SpatialViz-Bench: Automatically Generated Spatial Visualization Reasoning Tasks for MLLMs

Siting Wang[1,2] **Luoyang Sun**[1,2] **Cheng Deng**[3,†] **Kun Shao**[3] **Minnan Pei**[1,2]

Zheng Tian[4] **Haifeng Zhang**[1,†] **Jun Wang**[5,6,†]

[1]The Key Laboratory of Cognition and Decision Intelligence for Complex Systems
Institute of Automation, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences, Beijing, China
[3]Huawei Noah's Ark, UK
[4] School of Creativity and Art, ShanghaiTech University
[5]University College London
[6]UCL Centre for Artificial Intelligence

Abstract

Humans can directly imagine and manipulate visual images in their minds, a capability known as *spatial visualization*. While multi-modal Large Language Models (MLLMs) support imagination-based reasoning, *spatial visualization* remains insufficiently evaluated, typically embedded within broader mathematical and logical assessments. Existing evaluations often rely on IQ tests or math competitions that may overlap with training data, compromising assessment reliability. To this end, we introduce ***SpatialViz-Bench***, a comprehensive multi-modal benchmark for *spatial visualization* with *12* tasks across *4* sub-abilities, comprising *1,180* automatically generated problems. Our evaluation of *33* state-of-the-art MLLMs not only reveals wide performance variations and demonstrates the benchmark's strong discriminative power, but also uncovers counter-intuitive findings: models exhibit unexpected behaviors by showing difficulty perception that misaligns with human intuition, displaying dramatic 2D-to-3D performance cliffs, and defaulting to formula derivation despite spatial tasks requiring visualization alone. SpatialViz-Bench empirically demonstrates that state-of-the-art MLLMs continue to exhibit deficiencies in *spatial visualization* tasks, thereby addressing a significant lacuna in the field. The benchmark is publicly available. [*] [*]

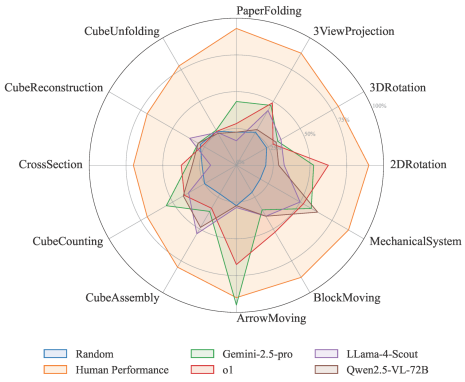
1 Introduction

In IQ tests, common tasks include visualizing objects after rotation or folding, which assess *spatial visualization* ability—the cognitive capacity to mentally manipulate spatial information and visual imagery.

Spatial visualization, as a distinct cognitive ability, was firstly identified by Thurstone on primary mental abilities in 1938, gaining widespread recognition and extensive investigation in psychology and cognitive science [1]. Completing reasoning tasks that require *spatial visualization* typically involves 2 other spatial abilities: *spatial perception* and *spatial memorization*. *Spatial perception* [2] aims

[†]Correspondence to Cheng Deng, Haifeng Zhang Jun Wang.
[*]Data: `https://huggingface.co/datasets/PLM-Team/Spatial-Visualization-Benchmark`
[*]Code: `https://github.com/wangst0181/Spatial-Visualization-Benchmark`

Preprint. Under review.



a

Benchmark	Categories	Source	2DR 3DR	3V	PF	CU CR	CS	CC	CA	AM BM	MS
MM-IQ	3D-Geometry	Web	✓	✓		✓	✓		✓		
MARVEL	3D-Geometry	Web				✓					
EMMA	3D Spatial Simulation 2D Transformation	Web		✓	✓						
BLINK	IQ-Test	Web				✓					
MATH-Vision	Descriptive Geometry, Transformation Geometry	Web	✓	✓	✓	✓					
LogicVista	3D Shapes, Mechanical	Web	✓	✓							✓
VisualPuzzles	Spatial Reasoning	Web	✓		✓	✓			✓		
Ours	Spatial Visualization	Auto-Gen	✓	✓	✓	✓	✓	✓	✓	✓	✓

b

Figure 1: **Overview of model performance and benchmark tasks.** Part a shows the zero-shot accuracy across tasks, highlighting gaps between leading proprietary and open-source models, human performance, and random baselines; part b shows the comparison of SpatialViz-Bench with existing spatial visualization benchmarks, while feature checkmarks indicate presence, they do not imply comprehensive or systematic coverage.

to perceive external spatial information and relationships, while *spatial memorization* [3] requires temporarily storing spatial transformation information mentally without accessing physical objects. To understand how MLLMs complete *spatial visualization* tasks, we subdivide this capability into 4 sub-abilities: *mental rotation*, *mental folding*, *visual penetration*, and *mental animation*, allowing for more precise task design.

Research in computer vision and embodied intelligence has consistently emphasized the study of spatial ability. Traditional benchmarks for *spatial perception* typically limit their evaluations to tasks such as depth estimation and relative position judgment. With the recent advances in multimodal large language models (e.g., o1, Gemini-2.5, Claude-3.7, Qwen2.5VL, Doubao), researchers investigat techniques to bolster general-purpose models' capabilities in complex spatial understanding and reasoning. As for *spatial perception*, SpatialRGPT-bench [4] proposes tasks of spatial relationships requiring world knowledge. Moreover, VSI-Bench [5] addresses the gap in *spatial memorization* assessment by drawing inspiration from humans' ability to mentally reconstruct home layouts when selecting furniture.

Despite the inclusion of *spatial visualization* items in existing multimodal reasoning benchmarks, these tasks are usually buried under broader categories such as mathematical or logical reasoning (see Figure 1) rather than treated as dedicated spatial-reasoning challenges. Consequently, they primarily test whether a model can "solve" a given problem, rather than driving research toward core spatial abilities. Moreover, most examples are drawn from publicly available sources, online IQ tests, psychological batteries, administrative exams, and math contests, which risks overlap between training and evaluation data and undermines reliability. The scarcity of items per subskill also magnifies random error, while heterogeneous formats make it hard to distinguish true reasoning failures from misreading. Even when models may have seen similar problems during pretraining, performance remains poor. State-of-the-art systems score just *27.64* on 3D Geometry in MM-IQ [6] and *26.00* on Descriptive Geometry in MathVision [7]. To address these issues, a scalable, standardized benchmark for *spatial visualization* is necessary and meaningful, ensuring fair evaluation, enabling diagnostic analysis of failure modes, and providing clear targets for model improvement.

Spatial visualization, a high-level spatial skill, is essential for MLLMs to succeed in diverse downstream tasks. In embodied intelligence, for instance, robots navigating 3D environments rely on *spatial visualization* for accurate scene understanding, action planning, and internal simulation. Directly measuring these skills in complex applications is difficult, since many factors can affect task outcomes. Like human exams that isolate core competencies, evaluating models on basic spatial tasks pinpoints their true abilities. And strong results on these foundational challenges reliably signal readiness for more advanced spatial problems.

Based on the research background and motivations outlined above, we put forward a novel multimodal spatial reasoning benchmark, ***SpatialViz-Bench***, specifically designed to evaluate the *spatial visualization* abilities of MLLMs. The main contributions of our work can be listed as follows:

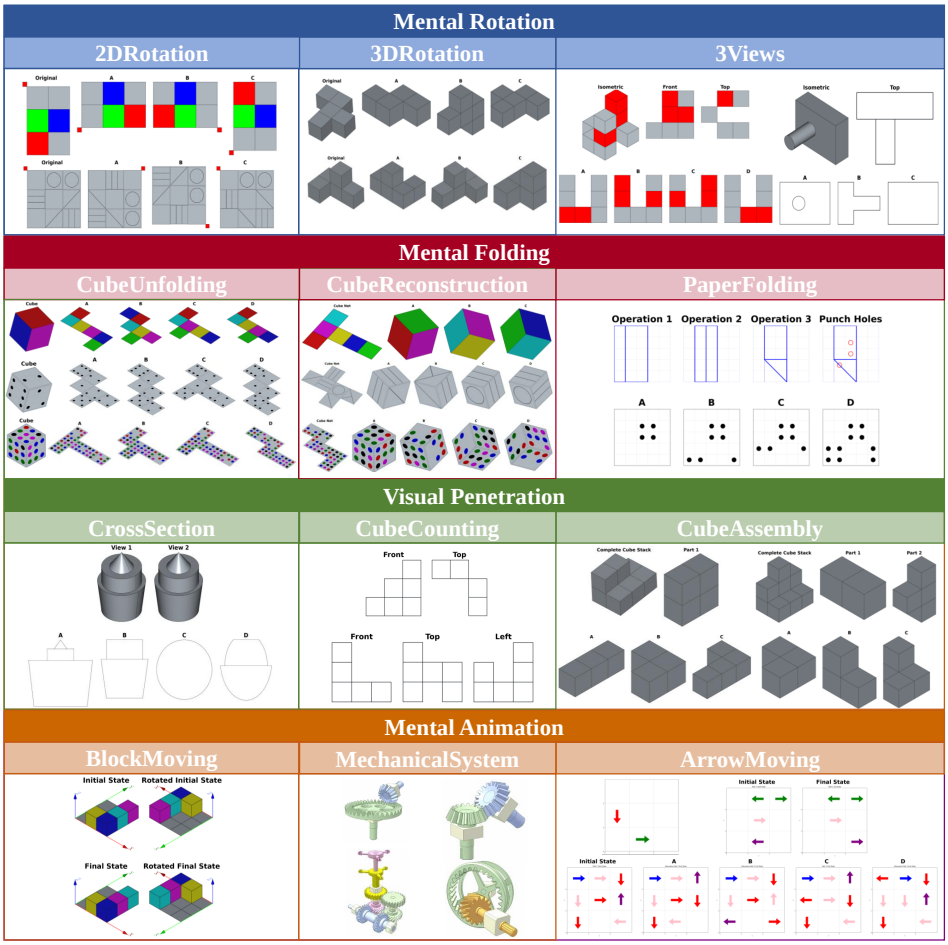


Figure 2: **The overview of SpatialViz-Bench.** SpatialViz-Bench evaluates 4 spatial sub-abilities, mental rotation, mental folding, visual penetration, and mental animation, via 3 tasks each (12 tasks total). Each task has 2–3 difficulty levels of 40–50 cases, yielding 1,180 question–answer pairs.

- We construct an enhanced framework of spatial visualization ability and summarized the process to complete *spatial visualization* tasks.
- To evaluate the ***4*** key sub-abilities of ***spatial visualization***, we design ***12*** basic assessment tasks and complete a pipeline for automatically generating problems. Each task has 2–3 difficulty levels, with 40–50 test cases per level, forming a total test set of ***1,180*** examples.
- We systematically evaluated ***33*** MLLMs, including 9 closed-source and 24 open-source models ranging from 3B to 108B parameters. Gemini-2.5-pro and ChatGPT o1 achieved the highest overall scores of *44.66* and *41.36*, outperforming others by about 10%.
- Our analysis reveals that models exhibit unexpected behaviors by showing difficulty perception that misaligns with human intuition, displaying dramatic 2D-to-3D performance cliffs, and defaulting to formula derivation despite spatial tasks requiring visualization alone.

2 Related Works

2.1 Spatial Tasks in Vision-Language Understanding and Reasoning

Early benchmarks in computer vision focused on perceptual-level understanding by extracting explicit geometric cues from visual inputs in the form of classification or regression, while recent MLLM evaluations are visual question answering tasks. Existing benchmarks [8, 9, 10, 11, 12] evaluate *spatial perception* in terms of object- or camera-centric spatial relationships, relative distances, and object size comparisons.

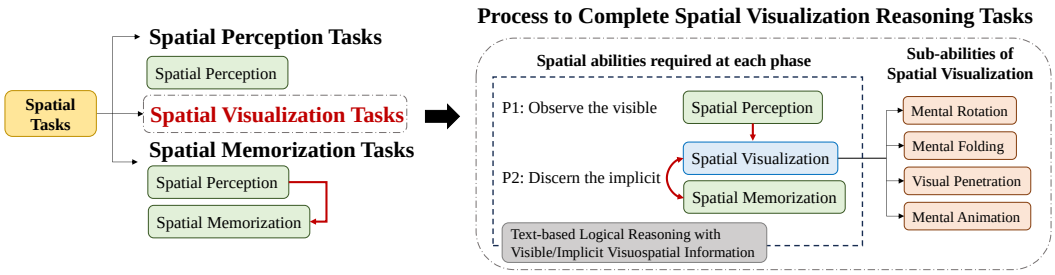


Figure 3: **Spatial visualization tasks process.** The process has 2 phases, first perceiving visible cues (spatial perception), then inferring hidden details (alternating visualization and memorization), with language-driven reasoning guiding each step. We define 4 sub-abilities, each tied to a specific task type.

With the development of MLLMs, benchmarks that emphasize complex reasoning have emerged recently. *Spatial perception* still remains the dominant focus. Benchmarks like Blink [13] pioneered the use of marks on images to evaluate spatial relationships, while SpatialRGPT-bench [4] extends this approach by incorporating multi-hop reasoning that depend on world knowledge. Video-based benchmarks such as VCBench [14] and VSI-bench [5] have been introduced to fill the gap in evaluating *spatial memorization* capabilities, assessing spatial working memory and perspective-shifting abilities in dynamic contexts.

However, *spatial visualization* remains underexplored. Few benchmarks like SPARE3D [15] and CLEVR-MRT [16] offer *mental rotation* tasks, while works like Stogiannidis et al. [17] and Xu et al. [18] inspired by Gardner's Theory [19] treat *spatial visualization* as an undifferentiated subskill of spatial ability, lacking finer-grained categorization and relying on a narrow set of tasks for evaluation.

2.2 Spatial Visualization Tasks in Existing Multimodal Reasoning Benchmarks

Current multimodal reasoning benchmarks typically fall into four categories: (1) world knowledge reasoning [20, 21], (2) mathematical and logical reasoning [6, 22, 23, 7, 24, 25, 26], (3) domain-specific reasoning [27, 28, 29, 30], and (4) spatial reasoning [4, 31, 32].

Notably, *spatial visualization* tasks are often treated as subcategories under mathematical and logical reasoning benchmarks as shown in Figure 1. These tasks primarily serve to evaluate whether models can solve such problems, rather than to guide the development of models specifically toward addressing spatial reasoning challenges. In contrast, our benchmark is structured around 4 core sub-abilities of *spatial visualization* identified in cognitive psychology, aiming to guide the direction for model refinement in *spatial visualization*. In addition, we summarized the essential phases to complete *spatial visualization* tasks, enabling us to identify the causes of reasoning errors.

3 Spatial Visualization

Spatial visualization is a core component of human cognitive systems and a critical capability for MLLMs to deploy in downstream applications. Research of *spatial visualization* began with Thurstone in 1938 [1], which is defined as one of the cognitive abilities to perform mental operations on visual images. Subsequently, Thurstone [2] proposed three spatial factors, including *spatial perception*, *spatial visualization* and *mental rotation*. After systematic research in neuropsychology, *spatial visualization* ability can be tested from multiple aspects, including *mental folding* [33], *mental animation* [34], and *visual penetration* [35]. Based on a comprehensive analysis of existing literature, we reclassify spatial abilities into *spatial perception*, *spatial memorization* [3], and *spatial visualization*, where the former two form the basis for performing tasks involving the latter.

As shown in Figure 3, we decompose the process of completing *spatial visualization* tasks into 2 phases: observing visible information and discerning implicit information, forming the necessary information for *spatial visualization* reasoning. The former only requires *spatial perception* abilities, while the latter demands alternating between *spatial visualization* and *spatial memorization* abilities. Throughout the entire process, language-inspired logical reasoning serves as the backbone facilitating progression through these phases.

Table 1: Sub-abilities, Corresponding Tasks, and Evaluation Focus.

Sub-ability	Tasks	Levels	Cases	Evaluation Focus
Mental Rotation	2D Rotation	2	80	Assess the model's ability to recognize object shapes after rotation and infer spatial structures from different viewpoints.
	3D Rotation	2	80	
	Three-view Projection	2	100	
Mental Folding	Paper Folding	3	120	Test the model's understanding of object transformations between different geometric forms.
	Cube Unfolding	3	120	
	Cube Reconstruction	3	120	
Visual Penetration	Cross-section	3	120	Evaluate the ability to infer internal object structures based on visible exterior features.
	Cube Counting	3	120	
	Cube Assembly	2	80	
Mental Animation	Arrow Moving	2	80	Examine the understanding of dynamic state changes and causal propagation across system components.
	Block Moving	2	80	
	Mechanical System	2	80	

- **P1-1 Spatial Perception:** Perceive visual objects for visible spatial information and relationships;
- **P2-1 Spatial Visualization:** Mentally manipulate the visual images to discern implicit spatial information beneath the surface;
- **P2-2 Spatial Memorization:** Temporarily store visuospatial information mentally to support ongoing spatial tasks.

Additionally, our research incorporates mental rotation within *spatial visualization* ability, instead of a parallel construct, thus covering 4 following sub-abilities: 1) **Mental Rotation**, representing and rotating two-dimensional and three-dimensional objects in space mentally while maintaining object features; 2) **Mental Folding**, folding two-dimensional patterns into three-dimensional objects or unfold three-dimensional objects into two-dimensional representations; 3) **Visual Penetration**, imagining the internal structure of objects based on external features; 4) **Mental Animation**, mentally visualizing the motion and movement of components within any form of system or in general.

4 SpatialViz-Bench

4.1 Overview

Existing benchmarks for *spatial visualization* abilities often rely on web-sourced tasks, resulting in disjointed and inconsistent problem formulations that lack standardization (see Figure 1). This availability-driven collection strategy overlooks systematic design grounded in cognitive theory, leading to incomplete coverage of key sub-abilities and limiting the robustness of evaluation results. To address these limitations, we propose a systematic, ability-centric methodology. Specifically, we construct a hierarchical classification framework grounded in cognitive principles, map existing tasks to this taxonomy, and design new tasks to fill underrepresented abilities. All tasks adopt a unified input format and standardized question templates to minimize confounding factors arising from task variation and to support fine-grained analysis of model reasoning errors (as shown in Section 5.3).

Following the systematic ability-based methodology described above, we first establish a theoretical framework of *spatial visualization* ability in Section 3. Based on the framework, we propose ***SpatialViz-Bench***, a benchmark designed to quantitatively evaluate the *spatial visualization* capabilities of MLLMs. *SpatialViz-Bench* encompasses 4 dimensions, corresponding to the 4 core sub-abilities of *spatial visualization*: *mental rotation*, *mental folding*, *visual penetration*, and *mental animation*. The evaluation of each sub-ability and their corresponding tasks is summarized in Table 1.

For each sub-ability, we have meticulously designed *3* assessment tasks that focus on that particular ability, forming a comprehensive evaluation system comprising 12 tasks in total. Each task is further divided into *2* or *3* difficulty levels, with each level containing *40* or *50* carefully designed test cases, resulting in a total of *1,180* question-answer pairs. Regarding data sources, except for Level 1 of the three-view projection task, which utilizes model files from the DeepCAD dataset [36], and the mechanical system task, which employs system materials acquired from open-source material websites, all other tasks are automatically generated using Python and FreeCAD.

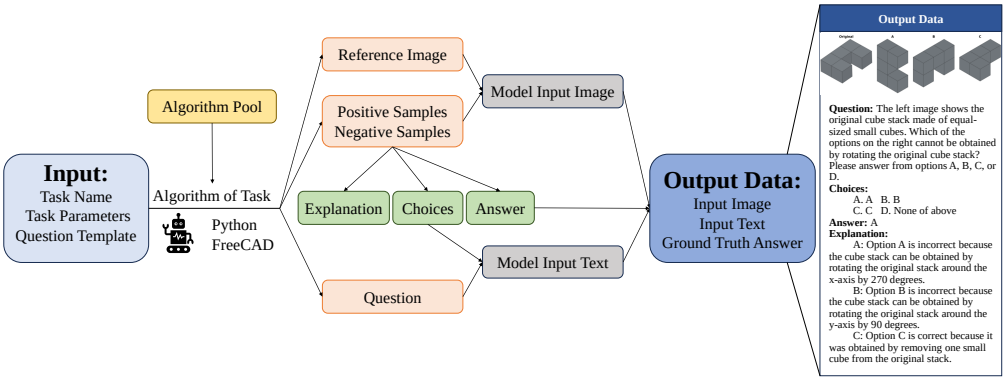


Figure 4: **The automated generation pipeline of a data instance.** We constructed the dataset using an automated generation system that integrates Python with FreeCAD, enabling precise control of difficulty, systematic generation of distractor options, and automatic recording of explanations for incorrect choices.

4.2 Benchmark Construction

The construction process of *SpatialViz-Bench* consists of two components: **automated generation and manual design**. For 11 tasks (i.e., all except the mechanical system task), we construct the dataset using an automated generation system that integrates Python with FreeCAD, as shown in Figure 4, enabling convenient expansion to new tasks. This automated generation method offers multiple advantages: 1) it introduces controlled randomness to increase problem diversity; 2) it enables precise control of difficulty parameters; 3) it systematically generates distractor options and automatically records explanations of incorrect options, providing deeper diagnostic information for model evaluation. For the mechanical system task, considering its complexity, we carefully select representative simulation examples from publicly available sources and manually design corresponding questions and answer options to ensure these problems accurately evaluate models' ability to understand the dynamic operational principles of mechanical systems. Figure 4 shows a benchmark data example, with additional examples available in Appendix C.

4.2.1 Mental Rotation

2D Rotation Task. A colored grid pattern with a red corner marker is rotated by $90°/180°/270°$ to generate positive samples. Negative samples involve horizontal/vertical mirroring. We further replace symmetric color fills with non-centrally symmetric patterns. Negatives include mirror flips and internal rotations of pattern components, increasing spatial reasoning difficulty. As shown in Algorithm 1.

3D Rotation Task. A connected cube stack is rotated along x/y/z axis to form positives. Negatives are created by removing one cube or mirroring the isometric view, ensuring no simple rotation can reproduce them. Spatial complexity is increased by enlarging assembly dimensions, requiring enhanced 3D rotational reasoning. As shown in Algorithm 2 and Algorithm 3.

Three-View Projection Task. This task has two categories. Firstly, given isometric, front, and top views of a connected cube stack with marked reference cubes, the task is to select the correct left view. Negatives involve altering reference cube positions or substituting the right view. We further introduce real engineering parts from the DeepCAD dataset [36], rendered into standard projections via FreeCAD. Negatives are crafted through random internal lines deletion, view flipping/rotation, or transformations on unseen views. As shown in Algorithm 4 and Algorithm 5.

4.2.2 Mental Folding

Paper Folding Task. A Python-based pipeline generates $m \times n$ grid patterns undergoing sequential folds (vertical/horizontal/diagonal), followed by hole-punching and unfolding. The task requires identifying the correct unfolded hole distribution. Negative samples are generated by mirroring, deleting, adding, or relocating holes to violate fold-induced symmetry. Task difficulty increases with more folds, larger grids, and denser hole placements. As shown in Algorithm 6 and Algorithm 7.

Cube Unfolding Task. Given a cube with six uniquely colored faces and a view from a corner (three visible faces), the task is to select the correct 2D net (11 possibilities as shown in Figure 5). Positives can be crafted either by using different cube nets of the same cube or by fixing the mapping of visible faces while randomly shuffling the remaining faces. Negatives are crafted by swapping visible face colors or flipping visible-opposite face pairs. We further replace solid colors with non-centrally symmetric patterns. View angles prioritize faces with asymmetric patterns. Internal rotations of pattern components are introduced to further increase the reasoning difficulty. To push the difficulty even further, all six faces feature random colored-dot patterns on a 3×3 grid. As shown in Algorithm 8, Algorithm 9 and Algorithm 10.

Cube Reconstruction Task. Cubes have six uniquely colored faces. Two task variants exist: (1) select the correct vertex view of a cube when given its net pattern, with negative samples created by mirroring the correct view; (2) identify the color of a face opposite to a given colored face. Difficulty progression follows the cube unfolding tasks. As shown in Algorithm 8 and Algorithm 11.

4.2.3 Visual Penetration

Cross-Section Task. Nine basic geometric solids (e.g., triangular/rectangular/circular prisms/pyramids/frustums) are combined in pairs with conical shapes on top. Cross-sections are generated by slicing the composite shapes using planes parallel to the XY/YZ/XZ planes. Negative samples are constructed by adjusting the relative geometric proportions within the composite. Task complexity is increased by introducing composites with three solids, which often produce disconnected cross-sections that demand enhanced visual reasoning. Additional complexity is introduced by generating oblique cross-sections at $45°/135°$. As shown in Algorithm 12.

Cube Counting Task. The task requires inferring the total cube count of a connected cube stack based on two orthogonal projection views. The minimum and maximum counts are mathematically derived to guide the construction of answer options. Constraints increase to three orthogonal projection views, reducing the number of possible solutions while increasing view integration complexity. Task difficulty further increases by expanding the spatial dimensions of the cubic assemblies. As shown in Algorithm 2 and Algorithm 13.

Cube Assembly Task. A pyramid-like cube stack is split into two connected parts. Tasks require identifying the complementary piece that fits the reference part. Negative samples are generated by modifying the correct piece through the addition or removal of cubic units. The difficulty is further increased by enlarging the spatial dimensions and dividing the structure into three parts instead of two. As shown in Algorithm 14 and Algorithm 15.

4.2.4 Mental Animation

Arrow Moving Task. For the easy version, an arrow with random initial position and orientation in a 3×3 grid operates by ego-centric rules: movement occurs in 4 directions (forward/backward/left/right), with "forward" always indicating the arrow's current orientation. The arrow reorients to the movement direction after each movement. Valid operation sequences are algorithmically generated; negative samples share the same initial state but yield incorrect endpoints. For the hard version, multiple colored arrows are introduced with extended rules: empty positions allow direct entry; occupied positions trigger object exchanges while maintaining Level 0 movement principles. Tasks include predicting final states from sequences, or inferring correct sequences from state pairs. As shown in Algorithm 16, Algorithm 17, Algorithm 18 and Algorithm 19.

Block Moving Task. Colored cube stack combines directional movement with gravity simulation. Cubes move along six directions with unsupported cubes falling until reaching support and swapping positions as same as Arrow Moving Task. Increased spatial complexity and longer sequences elevate reasoning difficulty. As shown in Algorithm 20 and Algorithm 21.

Mechanical System Task. We use open-source mechanical system simulations, classifying complexity by module quantity and designing appropriate questions. These tasks assess advanced mental animation abilities, particularly to understand how the motion of one component affects others.

Table 2: Comparison of open-source model performances. Tasks: 2D Rotation (2DR), 3D Rotation (3DR), Three-View Projection (3VP), Paper Folding (PF), Cube Unfolding (CU), Cube Reconstruction (CR), Cross-Section (CS), Cube Counting (CC), Cube Assembly (CA), Arrow Moving (AM), Block Moving (BM), Mechanical System (MS). The first and second highest accuracy of MLLMs are marked in red and blue.

