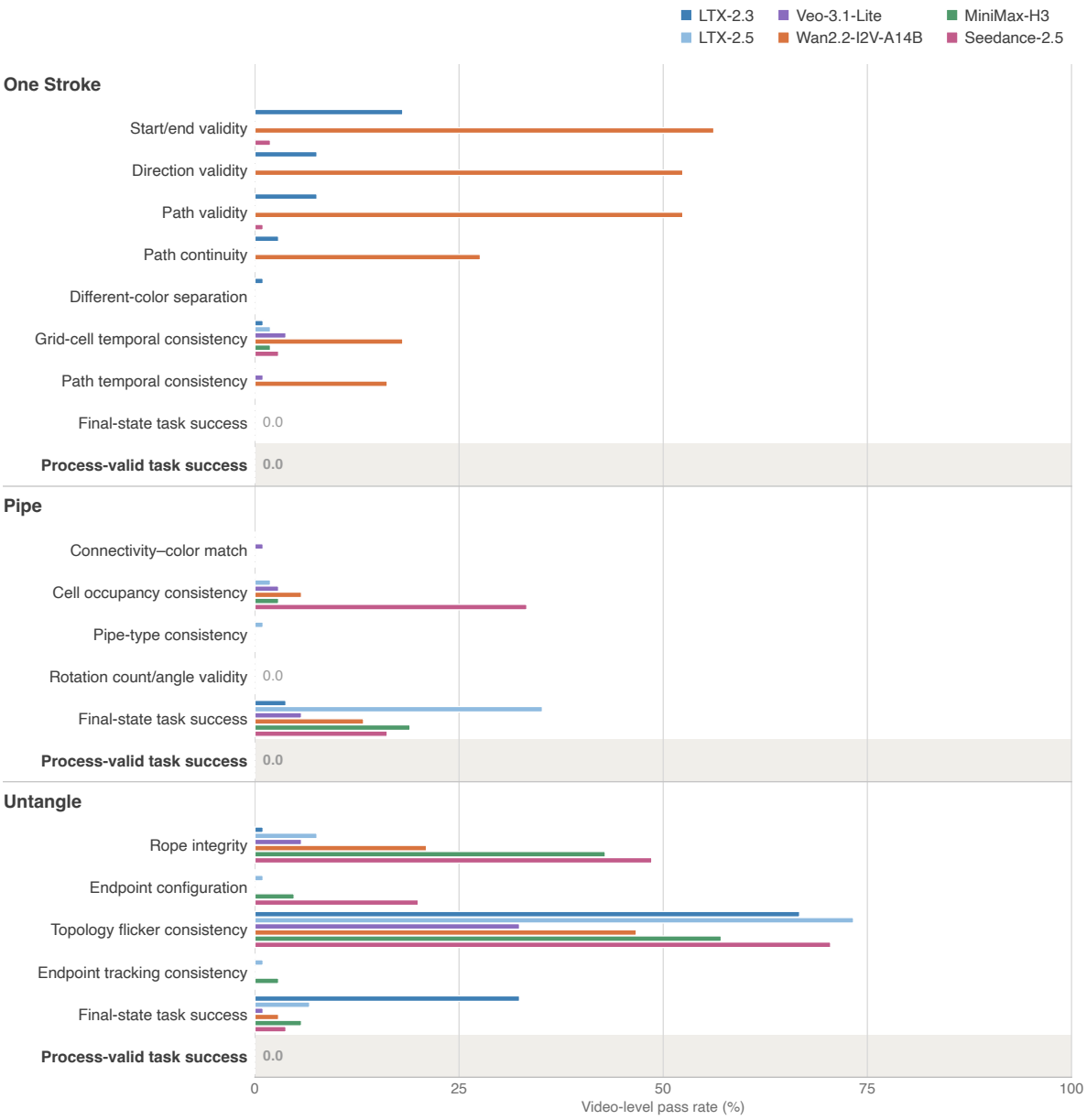


Figure 10 | **Per-check video-level pass rates from Table 5, ordered loosest to strictest.** The shaded *process-valid task success* row combines terminal-state success with every applicable static and dynamic check. The two per-group aggregates are omitted here for readability.

the terminal-state scorability gate. Its 77.8% accuracy is dominated by its 98.6% recall on the more frequent human-scorable class, whereas recall on human-unscorable videos is only 9.5%, yielding 54.0% balanced accuracy. In particular, 38 of the 40 errors arise when the parser accepts a terminal state that the reviewer considers unscorable. These errors show that the coverage gate frequently accepts terminal states that a human cannot score.

B.7. Camera and Viewpoint Controls

Controlled camera factors. The benchmark deliberately separates topological difficulty from viewpoint difficulty. Static generators store the scene seed and render camera metadata, so we can rerender the same underlying topology under a canonical top view, oblique view, wider field of view, tighter crop, or distractor-heavy view. For planning environments, the browser viewport is fixed and scene-only screenshots are captured at each step, which prevents accidental

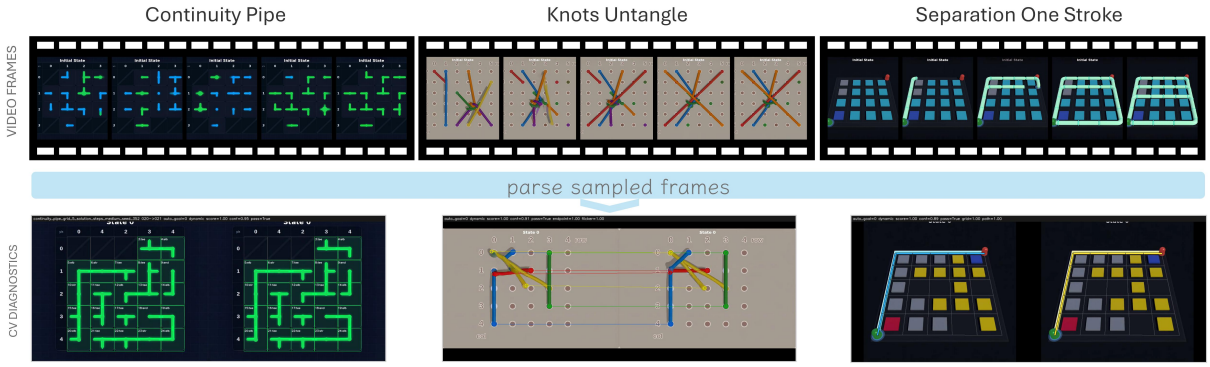


Figure 11 | **Video sampling and computer-vision transition diagnostics.** For Pipe, Untangle, and One Stroke, the top filmstrips show five sampled frames from representative generated rollouts, and the bottom panels show task-specific overlays for adjacent-frame checks.

Environment	Raw terminal status	Operational trigger	Scorable	LTX-2.3	Wan2.2-I2V-A14B
One Stroke	`ok`	At least one colored cell and both start and end markers are detected.	Yes	87	102
	`degraded`	Colored cells are detected, but either the start or end marker is missing.	Yes	11	3
	`failed`	No colored cells are detected in the terminal frame.	No	7	0
Pipe	`ok`	At least one active pipe cell is detected on the fitted board grid.	Yes	97	105
	`grid_not_found`	No active pipe cell is detected, so no pipe state can be constructed.	No	8	0
Untangle	`ok`	The hole lattice is fitted and every detected rope has exactly two connected, unambiguous endpoints resolved to lattice holes.	Yes	2	3
	`degraded`	The hole lattice and at least one expected rope are detected, but endpoint evidence is incomplete, extra, isolated, ambiguous, or not fully resolved.	Yes	96	102
	`failed`	The hole lattice cannot be fitted, or no expected rope mask survives the minimum-area detector threshold.	No	7	0

Table 31 | **Task-specific terminal parser statuses and their mapping to terminal-state scorability.** Counts are terminal frames among the 105 videos in each cell of environment and generator. Status is a detector-completeness label, so `failed` does not mean that a readable puzzle was merely unsolved.

resolution changes from becoming a hidden model-specific advantage. Table 35 summarizes the diagnostic role of each controlled camera factor.

B.8. Shortcut and Input Representation Controls

Protocol. We test whether performance depends on scene structure that determines the answer or on cues that remain after this structure is removed. Full input contains the rendered observation and the complete task prompt. Text-only removes the rendered observation while retaining the question for each instance. Answer-prior also removes the information specific to each question and therefore measures the signal carried by the answer distribution. Appearance-only retains visual style cues without the original scene structure. Within each row, all conditions use the same instances and decoding configuration. Table 36 reports these controls for the task and model pairs on which all three were evaluated. We additionally probe the input representation with a symbolic condition that replaces the rendering with a symbolic description of the same state. This condition covers every task and model with a completed matched symbolic run, and Table 37 compares it against the full input.

Type	Metric	Operational pass condition
Static	Different-color separation	On the final frame, the detected path partitions the board so that no connected region contains cells of more than one color. This isolates the separation constraint from the full task oracle.
Static	Direction validity	The detected arrowhead lies at the advancing path endpoint, points away from the start, and points toward the end. An orientable path without reliable arrowhead evidence is retained as a low-confidence pass.
Static	Path continuity	The stroke mask and skeleton each have one significant component, the snapped legal-edge graph is connected, fitted segment gaps remain below the grid-scaled tolerance, and combined confidence is at least 0.60.
Static	Path validity	The connected stroke snaps to one or more legal up/down/left/right board edges with snap confidence at least 0.55 and sufficient evidence that it does not cut through colored cells. A straight diagonal fails.
Static	Start/end validity	An intermediate path remains anchored at the start. On the final frame, one connected stroke must touch both the start and end markers.
Dynamic	Grid-cell temporal consistency	Adjacent frames preserve the number and color of cells and match each cell to a same-color position within the grid-scaled spatial tolerance.
Dynamic	Path temporal consistency	The current path retains the preceding legal-edge set except for edges removed by legal backtracking, adds no disconnected branch, and keeps sufficient adjacent-frame overlap. When edge snapping is unavailable, path area retention must be at least 0.75 and pixel overlap at least 0.45. Either branch is also gated by per-frame path continuity.
Outcome	Final-state task success	The final frame contains one connected start-to-end path on legal grid edges and passes the environment's exact region-constraint solver.

Table 32 | **Operational definitions of the One Stroke CV metrics.** Early frames before any path is drawn are marked not applicable for path-based checks rather than counted as passes.

Environment	Type	Metric	Operational pass condition
Pipe	Static	Connectivity and color match	For every detected pipe cell, green means reachable from the source under the parsed arm connections and blue means unreachable. The source must be green.
	Dynamic	Cell occupancy consistency	The set of occupied pipe-grid cells is identical in adjacent frames, and no pipe cell appears or disappears.
	Dynamic	Pipe-type consistency	Every occupied cell shared by adjacent frames keeps the same rotation-invariant pipe shape. Orientation may change, but endpoint, straight, elbow, and tee identities may not.
	Dynamic	Rotation count/angle validity	At most one pipe changes orientation in an adjacent transition, and it moves to a registered legal orientation no more than one quarter-turn away. The pixel-angle fallback likewise requires an absolute rotation below 90°.
	Outcome	Final-state task success	Every ground-truth active pipe cell is connected to the source in the parsed terminal configuration.
Untangle	Static	Endpoint configuration	Every detected rope has nonzero mask area, exactly two selected and resolved endpoints, no extra endpoint candidate, and no isolated endpoint cap.
	Static	Rope integrity	Each rope mask has sufficient area and is not visibly fragmented, meaning that its largest component covers at least 45% of rope area or it has at most two large components (each at least 12% of rope area). At least 90% of ropes must pass.
	Dynamic	Endpoint tracking consistency	Each rope persists across adjacent frames and its two endpoints can be matched within half the nearest hole spacing. Missing ropes, unreadable endpoint counts, or larger jumps fail, while ambiguity is retained as low confidence rather than an automatic hard failure.
	Dynamic	Topology flicker consistency	Crossing count may not change while endpoint-to-hole assignments are fixed or endpoints are nearly stationary, and endpoint-to-hole assignments may not change while endpoints are nearly stationary. A topology change accompanied by visible endpoint motion is not applicable to this flicker check.
	Outcome	Final-state task success	The terminal crossing count is readable and equals zero.

Table 33 | **Operational definitions of the Pipe and Untangle CV metrics.** Dynamic checks are evaluated on every adjacent pair of sampled frames for which the check is applicable.

Prompt and answer priors are concentrated in Sheep. Of the two reasoning tasks probed, Sheep carries the strongest prior. Gemini-3.1-Flash-Lite scores 61.0 with the full Sheep input

Panel A. Parse quality and CV metric agreement by environment and generator

Environment	Generator	*N*	Human-scorable view	Exact CV parse	Usable CV parse	CV metric agreement
Pipe	LTX-2.3	30	76.7%	96.7%	100.0%	100.0%
Pipe	Wan2.2-I2V-A14B	30	100.0%	100.0%	100.0%	100.0%
One Stroke	LTX-2.3	30	76.7%	43.3%	100.0%	100.0%
One Stroke	Wan2.2-I2V-A14B	30	100.0%	100.0%	100.0%	100.0%
Untangle	LTX-2.3	30	6.7%	3.3%	100.0%	100.0%
Untangle	Wan2.2-I2V-A14B	30	100.0%	23.3%	100.0%	100.0%
Overall	**Both**	**180**	**76.7%**	**61.1%**	**100.0%**	**100.0%**

Panel B. Human versus automatic terminal-state scorability

	Human scorable	Human unscorable
Parser scorable	75.6%	21.1%
Parser unscorable	1.1%	2.2%
Derived gate metrics		
Accuracy	77.8% (140/180 correctly classified videos)	
Scorable recall	98.6% (136/138 human-scorable videos accepted)	
Unscorable recall	9.5% (4/42 human-unscorable videos rejected)	
Balanced accuracy	54.0% (mean of scorable and unscorable recall)	

Table 34 | **Human audit of the computer-vision verifier and terminal-state scorability gate.** A balanced set of 180 videos spans three environments, the LTX-2.3 and Wan2.2-I2V-A14B generators, and three generator difficulty settings (10 per cell). **Panel A** reports parse quality and agreement between automatic video-level metric decisions and manual review. A usable parse may be partial but retains enough evidence for the diagnostic decision. **Panel B** compares automatic terminal parser acceptance with the human judgment of whether the visible state is scorable, and each cell is a percentage of all 180 videos.

