SOLIDGEOGEN: ADVANCING MULTIMODAL SOLID-GEOMETRY AND SPATIAL REASONING THROUGH VERIFIABLE SYMBOLIC GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Solid geometry requires multimodal large language models (MLLMs) to jointly interpret 3D structures and perform mathematical and spatial reasoning, yet high-quality data with aligned diagrams and verifiable solutions remains scarce. Existing data-generation methods focus primarily on plane geometry, while solid-geometry problem generation remains limited in both diversity and reliability. We present **SOLIDGEOGEN**, a symbolic framework for generating diverse solid-geometry problems with aligned diagrams, formal representations, and verifiable reasoning traces. This design ensures high-quality, internally consistent data for both training and evaluation. Building on SOLIDGEOGEN, we construct **S^{2}BENCH**, a **S**olid-**S**patial **Bench**mark comprising 2,000 problems across ten solid-geometry and spatial-reasoning tasks. We further build S^{2}TRAIN, a training corpus containing 10,000 examples. Leveraging S^{2}TRAIN, we propose **VISTA**, a **V**isual-structure-**I**nformed and **S**ymbolic-**T**race-**A**ssisted training framework. Extensive evaluation highlights the difficulty of S^{2}BENCH: all evaluated open-source MLLMs achieve below 30% average accuracy, while even the strongest proprietary model reaches only 67.2%. VISTA substantially improves accuracy on S^{2}BENCH at both 9B and 27B scales, while achieving consistent gains on external benchmarks for multimodal math reasoning and spatial intelligence. These results demonstrate that symbolic generation can provide scalable and reliable supervision for advancing complex multimodal reasoning in MLLMs.

1 INTRODUCTION

"There is no royal road to geometry." — Euclid

Geometry plays a fundamental role in mathematics and provides a natural testbed for symbolic and multimodal reasoning (Petersen, 2006; Trinh et al., 2024). Based on diagram structure, geometry is categorized into plane and solid geometry (Arana & Mancosu, 2012). While plane geometry focuses primarily on two-dimensional entities, solid geometry requires models to perceive and reason over three-dimensional structures and spatial transformations. These characteristics make solid geometry challenging for jointly studying spatial understanding and complex mathematical reasoning. With the development of MLLMs, substantial progress has been made in plane geometry (Zhang et al., 2024b; 2023; Chen et al., 2021; Xu et al., 2025); however, current models still struggle with solid geometry. Recent studies reveal a notable gap between MLLMs and human performance on such tasks (Wang et al., 2025b; Wu et al., 2025a), highlighting solid geometry as an important and underexplored domain for advancing multimodal spatial and mathematical reasoning.

Despite its importance, high-quality data for solid geometry remains scarce. Synthetic data generation has been explored in geometric reasoning, including template-based methods (Kazemi et al., 2023; Huang et al., 2025; Zhang et al., 2025a), LLM-based generation (Cai et al., 2024; Jiao et al., 2026), and symbolic approaches built on formal and theorem systems (Pan et al., 2025; Fu et al., 2025; Linger et al., 2025). These paradigms respectively face limitations in diversity, reasoning reliability, and system complexity, as shown in Figure 1. Nevertheless, existing efforts focus predominantly on plane geometry. In contrast, data generation for solid geometry remains at an early stage and faces substantially greater challenges. It requires richer formal representations of three-dimensional primitives and spatial relations, more complex theorem application over interactions

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

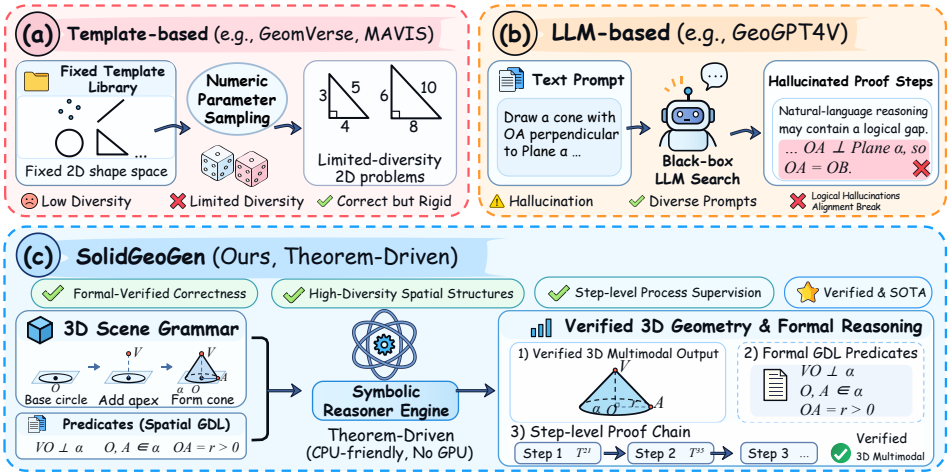


Figure 1: Solid-geometry data generation paradigms. SOLIDGEOGEN combines formal representations, theorem-driven reasoning, and rendering to produce diverse, aligned and verifiable data.

among lines, planes, and solids, and faithful rendering of 3D structures into diagrams (Wu et al., 2025a). These requirements make it considerably harder to maintain consistency among natural language problems, formal representations, rendered diagrams, answers, and stepwise reasoning traces, thereby limiting the scalability and reliability of solid geometry generation.