Model	Overall	Mental Rotation				Mental Folding				Visual Penetration				Mental Animation			
		2DR	3DR	3VP	Avg	PF	CU	CR	Avg	CS	CC	CA	Avg	AM	BM	MS	Avg
Random	25.08	23.75	27.50	31.00	27.69	19.17	20.00	25.83	21.67	30.00	25.00	30.00	28.12	28.75	16.25	25.00	23.33
3B																	
SAIL-VL-1.5-2B	24.15	22.50	22.50	22.00	22.31	20.00	27.50	20.00	22.50	24.17	26.67	32.50	27.19	21.25	25.00	27.50	24.58
InternVL3-2B	26.19	16.25	33.75	31.00	27.31	22.50	25.83	25.00	24.44	20.00	30.83	30.00	26.56	18.75	32.50	30.00	27.08
Deepseek-VL2-tiny(3B)	21.36	17.50	22.50	27.00	22.69	21.67	20.83	19.17	20.56	20.83	22.50	18.75	20.94	18.75	21.25	25.00	21.67
Qwen2.5-VL-3B-Instruct	26.10	20.00	18.75	21.00	20.00	25.00	25.83	21.67	24.17	25.83	23.33	30.00	25.94	35.00	30.00	42.50	35.83
7B																	
Qwen2.5-VL-7B-Instruct	27.97	25.00	16.25	29.00	23.85	34.17	21.67	30.00	28.61	16.67	36.67	28.75	27.19	22.50	23.75	51.25	32.50
Qwen2.5-Omni-7B	27.29	22.50	20.00	29.00	24.23	25.00	27.50	20.00	24.17	20.83	33.33	27.50	27.19	31.25	30.00	45.00	35.42
LLaVA-OneVision-Qwen2-7B-ov-hf	27.29	31.25	18.75	29.00	26.54	21.67	25.00	23.33	23.33	18.33	33.33	32.50	27.50	21.25	40.00	40.00	33.75
SAIL-VL-1.6-8B	25.00	18.75	21.25	25.00	21.92	28.33	25.00	18.33	23.89	21.67	19.17	23.75	21.25	25.00	35.00	45.00	35.00
InternVL2.5-8B	26.69	26.25	27.50	36.00	30.38	28.33	28.33	20.83	25.83	15.83	28.33	26.25	23.12	20.00	22.50	43.75	28.75
InternVL3-8B	30.08	20.00	38.75	28.00	28.85	28.33	23.33	25.00	25.56	15.83	40.83	38.75	30.94	30.00	30.00	51.25	37.08
16B																	
Kimi-VL-A3B-Instruct(16B)	23.90	16.25	30.00	36.00	28.08	25.83	20.00	26.67	24.17	21.67	5.00	28.75	17.19	15.00	31.25	37.50	27.92
Kimi-VL-A3B-thinking(16B)	28.14	13.75	20.00	25.00	20.00	23.33	24.17	26.67	24.72	25.00	36.67	25.00	29.38	30.00	43.75	47.50	40.42
Deepseek-VL2-small(16B)	25.17	31.25	16.25	26.00	24.62	22.50	25.00	26.67	24.72	9.17	35.00	35.00	25.31	26.25	23.75	28.75	26.25
32B																	
Deepseek-VL2(27B)	28.31	25.00	33.75	30.00	29.62	31.67	25.00	22.50	26.39	18.33	39.17	28.75	28.75	26.25	30.00	31.25	29.17
Qwen2.5-VL-32B-Instruct	32.12	31.25	35.00	38.00	35.00	21.67	25.00	27.50	24.72	25.83	36.67	43.75	34.38	28.75	27.50	55.00	37.08
InternVL2.5-38B	28.56	28.75	31.25	29.00	29.62	24.17	23.33	30.83	26.11	24.17	26.67	31.25	26.88	21.25	26.25	52.50	33.33
InternVL3-38B	30.34	22.50	33.75	29.00	28.46	20.83	29.17	30.83	26.94	21.67	32.50	41.25	30.63	25.00	30.00	56.25	37.08
72B																	
Qwen2.5-VL-72B-Instruct	33.31	28.75	31.25	28.00	29.23	22.50	20.00	30.00	24.17	30.00	41.67	48.75	39.06	27.50	40.00	63.75	43.75
QvQ-72B-preview	28.14	21.25	30.00	31.00	27.69	16.67	19.17	27.50	21.11	30.00	22.50	32.50	27.81	25.00	50.00	43.75	39.58
LLaVA-OneVision-Qwen2-72B-ov-hf	28.73	30.00	30.00	40.00	33.85	22.50	18.33	29.17	23.33	23.33	33.33	40.00	31.25	27.50	17.50	38.75	27.92
InternVL2.5-78B	27.20	22.50	23.75	30.00	25.77	26.67	16.67	28.33	23.89	23.33	30.00	27.50	26.88	33.75	28.75	40.00	34.17
InternVL3-78B	29.75	25.00	25.00	34.00	28.46	19.17	25.00	22.50	22.22	20.83	40.00	48.75	35.00	23.75	41.25	41.25	35.42
108B																	
Llama-4-Maverick-17B-128E-Instruct	31.78	20.00	40.00	40.00	33.85	16.67	29.17	29.17	25.00	19.17	35.00	47.50	32.19	35.00	40.00	42.50	39.17
LLama-4-Scout-17B-16E-Instruct	34.24	32.50	35.00	43.00	37.31	16.67	32.50	36.67	28.61	17.50	37.50	53.75	34.06	28.75	40.00	50.00	39.58

Table 3: Comparison of closed-source model performances.

Model	Overall	Mental Rotation				Mental Folding				Visual Penetration				Mental Animation			
		2DR	3DR	3VP	Avg	PF	CU	CR	Avg	CS	CC	CA	Avg	AM	BM	MS	Avg
GPT-4o	31.10	32.50	27.50	33.00	31.15	29.17	15.83	30.00	25.00	19.17	40.83	40.00	32.50	22.50	32.50	60.00	38.33
o1	41.36	62.50	28.75	49.00	46.92	28.33	34.17	26.67	29.72	37.50	40.83	33.75	37.81	67.50	52.50	52.50	57.50
Claude-3.5-sonnet	32.54	31.25	25.00	45.00	34.62	20.83	22.50	31.67	25.00	22.50	35.83	46.25	33.44	37.50	31.25	52.50	40.42
Claude-3.7-sonnet	33.90	32.50	36.25	44.00	38.08	18.33	26.67	29.17	24.72	24.17	30.83	43.75	31.56	66.25	28.75	43.75	46.25
Gemini-1.5-pro	28.81	32.50	28.75	34.00	31.92	13.33	24.17	30.83	22.78	20.83	26.67	32.50	25.94	37.50	26.25	51.25	38.33
Gemini-2.5-flash	36.86	42.50	30.00	35.00	35.77	26.67	30.00	40.83	32.50	30.00	38.33	28.75	32.81	67.50	33.75	48.75	50.00
Gemini-2.5-pro	44.66	52.50	32.50	47.00	44.23	43.33	31.67	30.00	35.00	33.33	55.00	36.25	42.19	95.00	35.00	58.75	62.92
Doubao-1-5-vision-pro	33.31	7.50	35.00	45.00	30.38	31.67	23.33	29.17	28.06	30.00	55.83	30.00	39.69	22.50	37.50	47.50	35.83
Qwen-VL-max	32.03	23.75	26.25	33.00	28.08	24.17	17.50	31.67	24.44	26.67	47.50	42.50	38.44	26.25	36.25	55.00	39.17

5 Evaluation

5.1 Evaluation Setup

Models We conducted comprehensive experiments on a diverse range of MLLMs, including 9 closed-source models and 23 open-source models. For **closed-source MLLMs**, we evaluated models from five major providers, including GPT-4o [37] and o1 [38] for OpenAI series, Gemini-1.5-pro [39], Gemini-2.5-flash and Gemini-2.5-pro [40] for Gemini series, Claude-3.5-sonnet [41] and Claude-3.7-sonnet [42] for Claude series, Qwen-VL-max [43], and Doubao-1.5-vision-pro [44]. For **open-source MLLMs**, we assessed the capabilities of Qwen2.5-VL series [45], QvQ [46], Qwen-Omni [47], InternVL-2.5 series [48], InternVL-3 series [49], Deepseek-VL2 series [50], SAIL-VL series [51], Kimi-VL-A3B series [52], LLama-4 series [53], and LLaVA-OneVision series [54].

Setting To maintain consistency across evaluations, all experiments were conducted in a zero-shot setting using identical prompts designed to encourage models to explicitly output their reasoning processes before providing final answers. This approach allowed us to not only assess the accuracy of responses but also gain insights into the underlying reasoning mechanisms employed by different models across our benchmark tasks.

Metric Design To evaluate models handling multimodal inputs and generating textual outputs, with most options presented as images, we formatted all tasks as Multiple-Choice Answer (MCA) with one correct answer. Option and reference images were integrated into a unified visual input. Model performance was assessed using accuracy, based on the match between predicted and ground-truth answers. This standardized approach ensures consistent evaluation across tasks and enables fair comparison of multimodal understanding across models.

More details of evaluation setup are provided in Appendix D.

5.2 Evaluation Results

We compare leading MLLMs on SpatialViz-Bench, as shown in Table 2, Table 3 and Appendix E.1.

Heavily Vision-Dependent Long-Chain Reasoning Tasks In our benchmark, 11 out of 12 tasks use fixed templates with algorithmically generated data to ensure consistency and minimize randomness. The manually designed task aligns with the same design principles. As the textual input alone is insufficient, visual input is essential for problem-solving, making the benchmark highly vision-dependent. Most options are image-based, requiring precise visual analysis rather than simple matching, thereby increasing reasoning complexity. For both humans and MLLMs, these tasks demand multi-step spatial reasoning and mental transformations, exemplifying complex chain-of-thought processes.

Overall Performance Overall Performance. All evaluated models performed well below human level, underscoring the benchmark's difficulty. The top performers—Gemini-2.5-pro and o1—achieved accuracies of 44.66% and 41.36%, roughly twice the random baseline. Among open-source models, LLaMA-4-Scout and Qwen2.5-VL-72B-Instruct performed best, though their accuracy rates remained about 10% below those of the top closed-source models. The clear performance gaps indicate the benchmark's strong discriminative power. The results suggest a positive correlation between performance and model scale in most tasks, as well as a consistent improvement in spatial reasoning across newer model versions. These findings support current training strategies while highlighting significant room for improvement.

Task-level Performance Models with higher overall accuracy generally perform well across individual tasks. Most models show near-random accuracy on 3D Rotation, Cube Unfolding, and Cube Reconstruction, indicating common perceptual and visualization limitations. Both proprietary models perform well on the Arrow Moving task, with Gemini-2.5-pro even surpassing human performance, while most of open-source models perform at near-random levels. This suggests that, despite its relatively low visual complexity, the task requires advanced reasoning—such as understanding object-centered motion—which open-source models still lack. In most cases, model performance matched our expected difficulty levels, though some discrepancies with human perception offer valuable insights for refining task design and guiding future research. Additional evaluation results and task-specific analysis are provided in Appendix E.1.

Evaluation on our proposed SpatialViz-Bench reveals significant limitations in current MLLMs' *spatial visualization* capabilities, with all evaluated models performing substantially below human-level performance. The results demonstrate the benchmark's strong discriminative power and highlight the substantial room and urgent need for improvement in reasoning with *spatial visualization*, particularly for tasks demanding complex visual-spatial transformations and precise spatial localization.

5.3 Analysis of Test Cases

We randomly selected representative samples of Gemini-2.5-pro across various tasks (detailed in Appendix E.2). The model exhibited strong reasoning capabilities, often following logically coherent and complete processes, validating the effectiveness of our evaluation results. However, we still observed instances where the model provided correct answers despite flawed reasoning processes.

The *spatial memorization* process is represented through intermediate conclusions presented in textual format, which greatly reduces errors in *spatial memorization*. Deeper analysis revealed that most errors occurred primarily during perception and visualization phases rather than logical reasoning stages. Specifically, the model showed notable deficiencies in color recognition, complex pattern identification, and understanding relative positions, such as failing to accurately identify the structure of cube nets and symmetrical relationships between faces. The model demonstrated particular

limitations in 3D spatial understanding, struggling to accurately identify the quantity, position, and spatial relationships of stacked cubes.

Comparing performance across tasks, Gemini-2.5-pro achieved 95% accuracy on the Arrow Moving task, showing full understanding of object-centered motion while maintaining logical consistency throughout. However, when questions with similar formulations involved 3D space, accuracy dropped to 35%, primarily because the model could not correctly identify cube structures.

For Mechanical System tasks, we deliberately designed questions requiring only *spatial visualization* abilities for qualitative analysis. However, observations showed the model tended to use theoretical formula derivation, which differs significantly from how human subjects approach similar problems. This behavior ikely results from pre-training data where mechanical tasks predominantly required theoretical derivation, conditioning the model into a fixed reasoning pattern.

Based on this analysis, we conclude that models trained with rich pre-training knowledge have significantly improved in knowledge-dependent reasoning abilities but still exhibit clear deficiencies in perception and visualization capabilities. This finding provides a clear direction for subsequent model training optimization: strengthening *spatial perception* and *visualization* abilities.

6 Conclusion

We introduce ***SpatialViz-Bench***, a cognitive-science–inspired benchmark for testing *spatial visualization* in multimodal models. It comprises 12 tasks (1,180 problems) across 4 core abilities, mental rotation, folding, visual penetration and animation. Our results show strong discriminative power and appropriate challenge, revealing that current models are primarily limited by visuospatial information acquisition rather than logical reasoning—guiding future MLLM optimizations in spatial skills. We discuss our future work in Appendix F.

References

[1] Louis Leon Thurstone. Primary mental abilities: Psychometric monographs no. 1. In *The measurement of intelligence*, pages 131–136. Springer, 1938.

[2] Louis Leon Thurstone. Some primary abilities in visual thinking. *Proceedings of the American Philosophical Society*, 94(6):517–521, 1950.

[3] Sergio Della Sala, Colin Gray, Alan Baddeley, Nadia Allamano, and Lindsey Wilson. Pattern span: A tool for unwelding visuo–spatial memory. *Neuropsychologia*, 37(10):1189–1199, 1999.

[4] An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. Spatialrgpt: Grounded spatial reasoning in vision-language models. In *NeurIPS*, 2024.

[5] 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. *arXiv preprint arXiv:2412.14171*, 2024.

[6] Huanqia Cai, Yijun Yang, and Winston Hu. Mm-iq: Benchmarking human-like abstraction and reasoning in multimodal models. *arXiv preprint arXiv:2502.00698*, 2025.

[7] Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with math-vision dataset. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL `https://openreview.net/forum?id=QWTCcxMpPA`.

[8] Fangyu Liu, Guy Edward Toh Emerson, and Nigel Collier. Visual spatial reasoning. *Transactions of the Association for Computational Linguistics*, 2023.

[9] Amita Kamath, Jack Hessel, and Kai-Wei Chang. What's "up" with vision-language models? investigating their struggle with spatial reasoning. In *EMNLP*, 2023.

[10] 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 (CVPR)*, pages 14455–14465, June 2024.

[11] Fatemeh Shiri, Xiao-Yu Guo, Mona Far, Xin Yu, Reza Haf, and Yuan-Fang Li. An empirical analysis on spatial reasoning capabilities of large multimodal models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21440–21455, 2024.

[12] Jiahao Nie, Gongjie Zhang, Wenbin An, Yap-Peng Tan, Alex C Kot, and Shijian Lu. Mmrel: A relation understanding benchmark in the mllm era. *arXiv preprint arXiv:2406.09121*, 2024.

[13] Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. *arXiv preprint arXiv:2404.12390*, 2024.

[14] Chenglin Li, Qianglong Chen, Zhi Li, Feng Tao, and Yin Zhang. Vcbench: A controllable benchmark for symbolic and abstract challenges in video cognition. *arXiv preprint arXiv:2411.09105*, 2024.

[15] Wenyu Han, Siyuan Xiang, Chenhui Liu, Ruoyu Wang, and Chen Feng. Spare3d: A dataset for spatial reasoning on three-view line drawings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14690–14699, 2020.

[16] Christopher Beckham, Martin Weiss, Florian Golemo, Sina Honari, Derek Nowrouzezahrai, and Christopher Pal. Visual question answering from another perspective: Clevr mental rotation tests. *Pattern Recognition*, 136:109209, 2023.

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

[18] Wenrui Xu, Dalin Lyu, Weihang Wang, Jie Feng, Chen Gao, and Yong Li. Defining and evaluating visual language models' basic spatial abilities: A perspective from psychometrics. *arXiv preprint arXiv:2502.11859*, 2025.

[19] Marc H Bornstein. Frames of mind: The theory of multiple intelligences, 1986.

[20] Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pages 3195–3204, 2019.

[21] Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. A-okvqa: A benchmark for visual question answering using world knowledge. In *European conference on computer vision*, pages 146–162. Springer, 2022.

[22] Yunzhuo Hao, Jiawei Gu, Huichen Will Wang, Linjie Li, Zhengyuan Yang, Lijuan Wang, and Yu Cheng. Can mllms reason in multimodality? emma: An enhanced multimodal reasoning benchmark. *arXiv preprint arXiv:2501.05444*, 2025.

[23] Yifan Jiang, Jiarui Zhang, Kexuan Sun, Zhivar Sourati, Kian Ahrabian, Kaixin Ma, Filip Ilievski, and Jay Pujara. Marvel: Multidimensional abstraction and reasoning through visual evaluation and learning. *arXiv preprint arXiv:2404.13591*, 2024.

[24] Yijia Xiao, Edward Sun, Tianyu Liu, and Wei Wang. Logicvista: Multimodal llm logical reasoning benchmark in visual contexts, 2024. URL `https://arxiv.org/abs/2407.04973`.

[25] Yueqi Song, Tianyue Ou, Yibo Kong, Zecheng Li, Graham Neubig, and Xiang Yue. Visualpuzzles: Decoupling multimodal reasoning evaluation from domain knowledge, 2025. URL `https://arxiv.org/abs/2504.10342`.

[26] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*, 2023.

[27] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9556–9567, 2024.

[28] Ge Zhang, Xinrun Du, Bei Chen, Yiming Liang, Tongxu Luo, Tianyu Zheng, Kang Zhu, Yuyang Cheng, Chunpu Xu, Shuyue Guo, et al. Cmmmu: A chinese massive multi-discipline multimodal understanding benchmark. *arXiv preprint arXiv:2401.11944*, 2024.

[29] Zheqi He, Xinya Wu, Pengfei Zhou, Richeng Xuan, Guang Liu, Xi Yang, Qiannan Zhu, and Hua Huang. Cmmu: A benchmark for chinese multi-modal multi-type question understanding and reasoning. *arXiv preprint arXiv:2401.14011*, 2024.

[30] Tanik Saikh, Tirthankar Ghosal, Amish Mittal, Asif Ekbal, and Pushpak Bhattacharyya. Scienceqa: A novel resource for question answering on scholarly articles. *International Journal on Digital Libraries*, 23 (3):289–301, 2022.

[31] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017.

[32] Zhuowan Li, Xingrui Wang, Elias Stengel-Eskin, Adam Kortylewski, Wufei Ma, Benjamin Van Durme, and Alan L Yuille. Super-clevr: A virtual benchmark to diagnose domain robustness in visual reasoning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14963–14973, 2023.

[33] Leila Glass, Frank Krueger, Jeffrey Solomon, Vanessa Raymont, and Jordan Grafman. Mental paper folding performance following penetrating traumatic brain injury in combat veterans: a lesion mapping study. *Cerebral Cortex*, 23(7):1663–1672, 2013.

[34] Valerie K Sims and Mary Hegarty. Mental animation in the visuospatial sketchpad: Evidence from dual-task studies. *Memory & Cognition*, 25:321–332, 1997.

[35] Sarah Titus and Eric Horsman. Characterizing and improving spatial visualization skills. *Journal of Geoscience Education*, 57(4):242–254, 2009.

[36] Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6772–6782, October 2021.

[37] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[38] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[39] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[40] Google Deepmind. Gemini 2.5: Our most intelligent AI model, March 2025. URL `https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/`.

[41] Anthropic. Claude 3.5 Sonnet, . URL `https://www.anthropic.com/news/claude-3-5-sonnet`.

[42] Anthropic. Claude 3.7 Sonnet, . URL `https://www.anthropic.com/claude/sonnet`.

[43] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023.

[44] ByteDance. ByteDance Releases Doubao Large Model 1.5 Pro, Performance Surpassing GPT-4o and Claude3.5Sonnet. URL https://www.aibase.com/news/www.aibase.com/news/14931.

[45] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[46] Qwen Team. Qvq: To see the world with wisdom, December 2024. URL https://qwenlm.github.io/blog/qvq-72b-preview/.

[47] Jin Xu, Zhifang Guo, Jinzheng He, Hangrui Hu, Ting He, Shuai Bai, Keqin Chen, Jialin Wang, Yang Fan, Kai Dang, et al. Qwen2. 5-omni technical report. *arXiv preprint arXiv:2503.20215*, 2025.

[48] Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024.

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

[50] Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, et al. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*, 2024.

[51] Hongyuan Dong, Zijian Kang, Weijie Yin, Xiao Liang, Chao Feng, and Jiao Ran. Scalable vision language model training via high quality data curation. *arXiv preprint arXiv:2501.05952*, 2025.

[52] Kimi Team, Angang Du, Bohong Yin, Bowei Xing, Bowen Qu, Bowen Wang, Cheng Chen, Chenlin Zhang, Chenzhuang Du, Chu Wei, et al. Kimi-vl technical report. *arXiv preprint arXiv:2504.07491*, 2025.

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

[54] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.

Appendix

A Detailed Related Works **15**

A.1 Spatial Tasks in Vision-Language Understanding and Reasoning 15

A.2 Spatial Visualization Tasks in Existing Multimodal Reasoning Benchmarks 15

B Data Curation Details **16**

B.1 Automated Data Generation Pipeline 16

B.2 Manul Design for Mechanical System Task 16

C Data Examples **35**

D Evaluation Details **41**

D.1 Prompts for Response Generation 41

D.2 Models 41

D.3 Methods for Answer Extraction 41

D.4 Human Performance 41

E Detailed Results **42**

E.1 Intra-Category Comparisons Across Levels 42

E.2 Test Cases 48

F Future Work **62**

A Detailed Related Works

A.1 Spatial Tasks in Vision-Language Understanding and Reasoning

Spatial reasoning plays a foundational role in embodied intelligence, supporting tasks such as navigation, object interaction, and scene understanding. Early benchmarks primarily target perceptual-level spatial understanding—such as monocular depth estimation, object localization, and relative position classification—by extracting explicit geometric cues from visual inputs. With the rise of MLLMs, these capabilities are increasingly evaluated via visual question answering tasks. Datasets such as VSR [8] and What'sUp [9] benchmark models' comprehension of object-centric spatial relationships, while SpatialVLM [10], Spatial-MM [11], and MMRel [12] further expand evaluation to include relative distances, camera-object perspectives, and object size comparisons. These tasks mostly emphasize recognition rather than complex reasoning.

Recent efforts have introduced more cognitively demanding benchmarks. Blink [13] includes a Multi-view Reasoning task to test rotational understanding from both object and camera viewpoints. SpatialRGPT-bench [4] advances spatial QA by incorporating world knowledge and multi-hop reasoning. Meanwhile, video-based benchmarks have begun to assess spatial memory in dynamic contexts. VCBench [14] evaluates this through tasks such as Flash Grid and 3D Navigator, which test a model's ability to retain 2D spatial positions and predict trajectories in 3D space. VSI-bench [5] focuses on egocentric-to-allocentric transformation and perspective-shifting abilities, targeting spatial reasoning skills essential for direction judgment and route planning.

Despite these advances, spatial visualization—defined as mentally manipulating shapes in space—remains underexplored. Only a few benchmarks, such as SPARE3D [15] and CLEVR-MRT [16], offer mental rotation or multi-view tasks, and these are limited in diversity and MLLM alignment. Recent research works like Stogiannidis et al. [17] and Xu et al. [18]—include spatial visualization as a subskill of spatial ability without further categorization, with the former completely collecting data from online psychological tests and the latter using only paper folding tasks to assess this ability, limiting the evaluation scope.

A.2 Spatial Visualization Tasks in Existing Multimodal Reasoning Benchmarks

As interest in systematically evaluating MLLMs' reasoning capabilities grows, existing multimodal reasoning benchmarks can be grouped into 4 categories: (1) world knowledge reasoning (e.g., OK-VQA [20], A-OKVQA [21]), (2) mathematical/logical reasoning (e.g., MathVista [26], Math-Vision [7], MM-IQ [6]), (3) domain-specific reasoning (e.g., MMMU [27], CMMMU [28], CMMU [29], ScienceQA [30]), and (4) spatial reasoning (e.g., CLEVR [31], Super-CLEVR [32], SpatialRGPT [4]).

Among them, spatial visualization tasks are often treated as subcategories under mathematical and logical reasoning benchmarks, such as 3D-Geometry category in MM-IQ [6] and MARVEL [23], the 3D Spatial Simulation category in EMMA [22], the 3D Shapes category in LogicVista [24], the IQ-Test category in Blink [13], and the Descriptive Geometry and Transformation Geometry categories in Math-Vision [7], Spatial Reasoning category in VisualPuzzles [25]. While these subcategories contribute to spatial reasoning evaluation, they are typically limited in scope and do not offer a comprehensive framework specifically tailored to spatial visualization, particularly in capturing the breadth and depth of this cognitive ability. In addition, these tasks primarily serve to evaluate whether models can solve such problems, rather than to guide the development of models specifically toward addressing spatial reasoning challenges.

In contrast, our benchmark is designed around 4 core sub-skills of spatial visualization identified in cognitive psychology, with curated tasks targeting each ability. This leads to a more structured and comprehensive evaluation protocol. Moreover, while prior benchmarks often rely on internet-sourced questions—raising issues of coverage and data leakage—our benchmark employs algorithmic generation for most tasks. This approach ensures greater reliability, reduces training-set overlap, and enables scalable data creation for both evaluation and training in spatial reasoning. More importantly, we summarized the essential phases to complete spatial visualization tasks, enabling us to identify the causes of reasoning errors.

B Data Curation Details

B.1 Automated Data Generation Pipeline

FreeCAD, an open-source Computer-Aided Design (CAD) software, provides deep integration with Python programming language, enabling parametric model construction through programming. We leveraged the synergy between FreeCAD and Python to successfully automate the generation of 9 spatial visualization tasks: 2DRotation, 3DRotation, 3ViewProjection, CubeFolding, CubeReconstruction, CrossSection, CubeCounting, CubeAssembly, and BlockMoving. Additionally, two tasks—PaperFolding and ArrowMoving—were implemented solely using Python. For the MechanicalSystem task, due to its complexity and specific requirements, we employed precise manual design methods. To supplement the task overview presented in Section 4.2, the following sections provide detailed pseudocode for each automatically generated task, offering more systematic and in-depth technical insights.

Mental Rotation Tasks. Algorithm 1 presents the pseudocode for the 2D Rotation Task. For the 3D Rotation Task, Three-View Projection Task, Cube Counting Task, and Block Moving Task, we need to construct connected cube stacks, with the core functions detailed in Algorithm 2. Algorithm 3 demonstrates the complete implementation process of the 3D Rotation Task. The method for generating three-view projections of marked cube stacks is elaborated in Algorithm 4. Algorithm 5 describes the process of importing models from the DeepCAD dataset and generating their three-view projections.

Mental Folding Tasks. Algorithm 6 implements a Paper class for simulating the dynamic processes of paper folding, holes punching, and unfolding. Based on this simulation framework, Algorithm 7 constructs the data for the Paper Folding Task. Algorithm 8 presents the core functions for transforming 11 standard cube nets (as shown in Figure 5) into three-dimensional cubes. Utilizing these transformation functions, while Algorithm 9 demonstrates how different unfolding patterns can produce the same cube.Algorithm 10 and Algorithm 11 provide the complete pseudocode implementations for the Cube Unfolding Task and Cube Reconstruction Task, respectively.