Camera factor	Diagnostic role
Top vs. oblique	Tests whether models preserve connectivity, enclosure, and knot over/under relations when Euclidean projection changes.
Field of view	Separates global topology errors from missed off-screen or cropped boundary segments.
Zoom/crop	Identifies failures caused by small gaps, thin walls, or subtle through-holes rather than by task semantics.
Multi-view pair	Checks whether an answer is stable when one view exposes depth or occlusion cues missing from another view.

Table 35 | Controlled camera factors in scene generation. The same seed can be rerendered under controlled view changes while keeping the topological ground truth fixed.

and retains 56.0 under both text-only and answer-prior input. InternVL3.5-241B shows the same pattern. Its full-input score is 28.0 and its answer-prior score is 28.0. These results show that the answer distribution explains most of the measured Sheep performance on this matched subset. The 2D Maze results differ. Removing the scene lowers Gemini-3.1-Flash-Lite from 21.0 to 8.0, so its full-input score depends on visual evidence from each instance. The appearance-only control, however, reaches 36.0 on the same matched subset, which is above the full input, so style cues alone carry nontrivial answer signal there, and we read these controls as a diagnostic rather than an exact decomposition of the score.

Explicit structure helps selected planning tasks. Symbolic input raises Gemini-3.1-Flash-Lite by approximately 33.3 percentage points on Swap (13.33 to 46.67) and by 15.0 points on Chat Noir (25.00 to 40.00), while InternVL3.5-241B gains on neither task. The effect therefore depends on both the environment and the model rather than following from symbolic input alone.

Explicit structure does not remove the general planning bottleneck. Pipe and One Stroke remain at zero or near zero under every tested input condition. Their failures persist after the

Task	Model	N	Full input	Shortcut controls		
				Text only	Answer prior	Appearance only
Reasoning tasks						
2D Maze	Gemini-3.1-Flash-Lite	100	21.00	8.00	8.00	36.00
	InternVL3.5-241B	100	10.00	14.00	8.00	6.00
Sheep	Gemini-3.1-Flash-Lite	100	61.00	56.00	56.00	43.00
	InternVL3.5-241B	100	28.00	23.00	28.00	18.00
Planning tasks						
Pipe	Gemini-3.1-Flash-Lite	60	0.00	0.00	0.00	0.00
	InternVL3.5-241B	60	1.67	0.00	0.00	0.00
One Stroke	Gemini-3.1-Flash-Lite	60	0.00	0.00	0.00	0.00
	InternVL3.5-241B	60	0.00	0.00	0.00	0.00

Table 36 | Shortcut controls on matched evaluation subsets, where full input denotes the rendered scene with the complete prompt. Values are accuracy (%) for reasoning tasks and episode success rate (%) for planning tasks, and each row compares conditions on the same instances. Rates use the same approximate convention as Table 37, with integer percentages for reasoning and the nearest multiple of 100/60 percentage points for planning, displayed to two decimals.

Task	Model	N	Full input	Symbolic input
Reasoning tasks				
2D Maze	Gemini-3.1-Flash-Lite	100	21.00	27.00
	InternVL3.5-241B	100	10.00	7.00
Sheep	Gemini-3.1-Flash-Lite	100	61.00	9.00
	InternVL3.5-241B	100	28.00	4.00
Planning tasks				
Pipe	Gemini-3.1-Flash-Lite	60	0.00	0.00
	InternVL3.5-241B	60	1.67	0.00
One Stroke	Gemini-3.1-Flash-Lite	60	0.00	0.00
	InternVL3.5-241B	60	0.00	0.00
Swap	Gemini-3.1-Flash-Lite	60	13.33	46.67
	InternVL3.5-241B	60	13.33	8.33
Chat Noir	Gemini-3.1-Flash-Lite	60	25.00	40.00
	InternVL3.5-241B	60	6.67	3.33

Table 37 | Symbolic input probe on matched evaluation subsets, where the rendered scene is replaced by a symbolic description of the same state and the rest of the prompt is fixed. Values are accuracy (%) for reasoning tasks and episode success rate (%) for planning tasks, and each row compares the two conditions on the same instances. Rates are approximate. Reasoning values are truncated to integer percentages, and planning values are rounded to the nearest multiple of 100/60 percentage points before formatting to two decimals.

rendered scene is replaced by an explicit state description. Input representation alone therefore does not account for these failures. The remaining errors are consistent with limits in state transition reasoning and action selection. Symbolic input also lowers Sheep performance for both models. It is therefore a representation probe rather than an oracle upper bound.

B.9. Do Foundation Models Have Topological Biases?

Percentages in this subsection are shares of the labeled failures of the two representative models described in Appendix C, pooled over both models (70 failures per environment) and labeled with the seven-category taxonomy of Appendix C.2.

Connectivity-by-default bias. In the two annotated models, a connection is frequently inferred from visual proximity. This appears in 2D/3D Maze errors where adjacent rooms, bars, or cells are treated as connected despite a blocking wall, and in planning runs where illegal transitions

attempt to move through occupied, blocked, or out-of-bounds cells. Across the two Maze reasoning environments, 52.9% of labeled failures are perception-grounding errors. Another 40.7% are instruction-following failures, most of them InternVL3.5-241B responses that are cut off before a committed answer. Pipe planning shows the same bias in an interactive setting, where 90.0% of labeled failures are perception-grounding errors in which end pieces are read as straight pipes or the source is given openings it does not have, so legal rotations are spent on a misread network.

Hole undercounting. Hole exposes a consistent tendency to undercount through-holes, especially in Hard-level scenes with occlusion, multiple cavities, or ambiguous shading. Under the causal-priority taxonomy, these failures are assigned primarily to perception grounding (64.3%), followed by task understanding (20.0%), mostly counting an opening spanned by a bar as a single hole, and instruction following (15.7%). None of the labeled Hole errors receive the later topological-invariance label. The dominant mistake is therefore grounding a visible depression, tunnel, or opening as the wrong kind of structure, before an invariance judgment can be credited.

Knot-structure collapse. Knots is the only reasoning environment whose failures include explicit topological-invariance labels, but these account for just 11.4% of labeled Knots failures. Perception grounding accounts for 52.9%, as models miss crossings or segment one rope into several rings, and instruction following for another 32.9%, often a final answer that contradicts the model's own reasoning. Untangle has a different profile, with instruction following accounting for 58.6% of labeled failures, including every InternVL3.5-241B failure, and action planning for 21.4%, so the interactive deficit cannot be summarized as the same static-recognition error. Table 38 summarizes these environment-specific error signatures.

Error signature	Where it appears	Current evidence
Connectivity grounding	2D Maze, 3D Maze, Pipe	Perception grounding accounts for 52.9% of Maze and 90.0% of Pipe failures.
Hole-type grounding	Hole	Perception grounding accounts for 64.3%, and topological invariance accounts for 0.0%.
Knot-structure classification	Knots	Perception grounding accounts for 52.9%, and topological invariance for 11.4%.
Environment-specific planning	Swap, One Stroke, Chat Noir, Untangle	The dominant label shifts by environment, to task understanding (67.1%), dynamic violations (38.6%), action planning (58.6%), and instruction following (58.6%), respectively.

Table 38 | Environment-specific error signatures in the labeled failures. Percentages are shares of the labeled failures of both models in each environment, sampled as described in Appendix C.

C. Error Analysis

C.1. Annotation Framework

Motivation. MINDTOPO evaluates whether foundation models can ground visual scenes, identify topological relations, predict state changes under manipulation, and plan valid actions. A single split into perception and reasoning is therefore too coarse, because an incorrect answer may come from a bad output format, a mistaken task objective, a visual grounding failure, a wrong belief about topological invariance, an incorrect final-state prediction, an impossible transition, or a poor action plan. We use the following seven primary categories to separate these failure sources.

Top-level categories. Table 39 summarizes the seven categories. Full definitions, subcategories, and labeled examples for each category appear in Appendix C.2.

Category	Definition
0. Instruction Following (Appendix C.2.1)	The response is unparseable or violates the protocol.
1. Task Understanding (Appendix C.2.2)	The model solves the wrong task or applies the wrong rule.
2. Perception Grounding (Appendix C.2.3)	The response relies on a wrong visual fact.
3. Topological Invariance (Appendix C.2.4)	The model misjudges whether a relation survives an allowed transformation.
4. Feature State Prediction (Appendix C.2.5)	The model predicts a wrong post-action state of a relevant feature.
5. Dynamic (Appendix C.2.6)	The model assumes an impossible transition during motion.
6. Action Planning (Appendix C.2.7)	The model selects a poor legal action or multi-step plan.

Table 39 | The seven primary error categories.

C.2. Error Category Definitions

C.2.1. 0. Instruction Following Error

The model fails to provide a response that can be evaluated under the prompt's requested protocol. This category captures answer-format and response-compliance failures rather than visual or topological capability failures. We assign category 0 after applying the benchmark's fallback parser. If the fallback parser can recover a valid answer, the prediction is analyzed under categories 1–6 instead.

0a. Invalid Answer Format. The output cannot be parsed into the required type or schema. Typical cases include prose returned when the prompt asks for `{"answer": 3}`, a missing JSON key, a cell index returned as a sentence, and a list returned when a scalar is required.

0b. Multiple, Hedged, or Missing Final Answer. The task expects one answer but the model gives several candidates, hedges between options, or never states a final answer. Typical cases include returning "A or C" for a multiple-choice task, listing two candidate moves, and explaining the image without an answer.

0c. Prompt-Protocol Violation. The model ignores explicit output constraints even though it appears to understand the broad task. Typical cases include emitting chain-of-thought when only the final answer is requested, using a free-form action when the prompt requires a fixed enum, and adding extra fields that break the evaluator.

0d. CoT/Output Internal Inconsistency. The chain-of-thought reaches one conclusion but the structured answer field reports a different value, so the parser commits a verdict that contradicts the model's own stated reasoning. This is distinct from 0a, 0b, and 0c because the response is well-formed and a single answer is emitted. The failure lies purely in the link between rationale and final field. Typical cases include a fence-repair trace concluding "there are zero gaps; done" followed by an answer field of 1, a hole-count walk-through that enumerates three holes followed by an answer field of 4, and a knot-classification thought block stating "this is an unknot" followed by the knot label B.

0e. Output-Budget Truncation. The model devotes its output budget to deliberation and is cut off before any structured answer is produced. We assign this label only when the fallback parser also fails to recover an answer from the available text. This differs from 0c (intentional protocol violation), because the model is trying to comply but never reaches the answer. Typical cases include a long endpoint enumeration on an Untangle step that hits the response cap mid-sentence and a multi-hundred-word deliberation on a separation question that ends without a final letter.

C.2.2. 1. Task Understanding Error

The response is syntactically usable, but the model has misunderstood the task objective, the permitted operation, or the relevant rule set. This category differs from category 0, because the answer may be parseable but is generated for the wrong problem.

1a. Wrong Objective. The model optimizes or answers the wrong target. Typical cases include counting objects when the task asks whether an object is enclosed, reporting the current state when the prompt asks for the post-action state, and selecting the visually closest option when the task requires topological equivalence.

1b. Wrong Rule Semantics. The model applies an incorrect rule for the environment. Typical cases include treating diagonal grid contact as connectivity when only four-neighbor connectivity is allowed, assuming a pipe piece can connect through a closed side, and misunderstanding whether a rope can be lifted over another rope in the specified setting.

1c. Wrong Entity or Reference Binding. The model answers for a different referenced object or option than the one requested by the prompt. Typical cases include comparing option B to the target when the prompt asks about option C, moving the red rope when the instruction refers to the blue rope, and using the goal state as the current state in Swap.

1d. Multi-Turn Task Drift. Specific to interactive tasks. The model identifies the task correctly on the first turn, but in subsequent turns its reasoning context drifts to a different problem framing, causing all later decisions to optimize the wrong objective. It differs from 1a and 1b because the failure is not a one-shot misreading of the prompt but a loss of task identity across turns despite the original instructions still being supplied. Typical cases include a One Stroke planner recognizing the rules at step 0 and then describing the same board as a robotic-arm pick-and-place scene from step 1 onward, a cat-and-mouse encloser switching to hypothesizing prime-number patterns in the cell labels by the second move, and a Pipe puzzle reframed as "Unblock Me" or as a sliding-block puzzle once intermediate states are shown.

C.2.3. 2. Perception Grounding Error

The model fails at the visual-grounding stage. The relevant task is understood, but the response relies on a wrong visual fact.

2a. Object, Color, or Attribute Misidentification. A visible element is detected but its identity or attribute is wrong. Typical cases include confusing the blue and red ropes, misreading a through-hole as filled, treating a blocked cell as free, and mistaking an open pipe side for a closed side.

2b. Location or Relative-Position Grounding Error. The model misreads where elements are or how they are positioned relative to each other. Typical cases include reversing the relative position of A and B, assigning an object to the wrong grid cell, missing that one endpoint lies inside a loop, and confusing left/right or above/below relations in the rendered scene.

2c. Missed or Hallucinated Visual Element. A relevant element is absent from or added to the model's internal scene representation, including fine-grained material or depth cues that distinguish a real opening from a surface marking. Typical cases include missing a hole, sheep, rope crossing, wall segment, or pipe piece. Others hallucinate an extra obstacle or connection, mistake a printed symbol or shaded depression for a through-hole, or misread interior shading as a solid floor.