Visual Penetration Tasks. Algorithm 12 details the implementation pseudocode for the Cross-Section Task. Algorithm 13 comprehensively presents the data generation procedure as well as the mathematical calculation process to guide the construction of answer options in the Cube Counting Task. Algorithm 14 contains the core functions for decomposing a complete cube stack into multiple connected parts. Building upon these functions, Algorithm 15 provides the complete construction pseudocode for the Cube Assembly Task.

Mental Animation Tasks. Algorithm 16 implements an ArrowPath class for simulating the movement process of an arrow centered on itself. Algorithm 17 implements an ArrowMap class that inherits from the ArrowPath class, designed to simulate movement and exchange operations in multi-arrow environments. Based on the ArrowPath class, Algorithm 18 details the data construction process for the single-arrow version of the Arrow Moving Task. Correspondingly, using the ArrowMap class, Algorithm 19 elucidates the data construction process for the multi-arrow version of the Arrow Moving Task. Algorithm 20 implements a Block class for simulating the movement and exchange processes of blocks that follow gravitational rules. Building upon this Block class, Algorithm 21 presents the complete pseudocode implementation of the Block Moving Task.

B.2 Manul Design for Mechanical System Task

We collected simulation materials from open-source platforms and recruited human annotators to label the data following a format similar to automatically generated data. Additional annotators independently reviewed the annotations before discussion and reaching consensus, ultimately producing 80 validated mechanical system data samples.

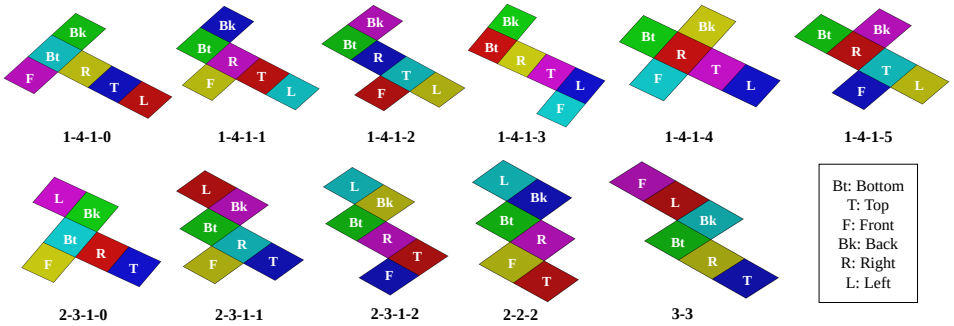


Figure 5: The eleven unfolded patterns of a cube with their corresponding numbered names. Assuming the square in row 1, position 0 represents the bottom face, and position 1 represents the right face, the corresponding arrangement of the remaining faces can be determined, facilitating the rotation of the square faces.

Algorithm 1 2D Rotation Task

Input: Color(Pattern) set C, grid size (H, W), unit length s, marker length s', task mode m
Initialize binary matrix $M \in \{0,1\}^{H \times W}$ with random values
Initialize empty lists $positive\_samples$, $negative\_samples$
function DRAWGRIDWITHMARKER($M,\ C,\ H,\ W,\ s,\ s',\ record = list()$)
    for $i \leftarrow 0$ to $H-1$ **do**
        for $j \leftarrow 0$ to $W-1$ **do**
            $pos \leftarrow (j \cdot s,\ (H-1-i) \cdot s,\ 0)$
            $square \leftarrow$ `FreeCAD.makePlane`$(s,\ s,\ (pos, 0°))$
            if $M[i][j] = 1$ **then**
                if record is empty **then**:
                    Randomly select $c \in C$ and assign c to $square$ at pos
                    Append c to $record$
                else
                    Assign `rotate`(`Pop`$(record, 0),\ 90°)$ to $square$ at pos
                end if
            end if
        end for
    end for
    Randomly select $corner \in$ {"top_left", "top_right", "bottom_left", "bottom_right"}
    $pos_{\text{marker}} \leftarrow$ `get_marker_pos`$(H,\ W,\ s,\ s',\ corner)$
    `FreeCAD.makePlane`$(s',\ s',\ (pos_{\text{marker}}, 0°))$ with red color
    $img \leftarrow$ `FreeCAD.saveImage()`
    return $img, record$
end function
$ref\_img, record \leftarrow$ `DrawGridWithMarker`(M, C, H, W, s, s')
if $m =$ "pattern" **then**
    $transform\_image, record \leftarrow$ `DrawGridWithMarker`$(M, C, H, W, s, s', record)$
    Append $transform\_img$ to $negative\_samples$
end if
for $angle \in \{90°, 180°, 270°\}$ **do**
    $img \leftarrow$ `rotate`$(ref\_img, angle)$
    Append img to $positive\_samples$
end for
for $flip\_dir \in$ {"horizontal", "vertical"} **do**
    $img \leftarrow$ `flip`$(ref\_img, flip\_dir)$
    Append img to $negative\_samples$
end for
$samples \leftarrow (positive\_samples, negative\_samples)$
Shuffle $samples$ to assign $[A, B, C, D]$ and record $answer\_id$
$data \leftarrow$ `create_data`$(ref\_img, samples, question, answer\_id)$

Algorithm 2 Fucntions for Creating Cubes with None-isolated Regions

Input: Spatial size (X, Y, Z), cube size s
Initialize zero value 3D tensors $placement \in \{0\}^{Z \times Y \times X}$, empty list $cubes$
function CREATECUBE($x,\ y,\ z$)
    $cube \leftarrow$ `FreeCAD.makebox`$(s, s, s, (x, y, z))$ and append $cube$ to $cubes$
    $placement[z][y][x] \leftarrow 1$
end function
function CREATECUBES($X,\ Y,\ Z$)
    for $z \leftarrow 0$ to $Z-1$ **do**
        for $y \leftarrow 0$ to $Y-1$ **do**
            for $x \leftarrow 0$ to $X-1$ **do**
                if $z = 0$ or $placement\_space[z-1][y][x] = 1$ **then**
                    With 50% probability `CreateCube`(x, y, z)
                end if
            end for
        end for
    end for
end function
function CONNECTISOLATEDCUBES($X,\ Y$)
    $cubes_{xy} \leftarrow \{(x, y) \mid placement[0][y][x] = 1\}$
    Initialize empty set $visited$, empty list $regions$
    $directions \leftarrow$ [(-1,0),(1,0),(0,-1),(0,1),(-1,-1),(-1,1),(1,-1),(1,1)]
    for all $(x, y) \in cubes_{xy}$ **do**
        if $(x, y) \notin visited$ **then**
            Initialize empty list $region$, empty queue $queue$
            Add (x, y) to $visited$, add (x, y) to $queue$
            while $queue$ is not empty **do**
                $(cx, cy) \leftarrow$ `popLeft`$(queue)$
                Append (cx, cy) to $region$
                for all $(dx, dy) \in directions$ **do**
                    $(nx, ny) \leftarrow (cx + dx, cy + dy)$
                    if $0 \le nx < X$ and $0 \le ny < Y$ and $(nx, ny) \notin visited$ and $placement[0][ny][nx] = 1$ **then**
                        Add (nx, ny) to $visited$, add (nx, ny) to $queue$
                    end if
                end for
            end while
            Append $region$ to $regions$
        end if
    end for
    if $|regions| > 1$ **then**
        for $i \leftarrow 0$ to $|regions| - 2$ **do**
            Find $(x_1, y_1), (x_2, y_2)$ with min L_1 distance between $regions[i]$ and $regions[i + 1]$
            $x \leftarrow x_1,\ y \leftarrow y_1$
            while $(x \neq x_2)$ or $(y \neq y_2)$ **do**
                if $x \neq x_2$ and $y \neq y_2$ **then**
                    $x \leftarrow x \pm 1,\ y \leftarrow y \pm 1$
                else if $x \neq x_2$ **then**
                    $x \leftarrow x \pm 1$
                else if $y \neq y_2$ **then**
                    $y \leftarrow y \pm 1$
                end if
                if $placement\_space[0][y][x] = 0$ **then**
                    `CreateCube`$(placement, x, y, 0)$
                end if
            end while
        end for
    end if
end function

Algorithm 3 3D Rotation Task

Input: Spatial size (X, Y, Z), cube size s
Initialize zero value 3D tensors $placement \in \{0\}^{Z \times Y \times X}$, empty list $cubes$
Initialize empty lists $positive\_samples$, $negative\_samples$
Update $placement, cubes$ with `CreateCubes`$(X,\ Y,\ Z)$
Update $placement, cubes$ with `ConnectIsolatedCubes`$(X,\ Y)$
$ref\_img \leftarrow$ `FreeCAD.saveImage`$(cubes)$

for $i \leftarrow 1$ to 4 **do**
    Randomly select $axis \in \{x, y, z\}$ and $angle \in \{90°, 180°, 270°\}$
    $rotated\_cubes \leftarrow$ `rotate`$(cubes,\ axis,\ angle)$
    $rotated\_img \leftarrow$ `FreeCAD.saveImage`$(rotated\_cubes)$
    Append $rotated\_img$ to $positive\_samples$
end for

$cubes' \leftarrow$Randomly remove a cube from $cubes$ and rotate the left cubes as above
$rotated\_removed\_img \leftarrow$ `FreeCAD.saveImage`$(cubes')$
Append $rotated\_removed\_img$ to $negative\_samples$

for $flip\_dir \in$ {"horizontal", "vertical"} **do**
    Randomly choose $sample$ from $positive\_samples$
    $img \leftarrow$ `flip`$(sample, flip\_dir)$
    Append img to $negative\_samples$
end for

$samples \leftarrow (positive\_samples, negative\_samples)$
Shuffle $samples$ to assign $[A, B, C, D]$ and record $answer\_id$
$data \leftarrow$ `create_data`$(ref\_img, samples, question, answer\_id)$

Algorithm 4 Three-View Projection Task with Marked Cube Stack

Input: Spatial size (X, Y, Z), cube size s
Initialize zero value 3D tensors $placement \in \{0\}^{Z \times Y \times X}$, empty list $cubes$
Initialize empty lists $positive\_samples$, $negative\_samples$
Update $placement, cubes$ with `CreateCubes`$(X,\ Y,\ Z)$
Update $placement, cubes$ with `ConnectIsolatedCubes`$(X,\ Y)$

function ColorVisibleFaces($X,\ Y,\ Z,\ colored\_num$)
    $cubes \leftarrow$ Find cubes that can be seen from front or top or left view
    Randomly color `min`$(colored\_num, |cubes|)$ cubes in red
end function

function SaveViews($cubes$)
    Initialize empty list $views$
    for all $view \in$ {"Isometric", "Top", "Front", "Left"} **do**
        $img \leftarrow$`FreeCAD.saveView`$(view)$ and append img to $views$
    end for
    return $views$
end function

Update $cubes$ with `ColorVisibleFaces`$(X, Y, Z, colored\_num)$
$views \leftarrow$ `SaveViews(`$cubes$`)`
Select $left\_view$ from $views$ to $positive\_samples$
Select $right\_view$ from $views$ to $negative\_samples$

Cleaer all colors and update $cubus$ with `ColorVisibleFaces`$(X, Y, Z, colored\_num)$ as above
$new\_views \leftarrow$ `SaveViews(`$cubes$`)`
Select $left\_view$ and $right\_view$ from new_views to $negative\_samples$

$samples \leftarrow (positive\_samples, negative\_samples)$
Shuffle $samples$ to assign $[A, B, C, D]$ and record $answer\_id$
$ref\_img \leftarrow (isometric\_view,\ top\_view,\ front\_view)$
$data \leftarrow$ `create_data`$(ref\_img, samples, question, answer\_id)$

Algorithm 5 Three-View Projection Task with Models from DeepCAD Datasets

Input: step file path pth
Initialize empty lists $positive\_samples$, $negative\_samples$
$shape \leftarrow$`Open`(pth)
$views \leftarrow$ `SaveViews(`$shape$`)`
function CREATEINCORRECTVIEW($view,\ mode$)
    if $mode = 0$ **then**
        $img' \leftarrow$Extract all internal lines and randomly delete 1 line
    else if $mode = 1$ **then**
        $img' \leftarrow$`rotate`$(view, 90°)$
    else if $mode = 2$ **then**
        $img' \leftarrow$ `flip`($view$, "horizontal" or "vertical')
    end if
    return img'
end function
$ref_view \leftarrow$Choose view from $views$ with max area
$(questioned\_view, other\_view) \leftarrow$ Randomly assign $views$ except for ref_view
Append $questioned\_view$ to $positive\_samples$
for $mode \leftarrow 0$ to 2 **do**
    $incorrect\_view \leftarrow$`CreateIncorrectView`($questioned\_view$ or $other\_view$, $mode$)
    Append $incorrect\_view$ to $negative\_samples$
end for
$samples \leftarrow (positive\_samples, negative\_samples)$
Shuffle $samples$ to assign $[A, B, C, D]$ and record $answer\_id$
$ref\_img \leftarrow (isometric\_view,\ top\_view,\ front\_view)$
$data \leftarrow$ `create_data`$(ref\_img, samples, question, answer\_id)$

Algorithm 6 Simulation for Paper Folding, Punching and Unfolding

Class Paper
Attributes:
    $grid$, $complete\_grid$: 2D arrays representing current and complete paper states
    $original\_rows$, $original\_cols$: initial dimensions
    $current\_rows$, $current\_cols$: current dimensions after folding
    $folds$: list of fold operations
function FOLD($direction$, $line$ or $diagonal\_points$)
    if $direction$ is horizontal **then**
        Calculate folded area
        Update $complete\_grid$ by marking folded area as -1
        Create new grid with updated dimensions
    else if $direction$ is vertical **then**
        Similar to horizontal but for columns
    else if $direction$ is diagonal **then**
        Calculate diagonal line equation
        Mark appropriate triangular area as -1
    end if
    Record fold operation in $folds$
end function

function PUNCH($points$)
    for each (x, y) in $points$ **do**
        Set $grid[x][y] \leftarrow 1$
        Set corresponding $complete\_grid$ position to 1
    end for
    Record punch operation in $folds$
end function

function UNFOLD
    for each $fold$ in reverse $folds$ **do**
        if $fold$ is horizontal **then**
            Mirror grid about fold line
        else if $fold$ is vertical **then**
            Mirror grid about fold line
        else if $fold$ is diagonal **then**
            Mirror grid about diagonal line
        end if
        Update current dimensions of paper
    end for
    Clear $folds$ list
end function

function CREATEINCORRECTVIEW($mode$)
    Create incorrect variant by:
    if $mode =$ "row" **then**
        Either remove a row of holes, add extra row, or swap rows
    else if $mode =$ "col" **then**
        Either remove a column of holes, add extra column, or swap columns
    else
        Combine row and column errors
    end if
    Update $paper$ with above changes
end function

Algorithm 7 Paper Folding Task

Input: Dimensions of paper $(rows, cols)$, number of folds $steps$, number of holes $punches$
Initialize $paper$ with dimensions $rows \times cols$
Initialize empty lists ref_imgs, $positive\_samples$, $negative\_samples$
for $step \leftarrow 1$ to $steps$ **do**
    if $step = steps$ **then**
        $direction \leftarrow$ "diagonal"
    else
        $direction \leftarrow$ Randomly select $direction \in$ ["horizontal","vertical"]
    end if
    if $direction =$ "horizontal" **then**
        $line \leftarrow \texttt{randomInt}(1, paper.current\_rows - 1)$
        $paper.\texttt{Fold}(direction, line)$
    else if $direction =$ "vertical" **then**
        $line \leftarrow \texttt{randomInt}(1, paper.current\_cols - 1)$
        $paper.\texttt{Fold}(direction, line)$
    else if $direction =$ "diagonal" **then**
        $diagonal\_points \leftarrow$ Randomly select one set of 45-degree line endpoints
        $paper.\texttt{Fold}(direction, diagonal\_points)$
    end if
    $img \leftarrow \texttt{draw\_paper}(paper)$ and append img to ref_imgs
end for

$points \leftarrow$ Randomly select $punches$ zero positions
$paper.\texttt{Punch}(points)$
$img \leftarrow \texttt{draw\_paper}(paper)$ and append img to ref_imgs

$paper.\texttt{Unfold}()$
$img \leftarrow \texttt{draw\_paper}(paper)$ and append img to $positive\_samples$

Initialize $paper'$ with same dimensions as $paper$
$paper'.grid \leftarrow paper.grid$ to copy the state of unfolded paper
Determine the incorrect view $mode$
for $i \leftarrow$1 to 3 **do**
    Update $paper'$ with $paper'.\texttt{CreateIncorrectView}(mode)$
    $img \leftarrow \texttt{draw\_paper}(paper')$ and append img to $negative\_samples$
end for

$samples \leftarrow (positive\_samples, negative\_samples)$
Shuffle $samples$ to assign $[A, B, C, D]$ and record $answer\_id$
$data \leftarrow \texttt{create\_data}(ref\_imgs, samples, question, answer\_id)$

Algorithm 8 Functions for Reconstruting Cube from 11 Kinds of Cube Nets

Input: cube size s
Define rotation operators:
  $R_x(\theta)$: Rotation about X-axis by θ degrees
  $R_y(\theta)$: Rotation about Y-axis by θ degrees
  $R_z(\theta)$: Rotation about Z-axis by θ degrees
function NET2CUBE($plane\_name,\ map,\ view,\ rot$)
  Initialize placement dictionary $planes$
  $planes$["Top"] $\leftarrow ((s/2, s/2, s),\ R_y(180°))$
  $planes$["Bottom"] $\leftarrow ((s/2, s/2, 0),\ Rx(0)$
  $planes$["Right"] $\leftarrow ((s, s/2, s/2),\ R_y(-90°))$
  $planes$["Left"] $\leftarrow ((0, s/2, s/2),\ R_y(90°) \circ R_z(90°))$
  $planes$["Back"] $\leftarrow ((s/2, s, s/2),\ R_x(90°))$
  if $plane\_name$ is "2-2-2" **then**
    $planes$["Top"] $\leftarrow (s/2, s/2, s),\ R_x(180°) \circ R_z(-90°)$
  else if $plane\_name$ is "1-4-1" **then**
    $planes$["Left"] $\leftarrow (0, s/2, s/2),\ R_y(90°)\circ$
  end if
  if $plane\_name \in$ ["1-4-1-0", "2-3-1-0"] **then**
    $planes$["Front"] $\leftarrow ((s/2, 0, s/2),\ R_x(-90°))$
  else if $plane\_name \in$ ["1-4-1-1", "1-4-1-4", "2-3-1-1", "2-2-2"] **then**
    $planes$["Front"] $\leftarrow ((s/2, 0, s/2),\ R_x(-90°) \circ R_z(-90°))$
  else if $plane\_name \in$ ["1-4-1-2", "1-4-1-5", "2-3-1-2", "3-3"] **then**
    $planes$["Front"] $\leftarrow ((s/2, 0, s/2),\ R_x(-90°) \circ R_z(180°))$
  else if $plane\_name$ is "1-4-1-3" **then**
    $planes$["Front"] $\leftarrow ((s/2, 0, s/2),\ R_x(-90°) \circ R_z(90°))$
  end if
  if $plane\_name \in$ ["1-4-1-4", "1-4-1-5"] **then**
    $planes$["Back"] $\leftarrow ((s/2, s, s/2),\ R_x(90°) \circ R_z(90°))$
  end if
  Form a cube by:
  for all $face\_name \in planes$ **do**
    $placement \leftarrow planes[face\_name]$
    $square \leftarrow$ `FreeCAD.makePlane`$(s,\ s,\ placement)$
    $c \leftarrow map[face\_name]$
    if rot is true **then**
      Assign `rotate`$(c,\ 90°)$ to $square$ at $placement$
    else
      Assign c to $square$ at $placement$
    end if
  end for
  $img \leftarrow$ `FreeCAD.saveView`$(view)$
  return img
end function
function DRAWNET($net,\ map,\ s,\ rot$)
  for $face\_name \in net$ **do**
    $i,\ j \leftarrow net[face\_name]$
    $pos \leftarrow (j \cdot s,\ (H - 1 - i) \cdot s,\ 0)$
    $square \leftarrow$ `FreeCAD.makePlane`$(s,\ s,\ (pos, 0°))$
    $c \leftarrow map[face\_name]$
    if rot is true **then**
      Assign `rotate`$(c,\ 90°)$ to $square$ at pos
    else
      Assign c to $square$ at pos
    end if
  end for
  $img \leftarrow$ `FreeCAD.saveImage`()
  return img
end function

Algorithm 9 Functions for Unfolding Cube to 11 kinds of Cube Nets

Using the same parameter definitions as those in Algorithm 8
function DRAWNETWIPIVOT($plane\_name, net, map, s, rot$)
    $pivot\_plane\_name \leftarrow$ "1-4-1-0"
    Initialize rotation dictionary $planes$
    if $plane\_name \in$ ["1-4-1-1", "1-4-1-4", "2-3-1-1", "2-2-2"] **then**
        $planes$["Front"] $\leftarrow R_z(90°)$)
    else if $plane\_name \in$ ["1-4-1-2", "1-4-1-5", "2-3-1-2", "3-3"] **then**
        $planes$["Front"] $\leftarrow R_z(-180°)$)
    else if $plane\_name$ is "1-4-1-3" **then**
        $planes$["Front"] $\leftarrow R_z(-90°)$)
    end if
    if $plane\_name \in$ ["1-4-1-4", "1-4-1-5"] **then**
        $planes$["Back"] $\leftarrow R_z(-90°)$)
    end if
    if $plane\_name \in$ ["2-3-1-0", "2-3-1-1", "2-3-1-2", "3-3", "2-2-2"] **then**
        $planes$["Left"] $\leftarrow R_z(-90°)$)
    end if
    if $plane\_name$ is "2-2-2" **then**
        $planes$["Top"] $\leftarrow R_z(-90°)$)
    end if
    Create a net which can form the same cube with pivot plane:
    for $face\_name \in net$ **do**
        $i,\ j \leftarrow net[face\_name]$
        $pos \leftarrow (j \cdot s,\ (H-1-i) \cdot s,\ 0)$
        $square \leftarrow$`FreeCAD.makePlane`$(s,\ s,\ (pos, 0°))$
        if rot is true **then**
            Assign `rotate`$(c,\ 90°)$ to square at pos
        else
            Assign c to $square$ at pos
        end if
        if $plane\_name \neq$ "1-4-1-0" **then**
            if $face\_name \in planes$ **then**
                $rotation \leftarrow planes[face\_name]$
                $square.Placement.Rotation \leftarrow rotation$
            end if
        end if
    end for
    $img \leftarrow$ `FreeCAD.saveImage`()
end function

Algorithm 10 Cube Unfolding Task

Input: Color(Pattern) set C, unit length s, task mode m
Initialize 11 cube nets
$nets : \{face\_name : (i, j) | face\_name \in \{\text{“Top”, “Bottom”, “Right”, “Left”, “Back”, “Front”}\}\}$
Initialize empty lists $positive\_samples$, $negative\_samples$
$map : \{face\_name : c | c \in C\}$ ←Randomly shuffle set C and assign it to six faces
Randomly select a $view$ ∈8 corner views of a cube
$pivot\_net\_name$ ← “1-4-1-0”
ref_img ←`Net2Cube`$(pivot\_net\_name, map, view, rot = \text{false})$
for $i \leftarrow 1$ to 2 **do**
    $plane\_name, net$ ← Randomly select net from $nets$
    img ←`DrawNetWiPivot`$(plane\_name, net, map, s, rot = \text{false})$
    Append img to $positive\_samples$
    if $m =$ “pattern” **then**
        img' ←`DrawNetWiPivot`$(plane\_name, net, map, s, rot = \text{true})$
        Append img' to $negative\_samples$
    end if
end for
map' ←Fix the mapping of $face\_name \in view$, and random shuffle the others
for $i \leftarrow 1$ to 2 **do**
    $plane\_name, net$ ← Randomly select net from $nets$
    img ←`DrawNetWiPivot`$(plane\_name, net, map, s, rot = \text{false})$
    Append img to $positive\_samples$
end for
map' ←Swap the colors(patterns) of a randomly selected $face \in view$ with its opposite face
$plane\_name, net$ ← Randomly select net from $nets$
img ←`DrawNetWiPivot`$(plane\_name, net, map', s, rot = \text{false})$
Append img to $negative\_samples$
$samples \leftarrow (positive\_samples, negative\_samples)$
Shuffle $samples$ to assign $[A, B, C, D]$ and record $answer\_id$
$data \leftarrow$ `create_data`$(ref\_img, samples, question, answer\_id)$

Algorithm 11 Cube Reconstruction Task

Input: Color(Pattern) set C, unit length s, task mode m
Initialize 11 cube nets
$nets : \{face\_name : (i, j) | face\_name \in \{\text{“Top”, “Bottom”, “Right”, “Left”, “Back”, “Front”}\}\}$
Initialize empty lists $positive\_samples$, $negative\_samples$
$map : \{face\_name : c | c \in C\}$ ←Randomly shuffle set C and assign it to six faces
$net \in \{0, 1\}^{3\times5}$ ← Randomly select net from $nets$
ref_img ←`DrawNet`$(net, map, s, rot = \text{false})$ and append img to $positive\_samples$
for $i \leftarrow 1$ to 3 **do**
    $view$ ← Randomly select a view from 8 corner views of a cube
    img ←`Net2Cube`$(net, map, view, rot = \text{false})$
    Append img to $positive\_samples$
end for
for $flip\_dir \in \{\text{“horizontal”, “vertical”}\}$ **do**
    Randomly choose $sample$ from $positive\_samples$
    $img \leftarrow$ `flip`$(sample, flip\_dir)$
    Append img to $negative\_samples$
end for
$samples \leftarrow (positive\_samples, negative\_samples)$
Shuffle $samples$ to assign $[A, B, C, D]$ and record $answer\_id$
$data \leftarrow$ `create_data`$(ref\_img, samples, question, answer\_id)$

Algorithm 12 Cross-Section Task

Input: Number of objects num, number of sections per mode k, whether rotate the slicing plane rot
Initialize candidate objects list $objects$, empty list $selected\_objects$
Initialize empty lists $positive\_samples$, $negative\_samples$
function GETSECTIONS($compound,\ k,\ plane$)
    Initialize empty list $imgs$
    Determine $coord_{min}$ and $coord_{max}$ from bounding box
    $step \leftarrow (coord_{max} - coord_{min})/(k+1)$
    for $i \leftarrow 1$ to k **do**
        $offset \leftarrow coord_{min} + i \times step$
        $normal\_vector \leftarrow$ unit vector normal to $plane$
        $section \leftarrow$ `FreeCAD.slice`($compound,\ normal\_vector,\ offset$)
        Rotate $section$ for better visualization
        $img \leftarrow$ `FreeCAD.savaImage`($section$) and append img to $imgs$
    end for
    return $imgs$
end function
function GETROTATEDSECTIONS($compound,\ axis,\ center$)
    $axis\_vector \leftarrow$ Corresponding unit vector of $axis$
    $plane \leftarrow$ Parallel to $axis$
    for $angle \in \{45°, 135°\}$ **do**
        $axix\_vector' \leftarrow$ `rotate`($axis\_vector, angle, plane$)
        $offset \leftarrow axix\_vector \cdot center$
        $section \leftarrow$ `FreeCAD.slice`($compound,\ axis\_vector,\ offset$)
        Rotate $section$ for better visualization
        $img \leftarrow$ `FreeCAD.savaImage`($section$) and append img to $imgs$
    end for
    return $imgs$
end function
$selected_objects \leftarrow$Randomly select num objects from $objects$
Randomly assign sizes to objects in $selected\_objects$
$compound \leftarrow$ Create objects in FreeCAD and compound objects
$center \leftarrow$Obtain the center of compound object
for $plane \in$ {"XY","XZ", "YZ"} **do**
    $imgs \leftarrow$ `GetSections`($compound,\ k,\ plane$)
    Append $imgs$ to $positive\_samples$
end for
if rot is true **then**
    for $axis \in$ {"x", "y", "z"} **do**
        for $angle \in \{45°,\ 135°\}$ **do**
            $imgs \leftarrow$ `GetRotatedSections`($compound,\ axis,\ center$)
            Append $imgs$ to $positive\_samples$
        end for
    end for
end if
$compound' \leftarrow$Randomly alter the relative ratios of objects in $compound$
$imgs \leftarrow$ Use any of the above approaches to obtain cross-sections of $compound'$