2d. Multi-Image Correspondence Error. The model parses individual images but fails to align corresponding objects, views, or states. Typical cases include mismatching the current and goal states in Swap, aligning the wrong point across 3D Maze views, and confusing a target object with a candidate option in separation tasks.

C.2.4. 3. Topological Invariance Understanding Error

The model grounds the scene, but misjudges whether the relevant topological relation is preserved or changed. This category targets topological concepts such as connectivity, enclosure, separation, overlap, crossing, linking, and inside/outside relations under transformations. A common failure is to rely on visible coordinates or shape similarity while missing that topology has changed, or to infer a topological change from a purely geometric deformation.

3a. False Invariance. The model says a topological relation is unchanged when it has changed. A typical case concludes that two configurations are equivalent because endpoints or object positions look unchanged, even though a rope crossing, overlap, enclosure, or connectivity relation has changed.

3b. False Change. The model says a topological relation changes under a transformation that preserves it. A typical case treats translation, rotation, stretching, or a viewpoint change as breaking connectivity, enclosure, or linkage when the topological relation remains the same.

3c. Wrong Topological Relation. The model reasons about the correct objects but assigns the wrong topological relation. Typical cases include linked vs. unlinked ropes, connected vs. separated pipe components, inside vs. outside a loop, and trapped vs. reachable in a maze-like enclosure.

C.2.5. 4. Feature State Prediction Error

The model understands the instruction and initial scene, but predicts the wrong final state of a feature after a specified manipulation. This category concerns the resulting state, not whether the intermediate motion was physically possible. For example, after the prompt describes lifting a blue rope and placing it in the upper-left region, the model may predict that the blue rope no longer overlaps the red rope, even though the ground-truth final state still contains an overlap.

4a. Wrong Post-Action Topological Feature. The predicted final connectivity, overlap, crossing, enclosure, or separation state is wrong. Typical cases include predicting that a moved rope no longer overlaps another rope when it still does, predicting that a rotated pipe network is connected when one joint remains disconnected, and predicting that a blocked cell traps the cat when an escape route remains.

4b. Wrong Post-Action Visual or Discrete State. The model applies the action to the wrong state variable. Typical cases include a wrong post-rotation orientation of a pipe cell, a wrong final slot after a swap, a wrong cell status after placing a block, and a wrong endpoint location after a move.

4c. Incomplete State Update. The model updates one feature but leaves another dependent feature in the old state. Typical cases include moving a rope endpoint but not updating crossings, rotating a pipe cell but not updating which cells become source-connected, and blocking a cell but not updating reachability.

C.2.6. 5. Dynamic Error

The model assumes an impossible intermediate transition during motion. Unlike category 4, which concerns the final state after an action, category 5 concerns whether the path from the initial state to the final state violates the environment's dynamics or physical constraints.

5a. Barrier or Collision Violation. The model moves through an obstacle, wall, boundary, or occupied cell. Typical cases include planning a maze step through a wall, sliding a block through another block, routing a path through a forbidden cell, and moving the cat through a blocked cell.

5b. Topological Barrier Violation. The model permits a transition that would require crossing, cutting, or teleporting through a topological constraint that the task forbids. Typical cases include passing a rope through another rope without an allowed lift, untangling a knot by crossing strands through each other, and separating linked components through an impossible motion.

5c. Unsupported Continuous Motion. The model predicts a valid-looking final arrangement but provides or assumes no legal continuous path to reach it. Typical cases include moving an object from one enclosed region to another without crossing a boundary, rotating a component through a collision, and placing a feature behind an occluding barrier as if it could pass through it.

C.2.7. 6. Action Planning Error

Specific to interactive tasks. The model follows the output protocol, understands the task, grounds the relevant scene facts, and does not assume an impossible transition, but still chooses a poor legal action or action sequence.

6a. Legal but Irrelevant Action. The action is allowed but does not advance the objective. Typical cases include rotating a non-critical pipe piece, blocking a cell that is not on any escape route, moving a puzzle tile that leaves the state equally far from the goal, and adjusting a rope segment that does not reduce crossings.

6b. Short-Horizon or Trap-Inducing Plan. The action appears locally reasonable but causes later failure. Typical cases include blocking a cell that lets the cat escape elsewhere, forming a One Stroke path that cuts off an unvisited region, making a swap that increases the minimum remaining distance, and untangling one crossing while creating a worse downstream crossing.

6c. State-Tracking Planning Error. The model chooses a later action based on a stale but otherwise valid state estimate. Typical cases include forgetting a previously rotated pipe piece, treating an already blocked cell as free, planning from the initial rather than current puzzle state, and repeating an action that has already been applied.

C.3. Cross-Model and Cross-Task Distributions

Sampling protocol. For each (model, task) pair we collected every incorrect prediction of the model on the 300 benchmark instances behind its cell in Table 1, 150 on the Easy level and 150 on the Hard level, using the same runs and the same scoring as that table. We then drew 35 of these failures uniformly at random with a fixed seed and assigned each sampled prediction one primary category. Labels follow the causal priority order instruction following, task understanding, perception grounding, dynamic violation, feature-state prediction, topological invariance, and action planning, so a failure receives the earliest category that explains it. Every pair has at least 175 failures, so each pair contributes exactly 35 labeled failures. This yields 560 labeled reasoning failures and 350 labeled planning failures, of which 415 lie on the Easy level and 495 on the Hard level. The reasoning split contains the 8 reasoning tasks (2D Maze, 3D Maze, Hole, Sheep, Knots, Bead, Origami Point, and Assembly), and the planning split contains the 5 planning tasks (Pipe, Chat Noir, Untangle, Swap, and One Stroke).

Aggregation. Every distribution in this appendix and in Section 3.4 is a share of the labeled failures. A distribution for a group of pairs, such as one model on one split or one environment across both models, pools the labeled failures of those pairs, so each pair enters with the same weight of 35 failures. Figures 12 and 13 provide the environment-level and model-level breakdowns, and Tables 40 and 41 give the labeled-sample counts for each model and for both models combined.

Per-environment breakdown. Figure 12 keeps the reasoning and planning tasks within a topological property distinct, which prevents perception, invariance, state-prediction, dynamic, and planning failures from being collapsed into one aggregate error rate.

Comparison with human annotators. The same category definitions can be applied to human responses. This lets us distinguish human-like task or perception mistakes from model-specific failures such as invalid output formatting, brittle topological invariance judgments, or impossible dynamic transitions.

D. Extended Related Work Discussion

This appendix expands the condensed Related Work in Section 4 into the longer, per-work discussion that did not fit in the main body. Table 42 compares MINDTOPO with related benchmarks.

Topological Cognition. Cognitive development positions topology as a primitive layer of spatial cognition that precedes projective and Euclidean reasoning. Piaget and Inhelder [3] introduced the original five-class taxonomy of proximity, separation, order, enclosure, and continuity, with subsequent mathematical and constructionist extensions by Beth and Piaget [68]

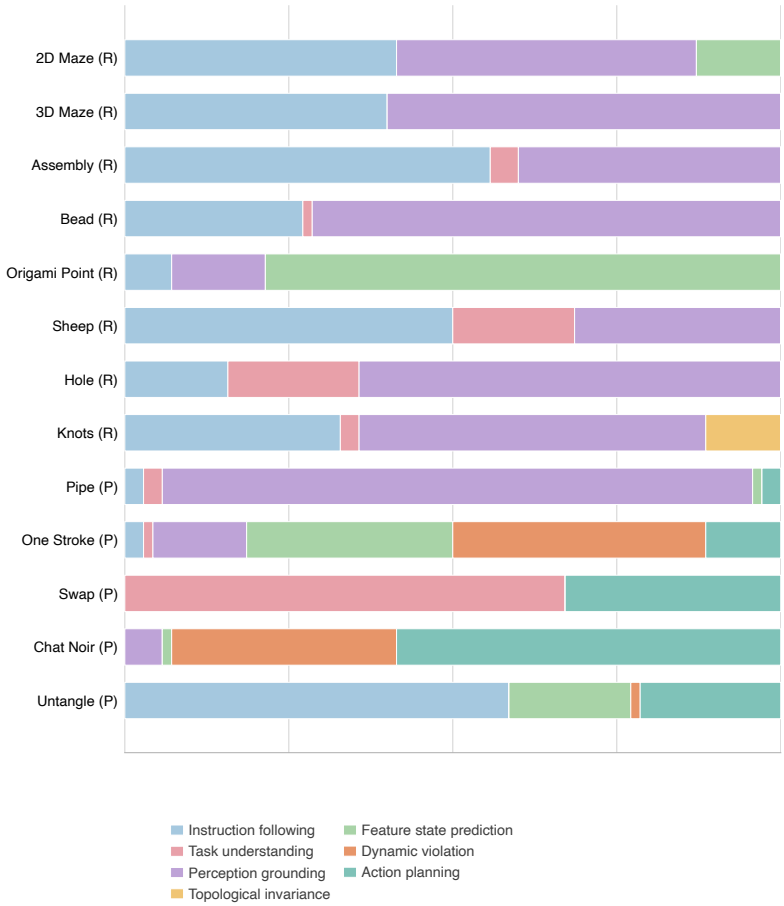


Figure 12 | **Environment-level error distributions.** Each bar pools the 35 labeled failures of Gemini-3.1-Pro and the 35 labeled failures of InternVL3.5-241B on one environment and shows the share of these 70 failures assigned to each error category.

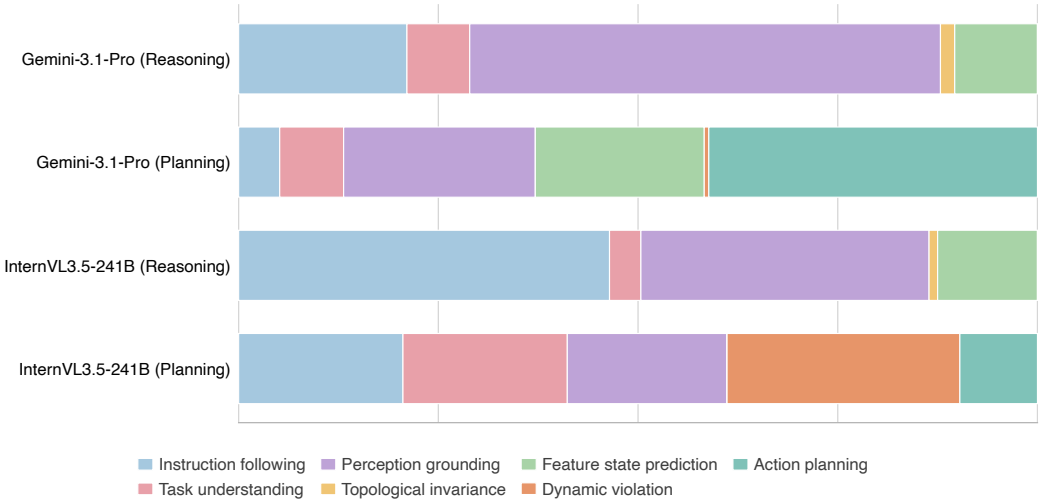


Figure 13 | **Model-level error distributions across reasoning and planning.** Each bar shows the category shares of the labeled failures of one model on one split, which number 280 for reasoning and 175 for planning.

Category	Gemini-3.1-Pro		InternVL3.5-241B		Combined	
	count	%	count	%	count	%
0. Instruction following	59	21.1	130	46.4	189	33.8
1. Task understanding	22	7.9	11	3.9	33	5.9
2. Perception grounding	165	58.9	101	36.1	266	47.5
3. Topological invariance	5	1.8	3	1.1	8	1.4
4. Feature state prediction	29	10.4	35	12.5	64	11.4
5. Dynamic violation	0	0.0	0	0.0	0	0.0
6. Action planning	0	0.0	0	0.0	0	0.0
Labeled failures	280	100	280	100	560	100

Table 40 | Category distribution of the labeled reasoning failures. Each model contributes 35 labeled failures on each of the 8 reasoning tasks, and the combined columns pool the 560 labeled failures of both models. Percentages are shares of the labeled failures in each column and may not sum to exactly 100 because of rounding.

Category	Gemini-3.1-Pro		InternVL3.5-241B		Combined	
	count	%	count	%	count	%
0. Instruction following	9	5.1	36	20.6	45	12.9
1. Task understanding	14	8.0	36	20.6	50	14.3
2. Perception grounding	42	24.0	35	20.0	77	22.0
3. Topological invariance	0	0.0	0	0.0	0	0.0
4. Feature state prediction	37	21.1	0	0.0	37	10.6
5. Dynamic violation	1	0.6	51	29.1	52	14.9
6. Action planning	72	41.1	17	9.7	89	25.4
Labeled failures	175	100	175	100	350	100

Table 41 | Category distribution of the labeled planning failures. Each model contributes 35 labeled failures on each of the 5 planning tasks, and the combined columns pool the 350 labeled failures of both models. Percentages are computed as in Table 40.

Benchmark	**Type**	**#Topo.**	**#Tasks**	**Size**	**QA**	**Inter.**	**Cog.**	**Diff.**	**Scale**
General visual and abstract reasoning									
BlindTest† [63]	Visual perception	–	7	4,860	✓	✗	✗	✓	✓
GameQA [64]	Game reasoning	–	158	~140K	✓	✗	✗	✓	✓
PuzzleVQA [65]	Abstract patterns	–	20	2,000	✓	✗	✓	✗	✓
Euclidean spatial reasoning									
CausalSpatial [51]	Causal spatial	–	4	1,012	✓	✗	✗	✓	✓
3DSRBench [41]	Euclidean	–	12	2,772	✓	✗	✗	✗	✗
iVISPAR [46]	Interactive spatial	–	1	300	✗	✓	✗	✓	✓
Mind the Gap [66]	Euclidean	–	6	1,800	✓	✗	✓	✓	✓
MindCube [8]	Euclidean	–	3	21,154	✓	✗	✓	✗	✗
OmniSpatial† [43]	Euclidean	–	50	8,400	✓	✗	✓	✗	✗
SITE [42]	Euclidean	–	6	8,068	✓	✗	✓	✗	✗
SpatialMQA [67]	Euclidean	–	6	5,392	✓	✗	✗	✗	✗
SpatialVLM [6]	Euclidean	–	2	546	✓	✗	✗	✗	✓
SpatialWorld [47]	Interactive spatial	–	6	760	✗	✓	✗	✓	✗
VSI-Bench [7]	Euclidean	–	8	5,130	✓	✗	✓	✗	✗
Topological spatial reasoning									
CurveBench [39]	Topological	2	1	756	✗	✗	✗	✓	✗
KnotGym [10]	Topological	1	3	Proc.	✗	✓	✗	✓	✓
TopoBench [40]	Topological	3	6	900	✗	✗	✗	✓	✓
MindTopo Core Set (ours)	**Topological**	**5**	**13**	**3,900**	✓	✓	✓	✓	✓

Table 42 | **Benchmark comparison.** #Topo. counts topological properties, Inter. marks interactive evaluation, Cog. marks an explicit human-cognition framework, Diff. marks difficulty levels, Scale marks automatic generation, and Proc. denotes procedural episodes. † Includes topology without making it the main focus.

and Papert [69]. Adult perception shows a related sensitivity. Chen [4, 5] showed that the visual system extracts topological invariants such as connectedness and number of holes ahead of feature-based attributes. Holes have been formalized as dependent entities by Casati and Varzi [70], with perceptual evidence on surrounded regions and hole shape from Nelson and Palmer [71] and Bertamini and Croucher [72]. Knots constitute a distinct cognitive domain,

characterized by Strohecker [11] as the "mother structure" that coordinates the other relations, and shown by Croom and Firestone [14] to strain adult intuitive physics. Hatcher [1] provides the algebraic-topology formalization, Martin [12] clarifies how Piagetian tasks map onto formal topological constructs, and Gärdenfors [73] argues that natural categories rest on topological notions such as connectedness and convexity. MINDTOPO translates this body of evidence into a unified visual taxonomy and asks whether modern foundation models exhibit the same topological primacy.

Cognitively Grounded and Topology-Adjacent Benchmarks. A closer line of work either revisits developmental psychology or tests topology through controlled spatial tasks. Piaget-inspired benchmarks diagnose developmental gaps in foundation models. CogDevelop2K [74] and CogLM [75] stage tests of reversed cognitive development. ConserveBench [76] and PerspectBench [77] target conservation and perspective-taking. KiVA [78] and BabyVision [37] extend the developmental lens to visual analogy and pre-linguistic vision. These benchmarks evaluate abilities adjacent to rather than organized around topological invariants. Individual spatial tasks include path connectivity in AlphaMaze [9], knots in KnotGym [10], folding in ORIGAMISPACE [79] and GamiBench [80], abstract positional reasoning in OPTiCAL [81], and constrained-manifold puzzles in Thinking in Structures [82]. Related work also probes how vision models respond to topological and geometric concepts [38].

TopoBench is the closest recent benchmark of global topology-focused constraint solving [40]. It contains 900 instances from six grid-puzzle families at three difficulty levels. The families cover path and network connectivity, loop closure, region partitioning under rotational symmetry, reflection-based visibility, and contiguity across intersecting axes. Puzzle-specific verifiers score complete solution grids. An analysis of 750 reasoning traces identifies recurring errors, and controlled interventions show that premature commitment and constraint forgetting directly reduce downstream accuracy. Additional experiments with cell-aligned representations and structured constraint tools attribute much of the remaining difficulty to extracting constraints from spatial representations. TopoBench presents symbolic grid inputs to reasoning LLMs and does not evaluate visual perception or closed-loop action. MINDTOPO instead grounds five Piagetian properties in rendered scenes and pairs reasoning questions with interactive planning environments.

Spatial Reasoning Benchmarks for MLLMs. Most spatial-reasoning benchmarks for multimodal large language models target Euclidean and viewpoint-centric competence rather than topological invariance. Static VQA benchmarks measure metric properties such as distance, direction, and inter-object relations. SpatialVLM [6] and SpatialMQA [67] ground this line. Broader aggregations including OmniSpatial [43], SITE [42], 3DSRBench [41], SPATIAL-DISE [83], and Mind the Gap [66] expand coverage but stay within a metric grammar. CausalSpatial moves from static relations to object-centric causal anticipation [51]. Its 1,012 multiple-choice examples cover collision, occlusion, compatibility, and trajectory at two difficulty levels. Each question pairs a rendered 3D scene with a hypothetical motion that changes one or more objects. The accompanying Causal Object World model derives object trajectories and renders videos that provide additional evidence to the MLLM. CausalSpatial therefore tests whether a model can predict the visual consequence of a specified intervention, while its fixed-question protocol does not test closed-loop action selection. Cognitive-map and viewpoint-integration benchmarks such as Thinking in Space [7], MindCube [8], and Theory of Space [44] probe spatial memory and limited-view inference. Interactive evaluations including iVISPAR [46], VSP [84], and RoboSpatial [85] test multi-step spatial control and affordance use.

SpatialWorld extends interactive spatial evaluation across heterogeneous environments [47]. It contains 760 human-annotated tasks from six scenario categories that are instantiated in eight simulation backends. Agents receive only egocentric RGB observations under partial

observability and express high-level decisions through a shared text-based action interface. Each task includes a human-validated initial state, a reference trajectory, and a terminal-state verifier. The benchmark evaluates 15 multimodal agents, with the strongest reaching a task success rate of 17.4%. Its categories organize application domains and action complexity rather than topological invariants. SpatialWorld therefore tests broad cross-environment interaction, while MINDTOPO pairs reasoning and planning under a five-property topological taxonomy. Neither SpatialWorld nor the other spatial benchmarks isolate invariants under continuous deformation. Their tasks hinge on quantitative geometry, viewpoint, intervention outcomes, or application goals rather than qualitative spatial structure.

Mechanistic studies offer a complementary account of spatial reasoning failures. Attention analysis finds that errors in spatial relation questions often coincide with attention directed toward irrelevant objects [86]. The resulting ADAPTVIS method adjusts visual attention using model confidence. This work motivates examining visual grounding alongside task accuracy, while our error annotations describe observable failures without identifying their internal attention mechanisms.

World Models and Physical Reasoning. A separate line of work evaluates MLLMs and video models as world simulators or physical reasoners, with focus on physical fidelity rather than topology preservation. World-model benchmarks such as PhyGenBench [87], Physion-Eval [88], EWMBench [89], WorldArena [90], World-in-World [91], and PhysicsMind [92] score image and video rollouts on commonsense and dynamics realism. Reasoning-oriented evaluations including MMGR [93], TiViBench [94], and VideoThinkBench [95] extend this view to multi-modal generative reasoning, and ENACT [52] casts embodied cognition as forward and inverse world modeling from egocentric interaction. Adjacent agentic benchmarks such as PhysBench [96], DeepPHY [97], and CHAIN [98] emphasize physics- and affordance-level reasoning. Phenomena such as objects passing through holes, surfaces being cut or sealed, and ropes tightening or untangling either appear only incidentally or are scored by metric and visual fidelity rather than by invariance under continuous deformation. MINDTOPO complements this line with topology-grounded probes that decouple topology preservation from physical realism, providing a controlled diagnostic for whether MLLMs internalize the qualitative structure of the physical world.

WorldAgen connects world modeling directly to action prediction through a shared model and adapts to new environments by learning from exploratory transitions at test time [99]. Its evaluation on robotic manipulation tasks addresses how improved dynamics prediction supports control. Our rollout audit examines a complementary requirement by testing whether generated state transitions obey the structural constraints of each environment.

E. Prompt Templates

E.1. Property-Level Prompt Templates

Representative property-level templates are shown below.

`[Task]` You are solving a continuity task. Determine which locations lie in the same connected component, or which single manipulation reconnects two disconnected locations.
`[Rules]` Use only the provided image(s). Do not cross walls, closed doors, blocking bars, or obstacle boundaries.
`[Question]` Return the requested reachable labels, removable bars, or connection decision.
`[Answer Format]` Output exactly one JSON object: `{"answer": value}`.

Figure 14 | Prompt for *Continuity* tasks.

`[Task]` You are solving a separation task. Identify whether visible parts belong to one connected object/sub-assembly, or choose a legal next action that keeps a single continuous stroke valid.
`[Rules]` Treat contact, overlap, and connected geometry according to the task definition; do not infer hidden connectors.
`[Question]` Select the matching option or next move.
`[Answer Format]` Output exactly one JSON object: `{"answer": value}`.

Figure 15 | Prompt for *Separation* tasks.

`[Task]` You are solving an order task. Read the sequence along a continuous carrier, track point labels through a fold sequence, or plan swaps that transform the current order into the goal order.
`[Rules]` Order is defined along the string, paper surface, or puzzle track rather than by raw image coordinates.
`[Question]` Return the requested sequence, relation, visible point set, or next action.
`[Answer Format]` Output exactly one JSON object: `{"answer": value}`.

Figure 16 | Prompt for *Order* tasks.

`[Task]` You are solving an enclosure task. Decide which objects are inside a closed boundary, whether an object can escape through gaps, how many through-holes a solid has, or where to place a blocking action.
`[Rules]` A region is enclosed only when every continuous path to the outside crosses a boundary; a hole must pass through the object.
`[Question]` Return the requested count, label set, repair choice, or action.
`[Answer Format]` Output exactly one JSON object: `{"answer": value}`.

Figure 17 | Prompt for *Enclosure* tasks.

`[Task]` You are solving a knots task. Determine whether the visible strand configuration is knotted, linked, unlinked, or composed of multiple components, or choose a legal endpoint move that untangles it.
`[Rules]` Use crossing order to infer the topology of static configurations. In Untangle, a legal move lifts one endpoint above the other strands and lowers it into the target hole.
`[Question]` Return the requested class, component count, link property, or next action.
`[Answer Format]` Output exactly one JSON object: `{"answer": value}`.

Figure 18 | Prompt for *Knots* tasks.

E.2. Task-Level Prompt Cards

The following cards reproduce the complete task-level prompts and qualitative examples, grouped by topological category and task. Each task's full specification appears in Appendix A.

E.2.1. Continuity

Reasoning Task: Continuity 2D Maze (Reachability Set)

[Task]
You are looking at a top-down view of a 2D maze. Several cells are marked with labeled circles (A, B, C, ...). Your job is to decide, for the target point, which of the OTHER labeled points you can reach from it through the open maze corridors. A point Y is reachable from X exactly when you can travel from X to Y without crossing any wall. You can only step from one cell to a neighboring cell across an open edge — you cannot cut diagonally through a wall corner.
Wall types: Walls in the maze come in three visual styles, all of which block movement equally: (a) full edge walls — solid lines along an entire cell edge; (b) partial edge walls — solid line segments shorter than a full edge; (c) diagonal walls — solid lines cutting across a cell along its diagonal. Any solid line, regardless of length or orientation, is a wall and cannot be crossed. Each labeled circle marks one cell.
The visual evidence for this question is provided below.
[Image 1]
Attached image.

[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
Which of the other labeled points can you reach from point A? The other labeled points are B, C, D, E, F.

[Answer Format]
Output exactly one JSON object using this schema: {"answer": {ans}} and nothing else.
Replace {ans} with the actual answer: a JSON array of point names, picked from the other labeled points shown in the image and NOT including the target itself, listed in alphabetical order. Use [] if no other point is reachable.

[Image]

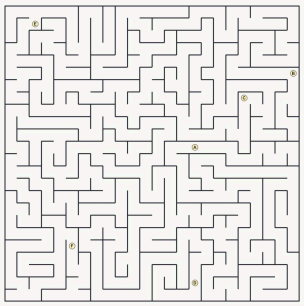


Figure 19 | **2D Maze** qualitative example (reachability set).

Reasoning Task: Continuity 2D Maze (Bar Removal)

[Task]
You are looking at a top-down view of a 2D maze. Two cells are marked with labeled circles A and B; in the current maze they are blocked from each other. Some of the maze's internal walls have been recolored as colored bars (purple / red / green / blue / yellow / orange). Suppose you are allowed to remove EXACTLY ONE bar — which colors of bar, when removed alone, would let A and B reach each other? Evaluate each bar independently: imagine removing only that one bar, leave every other bar in place, and check whether a path from A to B opens up. List EVERY color that works.
Wall types: Walls in the maze come in three visual styles, all of which block movement equally: (a) full edge walls — solid lines along an entire cell edge; (b) partial edge walls — solid line segments shorter than a full edge; (c) diagonal walls — solid lines cutting across a cell along its diagonal. Any solid line, regardless of length or orientation, is a wall and cannot be crossed. Each labeled circle marks one cell. A bar is a colored (non-black) wall; black walls are NOT bars.
Scoring: your answer is correct ONLY if you list every color whose single-bar removal reconnects A and B — no missing colors and no extras. A partial list, an extra color, or an empty answer when at least one qualifying bar exists, all count as wrong.
The visual evidence for this question is provided below.
[Image 1]
Attached image.

[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
Which colored bars, when removed individually, reconnect A and B? Evaluate purple, red, green, blue, yellow, and orange independently.

[Answer Format]
Output exactly one JSON object using this schema: {"answer": {ans}} and nothing else.
Replace {ans} with the actual answer: a JSON array of color names from the bars shown in the image, in alphabetical order. Use [] only if NO single-bar removal connects them.

[Image]

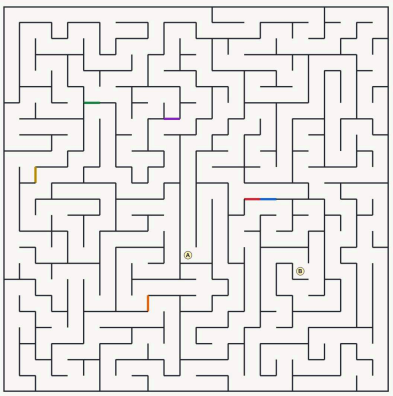


Figure 20 | **2D Maze** qualitative example (bar removal).