Append $imgs$ to $negative\_samples$
$samples \leftarrow (positive\_samples, negative\_samples)$
Shuffle $samples$ to assign $[A, B, C, D]$ and record $answer\_id$
$data \leftarrow$ `create_data`($ref\_img, samples, question, answer\_id$)

Algorithm 13 Cube Counting Task

Input: Spatial size (X, Y, Z), cube size s, number of constraint views num
Initialize zero value 3D tensors $placement \in \{0\}^{Z \times Y \times X}$, empty list $cubes$
Initialize empty list $samples$
function DETECTGRID($view,\ row\_num\, col\_num$)
    $contours \leftarrow$Find contours in $view$
    Initialize $grid$ matrix of size $row\_num \times col\_num$
    for $contour \in contours$ **do**
        $(x, y, w, h) \leftarrow$ Bounding rectangle of $contour$
        $row \leftarrow y/h$, $col \leftarrow x/w$
        if row and col within bounds **then**
            $grid[row][col] \leftarrow 1$
        end if
    end for
    return $grid$
end function

function GETCUBEANSWER($front,\ top,\ left,\ num$)
    $sum\_front\_col \leftarrow$ Column sums of $front$
    $sum\_top\_col \leftarrow$ Column sums of top
    $max\_2view \leftarrow sum\_front\_col \cdot sum\_top\_col$
    $min\_2view \leftarrow \texttt{sum}(sum\_top\_col - 1 + sum\_front\_col)$
    if $num = 2$ **then**
        return (max_2view, min_2view)
    end if
    $sum\_left\_col \leftarrow$ Column sums of $left$
    Initialize answer matrix with the same dimension as $top \in \{0\}^{H \times W}$
    for $row \leftarrow 0$ to $H - 1$ **do**
        for $col \leftarrow 0$ to $W - 1$ **do**
            if $top[row][col] = 1$ **then**
                $ans[row][col] \leftarrow \texttt{min}(sum\_front\_col[col], sum\_left\_col[row])$
            end if
        end for
    end for
    $max\_3view \leftarrow \texttt{sum}(ans)$
    $sum\_top\_row \leftarrow$ Row sums of top
    $min\_3view \leftarrow \texttt{max}(\texttt{sum}(sum\_top\_row - 1 + sum\_left\_col), min\_2view)$
    return (max_3view, min_3view)
end function

Update $placement, cubes$ with $\texttt{CreateCubes}(X,\ Y,\ Z)$
Update $placement, cubes$ with $\texttt{ConnectIsolatedCubes}(X,\ Y)$
$(front\_view,\ top\_view,\ left\_view) \leftarrow \texttt{SaveViews}(cubes)$
$front\_mat,\ top\_mat,\ left\_mat \leftarrow$
$\texttt{DetectGrid}(front\_view),\ \texttt{DetectGrid}(top\_view),\ \texttt{DetectGrid}(left\_view)$
if $num = 2$ **then**
    $ref\_img \leftarrow (top\_view,\ front\_view)$
    $(max\_view, min\_view) \leftarrow \texttt{GetCubeAnswer}(front\_mat,\ top\_mat,\ left\_mat,\ 2)$
else if $num = 3$ **then**
    $ref\_img \leftarrow (top\_view,\ front\_view,\ left_view)$
    $(max\_view, min\_view) \leftarrow \texttt{GetCubeAnswer}(front\_mat,\ top\_mat,\ left\_mat,\ 3)$
end if

$samples \leftarrow$Generate correct and incorrect nums based on the min_view to max_view range
Shuffle $samples$ to assign $[A, B, C, D]$ and record $answer\_id$
$data \leftarrow \texttt{create\_data}(ref\_img, samples, question, answer\_id)$

Algorithm 14 Functions for Splitting Cube Stack into Several Connected Parts

function GETNEIGHBORS($cube\_pos,\ cubes$)
    $(x, y, z) \leftarrow cube\_pos$
    Initialize empty list $neighbours$
    for $dx \in \{-1, 0, 1\}$ **do**
        for $dy \in \{-1, 0, 1\}$ **do**
            for $dz \in \{-1, 0, 1\}$ **do**
                if $|dx| + |dy| + |dz| = 1$ **then** ▷ 6-connected neighborhood
                    $neighbor\_pos \leftarrow (x + dx, y + dy, z + dz)$
                    if $neighbor\_pos \in cubes$ **then**
                        Append $neighbor\_pos$ to $neighbours$
                    end if
                end if
            end for
        end for
    end for
    return $neighbors$
end function

function REGIONGROWING($cubes,\ max\_cubes$)
    Initialize empty set $part$, empty list $queue$
    $start\_pos \leftarrow$ Randomly select a position from $cubes$ and append $start\_pos$ to $queue$
    while $queue$ not empty and $|part| < max\_cubes$ **do**
        $current\_pos \leftarrow \texttt{pop}(queue, 0)$
        if $current\_pos \notin part$ **then**
            Add $current\_pos$ to $part$
            $neighbors \leftarrow \texttt{GetNeighbors}(current\_pos,\ cubes)$
            Extend $[n \in neighbors \mid n \notin part]$ to $queue$
        end if
    end while
    return $part$
end function

function ISCONTINUOUS($part$)
    Initialize empty set $part$, empty list $queue$
    $start\_pos \leftarrow part[0]$ and append $start\_pos$ to $queue$
    while $queue$ not empty **do**
        $current\_pos \leftarrow \texttt{pop}(queue, 0)$
        if $current\_pos \notin visited$ **then**
            Add $current\_pos$ to $visited$
            $neighbors \leftarrow \texttt{GetNeighbors}(current\_pos,\ part)$
            Extend $[n \in neighbors \mid n \in part\ and\ n \notin visited]$ to $queue$
        end if
    end while
    return Whether $|visited| = |part|$
end function

function SPLITCUBES($cubes,\ max\_cubes,\ num\_parts$)
    $part1 \leftarrow \texttt{RegionGrowing}(cubes, max\_cubes)$
    if $\texttt{IsContinuous}(part1)$ **then**
        $remaining \leftarrow$ Remove $part1$ from $cubes$
    end if
    if $\texttt{IsContinuous}(remaining)$ **then**
        if $num\_parts = 2$ **then**
            return $\texttt{sort}([part1, remaining])$ by size
        else if $num\_parts = 3$ **then**
            Similarly find $part2$ from remaining cubes as above
            $part3 \leftarrow$ Remove $part2$ from $remaining$
            return $\texttt{sort}([part1, part2, part3])$ by size
        end if
    end if
end function

Algorithm 15 Cube Assembly Task

Input: Spatial size (X, Y, Z), cube size s, number of splitting parts k
Initialize zero value 3D tensors $placement \in \{0\}^{Z \times Y \times X}$, empty list $cubes$
Initialize empty lists ref_imgs, $positive\_samples$, $negative\_samples$
function CREATECUBESPYRAMID($X,\ Y,\ Z$)
    Initialize $num = 1$
    for $y \leftarrow 0$ to $Y - 1$ **do**
        $num = \texttt{randomInt}(num,\ \texttt{min}(y + 2,\ X))$
        for $x \leftarrow 0$ to $num - 1$ **do**
            $\texttt{CreateCube}(x,\ y,\ 0)$
        end for
    end for
    for $z \leftarrow 1$ to $Z - 2$ **do**
        Initialize $num = 0$
        for $y \leftarrow 0$ to $Y - 1$ **do**
            $num = \texttt{randomInt}(num,\ \texttt{max}(num, \texttt{sum}(placement[z-1][y])))$
            for $x \leftarrow 0$ to $num - 1$ **do**
                $\texttt{CreateCube}(x,\ y,\ z)$
            end for
        end for
    end for
    for $y \leftarrow 0$ to $Y - 1$ **do**
        for $x \leftarrow 0$ to $X - 1$ **do**
            With 50% probability $\texttt{CreateCube}(x,\ y,\ Z - 1)$
        end for
    end for
end function
Update $placement, cubes$ with $\texttt{CreateCubesPyramid}(X,\ Y,\ Z)$
$cubes\_img \leftarrow \texttt{FreeCAD.saveImage}(cubes)$ and append $cubes\_img$ to ref_imgs
$parts \leftarrow \texttt{SplitCubes}(cubes, max\_cubes,\ num\_parts)$
for $part \in parts[:-1]$ **do**
    $part\_img \leftarrow \texttt{FreeCAD.saveImage}(part)$ and append $part\_img$ to ref_imgs
end for
$part\_img \leftarrow \texttt{FreeCAD.saveImage}(parts[-1])$ and append $part\_img$ to $positive\_samples$
for $i \leftarrow 1$ to 2 **do**
    $part' \leftarrow$Randomly remove 1 cube from part[-1]
    $part'\_img \leftarrow \texttt{FreeCAD.saveImage}(part')$ and append $part\_img$ to $negative\_samples$
end for
$samples \leftarrow (positive\_samples, negative\_samples)$
Shuffle $samples$ to assign $[A, B, C, D]$ and record $answer\_id$
$data \leftarrow \texttt{create\_data}(ref\_img, samples, question, answer\_id)$

Algorithm 16 Simulation for Arrow Moving

Class ArrowPath
Attributes:
  W, H, k: Map width, height, and step count
  $max\_step \leftarrow \min(x, y)$
  $directions \leftarrow$ {(0,1),(1,0),(0,-1),(-1,0)} ▷ up, right, down, left
  $path$: Initialize with empty list to record relative moving direction and steps
  $states$: Initialize with empty list to record pos and orientation during transformation
function INITIALIZESTATE
    Reset $path,\ states$
    $orient\_id \leftarrow$ `randomInt`$(0, 3)$
    $pos \in \{(x, y)\} \leftarrow$ Randomly select a position in the map
    Append $(orient\_id,\ pos)$ to $states$
end function
function GETRELATIVEDIRECTION($orient\_id$)
    $forward \leftarrow directions[orient\_id]$
    $backward \leftarrow (-forward[0], -forward[1])$
    $left \leftarrow directions[(orient\_id - 1) \mod 4]$
    $right \leftarrow directions[(orient\_id + 1) \mod 4]$
    return {"forward":forward, "backward":backward, "left":left, "right":right}
end function
function UPDATEORIENTID($rel\_dir,\ orient\_id$)
    if rel_dir is "backward" **then**
        $orient\_id \leftarrow (orient\_id + 2) \mod 4$
    else if rel_dir is "left" **then**
        $orient\_id \leftarrow (orient\_id - 1) \mod 4$
    else if rel_dir is "right" **then**
        $orient\_id \leftarrow (orient\_id + 1) \mod 4$
    end if
    return $orient\_id$
end function
function MOVE($state,\ rel\_dir,\ steps$)
    $pos,\ orient\_id \leftarrow state$
    $move\_dir \leftarrow$ `GetRelativeDirection`$(orient\_id)[rel\_dir]$
    $new\_pos \leftarrow [pos[0] + move\_dir[0] \times steps, pos[1] + move\_dir[1] \times steps]$
    if new_pos is invalid **then**
        return false
    end if
    Append $(rel\_dir, steps)$ to $path$
    Append (`UpdateOrientId`$(rel\_dir,\ orient\_id),\ new\_pos$) to $states$
    return true
end function
function GENERATEPATH($k,\ end\_state$=None)
    for $i \leftarrow 1$ to k **do**
        repeat
            Randomly select $rel\_dir \in$ {"forward", "backward", "left", "right"}
            $steps \leftarrow$ `randomInt`$(1, max\_step)$
            $valid\_flag \leftarrow$ `Move`$(states[-1],\ rel\_dir,\ steps)$
            if end_state is not None and $i = k$ **then**
                $valid\_flag \leftarrow valid\_flag\ \ \&\ \ state[-1] \neq end\_state$
            end if
        until $valid\_flag$ is true
    end for
end function

Algorithm 17 Simulation for Arrows Moving

Class ArrowMap(Inherit from **Class** ArrowPath)
Attributes:
  $colors$: Color set
  $path$: Initialize with empty list to record arrow position, relative moving direction and steps
  $states$: Initialize with empty list to record map during transformation
function INITIALIZESTATE
  Initialize empty matrix $state$
  for $y \leftarrow$1 to H **do**
    for $x \leftarrow 1$ to W **do**
      With 50% probability:
      Randomly select $color \in colors$
      Randomly get $orient\_id \leftarrow$ `randomInt`$(0, 3)$
      $state[pos] \leftarrow$ Record $color$ and $orient\_id$ at $pos(x, y)$
    end for
  end for
  Append $state$ to $states$
end function

function MOVE($state, arrow\_pos,\ rel\_dir,\ steps$)
  $curr\_pos \leftarrow arrow\_pos$
  $curr\_orient\_id,\ curr\_color \leftarrow state[x][y]$
  $move\_dir \leftarrow$ `GetRelativeDirection`$(curr\_orient\_id)[rel\_dir]$
  $new\_pos \leftarrow [pos[0] + move\_dir[0] \times steps, pos[1] + move\_dir[1] \times steps]$
  if new_pos is invalid **then**
    return false
  end if
  $new\_orient\_id \leftarrow$ `UpdateOrientId`$(rel\_dir,\ orient\_id)$
  if $new\_pos = curr\_pos$ and $new\_orient\_id = curr\_orient\_id$ **then**
    return false
  end if
  Append $arrow\_pos,\ rel\_dir,\ steps$ to $path$
  if $state[new\_pos]$ is None **then**
    $state[curr\_pos] \leftarrow$ None
  else
    $target\_color,\ target\_orient\_id \leftarrow state[new\_pos]$
    $target\_move\_dir \leftarrow -move\_dir$
    $target\_rel\_directions \leftarrow$ `GetRelativeDirection`$(target\_orient\_id)$
    $taget\_rel\_dir \leftarrow$ Find $\{key \in target\_rel\_directions \mid value = target\_move\_dir\}$
    $new\_target\_orient\_id \leftarrow$ `UpdateOrientId`$(taget\_rel\_dir,\ target\_orient\_id)$
    $state[curr\_pos] \leftarrow target\_color$ and $new\_target\_orient\_id$
  end if
  $state[new\_pos] \leftarrow curr\_color$ and $curr\_orient\_id$
  return true
end function

function GENERATEPATH($k,\ end\_state$=None)
  for $i \leftarrow 1$ to k **do**
    repeat
      Randomly select $arrow\_pos \in \{pos \mid state[pos]$ is not None$\}$
      Randomly select $rel\_dir \in$ {"forward", "backward", "left", "right"}
      $steps \leftarrow$ `randomInt`$(1, max\_step)$
      $valid\_flag \leftarrow$ `Move`$(state, arrow\_pos, rel\_dir,\ steps)$
      if end_state is not None and $i = k$ **then**
        $valid\_flag \leftarrow valid\_flag\ \ \&\ \ state[-1] \neq end\_state$
      end if
    until $valid\_flag$ is true
  end for
end function

Algorithm 18 Arrow Moving Task in Easy Version

Input: Dimension of map (W, H), step count k
Initialize empty lists $positive\_samples$, $negative\_samples$
Initialize $arrow\_path$ with dimension $W \times H$
Initialize state with $arrow\_path$.`InitializeState`() and record as $initial\_state$
Update $path,\ states$ with $arrow\_path$.`GeneratePath`(k)
Append $path$ to $positive\_samples$
$ref\_img \leftarrow$ `draw_map`$(states[0], states[-1])$
Record $end\_state \leftarrow states[-1]$

From the same $initial\_state$
for $i \leftarrow 1$ to 3 **do**
    Update $path'$ with $arrow\_path$.`GeneratePath`$(k,\ end\_state)$
    Append $path'$ to $negative\_samples$
end for

$samples \leftarrow (positive\_samples, negative\_samples)$
Shuffle $samples$ to assign $[A, B, C, D]$ and record $answer\_id$
$data \leftarrow$ `create_data`$(ref\_img, samples, question, answer\_id)$

Algorithm 19 Arrow Moving Task in Hard Version

Input: Dimension of map (W, H), step count k, task mode m
Initialize empty lists $positive\_samples$, $negative\_samples$
Initialize $arrow\_map$ with dimension $W \times H$
Initialize state with $arrow\_map$.`InitializeState`() and record as $initial\_state$
Update $path,\ states$ with $arrow\_map$.`GeneratePath`(k)
Append $path$ to $positive\_samples$
if $m =$ "state" **then**
    $ref\_img \leftarrow$ `draw_map`$(states[0])$
    Append $states[-1]$ to $positive\_samples$
else if $m =$ "path" **then**
    $ref\_img \leftarrow$ `draw_map`$(states[0], state[-1])$
    Append $path$ to $positive\_samples$
end if
Record $end\_state \leftarrow states[-1]$

From the same $initial\_state$
for $i \leftarrow 1$ to 3 **do**
    Update $path',\ states'$ with $arrow\_map$.`GeneratePath`$(k,\ end\_state)$
    if $m =$ "state" **then**
        Append $states'[-1]$ to $negative\_samples$
    else if $m =$ "path" **then**
        Append $path'$ to $negative\_samples$
    end if
end for

$samples \leftarrow (positive\_samples, negative\_samples)$
Shuffle $samples$ to assign $[A, B, C, D]$ and record $answer\_id$
$data \leftarrow$ `create_data`$(ref\_img, samples, question, answer\_id)$

Algorithm 20 Simulation for Block Moving

Class Block
Attributes:
  X, Y, Z, k: Spatial size and step count
  $directions$: 6 directions
  $colors$: Color set
  $cubes\_info$: Initialize with empty list to record positions and colors of cube objects
  $transformation$: Initialize with empty list to record transformations
function INITIALIZESTATE
  Update $cubes$ with `CreateCubes`(X, Y, Z)
  Assign randomly selected colors to $cubes$ and record their colors and positions in $cubes\_info$
end function
function HASSUPPORT($x,\ y,\ z$)
  if $z = 0$ or there is cube at $(x, y, z-1)$ **then**
    return true
  end if
  return flase
end function
function DROPCUBES
  Sort $cubes\_info$ by z of pos in ascending order
  for $cube \in cubes\_info$ **do**
    $(x, y, z) \leftarrow$ Acquire position of $cube$ from $cubes\_info$
    while `HasSupport`(x, y, z) is flase **do**
      Change the position of $cube$ to $(x, y, z-1)$ and update $z \leftarrow z-1$
    end while
  end for
end function
function CHECKMOVE($from\_pos,\ to\_pos$)
  if (to_pos is invalid) or (`HasSupport`(to_pos) is false) or (there is no cube at $from\_pos$) or (there is no cube at to_pos and to_pos is on top of $from\_pos$) **then**
    return false
  end if
  return true
end function
function MOVECUBE($from\_pos,\ to\_pos$)
  if there is no cube at to_pos **then**
    Update $cubes\_info$ with changing the position of $cube$ at $from\_pos$ to to_pos
  else
    Update $cubes\_info$ with swapping the cube at $from\_pos$ and to_pos
  end if
  `DropCubes`()
  Append $(from\_pos,\ to\_pos - from\_pos)$ to $transformation$
end function
function GENERATETRANSFORMATION(k)
  for $i \leftarrow 1$ to k **do**
    Initialize empty list $possible\_moves$
    for all $cube \in cubes\_info$ **do**
      for all $direction \in directions$ **do**
        $to\_pos \leftarrow$ The position of cube $from\_pos + direction$
        if `CheckMove`$(from\_pos,\ to\_pos)$ is true **then**
          Append $(from\_pos,\ direction,\ to\_pos)$ to $possible\_moves$
        end if
      end for
    end for
    Randomly select $(from\_pos,\ direction,\ to\_pos) \in possible\_moves$
    `MoveCube`$(from\_pos,\ to\_pos)$
  end for
end function

Algorithm 21 Block Moving Task

Input: Spatial size $(X,\ Y,\ Z)$, step count k
Initialize empty lists ref_imgs, $positive\_samples$, $negative\_samples$
Initialize $block$ with size (X, Y, Z)
Initialize with $block$.`InitializeState`() and record as $initial\_cubes\_info$
$img \leftarrow$ `FreeCAD.saveImage`$(initial\_cubes)$ and append img to ref_imgs
Update $transformation,\ cubes\_info$ with $block$.`GenerateTransformation`(k)
Append $transformation$ to $positive\_samples$
Record $final\_cubes\_info$ after transformation
$img \leftarrow$ `FreeCAD.saveImage`$(final\_cubes)$ and append img to ref_imgs
From the same $initial\_cubes\_info$
for $i \leftarrow 1$ to 3 **do**
    repeat
        Update $transformation',\ cubes\_info'$ with $block$.`GenerateTransformation`(k)
    until $cubes\_info \neq final\_cubes\_info$
    Append $transformation$ to $negative\_samples$
end for
$samples \leftarrow (positive\_samples, negative\_samples)$
Shuffle $samples$ to assign $[A, B, C, D]$ and record $answer\_id$
$data \leftarrow$ `create_data`$(ref\_imgs, samples, question, answer\_id)$

C Data Examples

We present exemplars of varying difficulty levels for all tasks, with each sample containing an image, question, options, answer, and explanation.

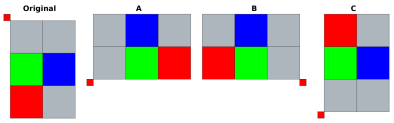
Mental Rotation 2DRotation: Figure 6, 3DRotation: Figure 7, 3ViewProjection: Figure 8;

Mental Folding PaperFolding: Figure 9, CubeUnfolding: Figure 10, CubeReconstruction: Figure 11;

Visual Penetration CrossSection: Figure 12, CubeCounting: Figure 13, CubeAssembly: Figure 14;

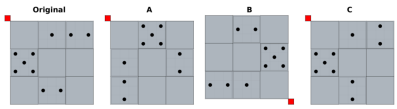
Mental Animation ArrowMoving: Figure 15, BlockMoving: Figure 16, MechanicalSystem: Figure 17.

2D Rotation Task-Level 0



Question: The left image shows a colored grid with a red square marking one corner. Which grid can be obtained by rotating the left grid only, without flipping or other changes? Please answer from options A, B, C, or D.
Choices:
A. A
B. B
C. C
D. All three other options are incorrect
Answer: A
Explanation:
A: Option A is correct because it was obtained by rotating the original image 270 degrees.
B: Option B is incorrect because it was obtained by rotating the original image 90 degrees and then flipping it horizontally.
C: Option C is incorrect because it was obtained by rotating the original image 180 degrees and then flipping it vertically.

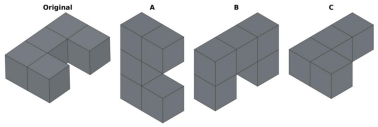
2D Rotation Task-Level 1



Question: The left image shows a colored grid with a red square marking one corner. Which grid can be obtained by rotating the left grid only, without flipping or other changes? Please answer from options A, B, C, or D.
Choices:
A. A
B. B
C. C
D. All three other options are incorrect
Answer: B
Explanation:
A: Option A is incorrect because it was obtained by rotating the original image 270 degrees and then flipping it horizontally.
B: Option B is correct because it was obtained by rotating the original image 180 degrees.
C: Option C is incorrect because it was obtained by rotating the asymmetric patterns in the image.

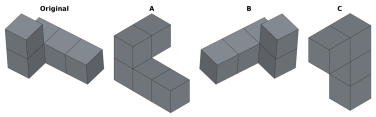
Figure 6: 2D Rotation Task.

3D Rotation Task-Level 0



Question: The left image shows the original cube stack made of equal-sized small cubes. Which of the options on the right cannot be obtained by rotating the original cube stack? Please answer from options A, B, C, or D.
Choices:
A. A
B. B
C. C
D. All three other options are incorrect
Answer: A
Explanation:
A: Option A is incorrect because the cube stack can be obtained by rotating the original stack around the x-axis by 270 degrees.
B: Option B is incorrect because the cube stack can be obtained by rotating the original stack around the y-axis by 90 degrees.
C: Option C is correct because it was obtained by removing one small cube from the original stack.

3D Rotation Task-Level 1



Question: The left image shows the original cube stack made of equal-sized small cubes. Which of the options on the right cannot be obtained by rotating the original cube stack? Please answer from options A, B, C, or D.
Choices:
A. A
B. B
C. C
D. All three other options are incorrect
Answer: A
Explanation:
A: Option A is correct because the cube stack can be obtained by rotating the original stack around the x-axis by 270 degrees.
B: Option B is incorrect because it is a vertically mirrored version of the original cube stack.
C: Option C is incorrect because it was obtained by removing one small cube from the original stack.

Figure 7: 3D Rotation Task.

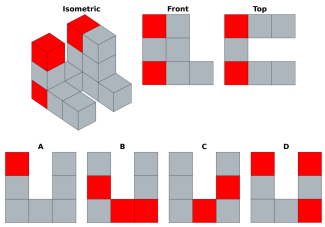
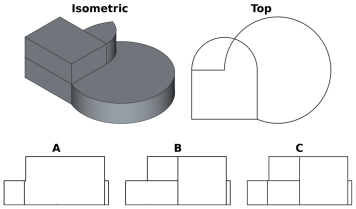
Three-View Projection Task-Level 0 Cubes	Three-View Projection Task-Level 1 CAD Model
	
Question: The cube stack is made of equal-sized small cubes, mostly gray with a few red ones. The top row shows its isometric view, front view, and top view from left to right. Which image in the bottom row is the left view of the cube stack? Please answer from options A, B, C, or D. **Choices:** A. A B. B C. C D. D **Answer:** D **Explanation:** A: Option A is incorrect because the image shows the right view of the cube stack instead of the left view. B: Option B is incorrect because the shape matches the right view instead of the left view, and the red cubes are not in the correct position. C: Option C is incorrect because the red cubes are not in the correct position in the view.	**Question:** The top row shows the isometric view (left) and the top view (right) of a 3D model. Which image in the bottom row is the left view of the model? Please answer from options A, B, C, or D. **Choices:** A. A B. B C. C D. All three other options are incorrect **Answer:** A **Explanation:** A: Option A is incorrect because the internal outlines are missing. B: Option B is incorrect because the internal outlines are missing.

Figure 8: Three-view Projection Task.

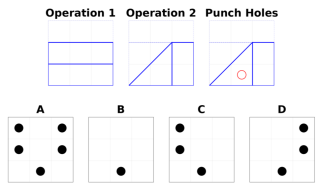
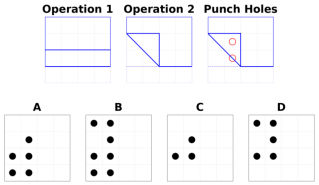
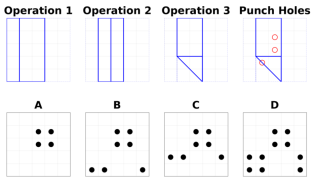
Paper Folding Task-Level 0	Paper Folding Task-Level 1	Paper Folding Task-Level 2
		
Question: The original paper is a 3×3 grid paper. The images in the top row show the results of 2 consecutive folding operations on the grid paper. Folding operations include folding along horizontal, vertical, or 45-degree direction. The rightmost image in the top row shows the result after punching holes in the folded paper. Which image represents the appearance of the paper after unfolding the punched paper? Black solid circles represent grid cells with holes. Please answer from options A, B, C, or D. **Choices:** A. A B. B C. C D. D **Answer:** C **Explanation:** A: Option A is incorrect because extra holes appear in column 3. B: Option B is incorrect because holes in column 1 are missing. D: Option D is incorrect because holes that should appear in column 1 appear in column 3.	**Question:** The original paper is a 4×4 grid paper. The images in the top row show the results of 2 consecutive folding operations on the grid paper. Folding operations include folding along horizontal, vertical, or 45-degree direction. The rightmost image in the top row shows the result after punching holes in the folded paper. Which image represents the appearance of the paper after unfolding the punched paper? Black solid circles represent grid cells with holes. Please answer from options A, B, C, or D. **Choices:** A. A B. B C. C D. D **Answer:** A **Explanation:** B: Option B is incorrect because extra holes appear in row 1. C: Option C is incorrect because holes in row 4 are missing. D: Option D is incorrect because holes that should appear in row 4 appear in row 1.	**Question:** The original paper is a 5×5 grid paper. The images in the top row show the results of 3 consecutive folding operations on the grid paper. Folding operations include folding along horizontal, vertical, or 45-degree direction. The rightmost image in the top row shows the result after punching holes in the folded paper. Which image represents the appearance of the paper after unfolding the punched paper? Black solid circles represent grid cells with holes. Please answer from options A, B, C, or D. **Choices:** A. A B. B C. C D. D **Answer:** C **Explanation:** A: Option A is incorrect because holes in row 4 are missing. B: Option B is incorrect because holes that should appear in row 4 appear in row 5. D: Option D is incorrect because extra holes appear in row 5.

Figure 9: Paper Folding Task.

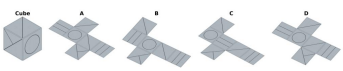
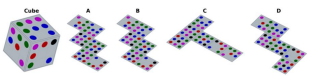
Cube Unfolding Task-Level 0	Cube Unfolding Task-Level 1	Cube Unfolding Task-Level 2
		
Question: The left image shows a colored cube from a particular viewing angle. The options are nets (unfolded patterns) of the cube, which are folded upward to form the cube. Which net, when folded, cannot form the cube shown in the left image? Please answer from options A, B, C, or D. **Choices:** A. A B. B C. C D. D **Answer:** C **Explanation:** A/D/B: Option A/D/B is incorrect because this net could be a valid net for the given cube, as the positions of red, pink, and blue match the shown cube. C: Option C is correct because this net cannot be a valid net for the given cube, as the positions of yellow and pink are reversed.	**Question:** The left image shows a cube with different patterns on its six faces from a particular viewing angle. The options are nets (unfolded patterns) of the cube, which are folded upward to form the cube. Which net, when folded, can form the cube shown in the left image? Please answer from options A, B, C, or D. **Choices:** A. A B. B C. C D. D **Answer:** D **Explanation:** A: Option A is incorrect because the squares with asymmetric patterns have been rotated. B: Option B is incorrect because the squares with asymmetric patterns have been rotated. C: Option C is incorrect because two faces have swapped positions. D: Option D is correct because the relative positions of three faces match the cube shown in the left image.	**Question:** The left image shows a cube with different patterns on its six faces from a particular viewing angle. The options are nets (unfolded patterns) of the cube, which are folded upward to form the cube. Which net, when folded, cannot form the cube shown in the left image? Please answer from options A, B, C, or D. **Choices:** A. A B. B C. C D. D **Answer:** B **Explanation:** A/C/D: Option A/C/D is incorrect because the relative positions of three faces match the cube shown in the left image. B: Option B is correct because two faces have swapped positions, so it cannot form the cube shown in the left image.

Figure 10: Cube Unfolding Task.

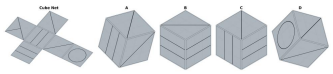
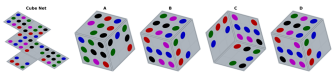
Cube Reconstruction Task-Level 0	Cube Reconstruction Task-Level 1	Cube Reconstruction Task-Level 2
		
Question: As shown, this is the net (unfolded pattern) of a cube, with six faces colored in different colors. The net is folded upward to form a cube. Which color face is opposite to the green face? Please answer from options A, B, C, or D. **Choices:** A. yellow B. pink C. All three other options are incorrect D. red **Answer:** B **Explanation:** A/B/C/D: Assuming the bottom face is the first cell in the second row of the net, then after folding, the front face is red, the back face is green, the left face is blue, the right face is cyan, the top face is yellow, the bottom face is pink.	**Question:** The left image shows the net (unfolded pattern) of a cube, with six faces having different patterns. The net is folded upward to form a cube. From an axonometric (3D) viewing angle of the cube, which combination of adjacent patterns is possible to see? Please answer from options A, B, C, or D. **Choices:** A. A B. B C. C D. D **Answer:** D **Explanation:** Assuming the bottom face is the first cell in the second row of the net, and the right face is the cell to its right. A: Option A is incorrect because it is a vertically mirrored version of the back-top-right view. B: Option B is incorrect because it includes rotated non-symmetric faces. C: Option C is correct because it shows the front-bottom-right view. D: Option D is incorrect because it is a horizontally mirrored version of the back-top-left view.	**Question:** The left image shows the net (unfolded pattern) of a cube, with six faces having different patterns. The net is folded upward to form a cube. From an axonometric (3D) viewing angle of the cube, which combination of adjacent patterns is possible to see? Please answer from options A, B, C, or D. **Choices:** A. A B. B C. C D. D **Answer:** D **Explanation:** Assuming the bottom face is the first cell in the second row of the net, and the right face is the cell to its right. A: Option A is correct because it shows the back-top-right view. B: Option B is incorrect because it includes rotated non-symmetric faces. C: Option C is incorrect because it is a horizontally mirrored version of the front-bottom-right view. D: Option D is incorrect because it includes rotated non-symmetric faces.

Figure 11: Cube Reconstruction Task.

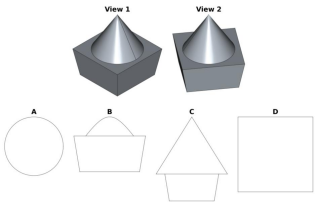
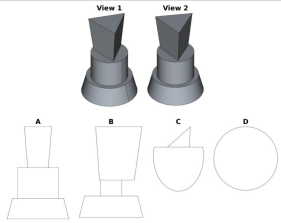
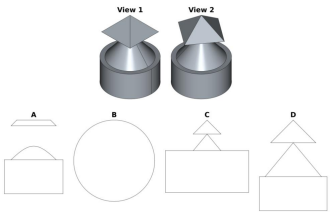
Cross-Section Task-Level 0	Cross-Section Task-Level 1	Cross-Section Task-Level 2
		
Question: The top row shows the combined shape viewed from two different angles. The shape consists of a cone on top of a square frustum. Which of the following images cannot be a cross-section of the shape? Please answer from options A, B, C, or D. **Choices:** A. A B. B C. C D. D **Answer:** C **Explanation:** A: Option A is incorrect because it is the cross-section of the shape made by a plane parallel to the XY plane. B: Option B is incorrect because it is the cross-section of the shape made by a plane parallel to the XZ plane. C: Option C is correct because the corresponding cross-section does not match the shape shown in the reference image. D: Option D is incorrect because it is the cross-section of the shape made by a plane parallel to the XY plane.	**Question:** The top row shows the combined shape viewed from two different angles. The shape consists of a triangular frustum, a cylinder, and a circular frustum from top to bottom. Which of the following images cannot be a cross-section of the shape? Please answer from options A, B, C, or D. **Choices:** A. A B. B C. C D. D **Answer:** B **Explanation:** A: Option A is incorrect because it is the cross-section of the shape made by a plane parallel to the XZ plane. B: Option B is correct because the corresponding cross-section does not match the shape shown in the reference image. C: Option C is incorrect because it is the cross-section made by a plane perpendicular to the XZ plane and rotated 45 degrees around the y-axis. D: Option D is incorrect because it is the cross-section of the shape made by a plane parallel to the XY plane.	**Question:** The top row shows the combined shape viewed from two different angles. The shape consists of a square pyramid, a cone, and a cylinder from top to bottom. Which of the following images cannot be a cross-section of the shape? Please answer from options A, B, C, or D. **Choices:** A. A B. B C. C D. D **Answer:** D **Explanation:** A: Option A is incorrect because it is the cross-section of the shape made by a plane parallel to the XZ plane. B: Option B is incorrect because it is the cross-section of the shape made by a plane parallel to the XY plane. C: Option C is correct because the corresponding cross-section does not match the shape shown in the reference image. D: Option D is incorrect because it is the cross-section of the shape made by a plane parallel to the XZ plane.

Figure 12: Cross-sectionn Task.

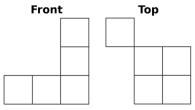
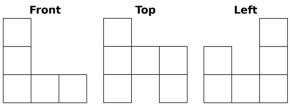
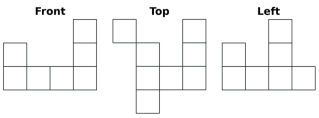
Cube Counting Task-Level 0	Cube Counting Task-Level 1	Cube Counting Task-Level 2
		
Question: Given two views, what is the minimum number of cubes required to satisfy the constraints shown in the images? Please answer from options A, B, C, or D. **Choices:** A. 5 B. All three other options are incorrect C. 7 D. 8 **Answer:** C **Explanation:** A/B/C/D: Given two views, at least 7 cubes and at most 9 cubes are required to satisfy the constraints.	**Question:** Given three views, what is the maximum number of cubes required to satisfy the constraints shown in the images? Please answer from options A, B, C, or D. **Choices:** A. 8 B. 11 C. 10 D. 9 **Answer:** D **Explanation:** A/B/C/D : Given three views, at least 9 cubes and at most 9 cubes are required to satisfy the constraints.	**Question:** Given three views, how many cubes could be needed to satisfy the constraints shown in the images? Please answer from options A, B, C, or D. **Choices:** A. All three other options are incorrect B. 7 C. 16 D. 11 **Answer:** D **Explanation:** A/B/C/D : Given three views, at least 11 cubes and at most 12 cubes are required to satisfy the constraints

Figure 13: Cube Counting Task.

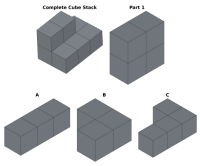
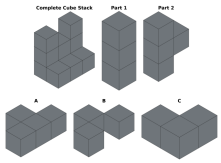
Cube Assembly Task-Level 0	Cube Assembly Task-Level 1
	
Question: The top left image shows the original complete cube stack made of equal-sized cubes. It can be formed by combining the small cube stack on the right(part 1) with one of the options below. Which option completes the original cube stack? Please answer from options A, B, C, or D. **Choices:** A. A B. B C. C D. All three other options are incorrect **Answer:** C **Explanation:** A/B: Option A/B is incorrect because one cube is missing, resulting in an incorrect cube stack shape.	**Question:** The top left image shows the original complete cube stack made of equal-sized cubes. It can be formed by combining the two small cube stacks on the right with one of the options below. Which option completes the original cube stack? Please answer from options A, B, C, or D. **Choices:** A. A B. B C. C D. All three other options are incorrect **Answer:** A **Explanation:** B/C: Option B/C is incorrect because one cube is missing, resulting in an incorrect cube stack shape.

Figure 14: Cube Assembly Task.

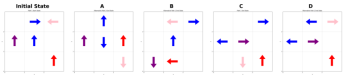
Arrow Moving Task-Level 0	Arrow Moving Task-Level 1(v1)	Arrow Moving Task-Level 1(v2)
		
Question: In the diagram, the red arrow is the initial arrow, and the green arrow is the final arrow. The arrow can move in four directions (forward, backward, left, right), where 'forward' always refers to the current direction the arrow is pointing. After each movement, the arrow's direction is updated to the direction of movement. Which of the following paths can make the arrow move from the starting position to the ending position? Please answer from options A, B, C, or D. **Choices:** A. (Left, 2 units)--(Left, 1 unit) B. (Forward, 1 unit)--(Backward, 1 unit) C. (Forward, 1 unit)--(Backward, 2 units) D. (Forward, 1 unit)--(Left, 1 unit) **Answer:** D **Explanation:** A/B/C: Option A/B/C is incorrect because the initial arrow cannot be transformed into the final arrow. D: Option D is correct because the initial arrow can be transformed into the final arrow.	**Question:** The left image shows the initial state. Arrows can move in four directions (forward, backward, left, right), where 'forward' always refers to the current direction the arrow is pointing. After each movement, the arrow's direction is updated to the direction of movement. If the target position is empty, the arrow can move there directly; otherwise, it needs to swap with the arrow at the target position, and both arrows' movements should satisfy the aforementioned requirements. After the transformations '((0, 1) Right, 1 unit)--((2, 2) Forward, 1 unit)--((1, 2) Left, 2 units)', which state from the options can be reached? Please answer from options A, B, C, or D. **Choices:** A. A B. B C. C D. D **Answer:** C **Explanation:** C: Option C is correct because the initial state can be transformed into the target state. A/B/D: Option A/B/D is incorrect because the initial state cannot be transformed into the target state.	**Question:** The left image shows the initial state, and the right image shows the final state. Arrows can move in four directions (forward, backward, left, right), where 'forward' always refers to the current direction the arrow is pointing. After each movement, the arrow's direction is updated to the direction of movement. If the target position is empty, the arrow can move there directly; otherwise, it needs to swap with the arrow at the target position, and both arrows' movements should satisfy the aforementioned requirements. Which of the following paths can transform the grid from the initial state to the final state? Please answer from options A, B, C, or D. **Choices:** A. ((1, 0) Backward, 1 unit)--((1, 2) Backward, 1 unit) B. ((1, 1) Left, 1 unit)--((1, 1) Forward, 1 unit) C. ((1, 1) Right, 1 unit)--((1, 1) Left, 1 unit) D. ((1, 2) Forward, 1 unit)--((0, 2) Backward, 1 unit) **Answer:** C **Explanation:** A: Option A is correct because the initial state can be transformed into the target state. B/C/D : Option B/C/D is incorrect because the initial state cannot be transformed into the target state.

Figure 15: Arrow Moving Task.

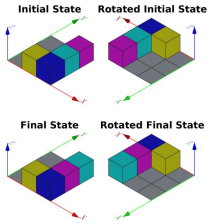
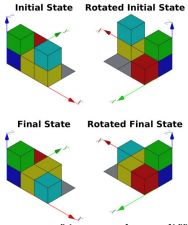
Blocks Moving Task-Level 0	Blocks Moving Task-Level 1
	
Question: The top row of images shows different views of the initial state of a cube stack, while the bottom row shows different views of the final state after transformation. During the transformation process, blocks can move one unit in any direction (forward, backward, left, right, up, down). If the target position is empty, the block can move there directly; if the target position already has a block, they swap places. Blocks cannot float in the air. If a block is moved away from a position, any block above it will fall down until reaching a supporting surface. The xyz axes are shown in the diagram, and each block's position can be precisely identified using coordinates (x,y,z). Which of the following transformation sequences can change the cube stack from the initial state to the final state shown in the diagram? Please answer from options A, B, C, or D. **Choices:** A. (1, 0, 0) y+ -- (1, 1, 0) y- -- (2, 1, 0) y+ B. (1, 0, 0) y+ -- (2, 1, 0) y+ -- (2, 0, 0) y+ C. (2, 2, 0) x- -- (2, 0, 0) x- D. (1, 0, 0) x- -- (2, 0, 0) y+ -- (2, 2, 0) y- **Answer:** A **Explanation:** B/C/D: Option BCD is incorrect because the initial state cannot be transformed into the final state.	**Question:** The top row of images shows different views of the initial state of a cube stack, while the bottom row shows different views of the final state after transformation. During the transformation process, blocks can move one unit in any direction (forward, backward, left, right, up, down). If the target position is empty, the block can move there directly; if the target position already has a block, they swap places. Blocks cannot float in the air. If a block is moved away from a position, any block above it will fall down until reaching a supporting surface. The xyz axes are shown in the diagram, and each block's position can be precisely identified using coordinates (x,y,z). Which of the following transformation sequences can change the cube stack from the initial state to the final state shown in the diagram? Please answer from options A, B, C, or D. **Choices:** A. (1, 0, 0) y+ -- (0, 0, 1) z- B. (1, 0, 0) x+ -- (1, 0, 0) y+ C. (2, 0, 0) x- -- (1, 0, 0) y+ -- (2, 0, 0) x- D. (0, 0, 0) x+ -- (0, 1, 0) y- -- (0, 0, 1) y+ **Answer:** C **Explanation:** A/B/D : Option A/B/D is incorrect because the initial state cannot be transformed into the final state.

Figure 16: Block Moving Task.

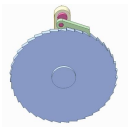
Mechanical System Task-Level 0	Mechanical System Task-Level 1
	
Question: When the red shaft connected to the green rod rotates clockwise, what is the motion of the centrally fixed blue gear? Please choose from options A, B, C, or D. **Choices:** A. Rotates clockwise B. Rotates counterclockwise C. Does not rotat D. Translates to the right **Answer:** A **Explanation:** A/B/C/D: The green rod drives the blue gear to rotate only in the clockwise direction.	**Question:** In the image, the green gear is fixed on a concentric shaft, while the yellow and pink gears are fixed on their own shafts. If the green gear rotates clockwise in the given view, what are the resulting motions of the yellow and pink gears? Please choose from options A, B, C, or D. **Choices:** A. Rotates clockwise, rotates clockwise B. Rotates clockwise, rotates counterclockwise C. Rotates counterclockwise, rotates counterclockwise D. Rotates counterclockwise, rotates clockwise **Answer:** D **Explanation:** A/B/C/D : The yellow gear directly meshes with the green gear and thus rotates in the opposite direction, while the pink gear is driven through two meshing steps and rotates in the same direction as the green gear.

Figure 17: Mechanical System Task.

D Evaluation Details

D.1 Prompts for Response Generation

We use the prompt template as follows: "You should first provide a reasoning process, then provide a single option (A, B, C or D) as the final answer. The reasoning process and the answer are enclosed within <think></think> and <answer></answer> tags, respectively, i.e., <think>reasoning process</think>, <answer>answer</answer>.\nQuestion: **<question here>**\nA.**<option A here>**\nB.**<option B here>**\nC.**<option C here>**\nD.**<option D here>**\n"

D.2 Models

For the DeepseekVL2 series, InternVL2.5 series, InternVL3 series, SAIL-VL series and LLaVA-OneVision series, we deployed these models on H100 servers and used the officially provided code to load the pre-trained models for inference. For all other models, we employed API calls through OpenAI's client service for inference. All closed-source models accessed via API in this study were used with specific, identifiable versions to ensure consistency and reproducibility. Specifically, we used the following model versions:

- `gpt-4o-2024-08-06` for GPT-4o
- `o1-2024-12-17` for o1
- `claude-3-5-sonnet-20240620` for Claude-3.5-Sonnet
- `claude-3-7-sonnet-20250219` for Claude-3.7-Sonnet
- `gemini-1.5-pro` for Gemini-1.5-pro
- `Gemini-2.5-flash-preview-04-17` for Gemini-2.5-flash
- `Gemini-2.5-pro-preview-03-25` for Gemini-2.5-pro
- `Doubao-1-5-vision-pro-32k-250115` for Doubao-1-5-vision-pro
- `qwen-vl-max-0408` for Qwen-VL-max

D.3 Methods for Answer Extraction

We employ a rule-based approach for answer extraction. When model responses follow the predetermined instruction format, we directly extract content between <answer></answer> tags using regular expressions. For responses that don't strictly adhere to the preset prompt, we apply various pattern matching strategies, including identifiers such as: "<answer>", "Answer:", "Final answer", "final answer", "Final Answer", "the answer is", "The answer is", "correct answer", "Correct answer", "Correct Answer", and "correct path". To avoid extracting extraneous content beyond option letters, we process the matching results and use periods "." as delimiters to extract the core answer. Regarding evaluation criteria, a response is deemed correct if and only if the extracted result contains exactly one uppercase option letter (A, B, C, or D) that matches the standard answer. This rule-based evaluation method ensures consistent judgment across various response formats. It is worth noting that despite being prompted to first output reasoning processes followed by answers, LLaVa-OneVision exhibited issues with certain questions, directly responding with single letter options. These cases fall outside our established matching logic and require separate handling.

D.4 Human Performance

To establish a robust human baseline analogous to tested MLLMs, we recruited participants with documented systematic training in mathematics and physics. This selection criterion mirrors the specialized knowledge domains inherent in the model's training data. The evaluation protocol required participants to solve problems without the use of scratch paper, thereby emphasizing inherent spatial visualization and mental manipulation capabilities, akin to assessing a model's internal reasoning processes without external memory aids. Furthermore, participants were allowed unlimited time per question and addressed problems in small batches per session. This design choice aimed to minimize the impact of cognitive fatigue and time constraints, isolating core reasoning abilities in a manner comparable to computational models which do not exhibit exhaustion.

E Detailed Results

In this section, we provide more evaluation results and test cases from Gemini-2.5-pro for each task.

E.1 Intra-Category Comparisons Across Levels

To provide deeper insight into the spatial visualization reasoning capabilities of Multi-modal Large Language Models (MLLMs), this section presents comprehensive experimental results that complement the aggregate performance assessment in Section 5.2. This analysis details the accuracy of each evaluated model across the four core sub-abilities—Mental Rotation, Mental Folding, Visual Penetration, and Mental Animation—defined in the SpatialViz-Bench benchmark, with results stratified by task type and difficulty level. This granular performance breakdown reveals specific strengths and weaknesses of the models when confronting various spatial reasoning challenges, offering targeted insights to guide future model improvements.

E.1.1 Mental Rotation

Table 4 documents model performance on 3 sub-tasks within the Mental Rotation category—2D Rotation (2DR), 3D Rotation (3DR), and 3-View Projection (3VP)—across different difficulty levels.

In the 2D Rotation (2DR) task, several models demonstrate foundational capabilities at Level 0, with ol (72.5%) and Gemini-2.5-pro (62.5%) achieving notable results. As difficulty increases to Level 1, most models show performance decline, though leading models maintain relatively high accuracy (ol: 52.5%, Gemini-2.5-pro: 42.5%).

For 3D Rotation (3DR), performance degradation with increased difficulty is more pronounced. At Level 0, ol (42.5%) and Gemini-2.5-pro (45.0%) perform adequately, but their accuracies decrease substantially to 15.0% and 20.0%, respectively, at Level 1. Many open-source models perform at or below random chance (25%-30%) at this higher difficulty level, highlighting the challenge of mental rotation in complex 3D space.

Interestingly, the 3-View Projection (3VP) task reveals a different pattern: when transitioning from Level 0 (cube stacks) to Level 1 (DeepCAD engineering models), some top-tier models like ol (improving from 40.0% to 58.0%) and Gemini-2.5-pro (increasing from 28.0% to 66.0%) demonstrate enhanced performance. This suggests certain Level 1 image features may be more amenable to these models' processing mechanisms, despite the presumed increase in complexity. Nevertheless, many other models show decreased performance from Level 0 to Level 1 in this sub-task. Overall, Mental Rotation tasks reveal a clear performance gradient across dimensions and geometric complexity while highlighting significant capability variations among model families.

E.1.2 Mental Folding

Table 5 documents model performance on 3 sub-tasks within the Mental Folding category—Paper Folding (PF), Cube Unfolding (CU), and Cube Reconstruction (CR)—at varying difficulty levels. These tasks assess models' capacity for continuous reasoning and dynamic visualization of 3D information throughout transformation processes.

In the Paper Folding (PF) task, as folding steps and hole-punching complexity increase (Level 0 to Level 2), most models perform near random chance, indicating significant challenges in tracking multi-step geometric operations and performing subsequent spatial reasoning.

The more complex Cube Unfolding (CU) and Cube Reconstruction (CR) tasks proved challenging for all models. These tasks require understanding the correspondence between 2D nets and 3D cubes, while also assessing the ability to mentally execute folding operations and continuously reason about transforming 3D structures. Even at Level 0, most models demonstrate low accuracy, often below random chance. In the CU task, Gemini-2.5-pro scored 37.5% (L0), 27.5% (L1), and 30.0% (L2), while ol achieved 37.5% (L0), 37.5% (L1), and 27.5% (L2).

For CR, Gemini-2.5-pro performed at 45.0% (L0), 10.0% (L1), and 35.0% (L2), and ol at 42.5% (L0), 12.5% (L1), and 25.0% (L2), both experiencing significant performance drops at Level 1. However, the surprising performance improvement at Level 2 contradicts human intuition, as Level 2 patterns are objectively more complex for humans. Analysis of sample solutions reveals that models approached

these tasks by employing clear textual descriptions to define patterns composed of differently colored dots, representing their positions in matrix form. Conversely, line patterns proved more challenging for models to describe, and internal rotations could not be easily represented through matrix transposition operations, which . This insight provides valuable direction for designing more challenging tests that effectively evaluate model limitations. The overall results reveal a severe deficiency in reasoning and visualization capabilities when finer-grained correspondence and transformation tracking are required. The introduction of asymmetric patterns further challenges models' ability to maintain precise visual perception and spatial-topological understanding. These results highlight current MLLMs' core weaknesses in handling spatial tasks involving geometric correspondence, topological transformations, and dynamic 3D reasoning.

E.1.3 Visual Penetration

Table 6 documents model performance on 3 sub-tasks within the Visual Penetration category—Cross-Section (CS), Cube Counting (CC), and Cube Assembly (CA)—at varying difficulty levels. This ability requires models to infer internal object structures from visible external features.

In the Cross-Section (CS) task, which requires models to visualize sectional shapes produced by cutting composite geometric solids with various planes, Gemini-2.5-pro and ol maintained relatively stable performance across Levels 0, 1, and 2, while most other models performed near random chance.

For the Cube Counting (CC) task, increasing constraints from two-view (Level 0) to three-view (Level 1), and subsequently expanding spatial dimensions (Level 2), progressively challenged models' view integration and counting inference capabilities. Gemini-2.5-pro's accuracy declined sharply from 80.0% (L0) to 52.5% (L1) and 32.5% (L2). Interestingly, ol's performance followed a pattern of 45.0% (L0), 32.5% (L1), and 45.0% (L2), recovering at Level 2 to match its Level 0 score. Most models struggled to effectively integrate multi-view information in this task.

The Cube Assembly (CA) task, which assesses the ability to identify complementary parts forming a complete structure, showed increasing difficulty as structures enlarged and constituent parts increased (Level 0 to Level 1). For example, Gemini-2.5-pro's accuracy dropped from 45.0% (L0) to 27.5% (L1), and ol's from 35.0% (L0) to 32.5% (L1). Collectively, these results reveal current models' limitations in inferring global internal structures and spatial occupancy from local surface information.

E.1.4 Mental Animation

Table 7 documents model performance on 3 sub-tasks within the Mental Animation category—Arrow Moving (AM), Block Moving (BM), and Mechanical System (MS)—at varying difficulty levels. These tasks assess understanding of dynamic state changes and causal propagation among system components.

In the Arrow Moving (AM) task, which requires understanding ego-centric movement rules and tracking state changes, the transition from simple single-arrow movements (Level 0) to multi-arrow environments involving swaps (Level 1) increasingly challenges models' rule comprehension and state tracking. A notable performance disparity exists between closed-source models (e.g., Gemini-2.5-pro and ol) and open-source counterparts: the former maintain high accuracy across both difficulty levels (almost 100% accuracy by Gemini-2.5-pro), while most open-source models perform significantly worse (near random), particularly in complex multi-arrow Level 1 scenarios. This suggests a capability gap, potentially stemming from differences in architecture or training data, when precise instruction following and multi-step dynamic spatial reasoning are required.