Reasoning Task: Continuity 3D Maze (Point List)

[Task]
You are solving a topological task called 3D Maze under the continuity category.
In this task, you must list every other labeled point that is connected to the target point under the shown door states. Two labeled points are connected only if an agent can travel between them without crossing walls, closed doors, furniture, or other solid obstacles. Open doors and open floor/corridor spaces are passable. Closed doors are not passable.
The visual evidence for this question is provided below.
[Image 1]
[Image 2]
[Image 3]
[Image 4]
[Image 5]

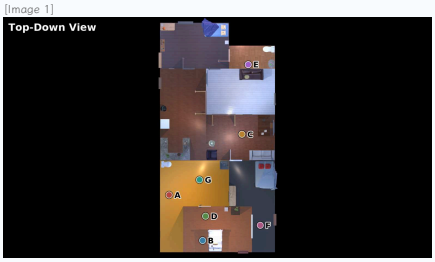
[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
List all other labeled points that are connected to point E under the shown door states. The other labeled points are: A, B, C, D, F, G.

[Answer Format]
Output exactly one JSON object: {"answer":{ans}} and nothing else.
Replace {ans} with the actual answer: a JSON array of point names, excluding the target point itself, listed in alphabetical order. Use [] if no other labeled point is connected to the target.

[Images]

[Image 1]



[Image 2]



[Image 3]



[Image 4]



[Image 5]

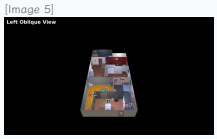


Figure 21 | **3D Maze** qualitative example (point list).

Reasoning Task: Continuity 3D Maze (Door Opening)

[Task]
You are solving a topological task called 3D Maze under the continuity category.
In this task, you must identify which colored doors need to be opened to connect two target points under the shown current door states.
Two labeled points are connected only if an agent can travel between them without crossing walls, closed doors, furniture, or other solid obstacles. Open doors and open floor/corridor spaces are passable. Closed doors are not passable.
The visual evidence for this question is provided below.
[Image 1]
[Image 2]
[Image 3]
[Image 4]
[Image 5]

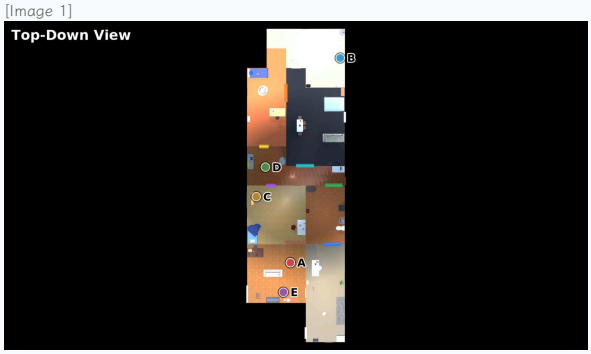
[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
Points A and B are not connected under the shown current door states. Open the unique minimum set of currently closed colored doors needed to make those two points connected. Count only doors that are currently closed; already-open doors need no action. Minimize the number of doors you open, not walking distance or the total doors along a route. The colored doors are: blue, green, yellow, purple, cyan, orange. Answer only with the color names of the doors that must be opened.

[Answer Format]
Output exactly one JSON object: {"answer":{ans}} and nothing else.
Replace {ans} with the actual answer: a JSON array of door color names. Use the unique minimum set of currently closed colored doors that must be opened, listed in alphabetical order.

[Images]

[Image 1]



[Image 2]



[Image 3]



[Image 4]



[Image 5]



Figure 22 | **3D Maze** qualitative example (door opening).

Interactive Task: Continuity Pipe

[Task]
You are solving Continuity Pipe. Rotate non-empty pipe cells until every pipe is connected to the green source. Grid coordinates are zero-based (x, y).

[Rules]
1. Choose exactly one non-empty pipe cell per turn.
2. The selected pipe rotates clockwise by 90 degrees.
3. If legal actions are listed, choose one of them; otherwise infer a legal non-empty cell from the current state.
4. The task succeeds when every pipe is connected to the source.

[Answer Format]
Output exactly one JSON object and nothing else: {"answer":{"x":X,"y":Y}}.

[Current Task]
The current state image is shown below.
[Image 1]
The board size is 5x5.
Legal actions for this turn are not provided; infer one non-empty pipe cell coordinate from the current state.
To solve this task, output the next legal action for the current state.

[Image]

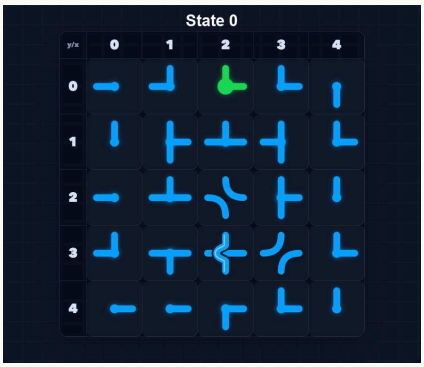


Figure 23 | **Pipe** qualitative example (first turn, 5×5 grid).

E.2.2. Separation

Reasoning Task: Separation Assembly (Bench Applaro)

[Task]
You are solving a topological task called Separation Objects under the separation category.
In this task, you must determine which candidate option shows the correct two-part decomposition of the complete object.
The correct option must split the complete object into exactly two candidate subassemblies that together form the complete object, with no missing parts, no extra parts, and no parts from another object.
Image 1 contains two labeled views of the complete object: Front upper right 45-degree oblique and Back upper left 45-degree oblique. Each option image contains the two candidate subassemblies for one option; each subassembly is shown from the front upper right 45-degree oblique view.
The visual evidence for this question is provided below.
[Image 1]
[Image 2]
[Image 3]
[Image 4]
[Image 5]
[Image 6]

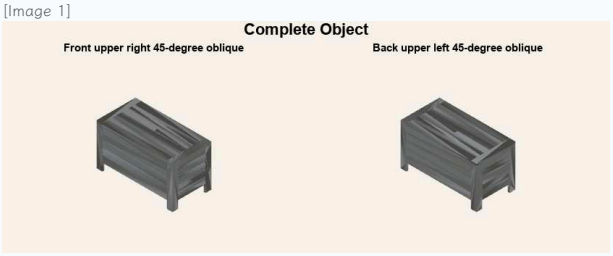
[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
Which option shows the correct two-part decomposition of the complete object? Choose one option from A, B, C, D, E.

[Answer Format]
Output exactly one JSON object: {"answer":"{ans}"} and nothing else.
Replace {ans} with the single legal answer for this task, chosen from "A", "B", "C", "D", "E".

[Images]

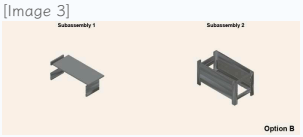
[Image 1]



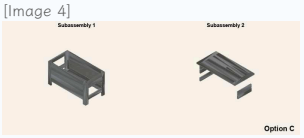
[Image 2]



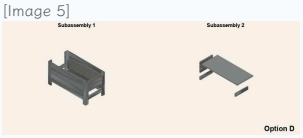
[Image 3]



[Image 4]



[Image 5]



[Image 6]

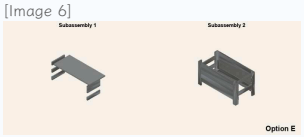


Figure 24 | **Assembly** qualitative example (bench, Applaro).

Reasoning Task: Separation Assembly (Chair Poang)

[Task]
You are solving a topological task called Separation Objects under the separation category.
In this task, you must determine which candidate option shows the correct two-part decomposition of the complete object.
The correct option must split the complete object into exactly two candidate subassemblies that together form the complete object, with no missing parts, no extra parts, and no parts from another object.
Image 1 contains two labeled views of the complete object: Front upper right 45-degree oblique and Back upper left 45-degree oblique. Each option image contains the two candidate subassemblies for one option; each subassembly is shown from the front upper right 45-degree oblique view.
The visual evidence for this question is provided below.
[Image 1]
[Image 2]
[Image 3]
[Image 4]
[Image 5]
[Image 6]

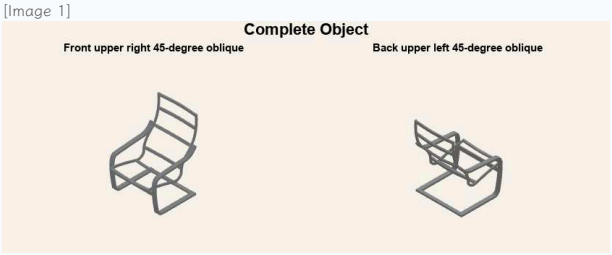
[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
Which option shows the correct two-part decomposition of the complete object? Choose one option from A, B, C, D, E.

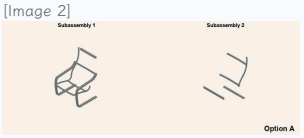
[Answer Format]
Output exactly one JSON object: {"answer":"{ans}"} and nothing else.
Replace {ans} with the single legal answer for this task, chosen from "A", "B", "C", "D", "E".

[Images]

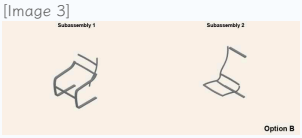
[Image 1]



[Image 2]



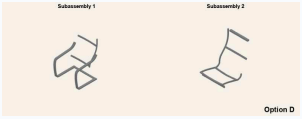
[Image 3]



[Image 4]



[Image 5]



[Image 6]

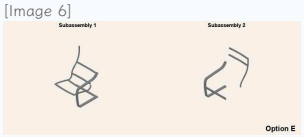


Figure 25 | **Assembly** qualitative example (chair, Poang).

Reasoning Task: Separation Assembly (Table Nesna)

[Task]
You are solving a topological task called Separation Objects under the separation category.
In this task, you must determine which candidate option shows the correct two-part decomposition of the complete object.
The correct option must split the complete object into exactly two candidate subassemblies that together form the complete object, with no missing parts, no extra parts, and no parts from another object.
Image 1 contains two labeled views of the complete object: Front upper right 45-degree oblique and Back upper left 45-degree oblique. Each option image contains the two candidate subassemblies for one option; each subassembly is shown from the front upper right 45-degree oblique view.
The visual evidence for this question is provided below.
[Image 1]
[Image 2]
[Image 3]
[Image 4]
[Image 5]
[Image 6]

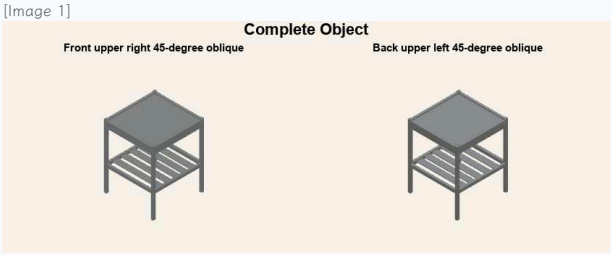
[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
Which option shows the correct two-part decomposition of the complete object? Choose one option from A, B, C, D, E.

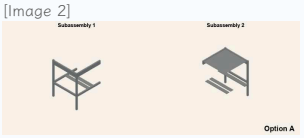
[Answer Format]
Output exactly one JSON object: {"answer":"{ans}"} and nothing else.
Replace {ans} with the single legal answer for this task, chosen from "A", "B", "C", "D", "E".

[Images]

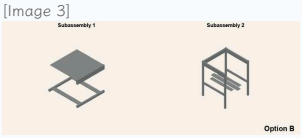
[Image 1]



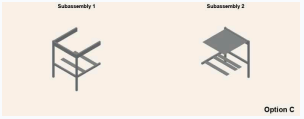
[Image 2]



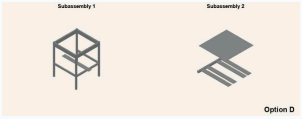
[Image 3]



[Image 4]



[Image 5]



[Image 6]

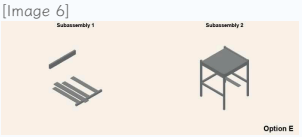


Figure 26 | **Assembly** qualitative example (table, Nesna).

Reasoning Task: Separation Assembly (Plant Stand Olivblad)

[Task]
You are solving a topological task called Separation Objects under the separation category.
In this task, you must determine which candidate option shows the correct two-part decomposition of the complete object.
The correct option must split the complete object into exactly two candidate subassemblies that together form the complete object, with no missing parts, no extra parts, and no parts from another object.
Image 1 contains two labeled views of the complete object: Front upper right 45-degree oblique and Back upper left 45-degree oblique. Each option image contains the two candidate subassemblies for one option; each subassembly is shown from the front upper right 45-degree oblique view.
The visual evidence for this question is provided below.
[Image 1]
[Image 2]
[Image 3]
[Image 4]
[Image 5]
[Image 6]

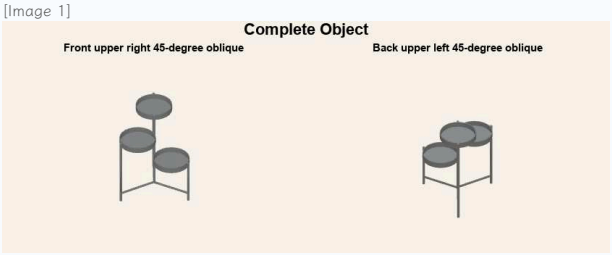
[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
Which option shows the correct two-part decomposition of the complete object? Choose one option from A, B, C, D, E.

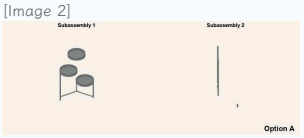
[Answer Format]
Output exactly one JSON object: {"answer":"{ans}"} and nothing else.
Replace {ans} with the single legal answer for this task, chosen from "A", "B", "C", "D", "E".

[Images]

[Image 1]



[Image 2]



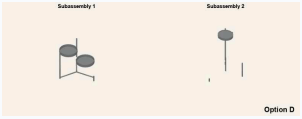
[Image 3]



[Image 4]



[Image 5]



[Image 6]



Figure 27 | **Assembly** qualitative example (plant stand, Olivblad).

Interactive Task: Separation One Stroke

[Task]
You are solving One-Stroke Color Grouping. Extend one continuous stroke from the bottom-left green start peg to the top-right red goal peg so same-colored cells share a region and different colors are separated. The white line is the current path.

[Rules]
1. Choose exactly one move from U, D, L, or R to extend the path by one edge.
2. Do not reuse an edge or create a closed loop. Backtracking over the most recent edge is allowed and undoes that move.
3. If legal actions are listed, choose one of them; otherwise infer a legal direction from the current state.
4. Illegal actions leave the state unchanged but still consume a step.
5. The task succeeds when the path reaches the goal and the final regions satisfy the color-separation rule.

[Answer Format]
Output exactly one JSON object and nothing else: {"answer":"D"}, replacing D with U, D, L, or R.

[Current Task]
The current state image is shown below.
[Image 1]
Legal actions for this turn: not provided; infer one legal action from the board state and the task rules.
To solve this task, output the next legal action for the current state.

[Image]



Figure 28 | **One Stroke** qualitative example (first turn, 6×6 board).

E.2.3. Order

Reasoning Task: Order Bead (Sequence Readout)

[Task]
You are solving the order task.
In this task, you must trace a bead string continuously from one endpoint to the other and report the bead colors in encounter order.
If this task depends on a specific visual definition, use this definition exactly: A bead string is a rope threaded through colored beads. The bead sequence is determined by following the rope continuously from one visible endpoint to the other and listing bead colors in the exact order encountered along that path.
The visual evidence for this question is provided below.
[Image 1]
Attached image.

[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.
4. Bead colors in these images are exactly: RED, BLUE, GREEN, YELLOW, ORANGE, PURPLE, WHITE, BROWN. Use these exact uppercase names; do not invent synonyms (e.g. write 'PURPLE', not 'VIOLET' or 'MAGENTA').

[Question]
Look at this 3D image of a curved string with colored beads threaded on it. Trace the string from one end to the other and list all bead colors in the order you encounter them.

[Answer Format]
Output exactly one JSON object: {"answer": {value}} and nothing else.
Replace {value} with the single legal answer for this task, chosen from a single JSON string containing the comma-separated bead-color sequence, for example "RED, BLUE, GREEN".

[Image]

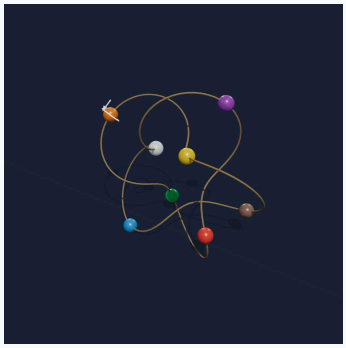


Figure 29 | **Bead** qualitative example (sequence readout).

Reasoning Task: Order Bead (Pair Relationship)

[Task]
You are solving the order task.
In this task, you must compare the bead-color sequences shown in two bead-string images and classify their relationship.
If this task depends on a specific visual definition, use this definition exactly: A bead string is a rope threaded through colored beads. IDENTICAL means the two sequences match exactly; REVERSED means one is the exact reverse of the other, such as RED, GREEN, BLUE, WHITE versus WHITE, BLUE, GREEN, RED; CYCLIC_ROTATION means one is a cyclic shift of the other without reversing the order, such as RED, GREEN, BLUE, WHITE versus BLUE, WHITE, RED, GREEN; DIFFERENT means none of the above.
The visual evidence for this question is provided below.
[Image 1]
Attached image.
[Image 2]
Attached image.

[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.
4. Bead colors in these images are exactly: RED, BLUE, GREEN, YELLOW, ORANGE, PURPLE, WHITE, BROWN. Use these exact uppercase names; do not invent synonyms (e.g. write 'PURPLE', not 'VIOLET' or 'MAGENTA').

[Question]
You are shown two images of bead strings. Each string has colored beads on a curved rope.
Compare the bead color sequences on both strings and determine their relationship.
Ignore the shape of the string. Focus only on the sequence of bead colors.
REVERSED means one sequence is the exact opposite order of the other; for example, RED, GREEN, BLUE, WHITE and WHITE, BLUE, GREEN, RED are REVERSED.
CYCLIC_ROTATION means the same circular order with a different starting bead, without reversing; for example, RED, GREEN, BLUE, WHITE and BLUE, WHITE, RED, GREEN are CYCLIC_ROTATION.

[Answer Format]
Output exactly one JSON object: {"answer": {value}} and nothing else.
Replace {value} with the single legal answer for this task, chosen from one of the JSON strings "IDENTICAL", "REVERSED", "CYCLIC_ROTATION", or "DIFFERENT".

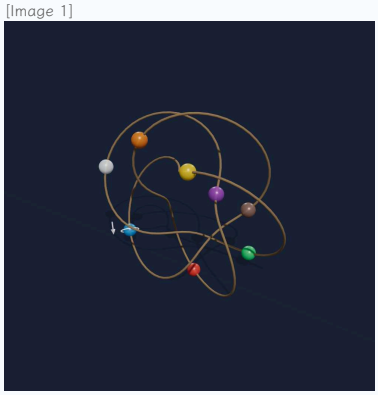
[Images]
[Image 1]

[Image 2]


Figure 30 | **Bead** qualitative example (pair relationship).

Reasoning Task: Origami Point Tracking (Bird-base)

[Task]
You are solving origami point tracking.
In this task, you must identify which labeled points on a folded piece of paper correspond to the unmarked dots placed on the original flat (unfolded) paper. The first image shows the flat paper with 2 unmarked dot(s) printed on it. The following images show the same paper being folded step by step, ending with the fully folded result where every visible point is labeled using the letters A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V. Paper is opaque: points on the back side may be hidden by overlying layers, and the same physical dot may appear in different positions across steps as the paper folds. The camera stays fixed, but the paper may rotate between images, sometimes by a large angle. Points are drawn only in the first and last images; the images in between show the paper without points. In the last image the dots are drawn at different sizes than before; a dot's size does not show which dot it is. You must track each unmarked dot through the fold sequence and report the letter(s) in the final image that correspond to the original unmarked dot(s).
If this task depends on a specific visual definition, use this definition exactly: not applicable
The visual evidence for this question is provided below.

[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
Which lettered point(s) in the final folded image correspond to the unmarked dot(s) shown on the flat paper in the first image? List the matching letter(s).

[Answer Format]
Output exactly one JSON object: {"answer": {json_answer_value}} and nothing else.
Replace {json_answer_value} with the single answer for this task, formatted according to a JSON array of uppercase letters chosen from [A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V], with no duplicates, e.g. ["A", "C"].

[Images]

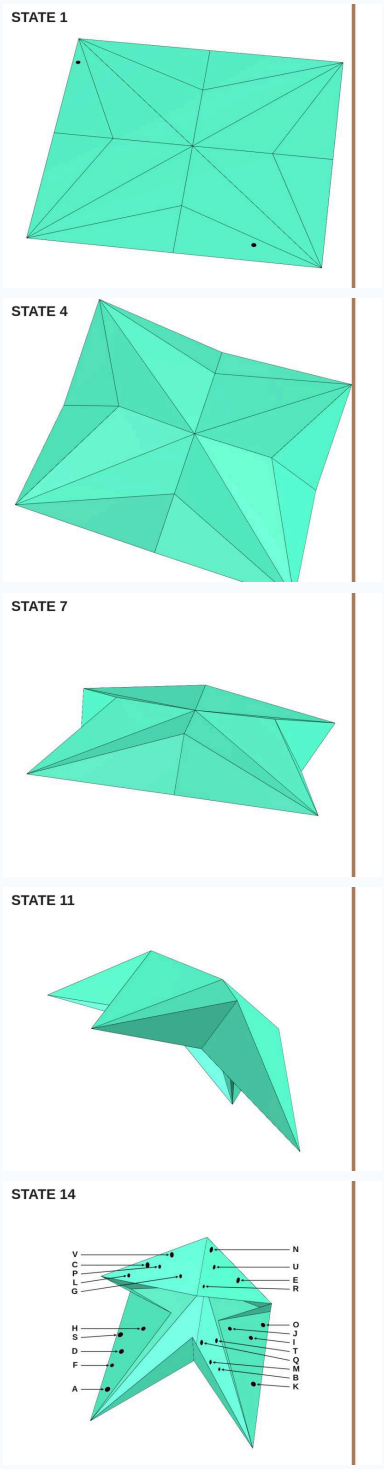


Figure 31 | **Origami Point** qualitative example (bird base).

Reasoning Task: Origami Point Tracking (Open-sink-base)

[Task]
You are solving origami point tracking.
In this task, you must identify which labeled points on a folded piece of paper correspond to the unmarked dots placed on the original flat (unfolded) paper. The first image shows the flat paper with 3 unmarked dot(s) printed on it. The following images show the same paper being folded step by step, ending with the fully folded result where every visible point is labeled using the letters A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T. Paper is opaque: points on the back side may be hidden by overlying layers, and the same physical dot may appear in different positions across steps as the paper folds. The camera stays fixed, but the paper may rotate between images, sometimes by a large angle. Points are drawn only in the first and last images; the images in between show the paper without points. In the last image the dots are drawn at different sizes than before; a dot's size does not show which dot it is. You must track each unmarked dot through the fold sequence and report the letter(s) in the final image that correspond to the original unmarked dot(s).
If this task depends on a specific visual definition, use this definition exactly: not applicable
The visual evidence for this question is provided below.

[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
Which lettered point(s) in the final folded image correspond to the unmarked dot(s) shown on the flat paper in the first image? List the matching letter(s).

[Answer Format]
Output exactly one JSON object: {"answer": {json_answer_value}} and nothing else.
Replace {json_answer_value} with the single answer for this task, formatted according to a JSON array of uppercase letters chosen from [A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T], with no duplicates, e.g. ["A", "C"].

[Images]

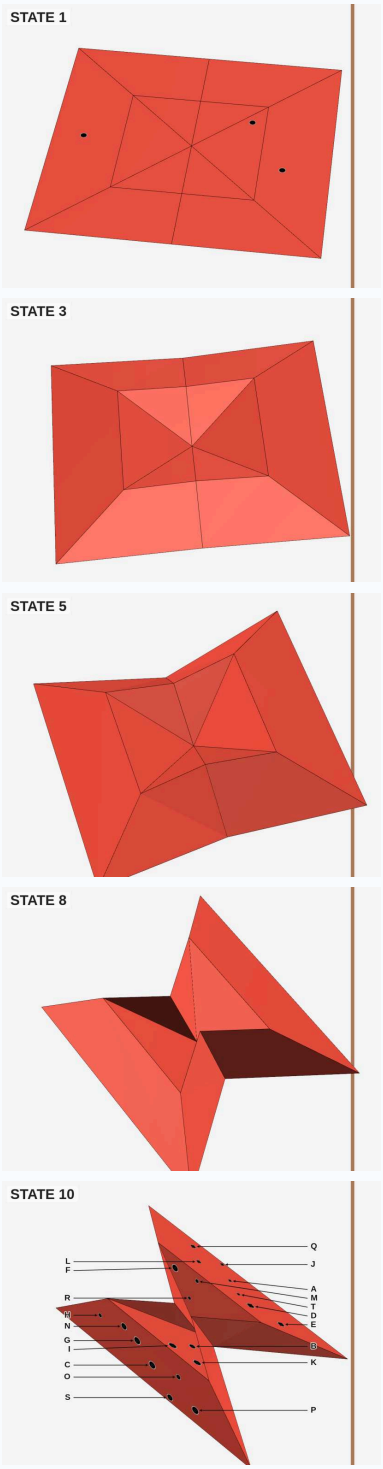


Figure 32 | **Origami Point** qualitative example (open-sink base).

Reasoning Task: Origami Point Tracking (Pinwheel-base)

[Task]
You are solving origami point tracking.
In this task, you must identify which labeled points on a folded piece of paper correspond to the unmarked dots placed on the original flat (unfolded) paper. The first image shows the flat paper with 1 unmarked dot(s) printed on it. The following images show the same paper being folded step by step, ending with the fully folded result where every visible point is labeled using the letters A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V. Paper is opaque: points on the back side may be hidden by overlying layers, and the same physical dot may appear in different positions across steps as the paper folds. The camera stays fixed, but the paper may rotate between images, sometimes by a large angle. Points are drawn only in the first and last images; the images in between show the paper without points. In the last image the dots are drawn at different sizes than before; a dot's size does not show which dot it is. You must track each unmarked dot through the fold sequence and report the letter(s) in the final image that correspond to the original unmarked dot(s).
If this task depends on a specific visual definition, use this definition exactly: not applicable
The visual evidence for this question is provided below.

[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
Which lettered point(s) in the final folded image correspond to the unmarked dot(s) shown on the flat paper in the first image? List the matching letter(s).

[Answer Format]
Output exactly one JSON object: {"answer": {json_answer_value}} and nothing else.
Replace {json_answer_value} with the single answer for this task, formatted according to a JSON array of uppercase letters chosen from [A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V], with no duplicates, e.g. ["A", "C"].