The Block Moving (BM) task combines directional movement with gravity simulation, increasing spatial complexity and operational sequence length, thereby challenging models' intuitive physics and 3D dynamic spatial reasoning. Gemini-2.5-pro's accuracy declined sharply from 95% to 35%, showing the difficulty in dealing with 3D scene.

For the Mechanical System (MS) task, which evaluates understanding of motion transmission and component linkage in complex mechanical systems, questions were designed to minimize reliance on formal physics formulas while emphasizing comprehension through observation and spatial imagination. Interestingly, some open-source models performed better than expected based on their performance in other 3D imagination tasks. This suggests these models may transform such

problems into more formalized reasoning processes similar to physical rule application, rather than relying solely on intuitive 3D mental simulation. While this strategy may yield relatively good scores in certain instances, it potentially deviates from the primary goal of assessing pure spatial visualization capabilities. Overall, mental animation tasks—especially those involving complex dynamic interactions and implicit physical laws—continue to pose significant challenges for current MLLMs, with models exhibiting considerable diversity in performance strategies and capabilities.

Table 4: Comparison of model performances on Mental Rotation tasks. The first and second highest accuracy of MLLMs are marked in red and blue, with open-source and closed-source models marked separately.

Model	Overall	2DRotation			3DRotation			3ViewProjection		
		L0	L1	Avg	L0	L1	Avg	L0	L1	Avg
Random	27.69	25.00	22.50	23.75	25.00	30.00	27.50	30.00	32.00	31.00
Open Source MLLMs										
3B										
SAIL-VL-1.5-2B	22.31	20.00	25.00	22.50	17.50	27.50	22.50	20.00	24.00	22.00
InternVL3-2B	27.31	12.50	20.00	16.25	32.50	35.00	33.75	24.00	38.00	31.00
Deepseek-VL2-tiny(3B)	22.69	10.00	25.00	17.50	20.00	25.00	22.50	22.00	32.00	27.00
Qwen2.5-VL-3B-Instruct	20.00	25.00	15.00	20.00	15.00	22.50	18.75	16.00	26.00	21.00
7B										
Qwen2.5-VL-7B-Instruct	23.85	25.00	25.00	25.00	20.00	12.50	16.25	14.00	44.00	29.00
Qwen2.5-Omni-7B	24.23	32.50	12.50	22.50	25.00	15.00	20.00	22.00	36.00	29.00
LLaVA-OneVision-Qwen2-7B-ov-hf	26.54	35.00	27.50	31.25	20.00	17.50	18.75	24.00	34.00	29.00
SAIL-VL-1.6-8B	21.92	25.00	12.50	18.75	27.50	15.00	21.25	24.00	26.00	25.00
InternVL2.5-8B	30.38	30.00	22.50	26.25	35.00	20.00	27.50	32.00	40.00	36.00
InternVL3-8B	28.85	22.50	17.50	20.00	35.00	42.50	38.75	18.00	38.00	28.00
16B										
Kimi-VL-A3B-Instruct(16B)	28.08	15.00	17.50	16.25	32.50	27.50	30.00	24.00	48.00	36.00
Kimi-VL-A3B-thinking(16B)	20.00	10.00	17.50	13.75	17.50	22.50	20.00	20.00	30.00	25.00
Deepseek-VL2-small(16B)	24.62	40.00	22.50	31.25	10.00	22.50	16.25	22.00	30.00	26.00
32B										
Deepseek-VL2(27B)	29.62	20.00	30.00	25.00	35.00	32.50	33.75	20.00	40.00	30.00
Qwen2.5-VL-32B-Instruct	35.00	35.00	27.50	31.25	32.50	37.50	35.00	22.00	54.00	38.00
InternVL2.5-38B	29.62	27.50	30.00	28.75	37.50	25.00	31.25	20.00	38.00	29.00
InternVL3-38B	28.46	25.00	20.00	22.50	32.50	35.00	33.75	22.00	36.00	29.00
72B										
Qwen2.5-VL-72B-Instruct	29.23	25.00	32.50	28.75	40.00	22.50	31.25	22.00	34.00	28.00
QvQ-72B-preview	27.69	15.00	27.50	21.25	27.50	32.50	30.00	32.00	30.00	31.00
LLaVA-OneVision-Qwen2-72B-ov-hf	33.85	37.50	22.50	30.00	35.00	25.00	30.00	36.00	44.00	40.00
InternVL2.5-78B	25.77	25.00	20.00	22.50	30.00	17.50	23.75	26.00	34.00	30.00
InternVL3-78B	28.46	20.00	30.00	25.00	25.00	25.00	25.00	20.00	48.00	34.00
108B										
Llama-4-Maverick-17B-128E-Instruct	33.85	25.00	15.00	20.00	45.00	35.00	40.00	26.00	54.00	40.00
LLama-4-Scout-17B-16E-Instruct	37.31	32.50	32.50	32.50	32.50	37.50	35.00	28.00	58.00	43.00
Closed Source MLLMs										
GPT-4o	31.15	20.00	45.00	32.50	30.00	25.00	27.50	20.00	46.00	33.00
o1	46.92	72.50	52.50	62.50	42.50	15.00	28.75	40.00	58.00	49.00
Claude-3.5-sonnet	34.62	27.50	35.00	31.25	32.50	17.50	25.00	36.00	54.00	45.00
Claude-3.7-sonnet	38.08	40.00	25.00	32.50	40.00	32.50	36.25	34.00	54.00	44.00
Gemini-1.5-pro	31.92	30.00	35.00	32.50	27.50	30.00	28.75	26.00	42.00	34.00
Gemini-2.5-flash	35.77	55.00	30.00	42.50	40.00	20.00	30.00	18.00	52.00	35.00
Gemini-2.5-pro	44.23	62.50	42.50	52.50	45.00	20.00	32.50	28.00	66.00	47.00
Doubao-1-5-vision-pro	30.38	7.50	7.50	7.50	42.50	27.50	35.00	28.00	62.00	45.00
Qwen-VL-max	28.08	12.50	35.00	23.75	30.00	22.50	26.25	22.00	44.00	33.00

Table 5: Comparison of model performances on Mental Folding tasks.

Model	Overall	PaperFolding				CubeUnfolding				CubeReconstruction			
		L0	L1	L2	Avg	L0	L1	L2	Avg	L0	L1	L2	Avg
Random	21.67	17.50	20.00	20.00	19.17	15.00	27.50	17.50	20.00	30.00	25.00	22.50	25.83
Open Source													
3B													
SAIL-VL-1.5-2B	22.50	12.50	25.00	22.50	20.00	30.00	27.50	25.00	27.50	22.50	20.00	17.50	20.00
InternVL3-2B	24.44	25.00	27.50	15.00	22.50	35.00	12.50	30.00	25.83	35.00	22.50	17.50	25.00
Deepseek-VL2-tiny(3B)	20.56	27.50	17.50	20.00	21.67	20.00	25.00	17.50	20.83	15.00	20.00	22.50	19.17
Qwen2.5-VL-3B-Instruct	24.17	20.00	37.50	17.50	25.00	25.00	25.00	27.50	25.83	25.00	32.50	7.50	21.67
7B													
Qwen2.5-VL-7B-Instruct	28.61	35.00	35.00	32.50	34.17	17.50	30.00	17.50	21.67	27.50	30.00	32.50	30.00
Qwen2.5-Omni-7B	24.17	27.50	30.00	17.50	25.00	32.50	37.50	12.50	27.50	17.50	27.50	15.00	20.00
LLaVA-OneVision-Qwen2-7B-ov-hf	23.33	25.00	27.50	12.50	21.67	15.00	27.50	32.50	25.00	30.00	20.00	20.00	23.33
SAIL-VL-1.6-8B	23.89	35.00	17.50	32.50	28.33	25.00	30.00	20.00	25.00	17.50	25.00	12.50	18.33
InternVL2.5-8B	25.83	25.00	37.50	22.50	28.33	25.00	17.50	42.50	28.33	20.00	22.50	20.00	20.83
InternVL3-8B	25.56	25.00	20.00	40.00	28.33	25.00	20.00	25.00	23.33	25.00	27.50	22.50	25.00
16B													
Kimi-VL-A3B-Instruct(16B)	24.17	27.50	22.50	27.50	25.83	22.50	15.00	22.50	20.00	15.00	27.50	37.50	26.67
Kimi-VL-A3B-thinking(16B)	24.72	10.00	25.00	35.00	23.33	20.00	20.00	32.50	24.17	35.00	17.50	27.50	26.67
Deepseek-VL2-small(16B)	24.72	25.00	22.50	20.00	22.50	27.50	25.00	22.50	25.00	22.50	25.00	32.50	26.67
32B													
Deepseek-VL2(27B)	26.39	22.50	35.00	37.50	31.67	32.50	15.00	27.50	25.00	17.50	30.00	20.00	22.50
Qwen2.5-VL-32B-Instruct	24.72	15.00	37.50	12.50	21.67	17.50	35.00	22.50	25.00	30.00	10.00	42.50	27.50
InternVL2.5-38B	26.11	30.00	20.00	22.50	24.17	25.00	25.00	20.00	23.33	37.50	25.00	30.00	30.83
InternVL3-38B	26.94	22.50	20.00	20.00	20.83	25.00	35.00	27.50	29.17	22.50	32.50	37.50	30.83
72B													
Qwen2.5-VL-72B-Instruct	24.17	12.50	27.50	27.50	22.50	15.00	17.50	27.50	20.00	30.00	25.00	35.00	30.00
QvQ-72B-preview	21.11	15.00	12.50	22.50	16.67	22.50	15.00	20.00	19.17	30.00	25.00	27.50	27.50
LLaVA-OneVision-Qwen2-72B-ov-hf	23.33	22.50	17.50	27.50	22.50	10.00	17.50	27.50	18.33	47.50	25.00	15.00	29.17
InternVL2.5-78B	23.89	32.50	25.00	22.50	26.67	22.50	10.00	17.50	16.67	35.00	17.50	32.50	28.33
InternVL3-78B	22.22	15.00	30.00	12.50	19.17	35.00	22.50	17.50	25.00	30.00	20.00	17.50	22.50
108B													
Llama-4-Maverick-17B-128E-Instruct	25.00	15.00	17.50	17.50	16.67	30.00	25.00	32.50	29.17	30.00	32.50	25.00	29.17
LLama-4-Scout-17B-16E-Instruct	28.61	15.00	17.50	17.50	16.67	35.00	32.50	30.00	32.50	42.50	32.50	35.00	36.67
Closed Source													
GPT-4o	25.00	25.00	35.00	27.50	29.17	25.00	12.50	10.00	15.83	30.00	17.50	42.50	30.00
o1	29.72	27.50	30.00	27.50	28.33	37.50	37.50	27.50	34.17	42.50	12.50	25.00	26.67
Claude-3.5-sonnet	25.00	7.50	35.00	20.00	20.83	25.00	17.50	25.00	22.50	32.50	20.00	42.50	31.67
Claude-3.7-sonnet	24.72	20.00	20.00	15.00	18.33	32.50	25.00	22.50	26.67	32.50	17.50	37.50	29.17
Gemini-1.5-pro	22.78	10.00	10.00	20.00	13.33	22.50	20.00	30.00	24.17	37.50	25.00	30.00	30.83
Gemini-2.5-flash	32.50	15.00	37.50	27.50	26.67	32.50	30.00	27.50	30.00	55.00	27.50	40.00	40.83
Gemini-2.5-pro	35.00	57.50	40.00	32.50	43.33	37.50	27.50	30.00	31.67	45.00	10.00	35.00	30.00
Doubao-1-5-vision-pro	28.06	25.00	37.50	32.50	31.67	22.50	22.50	25.00	23.33	45.00	17.50	25.00	29.17
Qwen-VL-max	24.44	27.50	25.00	20.00	24.17	12.50	15.00	25.00	17.50	42.50	22.50	30.00	31.67

Table 6: Comparison of model performances on Visual Penetration tasks.

Model	Overall	CrossSection				CubeCounting				CubeAssembly		
		L0	L1	L2	Avg	L0	L1	L2	Avg	L0	L1	Avg
Random	28.12	32.50	27.50	30.00	30.00	30.00	20.00	25.00	25.00	22.50	37.50	30.00
Open Source												
3B												
SAIL-VL-1.5-2B	27.19	37.50	20.00	15.00	24.17	40.00	20.00	20.00	26.67	32.50	32.50	32.50
InternVL3-2B	26.56	22.50	22.50	15.00	20.00	22.50	32.50	37.50	30.83	27.50	32.50	30.00
Deepseek-VL2-tiny(3B)	20.94	17.50	25.00	20.00	20.83	25.00	25.00	17.50	22.50	17.50	20.00	18.75
Qwen2.5-VL-3B-Instruct	25.94	25.00	25.00	27.50	25.83	17.50	35.00	17.50	23.33	30.00	30.00	30.00
7B												
Qwen2.5-VL-7B-Instruct	27.19	12.50	12.50	25.00	16.67	32.50	50.00	27.50	36.67	35.00	22.50	28.75
Qwen2.5-Omni-7B	27.19	15.00	22.50	25.00	20.83	37.50	27.50	35.00	33.33	25.00	30.00	27.50
LLaVA-OneVision-Qwen2-7B-ov-hf	27.50	22.50	15.00	17.50	18.33	37.50	35.00	27.50	33.33	37.50	27.50	32.50
SAIL-VL-1.6-8B	21.25	17.50	22.50	25.00	21.67	22.50	17.50	17.50	19.17	30.00	17.50	23.75
InternVL2.5-8B	23.12	25.00	7.50	15.00	15.83	25.00	35.00	25.00	28.33	20.00	32.50	26.25
InternVL3-8B	30.94	17.50	15.00	15.00	15.83	25.00	45.00	52.50	40.83	45.00	32.50	38.75
16B												
Kimi-VL-A3B-Instruct(16B)	17.19	17.50	25.00	22.50	21.67	7.50	2.50	5.00	5.00	27.50	30.00	28.75
Kimi-VL-A3B-thinking(16B)	29.38	27.50	17.50	30.00	25.00	45.00	40.00	25.00	36.67	20.00	30.00	25.00
Deepseek-VL2-small(16B)	25.31	7.50	12.50	7.50	9.17	30.00	32.50	42.50	35.00	30.00	40.00	35.00
32B												
Kimi-VL-A3B-Instruct(16B)	17.19	17.50	25.00	22.50	21.67	7.50	2.50	5.00	5.00	27.50	30.00	28.75
Kimi-VL-A3B-thinking(16B)	29.38	27.50	17.50	30.00	25.00	45.00	40.00	25.00	36.67	20.00	30.00	25.00
Deepseek-VL2-small(16B)	25.31	7.50	12.50	7.50	9.17	30.00	32.50	42.50	35.00	30.00	40.00	35.00
72B												
Qwen2.5-VL-72B-Instruct	39.06	27.50	40.00	22.50	30.00	32.50	50.00	42.50	41.67	55.00	42.50	48.75
QvQ-72B-preview	27.81	32.50	30.00	27.50	30.00	35.00	25.00	7.50	22.50	40.00	25.00	32.50
LLaVA-OneVision-Qwen2-72B-ov-hf	31.25	35.00	7.50	27.50	23.33	22.50	37.50	40.00	33.33	45.00	35.00	40.00
InternVL2.5-78B	26.88	22.50	27.50	20.00	23.33	35.00	25.00	30.00	30.00	25.00	30.00	27.50
InternVL3-78B	35.00	17.50	25.00	20.00	20.83	37.50	52.50	30.00	40.00	42.50	55.00	48.75
108B												
Llama-4-Maverick-17B-128E-Instruct	32.19	27.50	15.00	15.00	19.17	27.50	47.50	30.00	35.00	52.50	42.50	47.50
LLama-4-Scout-17B-16E-Instruct	34.06	17.50	17.50	17.50	17.50	35.00	47.50	30.00	37.50	50.00	57.50	53.75
Closed Source												
GPT-4o	32.50	25.00	25.00	7.50	19.17	40.00	45.00	37.50	40.83	52.50	27.50	40.00
o1	37.81	40.00	42.50	30.00	37.50	45.00	32.50	45.00	40.83	35.00	32.50	33.75
Claude-3.5-sonnet	33.44	35.00	20.00	12.50	22.50	35.00	45.00	27.50	35.83	47.50	45.00	46.25
Claude-3.7-sonnet	31.56	20.00	35.00	17.50	24.17	30.00	32.50	30.00	30.83	40.00	47.50	43.75
Gemini-1.5-pro	25.94	32.50	17.50	12.50	20.83	32.50	25.00	22.50	26.67	42.50	22.50	32.50
Gemini-2.5-flash	32.81	32.50	35.00	22.50	30.00	52.50	32.50	30.00	38.33	30.00	27.50	28.75
Gemini-2.5-pro	42.19	32.50	35.00	32.50	33.33	80.00	52.50	32.50	55.00	45.00	27.50	36.25
Doubao-1-5-vision-pro	39.69	35.00	30.00	25.00	30.00	62.50	65.00	40.00	55.83	42.50	17.50	30.00
Qwen-VL-max	38.44	32.50	20.00	27.50	26.67	57.50	62.50	22.50	47.50	50.00	35.00	42.50

Table 7: Comparison of model performances on Mental Animation tasks.

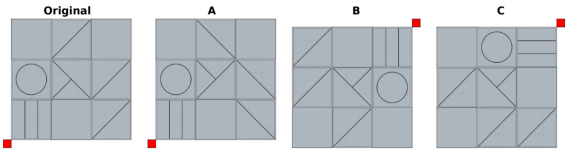
Model	Overall	ArrowMoving			BlockMoving			MechanicalSystem		
		L0	L1	Avg	L0	L1	Avg	L0	L1	Avg
Random	23.33	32.50	25.00	28.75	10.00	22.50	16.25	30.00	20.00	25.00
Open Source										
3B										
SAIL-VL-1.5-2B	24.58	15.00	27.50	21.25	22.50	27.50	25.00	35.00	20.00	27.50
InternVL3-2B	27.08	22.50	15.00	18.75	37.50	27.50	32.50	25.00	35.00	30.00
Deepseek-VL2-tiny(3B)	21.67	25.00	12.50	18.75	25.00	17.50	21.25	25.00	25.00	25.00
Qwen2.5-VL-3B-Instruct	35.83	35.00	35.00	35.00	32.50	27.50	30.00	57.50	27.50	42.50
7B										
Qwen2.5-VL-7B-Instruct	32.50	22.50	22.50	22.50	22.50	25.00	23.75	67.50	35.00	51.25
Qwen2.5-Omni-7B	35.42	27.50	35.00	31.25	32.50	27.50	30.00	67.50	22.50	45.00
LLaVA-OneVision-Qwen2-7B-ov-hf	33.75	25.00	17.50	21.25	45.00	35.00	40.00	52.50	27.50	40.00
SAIL-VL-1.6-8B	35.00	12.50	37.50	25.00	37.50	32.50	35.00	52.50	37.50	45.00
InternVL2.5-8B	28.75	12.50	27.50	20.00	20.00	25.00	22.50	52.50	35.00	43.75
InternVL3-8B	37.08	30.00	30.00	30.00	30.00	30.00	30.00	62.50	40.00	51.25
16B										
Kimi-VL-A3B-Instruct(16B)	27.92	17.50	12.50	15.00	27.50	35.00	31.25	57.50	17.50	37.50
Kimi-VL-A3B-thinking(16B)	40.42	22.50	37.50	30.00	35.00	52.50	43.75	62.50	32.50	47.50
Deepseek-VL2-small(16B)	26.25	25.00	27.50	26.25	25.00	22.50	23.75	47.50	10.00	28.75
32B										
Deepseek-VL2(27B)	29.17	20.00	32.50	26.25	35.00	25.00	30.00	40.00	22.50	31.25
Qwen2.5-VL-32B-Instruct	37.08	22.50	35.00	28.75	27.50	27.50	27.50	62.50	47.50	55.00
InternVL2.5-38B	33.33	17.50	25.00	21.25	32.50	20.00	26.25	60.00	45.00	52.50
InternVL3-38B	37.08	25.00	25.00	25.00	25.00	35.00	30.00	65.00	47.50	56.25
72B										
Qwen2.5-VL-72B-Instruct	43.75	27.50	27.50	27.50	45.00	35.00	40.00	67.50	60.00	63.75
QvQ-72B-preview	39.58	27.50	22.50	25.00	40.00	60.00	50.00	42.50	45.00	43.75
LLaVA-OneVision-Qwen2-72B-ov-hf	27.92	25.00	30.00	27.50	22.50	12.50	17.50	60.00	17.50	38.75
InternVL2.5-78B	34.17	35.00	32.50	33.75	27.50	30.00	28.75	55.00	25.00	40.00
InternVL3-78B	35.42	25.00	22.50	23.75	35.00	47.50	41.25	55.00	27.50	41.25
108B										
Llama-4-Maverick-17B-128E-Instruct	39.17	35.00	35.00	35.00	40.00	40.00	40.00	45.00	40.00	42.50
LLama-4-Scout-17B-16E-Instruct	39.58	15.00	42.50	28.75	47.50	32.50	40.00	57.50	42.50	50.00
Closed Source										
GPT-4o	38.33	32.50	12.50	22.50	25.00	40.00	32.50	62.50	57.50	60.00
o1	57.50	75.00	60.00	67.50	50.00	55.00	52.50	62.50	42.50	52.50
Claude-3.5-sonnet	40.42	42.50	32.50	37.50	25.00	37.50	31.25	57.50	47.50	52.50
Claude-3.7-sonnet	46.25	72.50	60.00	66.25	25.00	32.50	28.75	55.00	32.50	43.75
Gemini-1.5-pro	38.33	37.50	37.50	37.50	25.00	27.50	26.25	62.50	40.00	51.25
Gemini-2.5-flash	50.00	82.50	52.50	67.50	32.50	35.00	33.75	62.50	35.00	48.75
Gemini-2.5-pro	62.92	97.50	92.50	95.00	27.50	42.50	35.00	75.00	42.50	58.75
Doubao-1-5-vision-pro	35.83	22.50	22.50	22.50	32.50	42.50	37.50	57.50	37.50	47.50
Qwen-VL-max	39.17	25.00	27.50	26.25	35.00	37.50	36.25	65.00	45.00	55.00

E.2 Test Cases

This section presents test cases from Gemini-2.5-pro across various tasks, helping us identify error sources and provide direction for model improvement.

Mental Rotation-2D Rotation-Level 1

Prompt: You should first provide a reasoning process, then provide a single option(A, B, C or D) as the final answer. The reasoning process and the answer are enclosed within <think></think> and <answer></answer> tags, respectively, i.e., <think>reasoning process</think>, <answer>answer</answer>.
Question: The left image shows a patterned grid with a red square marking one corner. Which grid can be obtained by rotating the left grid only? Please answer from options A, B, C, or D.
Choices:
A. A B. B
C. C D. All three other options are incorrect
Ground Truth Answer: B

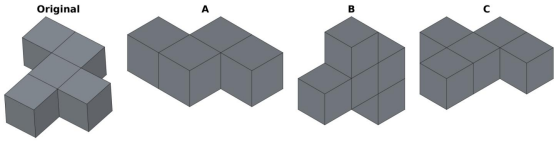


Thinking Process:
1. Analyze the Original Grid:
The grid is 3x3. Note the pattern in each cell and the position of the red square marker in the bottom-left corner.
Top row: Blank, Diagonal, Blank
Middle row: Circle, X-shape, Diagonal
Bottom row: Vertical lines, Blank, Diagonal
Red square: Bottom-left.
2. Analyze Option A: → Wrong incorrect pattern details
The grid pattern in A is identical to the Original grid. The red square is also in the bottom-left corner. This corresponds to a 0-degree (or 360-degree) rotation. While technically a rotation, it doesn't change the orientation.
3. Analyze Option B:
The red square is in the top-right corner. This position corresponds to a 180-degree rotation from the original bottom-left position. Let's check if the pattern matches a 180-degree rotation: Rotating the Original grid 180 degrees means the cell at (row, col) moves to (3-row+1, 3-col+1) = (4-row, 4-col), and the content of the cell is also rotated 180 degrees.
Original (3,1) [Vertical lines] -> Rotated (1,3) [Vertical lines] (Matches B)
Original (3,2) [Blank] -> Rotated (1,2) [Blank] (Matches B)
Original (3,3) [Diagonal] -> Rotated (1,1) [Diagonal] (Matches B)
Original (2,1) [Circle] -> Rotated (2,3) [Circle] (Matches B)
Original (2,2) [X-shape] -> Rotated (2,2) [X-shape] (Matches B)
Original (2,3) [Diagonal] -> Rotated (2,1) [Diagonal] (Matches B)
Original (1,1) [Blank] -> Rotated (3,3) [Blank] (Matches B)
Original (1,2) [Diagonal] -> Rotated (3,2) [Diagonal] (Matches B)
Original (1,3) [Blank] -> Rotated (3,1) [Blank] (Matches B)
The pattern in B perfectly matches a 180-degree rotation of the Original grid.
4. Analyze Option C:
The red square is in the top-right corner, also suggesting a 180-degree rotation. However, the pattern does not match the expected 180-degree rotation derived above. For example, the top-left cell in C is Blank, but the expected cell is Diagonal . The top-right cell in C has Horizontal lines, but the expected cell has Vertical lines. Grid C is not a rotation of the Original.
5. Conclusion:
Grid A is a 0-degree rotation (identical), and Grid B is a 180-degree rotation. Grid C is not a rotation. The question asks which grid can be obtained by rotating. Both 0-degree and 180-degree rotations are rotations. However, usually, such questions imply a change in orientation (non-zero rotation). Grid B represents a non-trivial rotation (180 degrees) of the original grid.
Final Answer: B

Figure 18: Test case of 2D Rotation Task from Gemini-2.5-pro.

Mental Rotation-3D Rotation-Level 0

Prompt： You should first provide a reasoning process, then provide a single option(A, B, C or D) as the final answer. The reasoning process and the answer are enclosed within <think></think> and <answer></answer> tags, respectively, i.e., <think>reasoning process</think>, <answer>answer</answer>.
Question: The left image shows the original cube stack made of equal-sized small cubes. Which of the options on the right **cannot** be obtained by rotating the original cube stack? Please answer from options A, B, C, or D.
Choices:
A. A B. B
C. C D. All three other options are incorrect
Ground Truth Answer: A



Thinking Process: → **Confused description**

1. Analyze the Original Shape:
The original shape consists of 5 equal-sized cubes. Visual inspection suggests a central cube connected to four other cubes. Given the 3D isometric view, it's reasonable to assume the central cube (which might be hidden) has cubes attached to four of its faces. Let's assume the structure is a central cube C with neighbours North (N), South (S), East (E), and Top (T). This structure, {C, N, S, E, T}, is inherently 3-dimensional (non-planar). The central cube C has 4 connections (neighbours), and the other 4 cubes (N, S, E, T) each have only 1 connection (to C).

2. Analyze Shape A:
Shape A consists of 5 cubes. It appears non-planar. Let's determine its connectivity (number of neighbours for each cube). By visual inspection, it seems to have a 'backbone' of 3 cubes and two 'legs'. The central cube of the backbone appears connected to 3 other cubes. No cube appears connected to 4 others. The connectivity pattern seems to be (1 cube with 3 neighbours, 1 cube with 2 neighbours, 3 cubes with 1 neighbour). This differs from the Original's pattern (1 cube with 4 neighbours, 4 cubes with 1 neighbour). Therefore, A cannot be obtained by rotating the Original.