[Images]

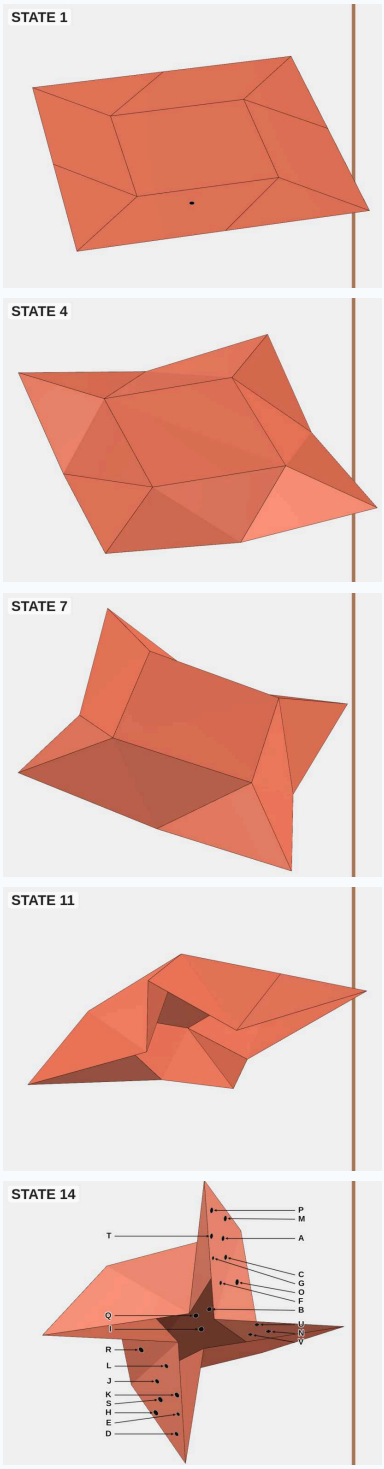


Figure 33 | **Origami Point** qualitative example (pinwheel base).

Reasoning Task: Origami Point Tracking (Waterbomb-base)

[Task]
You are solving origami point tracking.
In this task, you must identify which labeled points on a folded piece of paper correspond to the unmarked dots placed on the original flat (unfolded) paper. The first image shows the flat paper with 1 unmarked dot(s) printed on it. The following images show the same paper being folded step by step, ending with the fully folded result where every visible point is labeled using the letters A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P. Paper is opaque: points on the back side may be hidden by overlying layers, and the same physical dot may appear in different positions across steps as the paper folds. The camera stays fixed, but the paper may rotate between images, sometimes by a large angle. Points are drawn only in the first and last images; the images in between show the paper without points. In the last image the dots are drawn at different sizes than before; a dot's size does not show which dot it is. You must track each unmarked dot through the fold sequence and report the letter(s) in the final image that correspond to the original unmarked dot(s).
If this task depends on a specific visual definition, use this definition exactly: not applicable
The visual evidence for this question is provided below.

[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
Which lettered point(s) in the final folded image correspond to the unmarked dot(s) shown on the flat paper in the first image? List the matching letter(s).

[Answer Format]
Output exactly one JSON object: {"answer": {json_answer_value}} and nothing else.
Replace {json_answer_value} with the single answer for this task, formatted according to a JSON array of uppercase letters chosen from [A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P], with no duplicates, e.g. ["A", "C"].

[Images]

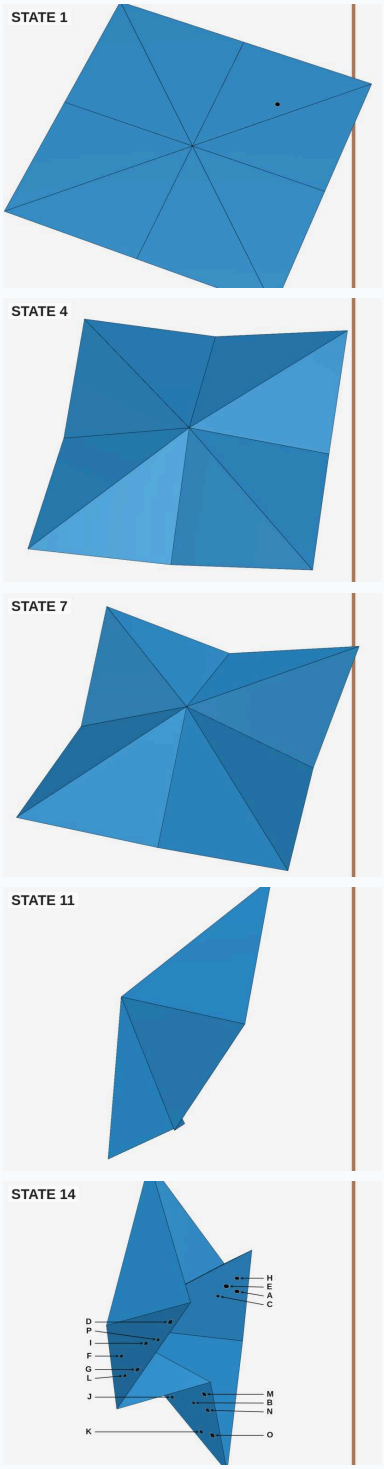


Figure 34 | **Origami Point** qualitative example (waterbomb base).

Interactive Task: Order Swap 2D Puzzle

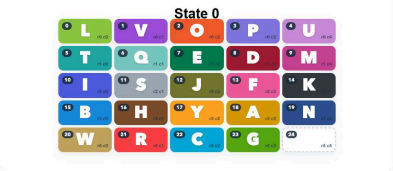
[Task]
You are solving a 2D grid swapping puzzle. Swap one colored block with the blank cell per turn until the current grid exactly matches the goal grid. Rows increase top-to-bottom and columns left-to-right; both are zero-based.

[Rules]
1. Choose exactly one non-empty current-grid cell per turn.
2. The selected block swaps with the blank cell.
3. If legal actions are listed, choose one of them; otherwise infer a legal non-empty cell from the current grid.
4. Illegal actions leave the state unchanged but still consume a step.

[Answer Format]
Output exactly one JSON object and nothing else: {"answer":{"row":R,"col":C}}.

[Current Task]
The current state image is shown below.
[Image 1]
The goal state image is shown below.
[Image 2]
Legal actions for this turn: not provided; infer one legal non-empty cell from the current grid.
To solve this task, output the next legal action for the current state.

[Images]
[Image 1]



[Image 2]



Figure 35 | **Swap** qualitative example (first turn, 5×5 grid).

E.2.4. Enclosure

Reasoning Task: Enclosure Hole (Hole Count)

[Task]
You are solving a topological task called Hole Detection under the enclosure category.
In this task, you must determine how many holes are visible in a top-down image of a board. Count only openings that pass through the board to open space.
A hole is an opening that connects through the board to open space. A pit or shallow depression that does not pass through the board is not a hole.
If another board is visible beneath it, evaluate only the top board and count holes on the top board only.
The visual evidence for this question is provided below.
[Image]

[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
How many holes are visible in this top-down view of the board?

[Answer Format]
Output exactly one JSON object: {"answer":{ans}} and nothing else.
Replace {ans} with the single legal answer for this task, chosen from the valid integers for this task.

[Image]

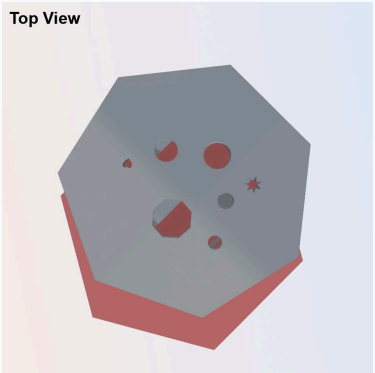


Figure 36 | **Hole** qualitative example (hole count).

Reasoning Task: Enclosure Sheep (Enclosed Count)

[Task]
You are solving the enclosure task.
In this task, you must look at a 3D scene with fences and numbered sheep, and count how many sheep cannot escape to the outside.
If this task depends on a specific visual definition, use this definition exactly: A sheep cannot escape if it is trapped inside fences and there is no continuous path from the sheep to the outside without crossing a fence segment. Sheep already outside all fences do not count. Sheep inside a fence with a usable gap to the outside also do not count.
The visual evidence for this question is provided below.
[Image 1]
Attached image.

[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
How many sheep cannot escape to the outside? Count sheep that are trapped inside fences. Do not count sheep that are already outside all fences or sheep that can escape through a gap.

[Answer Format]
Output exactly one JSON object: {"answer": {value}} and nothing else.
Replace {value} with the single legal answer for this task, chosen from a single JSON integer representing the count of sheep that cannot escape to the outside, for example 3.

[Image]

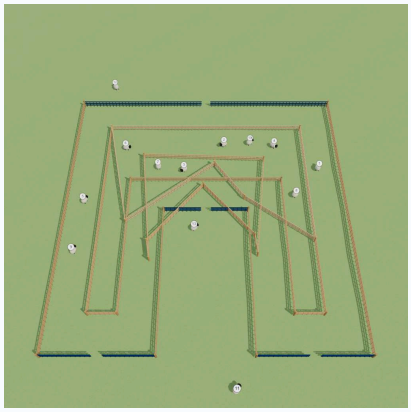


Figure 37 | **Sheep** qualitative example (enclosed count).

Reasoning Task: Enclosure Sheep (Escaping Sheep)

[Task]
You are solving the enclosure task.
In this task, you must look at a 3D scene with fences and numbered sheep, and identify which sheep can escape to the outside through fence gaps.
If this task depends on a specific visual definition, use this definition exactly: An enclosure is a closed region formed by connected fence segments. A gap is a missing fence segment that creates an opening. A sheep can escape if there exists a continuous path from that sheep to the exterior that passes only through gaps and does not cross any fence segment.
The visual evidence for this question is provided below.
[Image 1]
Attached image.

[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
Which sheep can escape to the outside through gaps in the fence? List their IDs.

[Answer Format]
Output exactly one JSON object: {"answer": {value}} and nothing else.
Replace {value} with the single legal answer for this task, chosen from the JSON string "NONE" or a single JSON string containing the escaping sheep IDs in ascending order, separated by commas, for example "1, 3, 5".

[Image]



Figure 38 | **Sheep** qualitative example (escaping sheep).

Reasoning Task: Enclosure Sheep (Densest Cell)

[Task]
You are solving the enclosure task.
In this task, you must look at a partitioned enclosure scene where inner fences divide the enclosure into labeled regions, and determine which region contains the most sheep.
If this task depends on a specific visual definition, use this definition exactly: A partitioned enclosure is divided into multiple labeled cells by inner fences. If multiple regions tie for the largest count, any tied label is acceptable.
The visual evidence for this question is provided below.
[Image 1]
Attached image.

[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
This scene shows a partitioned fenced area with labeled regions (A, B, C, ...). Which region contains the most sheep?

[Answer Format]
Output exactly one JSON object: {"answer": {value}} and nothing else.
Replace {value} with the single legal answer for this task, chosen from a single JSON string containing one region label visible in the image, such as "A", "B", or "C".

[Image]

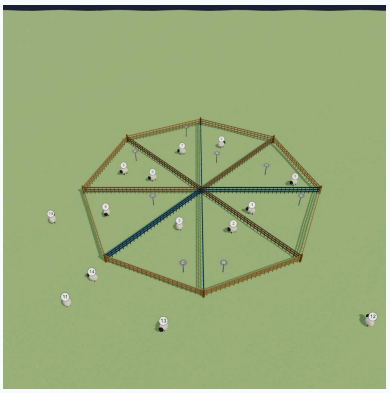


Figure 39 | **Sheep** qualitative example (densest cell).

Reasoning Task: Enclosure Sheep (Gap Repair)

[Task]
You are solving the enclosure task.
In this task, you must look at a 3D scene with a partially broken outermost fence and determine the minimum number of fence gaps that must be repaired to fully close it.
If this task depends on a specific visual definition, use this definition exactly: An enclosure is a closed region formed by connected fence segments. A gap is a missing fence segment that breaks the outermost fence. Each distinct gap counts as one repair, and the answer is the minimum number of repairs needed so that the outermost fence becomes fully closed.
The visual evidence for this question is provided below.
[Image 1]
Attached image.

[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
What is the minimum number of fence gaps that must be repaired to fully close the outermost fence?

[Answer Format]
Output exactly one JSON object: {"answer": {value}} and nothing else.
Replace {value} with the single legal answer for this task, chosen from a single JSON integer representing the minimum number of fence gaps that must be repaired, for example 2.

[Image]

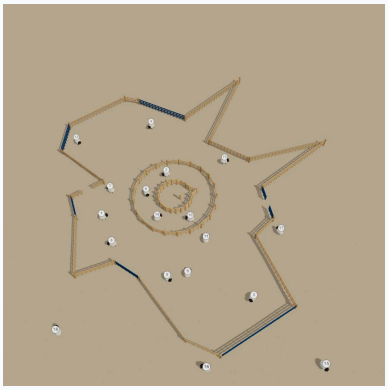


Figure 40 | **Sheep** qualitative example (gap repair).

Interactive Task: Enclosure Chat Noir

[Task]
You are solving Chat Noir / Encircle the Cat. Block one indexed hex cell per turn to trap the cat before it reaches the boundary. Cell indices are printed on the board.

[Rules]
1. Choose exactly one open non-cat cell to block.
2. After the block, the cat may stay still or move one hex step according to its policy.
3. If legal actions are listed, choose one of them; otherwise infer a legal open non-cat cell from the current state.
4. Illegal actions leave the state unchanged but still consume a step.
5. The task succeeds when the cat has no path to a boundary and fails if it reaches one.

[Answer Format]
Output exactly one JSON object and nothing else: {"answer":CELL_INDEX}.

[Current Task]
The current state image is shown below.
[Image 1]
The board radius is 4.
The current cat index is 30.
Legal actions for this turn: not provided; infer one legal open non-cat cell index from the board state and cell labels.
To solve this task, output the next legal action for the current state.

[Image]



Figure 41 | **Chat Noir** qualitative example (first turn, board radius 4).

E.2.5. Knots

Reasoning Task: Knots Static (Structure Classification)

[Task]
You are solving the knots task.
In this task, you must look at all the ropes in the image and decide which of the six categories below the whole scene fits into.
If this task depends on a specific visual definition, use this definition exactly:
A) Simple closed ring
- exactly ONE rope, its two ends are joined into a closed ring, AND it has NO knot tied in it.
- If you smoothed it out it would just be a circle with no self-overlap or crossings.

B) Knot
- exactly ONE rope tied into a knot. It may be a closed knotted ring, OR an open-ended knotted rope whose two ends are tied to sticks.
- Sticks may be present for open-ended knots, but are not required for this category. If sticks are shown, they are FIXED and IMMOVABLE, so the rope endpoints cannot be moved.
- The rope cannot be simplified into category A or C without cutting, or without moving fixed endpoints when sticks are present.

C) Open-ended rope with no knot
- exactly ONE open-ended rope, its two ends are tied to fixed sticks. The sticks are FIXED and IMMOVABLE in space, so the rope endpoints cannot be moved.
- With the endpoint constraints, the rope can still be pulled into a simple open arc without cutting.

D) Link
- TWO OR MORE closed-ring ropes, AND every single rope is linked to at least one other rope.
- No rope is completely free, every rope passes through some other rope and cannot be pulled away on its own.

E) Unlinked multiple ropes
- TWO OR MORE separate ropes; each individual rope may be open-ended or closed-ended, and may itself be tied into a knot.
- NONE of the ropes is linked to any OTHER rope. You could pick up each whole rope and carry it away from all the others without cutting anything.

F) Others
- the picture combines rope structures of two or more DIFFERENT categories from A-E in the same scene.
- Key test: if you cannot describe the WHOLE picture with a single A/B/C/D/E label and would need more category labels to cover everything, the answer is F.
The visual evidence for this question is provided below.
[Image 1]
Attached image.

[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
Using the six categories above, decide which one the entire scene in this image fits into.

Answer options:
A) Simple closed ring
B) Knot
C) Open-ended rope with no knot
D) Link
E) Unlinked multiple ropes
F) Others

[Answer Format]
Output exactly one JSON object: {"answer": {value}} and nothing else.
Replace {value} with the single legal answer for this task, chosen from one of the JSON strings "A", "B", "C", "D", "E", or "F".

[Image]

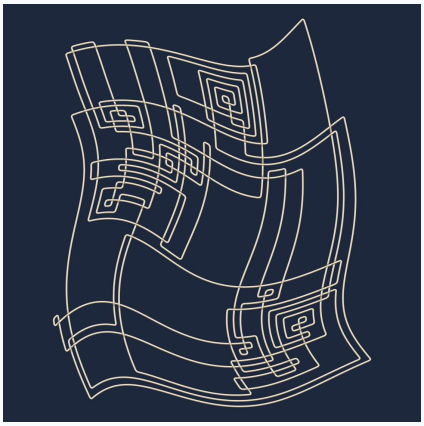


Figure 42 | **Knots** qualitative example (structure classification).

Reasoning Task: Knots Static (Component Count)

[Task]
You are solving the knots task.
In this task, you must count how many separate ropes are in the picture.
If this task depends on a specific visual definition, use this definition exactly:
A 'rope' is a single continuous strand.
Important:
- If two ropes are tied together, tangled up, or just lying on top of each other in the picture, they still count as TWO. We are counting physical strands, NOT how many groups would fall apart if you shook them.
- A single rope that crosses over itself is still ONE, not several ones.
- Sticks are FIXED and IMMOVABLE anchors in space; they are NOT ropes and must NOT be counted.

The visual evidence for this question is provided below.
[Image 1]
Attached image.

[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
How many separate ropes are there in this image?

[Answer Format]
Output exactly one JSON object: {"answer": {value}} and nothing else.
Replace {value} with the single legal answer for this task, chosen from a single non-negative JSON integer giving the number of ropes, for example 3.

[Image]

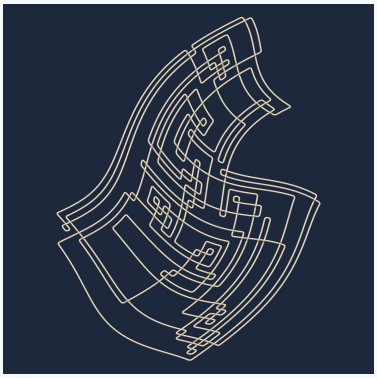


Figure 43 | **Knots** qualitative example (component count).

Reasoning Task: Knots Static (Link Topology)

[Task]
You are solving the knots task.
In this task, you must look at how the rings in the image are caught onto each other and pick which pattern matches the whole picture.
If this task depends on a specific visual definition, use this definition exactly:
All ropes here are closed rings, like rings on a key chain. Two rings are 'linked' when they go through each other so they cannot be pulled apart without cutting one open. Rings that just touch, sit next to each other, or look tangled but would slide apart with enough wiggling are NOT linked.

A) Not linked
- no two rings go through each other.
- Every ring could be lifted away from all the others without cutting.

B) Chain
- TWO OR MORE rings are hooked together one after another, like a metal chain.
- A two-ring paired link counts as a chain with two rings.
- Each ring is caught only on its direct neighbour(s) in its chain group, not on rings farther down the chain.
- If the picture contains multiple separate chain groups and no free rings or all-interlocked groups, the answer is still B.

C) All interlocked
- exactly THREE rings are woven together as a group.
- Any TWO rings by themselves would NOT be caught on each other, but the three together cannot be pulled apart unless one is cut.

D) Mixed
- no single A/B/C label describes the WHOLE picture.
- This also includes scenes where some rings are caught together while other rings are entirely free, such as a chain plus loose rings or an all-interlocked trio plus a free ring.
- This also includes scenes that combine chain-style links with all-interlocked groups.
- Key test: if you cannot describe the WHOLE picture with a single A/B/C label and would need more pattern labels to cover everything, the answer is D.
The visual evidence for this question is provided below.
[Image 1]
Attached image.

[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
Using the four patterns above, decide which one the entire scene in this image fits into.

Answer options:
A) Not linked
B) Chain
C) All interlocked
D) Mixed

[Answer Format]
Output exactly one JSON object: {"answer": {value}} and nothing else.
Replace {value} with the single legal answer for this task, chosen from one of the JSON strings "A", "B", "C", or "D".

[Image]

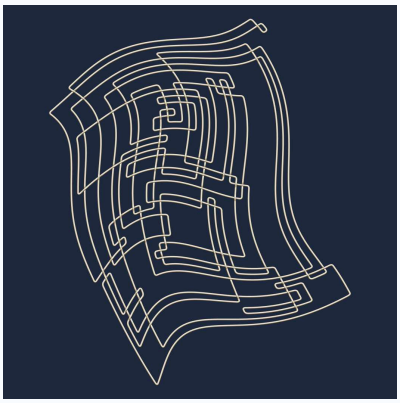


Figure 44 | **Knots** qualitative example (link topology).

Reasoning Task: Knots Static (Link Property)

[Task]
You are solving the knots task.
In this task, you must imagine cutting and removing the ring of a specified color, then list the colors of every remaining ring that is now FREE (not linked to any other remaining ring).
If this task depends on a specific visual definition, use this definition exactly:
Setup:
- Each ring in this picture has a distinct solid color, drawn from the palette: red, blue, green, brown, white, purple.
- You will be told the color of one specific ring. Imagine you cut that ring open and pull it out of the picture.
- Look at all the rings that REMAIN.

Definitions:
- 'Linked' = two rings pass through each other and cannot be pulled apart no matter how much you wiggle them. Rings that just touch, sit next to each other, or look tangled but would slide apart with enough wiggling are NOT linked.
- A remaining ring is FREE if it is not linked to ANY other remaining ring; you could pick it up and carry it away from all the other remaining rings without cutting anything.
- A remaining ring is NOT free if it is still linked to at least one other remaining ring.

What to output:
- List the colors of every remaining ring that is now FREE.
- Use only the color names from the palette: red, blue, green, brown, white, purple.
- The colors may be listed in any order.
- Do NOT include the color of the ring that was removed.
- If NO remaining ring is free (every remaining ring is still linked to at least one other), output exactly ["none"]. Do NOT output an empty list.

The visual evidence for this question is provided below.
[Image 1]
Attached image.

[Rules]
1. Use only the images and text provided in this prompt.
2. If answer options are provided, choose only from the provided options.
3. Do not output explanation beyond the required final answer.

[Question]
Imagine you cut and remove the green ring from this configuration. After removing it, which of the remaining rings are now FREE (not linked to any other remaining ring)? List the colors of all rings that are now free.

[Answer Format]
Output exactly one JSON object: {"answer": {value}} and nothing else.
Replace {value} with the single legal answer for this task, chosen from a JSON list of one or more strings drawn from {"red", "blue", "green", "brown", "white", "purple", "none"}; list every remaining ring that is now free; colors may appear in any order; for example ["red", "blue"]; if NO remaining ring is free output exactly ["none"] (never an empty list).

[Image]

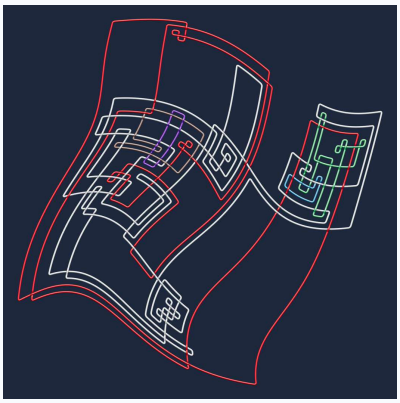


Figure 45 | **Knots** qualitative example (link property).

Interactive Task: Knots Untangle

[Task]
You are solving Knots Untangle. Move one rope endpoint at a time on the indexed pegboard until no two rope paths cross. Coordinates use zero-based (row, col).

[Rules]
1. Choose exactly one move per turn. If legal actions are listed, choose one of them; otherwise infer a legal move from the current state.
2. Move one endpoint from its occupied source hole to an empty target hole.
3. Illegal actions leave the state unchanged but still consume a step.
4. The task succeeds when zero crossings remain.

[Answer Format]
Output exactly one JSON object and nothing else:
{"answer":{"src_row":R,"src_col":C,"tgt_row":R,"tgt_col":C}}.

[Current Task]
The current state image is shown below.
[Image 1]
Attached image.

Current state summary:
- State index: 0
- Difficulty: hard
- Pegboard: 6x6 grid (rows/columns 0..5)
- Coordinate convention: holes are addressed as (row, col). Row numbers are shown along the top edge and increase from left to right; column numbers are shown along the left edge and increase from top to bottom. The top-left hole is (row=0, col=0).
- Rope count: 6
- Crossings remaining: 6
- Goal: reduce crossings to 0

Ropes on the board:
- Rope 0 (red) endpoints at (row=3,col=3) and (row=3,col=0)
- Rope 1 (green) endpoints at (row=5,col=0) and (row=1,col=3)
- Rope 2 (blue) endpoints at (row=0,col=4) and (row=4,col=5)
- Rope 3 (yellow) endpoints at (row=0,col=1) and (row=5,col=1)
- Rope 4 (purple) endpoints at (row=0,col=3) and (row=5,col=3)
- Rope 5 (orange) endpoints at (row=3,col=4) and (row=5,col=5)

Occupied holes: (row=3,col=3), (row=3,col=0), (row=5,col=0), (row=1,col=3), (row=0,col=4), (row=4,col=5), (row=0,col=1), (row=5,col=1), (row=0,col=3), (row=5,col=3), (row=3,col=4), (row=5,col=5)

Legal actions for this turn: not provided; infer one legal move from the current state image and task rules
Output ONLY a single JSON object in this exact format and nothing else:
{"answer":{"src_row":R,"src_col":C,"tgt_row":R,"tgt_col":C}}

[Image]

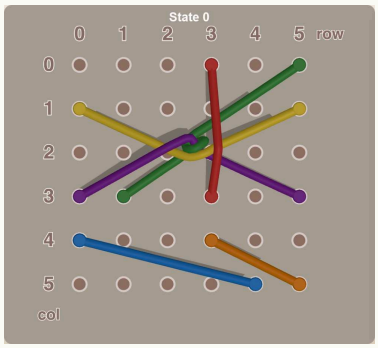


Figure 46 | **Untangle** qualitative example (first turn, 6×6 pegboard).

F. The Use of Large Language Models

We used large language models (LLMs), including Google's Gemini-3.1-Pro and OpenAI's GPT-5.5, as auxiliary tools to assist with writing, editing, and conducting the literature review for this manuscript. All content was critically revised and fact-checked by the human authors to ensure its scientific validity and originality. The authors are fully responsible for all statements and conclusions presented in this paper.

G. Extended Set

Composition. The Core Set contains 3,900 instances, with 150 Easy and 150 Hard instances for every task. The Extended Set contains 8,780 additional instances across the same 13 tasks. Table 2a gives the per-task composition in the main paper.

Exact-input deduplication. The two sets have no exact model-input overlap. Starting from the 11,035-row original pool, we remove 2,250 inputs that appear in Core and five repeated Swap inputs. For reasoning, input equality requires identical whitespace-normalized question text and an identical ordered sequence of image-file SHA256 hashes. For planning, it requires an identical saved initial scene and reset configuration together with the same action budget. Generation seeds and difficulty tags do not affect this comparison. Shared identifiers do not establish duplicate inputs: 150 Origami identifiers refer to different questions and images and remain in both sets under distinct release IDs. This rule checks exact equality, not perceptual similarity or topological isomorphism.

Evaluation results. Table 2b reports the evaluations of the Extended Set in the main paper. We recompute each score after removing Core overlaps and the five repeated Swap rows. The table includes the 11 models whose removed rows are recoverable from per-instance logs or matched Core subset results. Core and Extended differ in difficulty distribution, so their scores are reported separately and are not pooled. The Extended Set retains its source generators' difficulty labels, which are not a separately calibrated difficulty level.

Traceability. Each release manifest records the task, instance identifier, source record, record hash, and input hash. The Core manifest also records the Easy or Hard level. The removal manifest records every excluded row and whether it matched Core or repeated another Extended input. The result provenance file records the integer counts and source hashes used to recompute Table 2b.