3. Analyze Shape B:
Shape B consists of 5 cubes. It appears non-planar. Let's determine its connectivity. It looks like an L-shape base of 3 cubes, with two cubes stacked vertically on two of the base cubes (one on the corner, one on an end). By visual inspection, the maximum number of neighbours any cube has is 2 (the corner cube of the L and the two cubes adjacent to it in the L). The connectivity pattern seems to be (3 cubes with 2 neighbours, 2 cubes with 1 neighbour). This differs from the Original's pattern (1x4, 4x1). Therefore, B cannot be obtained by rotating the Original.

4. Analyze Shape C:
Shape C consists of 5 cubes. This shape appears to be planar, meaning all cubes lie in a single layer. It forms a shape known as the 'P' pentomino (a 2x2 square with one cube attached to the side). A planar shape consists of cubes whose centers all lie on the same plane.

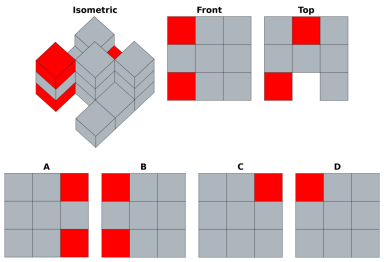
5. Compare Shapes with Original:
The Original shape {C, N, S, E, T} is non-planar because the Top cube (T) is in a different layer than the N, S, E cubes.
Shape C is planar. Rotation is a rigid transformation. It cannot change a non-planar object into a planar object.
Since the Original shape is non-planar and Shape C is planar, Shape C cannot be obtained by rotating the Original shape.
While analysis also suggested A and B cannot be obtained by rotating the Original (due to different connectivity patterns), the difference in planarity between the Original and C is a more fundamental geometric property preserved under rotation. Given that usually only one option is correct in such problems, the planarity difference provides the strongest reason.

6. Conclusion:
Shape C cannot be obtained by rotating the original cube stack because the original stack is non-planar and shape C is planar.
Final Answer: C

Figure 19: Test case of 3D Rotation Task from Gemini-2.5-pro.

Mental Rotation-3View Projection-Level 0

Prompt: You should first provide a reasoning process, then provide a single option(A, B, C or D) as the final answer. The reasoning process and the answer are enclosed within <think></think> and <answer></answer> tags, respectively, i.e., <think>reasoning process</think>, <answer>answer</answer>.
Question: The cube stack is made of equal-sized small cubes, mostly gray with a few red ones. The top row shows its isometric view, front view, and top view from left to right. Which image in the bottom row is the left view of the cube stack? Please answer from options A, B, C, or D.
Choices:
A. A B. B C. C D. D
Ground Truth Answer: A



Thinking Process:

1. Analyze the Isometric View:

The object is composed of stacks of cubes on a 3x3 base grid (viewed from top-down, let's label rows as Front, Middle, Back and columns as Left, Middle, Right).

Front-Left stack: 3 cubes high (Bottom Red, Middle Grey, Top Red).

Front-Middle stack: 3 cubes high (All Grey).

Front-Right stack: 1 cube high (Grey).

Middle-Left stack: 3 cubes high (All Grey).

Middle-Middle stack: 4 cubes high (3 Grey base + 1 Red on top).

Middle-Right stack: 1 cube high (Grey).

Back-Left stack: 3 cubes high (All Grey, implied by structure).

Back-Middle stack: 3 cubes high (All Grey, implied base for MM red cube).

Back-Right stack: 1 cube high (All Grey, implied by structure).

2. Determine the Left View:

The left view shows the object as seen from the left side. The 2D view grid will have columns representing the depth (Back, Middle, Front) and rows representing the height (Bottom, Middle, Top). We need to determine the color of the cube face visible at each position in this grid. Since the options are 3x3, we consider heights up to 3.

3. Looking from the left:

We see the leftmost faces of the stacks in each depth layer (Front, Middle, Back).

Front layer (Closest depth to the right in the left view): The leftmost stack is the Front-Left stack. It is 3 high (Bottom Red, Middle Grey, Top Red). Its left face is visible. So, the rightmost column of the left view is R (Top), G (Middle), R (Bottom).

Middle layer (Middle depth): The leftmost stack is the Middle-Left stack. It is 3 high (All Grey). Its left face is visible. So, the middle column of the left view is G (Top), G (Middle), G (Bottom).

Back layer (Furthest depth to the left in the left view): The leftmost stack is the Back-Left stack. It is 3 high (All Grey). Its left face is visible. So, the leftmost column of the left view is G (Top), G (Middle), G (Bottom).

4. Construct the 3x3 Left View Grid:

Row 3 (Top): Grey (Back), Grey (Middle), Red (Front)
Row 2 (Middle): Grey (Back), Grey (Middle), Grey (Front)
Row 1 (Bottom): Grey (Back), Grey (Middle), Red (Front)

5. Compare with Options:

Option A: Top Row (G, G, R), Middle Row (G, G, G), Bottom Row (G, G, R). This matches the derived left view.
Option B: Top Row (R, G, G), Middle Row (G, G, G), Bottom Row (R, G, G). Mismatch.
Option C: Top Row (G, G, R), Middle Row (G, G, G), Bottom Row (G, G, G). Mismatch.
Option D: Top Row (R, G, G), Middle Row (G, G, G), Bottom Row (G, G, G). Mismatch.

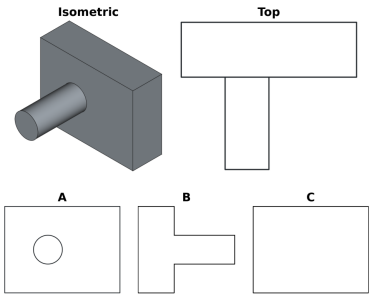
Note on discrepancies: The provided Front and Top views in the question appear inconsistent with the Isometric view (e.g., the height of the Front-Right stack, the position of the topmost Red cube, the empty space in the Top view). However, the question asks for the left view of the cube stack shown in the isometric view. Therefore, we derive the left view from the isometric drawing. The derivation based on the isometric view matches Option A.

Final Answer: A

Figure 20: Test case of Three-View Projection (Level 0) Task from Gemini-2.5-pro.

Mental Rotation-3View Projection-Level 1

Prompt： You should first provide a reasoning process, then provide a single option(A, B, C or D) as the final answer. The reasoning process and the answer are enclosed within <think></think> and <answer></answer> tags, respectively, i.e., <think>reasoning process</think>, <answer>answer</answer>.
Question: The top row shows the isometric view (left) and the top view (right) of a 3D model. Which image in the bottom row is the front view of the model? Please answer from options A, B, C, or D.
Choices:
A. A B. B C. C D. All three other options are incorrect
Ground Truth Answer: A



Thinking Process:

1. Understand the Task:
The goal is to identify the correct front view of the 3D object, given its isometric view and top view.

2. Analyze the Isometric View:
The isometric view shows a 3D representation of the object. It consists of a main rectangular prism (or block) and a cylinder attached to one of the larger faces of the prism. The cylinder extends outwards from this face.

3. Analyze the Top View:
The top view shows the object as seen from directly above. It appears as a 'T' shape. The wider horizontal part of the 'T' corresponds to the top surface of the rectangular prism. The narrower vertical part of the 'T' corresponds to the top view of the cylinder. This confirms that the cylinder is centered horizontally on the face it's attached to. It also shows the relative width of the prism and the diameter of the cylinder, as well as the length the cylinder protrudes.

4. Determine the Orientation for the Front View:
The isometric view typically shows the front, top, and one side (usually right). The top view is given. The front view would be looking at the face from which the cylinder protrudes.

5. Visualize the Front View:
Imagine looking directly at the face of the rectangular prism where the cylinder is attached. You would see the rectangular outline of that face. You would also see the end of the cylinder, which is a circle, superimposed onto that rectangular face.
From the isometric view, the cylinder appears centered vertically on the face.
From the top view, the cylinder is centered horizontally on the face.
Therefore, the front view should be a rectangle with a circle located in its center.

6. Compare with the Options:
Option A: Shows a rectangle with a circle inside, centered horizontally and vertically. This matches the visualization of the front view derived from the isometric and top views.
Option B: Shows a 'T' shape lying on its side. This does not match the front view. It resembles the top view rotated.
Option C: Shows only a rectangle. This would be the back view of the object, or the front view if the cylinder were absent.

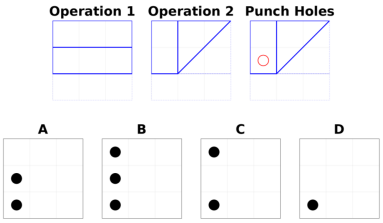
7. Conclusion:
Option A is the correct representation of the front view of the given 3D object.

Final Answer: A

Figure 21: Test case of Three-View Projection (Level 1) Task from Gemini-2.5-pro.

Mental Folding-Paper Folding-Level 0

Prompt: You should first provide a reasoning process, then provide a single option(A, B, C or D) as the final answer. The reasoning process and the answer are enclosed within <think></think> and <answer></answer> tags, respectively, i.e., <think>reasoning process</think>, <answer>answer</answer>.
Question: The original paper is a 3×3 grid paper. The images in the top row show the results of 2 consecutive folding operations on the grid paper. Folding operations include folding along horizontal, vertical, or 45-degree direction. The rightmost image in the top row shows the result after punching holes in the folded paper. Which image represents the appearance of the paper after unfolding the punched paper? Black solid circles represent grid cells with holes. Please answer from options A, B, C, or D.



Choices:
A. A B. B C. C D. D
Ground Truth Answer: A

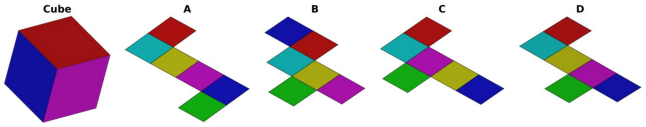
Thinking Process:
Step 1: Analyze the starting paper.
The paper is a 3x3 grid. Let's label the cells by (row, column) starting from the top-left as (1,1).
(1,1) (1,2) (1,3); (2,1) (2,2) (2,3); (3,1) (3,2) (3,3)
Step 2: Analyze Operation 1.
A horizontal fold is made across the middle of the original 3x3 grid. The diagram shows the bottom third (row 3) being folded upwards, lying underneath the middle third (row 2). The fold line is between row 2 and row 3.
The state after Operation 1 is a 3x2 grid shape (rows 1 and 2 are visible).
- Row 1 contains cells (1,1), (1,2), (1,3) (single layer).
- Row 2 (visible) contains cells (2,1), (2,2), (2,3) as the top layer, and cells (3,1), (3,2), (3,3) as the bottom layer, folded underneath.
Step 3: Analyze Operation 2.
Starting from the 3x2 shape after Operation 1. A diagonal fold is made within the right 2x2 section (columns 2 and 3). The fold line goes from the bottom-left corner of cell (2,2) to the top-right corner of cell (1,3). The top-right triangular part (involving cells (1,2), (1,3), (2,3)) is folded down/left onto the bottom-left triangular part (involving cells (1,2), (2,2), (2,3)).
The state after Operation 2 is shown. Note the layers:
- The leftmost column (original cells (1,1) and (2,1)/(3,1)) is unaffected by this fold.
Cell (1,1) is single layer. Cell (2,1) is the top layer, (3,1) is the bottom layer.
- The right part is now folded.
Step 4: Analyze Punch Holes.
A single hole is punched. Its location is in the bottom-left square of the shape shown after Operation 2. This corresponds to the cell location (2,1) in the configuration after Operation 1. Since this location (2,1) had two layers of paper from Operation 1 (original cell (2,1) on top and original cell (3,1) underneath), punching a hole here creates holes through both layers. This area was not affected by the fold in Operation 2.
Step 5: Unfold Operation 2.
We reverse Operation 2 by unfolding the diagonal fold. Since the punched hole was in the part not involved in this fold ((2,1) area), unfolding Operation 2 does not change the location of the holes or create new ones. The paper returns to the state after Operation 1 (the 3x2 shape). We have holes through both layers at position (2,1). That means original cell (2,1) has a hole, and original cell (3,1) has a hole.
Step 6: Unfold Operation 1.
We reverse Operation 1 by unfolding the bottom third (row 3) back down. The fold line was between row 2 and row 3. The holes are currently in cell (2,1) and cell (3,1). Unfolding separates these layers. The final unfolded 3x3 paper will have a hole in cell (2,1) (middle row, left column) and a hole in cell (3,1) (bottom row, left column).
Step 7: Compare with options.
- Option A shows holes at (2,1) and (3,1). This matches our result.
- Option B shows holes at (1,1), (2,1), (3,1). Incorrect.
- Option C shows holes at (1,1) and (3,1). Incorrect.
- Option D shows a hole at (3,1). Incorrect.
Therefore, the correct unfolded pattern is shown in Option A.
Final Answer: A

Figure 22: Test case of Paper Folding Task from Gemini-2.5-pro.

Mental Folding-Cube Unfolding-Level 0

Prompt： You should first provide a reasoning process, then provide a single option(A, B, C or D) as the final answer. The reasoning process and the answer are enclosed within <think></think> and <answer></answer> tags, respectively, i.e., <think>reasoning process</think>, <answer>answer</answer>.

Question: The left image shows a colored cube from a particular viewing angle. The options are nets (unfolded patterns) of the cube, which are folded upward to form the cube. Which net, when folded, **cannot** form the cube shown in the left image? Please answer from options A, B, C, or D.



Choices:

A. A B. B C. C D. D

Ground Truth Answer: C

Thinking Process:

1. Analyze the Target Cube:

The image shows a cube with three visible faces meeting at a vertex: Red (Top), Blue (Front), and Green (Right). This means Red, Blue, and Green must be adjacent faces in any valid net representation. → **Wrong color detection**

2. Analyze the Nets (A, B, C, D):

Each net represents an unfolded cube. To determine if a net can form the target cube, we need to check if the required faces (Red, Blue, Green) are adjacent and if their spatial relationship matches the target cube when folded. A key property of cube nets is identifying opposite faces. Faces are opposite if they are separated by one face in a straight line in the net, or by other standard rules depending on the net's shape.

3. Check Net A:

Identify opposite faces: Trace paths or visualize folding. In Net A, Teal is opposite Blue, Red is opposite Yellow, and Green is opposite Purple. → **Wrong opposite faces**

Check required adjacencies: The target cube requires Red, Blue, and Green to be adjacent. Since none of these are opposite pairs in Net A (Red/Yellow, Blue/Teal, Green/Purple), they can be adjacent.

Visualize folding: It's possible to fold Net A such that Red is Top, Blue is Front, and Green is Right. (For example, place Red top, Yellow bottom. Arrange neighbours Teal, Purple, Green around Red. Blue is opposite Teal. One possible fold results in Top:Red, Bottom:Yellow, Front:Purple, Right:Green, Back:Blue, Left:Teal. Rotating this gives the target view.)

Thus, Net A can form the cube.

4. Check Net B:

Identify opposite faces: In Net B, Yellow is opposite Red, Teal is opposite Green, and Purple is opposite Blue.

Check required adjacencies: The target cube requires Red, Blue, and Green to be adjacent. None of these are opposite pairs in Net B (Red/Yellow, Blue/Purple, Green/Teal), so they can be adjacent.

Visualize folding: It's possible to fold Net B to match the target cube. (For example, one fold gives Top:Blue, Bottom:Purple, Front:Green, Back:Teal, Left:Yellow, Right:Red. This can be rotated to match the target view: Red-Top, Blue-Front, Green-Right).

Thus, Net B can form the cube.

5. Check Net C:

Identify opposite faces: In Net C, Yellow is opposite Blue, Red is opposite Purple, and Green is opposite Teal.

Check required adjacencies: The target cube requires Red, Blue, and Green to be adjacent. None of these are opposite pairs in Net C (Red/Purple, Blue/Yellow, Green/Teal), so they can be adjacent.

Visualize folding: It's possible to fold Net C to match the target cube. (For example, one fold gives Top:Teal, Bottom:Green, Front:Red, Back:Purple, Left:Yellow, Right:Blue. This configuration can be rotated into the target view Red-Top, Blue-Front, Green-Right).

Thus, Net C can form the cube.

6. Check Net D:

Identify opposite faces: In Net D, Blue is opposite Red, Purple is opposite Teal, and Green is opposite Yellow.

Check required adjacencies: The target cube shows Red, Blue, and Green meeting at a vertex, which requires Red and Blue to be adjacent. However, in Net D, Red and Blue are opposite faces.

Conclusion:

Opposite faces cannot meet at a vertex or share an edge. Since Net D places Red and Blue opposite each other, it is impossible to fold Net D into the cube shown on the left where Red and Blue are adjacent. Thus, Net D cannot form the cube.

7. Final Answer:

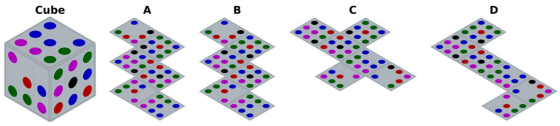
Based on the analysis, Nets A, B, and C can form the target cube, while Net D cannot because it places two adjacent faces from the target cube (Red and Blue) in opposite positions.

Final Answer: D

Figure 23: Test case of Cube Unfolding Task from Gemini-2.5-pro.

Mental Folding-Cube Unfolding-Level 2

Prompt: You should first provide a reasoning process, then provide a single option(A, B, C or D) as the final answer. The reasoning process and the answer are enclosed within <think></think> and <answer></answer> tags, respectively, i.e., <think>reasoning process</think>, <answer>answer</answer>.
Question: The left image shows a cube with different patterns on its six faces from a particular viewing angle. The options are nets (unfolded patterns) of the cube, which are folded upward to form the cube. Which net, when folded, cannot form the cube shown in the left image? Please answer from options A, B, C, or D.
Choices:
A. A B. B C. C D. D
Ground Truth Answer: A

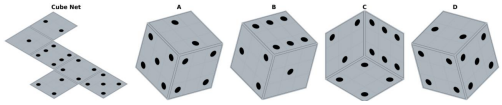


Thinking Process:
Let's analyze the visible faces of the cube:
Top face (T): Pattern of Blue, Purple, Green dots. Specifically, (B B B / P . P / G G G). The bottom-right corner dot is Green (G).
Front face (F): Pattern of Purple, Green, Red, Blue dots. Specifically, (P G / G R / G B). The top-right corner dot is Green (G).
Right face (R): Pattern of Green, Purple, Black, Blue, Red dots. Specifically, (G P . / G Bl B / R B R). The top-left corner dot is Green (G).
The common vertex where these three faces (T, F, R) meet has a Green dot from each face. So, this vertex is (G, G, G).
Now let's examine each net:
Net A:
Let's identify the patterns T, F, R in Net A if they exist.
Face 2 of Net A matches T. (B B B / P . P / G G G)
Face 3 of Net A matches F. (P G / G R / G B)
Now, we need to find face R: (G P . / G Bl B / R B R).
Let's list the patterns of the other faces in Net A:
Face 1: (B G . / M R . / Bl B .) (M=Magenta/Purple). Face 4: (Bl B . / P R . / G P .) Face 5: (P Bl . / B R . / G M .) Face 6: (P G . / B M . / B R .) Comparing R with faces 1, 4, 5, 6: None of them match the pattern of R. Since face R is not present in Net A, Net A cannot form the cube shown.
Let's verify other nets, assuming this is the primary criterion. If other nets contain T, F, R and can be arranged correctly, then A is the answer.
Net B:
Face 2 is T. Face 3 is F. Face 4 is R. These faces are arranged such that T is above F, and R is to the right of F. This is the standard T, F, R configuration.
Let's check the common vertex: T (face 2): bottom-right dot is G. F (face 3): top-right dot is G. R (face 4): top-left dot is G. So, the common vertex is (G,G,G). This is consistent with the cube. Thus, Net B can form the cube.
Net C:
Face 1 is T. Face 3 is F. Face 4 is R. These faces are arranged such that T is above F, and R is to the right of F. This is the standard T, F, R configuration. Let's check the common vertex:
T (face 1): bottom-right dot is G. F (face 3): top-right dot is G. R (face 4): top-left dot is G. So, the common vertex is (G,G,G). This is consistent with the cube. Thus, Net C can form the cube.
Net D:
The topmost face is R. The face below it is T. The face to the left of T is F. So, R, T, F are present. Arrangement when folded: R is Top, T is Front, F is Left.
Let's find the common vertex for this configuration: R (Top): bottom-left dot is R (from R B R). T (Front): top-left dot is B (from B B B). F (Left): top-right dot is G (from P G). So, the common vertex is (R, B, G). For this to match the (G,G,G) vertex of the cube, we would need R=G and B=G, which is false as the colors are distinct. Therefore, Net D cannot form the cube shown with this specific corner.
However, the primary reason for Net A being unable to form the cube is the absence of face R. This is a more fundamental issue than corner matching if all faces are present. If a face is missing, it absolutely cannot form the cube.
If we assume that there's only one incorrect option, and the "missing face" criterion for A is decisive, then A is the answer. The issues with D (and potential edge matching issues with B and C) might be secondary or based on misinterpretations of "forming the cube". But missing a face is a definitive failure. Let's reconfirm the "missing face R in Net A" argument. R = (G P . / G Bl B / R B R)
Net A, Face 1: (B G / M R / Bl B) - No match (e.g., G at top-left of R, B at top-left of Face 1)
Net A, Face 4: (Bl B / P R / G P) - No match (e.g., G at top-left of R, Bl at top-left of Face 4)
Net A, Face 5: (P Bl / B R / G M) - No match (e.g., G at top-left of R, P at top-left of Face 5)
Net A, Face 6: (P G / B M / B R) - No match (e.g., G at top-left of R, P at top-left of Face 6)
The conclusion that face R is not present in Net A seems correct. Therefore, Net A cannot form the cube. The final answer is $\\boxed{A}$.
Final Answer: A

Figure 24: Test case of Cube Unfolding Task from Gemini-2.5-pro.

Mental Folding-Cube Reconstruction-Level 1

Prompt： You should first provide a reasoning process, then provide a single option(A, B, C or D) as the final answer. The reasoning process and the answer are enclosed within <think></think> and <answer></answer> tags, respectively, i.e., <think>reasoning process</think>, <answer>answer</answer>.
Question: The left image shows the net (unfolded pattern) of a cube, with six faces having different patterns. The net is folded upward to form a cube. From an axonometric (3D) viewing angle of the cube, which combination of adjacent patterns is possible to see? Please answer from options A, B, C, or D.
Choices:
A. A B. B C. C D. D
Ground Truth Answer: A

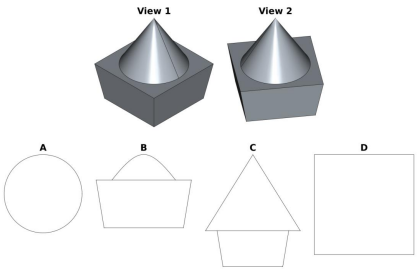


Thinking Process:
1. Identify the faces and their opposite pairs:
Label the faces in the net based on the number of dots: Top row: 4; Middle row: 3, 2, 6; Bottom row: 1, 5 (where 1 is below 3, and 5 is below 6) → **Wrong pattern detection**
Determine opposite faces by folding mentally or noting faces separated by one other face in a straight line (or around corners):
The face with 4 dots is opposite the face with 1 dot.
The face with 3 dots is opposite the face with 6 dots.
The face with 2 dots is opposite the face with 5 dots.
2. Evaluate the options based on opposite faces:
A cube view can only show faces that are adjacent to each other. Opposite faces can never be seen together.
Option A: Shows faces with 4 dots, 3 dots, and 1 dot. Since 4 and 1 are opposite faces, this view is impossible.
Option B: Shows faces with 5 dots, 3 dots, and 2 dots. Since 2 and 5 are opposite faces, this view is impossible.
Option D: Shows faces with 3 dots, 6 dots, and 5 dots. Since 3 and 6 are opposite faces, this view is impossible.
Option C: Shows faces with 2 dots, 4 dots, and 3 dots. None of these pairs (2-4, 2-3, 4-3) are opposite pairs (Opposites: 2-5, 4-1, 3-6). Therefore, this combination of adjacent faces is possible.
3. Verify Option C with adjacency and orientation (Optional but recommended):
Consider the face with 2 dots (let's call it F2) as the front face.
In the net, the face with 4 dots (F4) is attached to the top edge of F2 (relative to the net layout shown). When folded, F4 becomes the top face.
In the net, the face with 3 dots (F3) is attached to the left edge of F2. When folded, F3 becomes the left face.
So, the faces 2, 4, and 3 can indeed meet at the top-front-left vertex.
Check orientations shown in C:
Front (2 dots): The dots run diagonally from top-left to bottom-right. This matches the orientation on the net relative to the edges connected to F4 and F3.
Top (4 dots): The dots form a square. This matches the orientation on the net relative to the edge connected to F2.
Left (3 dots): The dots run vertically. The middle dot is closer to the edge shared with F2 (the front face). This matches the orientation on the net relative to the edge connected to F2.
The adjacencies and orientations in Option C are consistent with the net.
Conclusion:
Options A, B, and D are impossible because they show opposite faces.
Option C shows three faces that are adjacent in the net and whose orientations are consistent with folding.
Final Answer: C

Figure 25: Test case of Cube Reconstruction Task from Gemini-2.5-pro.

Visual Penetration-Cross Section-Level 0

Prompt： You should first provide a reasoning process, then provide a single option(A, B, C or D) as the final answer. The reasoning process and the answer are enclosed within <think></think> and <answer></answer> tags, respectively, i.e., <think>reasoning process</think>, <answer>answer</answer>.
Question: The top row shows the combined shape viewed from two different angles. The shape consists of a cone on top of a square frustum. Which of the following images cannot be a cross-section of the shape? Please answer from options A, B, C, or D.
Choices:
A. A B. B C. C D. D
Ground Truth Answer: C



Thinking Process:

1. Analyze the 3D Shape:
The object consists of a cone positioned inside a circular depression on the top face of a square frustum. The base of the cone is circular and lies on the plane of the top face of the frustum. The frustum has a square base and tapers upwards to a smaller square top face. The circular depression (and thus the cone's base) is contained entirely within this top square face.

2. Analyze Option A (Circle):
A horizontal cross-section plane passing through the cone part (above the frustum's top face) will intersect the cone to form a circle. This is a possible cross-section.

3. Analyze Option D (Square):
A horizontal cross-section plane passing through the frustum part below the level of the cone's base (and below the depression) will intersect the frustum to form a square. This is a possible cross-section.

4. Analyze Option C (Triangle on Trapezoid):
A vertical cross-section plane passing through the central axis of both the cone and the frustum, and parallel to two opposite side faces of the square frustum, will intersect the cone to form a triangle and the frustum to form an isosceles trapezoid. The triangle will sit atop the trapezoid. This is a possible cross-section.

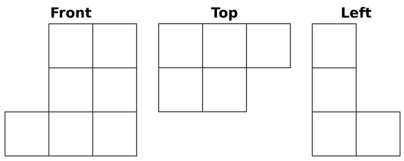
5. Analyze Option B (Curve on Trapezoid): → Correct and detailed analysis
This shape represents a cross-section formed by a plane cutting through both the cone and the frustum. The curved top part represents the intersection with the cone. If the plane is vertical and off-center, this curve is typically a hyperbola segment. If the plane is tilted, it could be an ellipse or parabola segment. The trapezoidal bottom part represents the intersection with the frustum. A vertical plane cutting the frustum can result in a rectangle (if parallel to a side face) or a trapezoid (if cutting diagonally or at an angle relative to the side faces). A tilted plane could also result in a trapezoid.
The key feature to examine is the junction between the curved top and the trapezoidal base. This junction line represents the intersection of the cutting plane with the plane containing the cone's base (the top surface of the frustum). The width of the curved section at its base corresponds to the length of the chord formed by the cutting plane intersecting the cone's circular base. The width of the top side of the trapezoid corresponds to the length of the line segment formed by the cutting plane intersecting the frustum's top square face. Since the cone's circular base is strictly smaller than and contained within the frustum's top square face (as seen in Views 1 and 2), any cutting plane that intersects both the circle and the square will create a chord across the circle that is strictly shorter than the segment across the square (unless the plane is merely tangent to the circle). Therefore, the base width of the curved (cone) section must be strictly narrower than the top width of the trapezoidal (frustum) section. However, in option B, the base width of the curve appears to be equal to the top width of the trapezoid. This configuration is geometrically impossible given that the cone's base circle lies inside the top square face of the frustum.

6. Conclusion: → Incorrect identification of the relative length relationship of options
Options A, C, and D represent possible cross-sections of the described 3D object.
Option B depicts a configuration where the width of the cone section matches the width of the frustum section at the joining plane, which contradicts the geometry shown in Views 1 and 2 where the cone base is smaller than the top face of the frustum.
Therefore, B cannot be a cross-section.
Final Answer: B

Figure 26: Test case of Cross-Section Task from Gemini-2.5-pro.

Visual Penetration-Cube Counting-Level 1

Prompt: You should first provide a reasoning process, then provide a single option(A, B, C or D) as the final answer. The reasoning process and the answer are enclosed within <think></think> and <answer></answer> tags, respectively, i.e., <think>reasoning process</think>, <answer>answer</answer>.
Question: Given three views, what is the **minimum** number of cubes required to satisfy the constraints shown in the images? Please answer from options A, B, C, or D.
Choices:
A. All three other options are incorrect B. 9 C. 11 D. 10
Ground Truth Answer: B



Thinking Process: → **Incorrect details, but based on the wrong details, the analysis is correct**

1. Understand the Goal:
The objective is to find the minimum number of unit cubes required to build a 3D structure whose orthographic projections (Front, Top, Left views) match the given diagrams.

2. Analyze the Top View:
The Top view shows the layout of the cubes on the base level (or the projection of all cubes onto the base). It occupies a 2x3 grid area, with the cell in the top-right corner (row 1, column 3) and the bottom-right cell (row 2, column 3) being potentially different.
The cells occupied are: Top row (back): (1,1), (1,2), (1,3); Bottom row (front): (2,1), (2,2); Cell (2,3) is empty.
This means there must be at least one cube stacked at each of the locations (1,1), (1,2), (1,3), (2,1), and (2,2).
This establishes a minimum of 5 cubes based on the footprint.

3. Analyze the Front View: → **Wrong order**
The Front view shows the maximum height of the stacks in each column (left-to-right).
Column 1 (left): Height is 3. Corresponds to locations (1,1) and (2,1). So, max(Height(1,1), Height(2,1)) = 3.
Column 2 (middle): Height is 3. Corresponds to locations (1,2) and (2,2). So, max(Height(1,2), Height(2,2)) = 3.
Column 3 (right): Height is 1. Corresponds to location (1,3) (since (2,3) is empty according to the top view). So, Height(1,3) = 1.

4. Analyze the Left View:
The Left view shows the maximum height of the stacks in each row (front-to-back, viewed from the left).
Back Row (left side of the Left view): Height is 3. Corresponds to locations (1,1), (1,2), (1,3). So, max(Height(1,1), Height(1,2), Height(1,3)) = 3.
Front Row (right side of the Left view): Height is 1. Corresponds to locations (2,1), (2,2). So, max(Height(2,1), Height(2,2)) = 1.

5. Combine Information and Minimize Cubes:
Let H(r, c) denote the height of the stack at row 'r' and column 'c'.
From Top view: H(1,1)>=1, H(1,2)>=1, H(1,3)>=1, H(2,1)>=1, H(2,2)>=1.
From Front view: Height(1,3) = 1.
From Left view: max(H(2,1), H(2,2)) = 1.
Since we know H(2,1)>=1 and H(2,2)>=1, this forces H(2,1) = 1 and H(2,2) = 1.
Now use the Front view constraints again:
max(H(1,1), H(2,1)) = 3. Since H(2,1) = 1, we must have H(1,1) = 3 to satisfy the maximum height and minimize cubes.
max(H(1,2), H(2,2)) = 3. Since H(2,2) = 1, we must have H(1,2) = 3 to satisfy the maximum height and minimize cubes.
Check the remaining Left view constraint: max(H(1,1), H(1,2), H(1,3)) = max(3, 3, 1) = 3.
This is satisfied by our determined heights.

6. Calculate Total Cubes: → **The error does not affect the final numerical result**
We have determined the minimum required heights for each stack: H(1,1) = 3, H(1,2) = 3, H(1,3) = 1, H(2,1) = 1, H(2,2) = 1
Total number of cubes = H(1,1) + H(1,2) + H(1,3) + H(2,1) + H(2,2)
Total cubes = 3 + 3 + 1 + 1 + 1 = 9

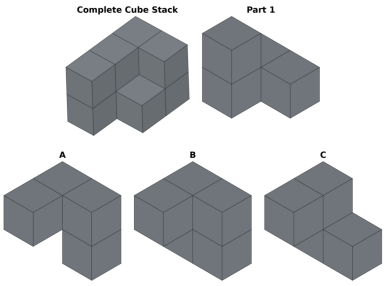
7. Compare with Options:
The calculated minimum number of cubes is 9. This matches option B.

Final Answer: B

Figure 27: Test case of Cube Counting Task from Gemini-2.5-pro.

Visual Penetration-Cube Assembly-Level 0

Prompt： You should first provide a reasoning process, then provide a single option(A, B, C or D) as the final answer. The reasoning process and the answer are enclosed within <think></think> and <answer></answer> tags, respectively, i.e., <think>reasoning process</think>, <answer>answer</answer>.
Question: The top left image shows the original complete cube stack made of equal-sized cubes. It can be formed by combining the small cube stack on the right(part 1) with one of the options below. Which option completes the original cube stack? Please answer from options A, B, C, or D.
Choices:
A. A B. B C. C D. All three other options are incorrect
Ground Truth Answer: B



Thinking Process: → **Most of the identifications of the number of cube stacks are incorrect**

1. Analyze the Complete Cube Stack:

Visualize the stack layer by layer or column by column. Let's use columns from back-left to front-right (like reading order if viewed from slightly above).

Back-Left Column: 2 cubes high.
Back-Right Column: 2 cubes high.
Middle-Left Column: 2 cubes high.
Middle-Right Column: 1 cube high (only bottom layer).
Front-Left Column: 1 cube high (only bottom layer).
Front-Right Column: 1 cube high (only bottom layer).
Total Cubes = 2 + 2 + 2 + 1 + 1 + 1 = 9 cubes.

2. Analyze Part 1:

Using the same column description:
Back-Left Column: 2 cubes high.
Back-Right Column: 0 cubes.
Middle-Left Column: 1 cube high (only bottom layer).
Middle-Right Column: 1 cube high (only bottom layer).
Front-Left Column: 1 cube high (only bottom layer).
Front-Right Column: 0 cubes.
Total Cubes = 2 + 0 + 1 + 1 + 1 + 0 = 5 cubes.

3. Determine the Missing Part (Complete Stack - Part 1): → **Incorrect approach to calculation**

Subtract the cubes in Part 1 from the Complete Stack for each column position:
Back-Left Column: 2 (Complete) - 2 (Part 1) = 0 cubes.
Back-Right Column: 2 (Complete) - 0 (Part 1) = 2 cubes (column 2 high).
Middle-Left Column: 2 (Complete) - 1 (Part 1) = 1 cube (must be the top cube).
Middle-Right Column: 1 (Complete) - 1 (Part 1) = 0 cubes.
Front-Left Column: 1 (Complete) - 1 (Part 1) = 0 cubes.
Front-Right Column: 1 (Complete) - 0 (Part 1) = 1 cube (must be the bottom cube).
Total Cubes in Missing Part = 0 + 2 + 1 + 0 + 0 + 1 = 4 cubes.
Structure of Missing Part: A column of 2 cubes in the back-right position, a single cube in the middle-left top position, and a single cube in the front-right bottom position.

4. Analyze the Options:

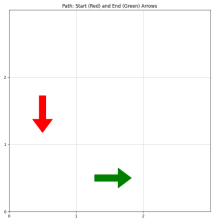
Option A: Has a column of 2 in the back-left, 1 in middle-left (bottom), 1 in front-left (bottom). Total 4 cubes. Does not match the required structure.
Option B: Has a column of 2 in back-left, 1 in back-right (bottom), 1 in middle-left (bottom), 1 in middle-right (bottom). Total 5 cubes. Incorrect number of cubes and structure.
Option C: Has a column of 2 in the back-right position, a single cube that aligns with the middle-left top position, and a single cube that aligns with the front-right bottom position. Total 4 cubes. This matches the required structure derived in step 3.

5. Conclusion:

Option C is the missing part that combines with Part 1 to form the Complete Cube Stack.
Final Answer: C

Figure 28: Test case of Cube Assembly Task from Gemini-2.5-pro.

Mental Animation-Arrow Moving-Level 0



Prompt： You should first provide a reasoning process, then provide a single option(A, B, C or D) as the final answer. The reasoning process and the answer are enclosed within <think></think> and <answer></answer> tags, respectively, i.e., <think>reasoning process</think>, <answer>answer</answer>.
Question: In the diagram, the red arrow is the initial arrow, and the green arrow is the final arrow. The arrow can move in four directions (forward, backward, left, right), where 'forward' always refers to the current direction the arrow is pointing. After each movement, the arrow's direction is updated to the direction of movement. Which of the following paths can make the arrow move from the starting position to the ending position? Please answer from options A, B, C, or D.
Choices:

A. (Left, 2 units)--(Left, 1 unit)
B. (Forward, 1 unit)--(Backward, 1 unit)
C. (Forward, 1 unit)--(Backward, 2 units)
D. (Forward, 1 unit)--(Left, 1 unit)\

Ground Truth Answer: D

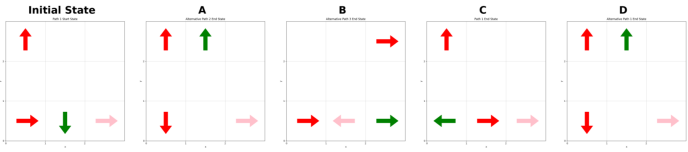
Thinking Process:

1. Analyze Initial State:
The red arrow (start) is located in the top-left quadrant. Its center appears to be at coordinates approximately (0.5, 1.5).
The direction of the red arrow is pointing downwards (South).
2. Analyze Final State:
The green arrow (end) is located in the bottom-right quadrant. Its center appears to be at coordinates approximately (1.5, 0.5).
The direction of the green arrow is pointing to the right (East).
3. Define Movement Rules:
The arrow can move Forward, Backward, Left, or Right relative to its current orientation. After each move, the arrow's orientation changes to the direction of the move. Cardinal Directions relative to current orientation:
If facing North (Up): Forward=Up, Backward=Down, Left=West, Right=East.
If facing South (Down): Forward=Down, Backward=Up, Left=East, Right=West.
If facing East (Right): Forward=Right, Backward=Left, Left=Up, Right=Down.
If facing West (Left): Forward=Left, Backward=Right, Left=Down, Right=Up. → **Fully comprehend the rules**
4. Evaluate Option A: (Left, 2 units) -- (Left, 1 unit)
Start: Position (0.5, 1.5), Direction Down.
Move 1: Left, 2 units. Relative to Down, Left is East (Right). Move 2 units East.
-- New Position: (0.5 + 2, 1.5) = (2.5, 1.5). New Direction: East (Right).
Move 2: Left, 1 unit. Relative to East (Right), Left is North (Up). Move 1 unit North.
-- New Position: (2.5, 1.5 + 1) = (2.5, 2.5).
New Direction: North (Up).
Result: Position (2.5, 2.5), Direction Up. Does not match the final state.
5. Evaluate Option B: (Forward, 1 unit) -- (Backward, 1 unit)
Start: Position (0.5, 1.5), Direction Down.
Move 1: Forward, 1 unit. Relative to Down, Forward is Down. Move 1 unit Down.
-- New Position: (0.5, 1.5 - 1) = (0.5, 0.5). New Direction: Down.
Move 2: Backward, 1 unit. Relative to Down, Backward is Up. Move 1 unit Up.
-- New Position: (0.5, 0.5 + 1) = (0.5, 1.5). New Direction: Up.
Result: Position (0.5, 1.5), Direction Up. Does not match the final state.
6. Evaluate Option C: (Forward, 1 unit) -- (Backward, 2 units)
Start: Position (0.5, 1.5), Direction Down.
Move 1: Forward, 1 unit. Relative to Down, Forward is Down. Move 1 unit Down.
-- New Position: (0.5, 1.5 - 1) = (0.5, 0.5). New Direction: Down.
Move 2: Backward, 2 units. Relative to Down, Backward is Up. Move 2 units Up.
-- New Position: (0.5, 0.5 + 2) = (0.5, 2.5). New Direction: Up.
Result: Position (0.5, 2.5), Direction Up. Does not match the final state.
7. Evaluate Option D: (Forward, 1 unit) -- (Left, 1 unit)
Start: Position (0.5, 1.5), Direction Down.
Move 1: Forward, 1 unit. Relative to Down, Forward is Down. Move 1 unit Down.
-- New Position: (0.5, 1.5 - 1) = (0.5, 0.5). New Direction: Down.
Move 2: Left, 1 unit. Relative to Down, Left is East (Right). Move 1 unit East.
-- New Position: (0.5 + 1, 0.5) = (1.5, 0.5). New Direction: East (Right).
Result: Position (1.5, 0.5), Direction East (Right). Matches the final state (green arrow).
8. Conclusion: Option D correctly describes the path from the red arrow to the green arrow.
Final Answer: D

Figure 29: Test case of Arrow Moving Task (Level 0) from Gemini-2.5-pro.

Mental Animation-Arrow Moving-Level 1

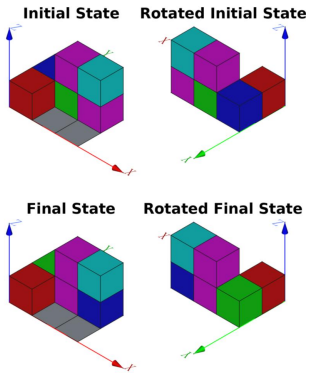
Prompt： You should first provide a reasoning process, then provide a single option(A, B, C or D) as the final answer. The reasoning process and the answer are enclosed within <think></think> and <answer></answer> tags, respectively, i.e., <think>reasoning process</think>, <answer>answer</answer>.
Question: The left image shows the initial state. Arrows can move in four directions (forward, backward, left, right), where 'forward' always refers to the current direction the arrow is pointing. After each movement, the arrow's direction is updated to the direction of movement. If the target position is empty, the arrow can move there directly; otherwise, it needs to swap with the arrow at the target position, and both arrows' movements should satisfy the aforementioned requirements. After the transformations '((0, 0) Left, 2 units)--((1, 0) Right, 1 unit)', which state from the options can be reached? Please answer from options A, B, C, or D.
Choices:
A. A B. B C. C D. D
Ground Truth Answer: C



Thinking Process:
1. Analyze Initial State:
Grid coordinates range from x=0 to x=2 and y=0 to y=2 (implicitly, based on positions).
Arrow 1: Red at (0, 0), Orientation: Right (East).
Arrow 2: Green at (1, 0), Orientation: Down (South).
Arrow 3: Pink at (2, 0), Orientation: Right (East).
Arrow 4: Red at (0, 2), Orientation: Up (North).
2. Analyze Transformation 1: ((0, 0) Left, 2 units)
Identify the arrow: The arrow starting at (0, 0) is Arrow 1 (Red, facing Right).
Determine the movement direction: 'Left' relative to the arrow's orientation (Right). If facing Right, 'Left' is Up (North).
Determine the target position: Move 2 units Up from (0, 0). Target is (0, 0 + 2) = (0, 2).
Check the target position: Position (0, 2) is occupied by Arrow 4 (Red, facing Up).
Execute swap: Since the target is occupied, Arrow 1 and Arrow 4 must swap positions.
Arrow 1 moves from (0, 0) to (0, 2). Movement direction: Up. New orientation for Arrow 1: Up.
Arrow 4 must move from (0, 2) to (0, 0). Movement direction: Down.
Check if Arrow 4's move is valid: Arrow 4 is facing Up.
Moving Down is 'Backward'. This is a valid relative move.
New orientation for Arrow 4: Down (direction of movement).
State after Transformation 1: Arrow 4: Red at (0, 0), Orientation: Down. Arrow 2: Green at (1, 0), Orientation: Down. Arrow 3: Pink at (2, 0), Orientation: Right. Arrow 1: Red at (0, 2), Orientation: Up.
3. Analyze Transformation 2: ((1, 0) Right, 1 unit)
Identify the arrow: The arrow currently at (1, 0) is Arrow 2 (Green, facing Down).
Determine the movement direction: 'Right' relative to the arrow's orientation (Down). If facing Down, 'Right' is Left (West).
Determine the target position: Move 1 unit Left from (1, 0). Target is (1 - 1, 0) = (0, 0).
Check the target position: Position (0, 0) is occupied by Arrow 4 (Red, facing Down).
Execute swap: Since the target is occupied, Arrow 2 and Arrow 4 must swap positions.
Arrow 2 moves from (1, 0) to (0, 0). Movement direction: Left. New orientation for Arrow 2: Left.
Arrow 4 must move from (0, 0) to (1, 0). Movement direction: Right.
Check if Arrow 4's move is valid: Arrow 4 is facing Down. Moving Right is 'Right'. This is a valid relative move.
New orientation for Arrow 4: Right (direction of movement).
State after Transformation 2 (Final State): Arrow 2: Green at (0, 0), Orientation: Left. Arrow 4: Red at (1, 0), Orientation: Right. Arrow 3: Pink at (2, 0), Orientation: Right. Arrow 1: Red at (0, 2), Orientation: Up.
4. Compare Final State with Options:
Option A: Incorrect arrows/positions/orientations.
Option B: Incorrect arrows/positions/orientations.
Option C: Green at (0, 0), facing Left. (Match); Red at (1, 0), facing Right. (Match); Pink at (2, 0), facing Right. (Match); Red at (0, 2), facing Up. (Match). This matches the derived final state.
Option D: Incorrect arrows/positions/orientations.
5. Conclusion: The state reached after the transformations matches Option C.
Final Answer: C

Figure 30: Test case of Arrow Moving Task (Level 1) from Gemini-2.5-pro.

Mental Animation-Block Moving-Level 1



Prompt: You should first provide a reasoning process, then provide a single option(A, B, C or D) as the final answer. The reasoning process and the answer are enclosed within <think></think> and <answer></answer> tags, respectively, i.e., <think>reasoning process</think>, <answer>answer</answer>.
Question: The top row of images shows different views of the initial state of a cube stack, while the bottom row shows different views of the final state after transformation. During the transformation process, blocks can move one unit in any direction (forward, backward, left, right, up, down). If the target position is empty, the block can move there directly; if the target position already has a block, they swap places. Blocks cannot float in the air. If a block is moved away from a position, any block above it will fall down until reaching a supporting surface. The xyz axes are shown in the diagram, and each block's position can be precisely identified using coordinates (x1,y1,z1). Which of the following transformation sequences can change the cube stack from the initial state to the final state shown in the diagram? Please answer from options A, B, C, or D.
Choices:

A. (1, 1, 1) x- -- (0, 0, 0) x+
B. (1, 1, 0) x- -- (1, 1, 0) x+ -- (1, 1, 1) z-
C. (1, 1, 1) x+ -- (2, 1, 0) x- -- (1, 1, 1) x+
D. (0, 0, 0) x+ -- (2, 1, 0) x-

Ground Truth Answer: B

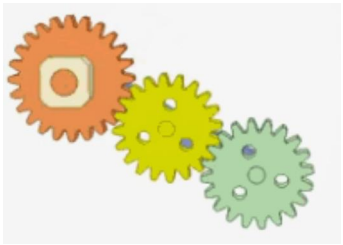
Thinking Process:

1. Analyze States:
Identify block positions (R, G, DB, P, C) in Initial and Final states using a consistent coordinate system (e.g., R at 0,0,0).
Initial: R(0,0,0), G(1,0,0), DB(0,1,0), P(1,1,0), C(1,1,1). → **Missing 1 block and the positions of blocks are wrong**
Final: R(0,0,0), DB(1,0,0), G(0,1,0), P(1,1,0), C(1,1,1).
The change is G swaps with DB.
2. Analyze Rules:
Moves are 1 unit. Target empty = move. Target occupied = swap. Gravity applies if support is removed.
3. Evaluate Options:
A: (1,1,1)x- moves C to (0,1,1). (0,0,0)x+ swaps R and G.
Result: G(0,0,0), R(1,0,0), DB(0,1,0), P(1,1,0), C(0,1,1). Incorrect state.
B: (1,1,0)x- swaps P and DB. (1,1,0)x+ moves DB(now at 1,1,0) to (2,1,0), causing C(1,1,1) to fall to (1,1,0). (1,1,1)z- attempts to move from empty (1,1,1). Invalid.
C: (1,1,1)x+ moves C(1,1,1) to (2,1,1), causing C to fall to (2,1,0). (2,1,0)x- swaps C(2,1,0) and P(1,1,0). (1,1,1)x+ attempts to move from empty (1,1,1). Invalid.
D: (0,0,0)x+ swaps R(0,0,0) and G(1,0,0). State: G(0,0,0), R(1,0,0), DB(0,1,0), P(1,1,0), C(1,1,1). (2,1,0)x- attempts to move from empty (2,1,0). Invalid.
4. Conclusion:
All options appear invalid or incorrect based on the rules. B, C, D reference empty coordinates. A leads to the wrong state.
5. Reassess:
The problem likely contains an error. However, if forced to choose, D's first move is valid. The required transformation (swapping G and DB) needs multiple steps. I found a 3-step sequence: (0,1,0)y-, (0,0,0)x+, (0,0,0)y+. Another 3-step sequence starting like D: (0,0,0)x+, (0,1,0)x+, (0,0,0)y+. Since Option D has only 2 steps and the second is invalid, it cannot be correct as written. But in multiple-choice questions with flaws, sometimes the intended answer has the most 'reasonable' start or structure. Option D starts with a valid swap. Without clarification, choosing D assumes it's the intended answer despite the error.
Final Answer: D

Figure 31: Test case of Block Moving Task from Gemini-2.5-pro.

Mental Animation-Mechanical System-Level 0

Prompt: You should first provide a reasoning process, then provide a single option(A, B, C or D) as the final answer. The reasoning process and the answer are enclosed within <think></think> and <answer></answer> tags, respectively, i.e., <think>reasoning process</think>, <answer>answer</answer>.
Question: In the coaxial system shown, the orange gear is fixed, and the yellow and green gears are identical in shape. If the yellow gear rotates clockwise, what is the motion of the green gear relative to the orange gear? Please choose from options A, B, C, or D.



Choices:

A. Clockwise rotation, clockwise revolution
B. Clockwise rotation, counterclockwise revolution
C. No rotation, clockwise revolution
D. No rotation, counterclockwise revolution

Ground Truth Answer: C

Thinking Process: **Can't intuitively thinking without theoretical derivation**

1. Define the system and motion:

We have an epicyclic gear train. The orange gear (Sun) is fixed (ω_o = 0). The yellow gear (Planet 1) meshes with the orange gear and rotates clockwise (ω_y < 0). The green gear (Planet 2) meshes with the yellow gear, and is identical to it (N_g = N_y). The yellow and green gears are carried by an implicit arm that revolves around the center of the orange gear. We need to find the absolute rotation (spin) and revolution (orbit) of the green gear relative to the fixed orange gear.

2. Determine the arm's revolution (ω_arm):

The relationship between the sun, planet, and arm is given by the Willis equation: (ω_y - ω_arm) / (ω_o - ω_arm) = -N_o / N_y. Substituting ω_o = 0: (ω_y - ω_arm) / (-ω_arm) = -N_o / N_y. Simplifying: ω_y - ω_arm = (-ω_arm) (-N_o / N_y) = ω_arm (N_o / N_y). Rearranging for ω_y: ω_y = ω_arm + ω_arm (N_o / N_y) = ω_arm (1 + N_o / N_y). We are given that ω_y is clockwise (negative). Since N_o and N_y are positive tooth counts, the term (1 + N_o / N_y) is positive. Therefore, for ω_y to be negative, ω_arm must also be negative. This means the arm, carrying the yellow and green gears, revolves clockwise around the orange gear.

3. Determine the green gear's rotation (ω_g):

Now consider the meshing between the yellow and green gears. They are carried by the same arm. The relative motion formula applies: (ω_g - ω_arm) / (ω_y - ω_arm) = -N_y / N_g. Since the yellow and green gears are identical, N_y = N_g. The ratio -N_y / N_g = -1. So, (ω_g - ω_arm) / (ω_y - ω_arm) = -1. ω_g - ω_arm = -(ω_y - ω_arm) = -ω_y + ω_arm. Rearranging for ω_g: ω_g = -ω_y + 2 ω_arm.

4. Express ω_g in terms of ω_y and gear ratio:

Substitute the expression for ω_arm from step 2 [ω_arm = ω_y / (1 + N_o / N_y)] into the equation for ω_g: ω_g = -ω_y + 2 [ω_y / (1 + N_o / N_y)]. Factor out ω_y: ω_g = ω_y [-1 + 2 / (1 + N_o / N_y)]. Combine terms inside the bracket: ω_g = ω_y [(-1 (1 + N_o / N_y) + 2) / (1 + N_o / N_y)]. ω_g = ω_y [(-1 - N_o / N_y + 2) / (1 + N_o / N_y)]. ω_g = ω_y [(1 - N_o / N_y) / (1 + N_o / N_y)]

5. Analyze the rotation based on relative sizes:

Case 1: If N_o > N_y (Orange gear larger than Yellow, as suggested by the image). Then (1 - N_o / N_y) is negative. Since ω_y is negative (clockwise), ω_g = (negative) (negative / positive) = positive. Green gear rotates counter-clockwise. Motion: CCW rotation, CW revolution. (Not an option).

Case 2: If N_o < N_y (Orange gear smaller than Yellow). Then (1 - N_o / N_y) is positive. ω_g = (negative) (positive / positive) = negative. Green gear rotates clockwise. Motion: CW rotation, CW revolution. (Option A).

Case 3: If N_o = N_y (Orange gear same size as Yellow). Then (1 - N_o / N_y) = 0. ω_g = ω_y (0 / positive) = 0. Green gear has no rotation (absolute rotation is zero). Motion: No rotation, CW revolution. (Option C).

6. Select the most plausible answer:

Although the image suggests N_o > N_y, this leads to a result not listed in the options. Gear train problems in multiple-choice format often rely on specific configurations, sometimes simplifying assumptions (like equal gear sizes if not explicitly stated otherwise) that lead to one of the provided answers. Assuming the problem intends for the case where N_o = N_y allows us to match option C.

7. Final Motion (assuming N_o = N_y):

Rotation of green gear (ω_g): 0 (No rotation). Revolution of green gear (ω_arm): Clockwise

Final Answer: C

Figure 32: Test case of Mechanical System Task from Gemini-2.5-pro.

F Future Work

Although we developed a fully automated data generation pipeline, some of the generated cube-stacking scenes still required manual verification to ensure that the spatial relations were not overly ambiguous. As a result, we currently provide a benchmark with 1,180 samples instead of a larger-scale dataset for training. Expanding the dataset is a promising direction for future work.