To address these challenges, we propose **SOLIDGEOGEN**, a theorem-driven framework integrating formal representation, symbolic reasoning, verification, and rendering for scalable and reliable solid geometry generation. At its core, SOLIDGEOGEN builds a dedicated formal system for solid geometry, including a typed geometry description language (GDL) for geometric entities, spatial relations, and metric attributes, together with an executable theorem library spanning K-12 to Olympiad-level problems. Beyond conventional geometry calculation, the framework also generates diverse spatial reasoning tasks involving structural relations, projections, folding, and other transformations of 3D configurations. Unlike LLM-based generators (Jiao et al., 2026), SOLIDGEOGEN derives solutions through explicit theorem application, making each reasoning step grounded in valid geometric principles and directly verifiable. To the best of our knowledge, it is the first theorem-driven symbolic framework for large-scale solid geometry generation, producing aligned natural language problems, formal representations, rendered diagrams, final answers, and step-level reasoning traces. These verifiable outputs provide reliable supervision for training MLLMs on both solid geometry reasoning and broader structured spatial reasoning.

Based on SOLIDGEOGEN, we construct the **S**olid-**S**patial **Bench** (**S^{2}BENCH**), a benchmark comprising 2,000 problems across 10 task categories spanning solid-geometry reasoning and spatial understanding. Extensive evaluation on S^{2}BENCH reveals substantial limitations of current MLLMs: all evaluated open-source models achieve below 30% average accuracy, even the frontier model GPT-5.6-Sol reaches only 67.2%, highlighting the challenge of S^{2}BENCH. We further build the **S**olid-**S**patial **Train**ing Corpus (**S^{2}TRAIN**), a training corpus with structured visual descriptions and reasoning traces derived from symbolic deduction. Leveraging S^{2}TRAIN, we propose **VISTA**, a **V**isual-structure-**I**nformed and **S**ymbolic-**T**race-**A**ssisted training framework. It uses the corpus's structural hints to synthesize supervised fine-tuning (SFT) data, then improves the resulting models through reinforcement learning (RL). We also explore the effect of on-policy distillation (OPD). VISTA raises average accuracy from 14.0% to 42.4% at 9B and from 19.4% to 50.9% at 27B. Crucially, the resulting models also show consistent gains on external multimodal mathematical and spatial reasoning benchmarks, demonstrating transfer beyond the synthetic distribution. In summary, our contributions are as follows:

- **Formal Foundation.** We establish a formal foundation for solid geometry, comprising 472 GDL symbols and an executable library of 680 theorem families with 722 branches.
- **Symbolic Generation Framework.** Building on this foundation, we develop **SOLIDGEOGEN**, a theorem-driven symbolic framework that automatically generates aligned problems, diagrams, formal representations, and step-level reasoning traces for solid geometry and spatial reasoning.

Method	Domain	Level	Paradigm	SG FL	Theorem Engine	SG Diagram	Verified Steps
GeomVerse (Kazemi et al., 2023)	PG	K	Template-based	✗	✗	✗	✗
AutoGeo (Huang et al., 2025)	PG	K	Template-based	✗	✗	✗	✗
MAVIS (Zhang et al., 2025a)	PG	K	Template-based	✗	✗	✗	✗
GeoGPT4V (Cai et al., 2024)	PG	K	LLM-assisted	✗	✗	✗	✗
MagicGeo (Wang et al., 2025a)	PG	K	LLM-assisted	✗	✗	✗	✗
Socratic-Geo (Jiao et al., 2026)	PG	K	LLM-assisted	✗	✗	✗	✗
GeoCode (Lin et al., 2026)	PG	K C	Formal/Symbolic	✗	✓	✗	✓
GeoGen (Pan et al., 2025)	PG	K	Formal/Symbolic	✗	✓	✗	✓
GeoFM (Zhang et al., 2025b)	PG	K	Formal/Symbolic	✗	✓	✗	✓
TrustGeoGen (Fu et al., 2025)	PG	K C	Formal/Symbolic	✗	✓	✗	✓
NeSyGeo (Wu et al., 2025b)	PG	K	Formal/Symbolic	✗	✓	✗	✓
DynaSolidGeo (Wu et al., 2025a)	SG	K C	Template-based	✗	✗	✓	✗
SOLIDGEOGEN (Ours)	SG	K C	Formal/Symbolic	✓	✓	✓	✓

Table 1: Comparison of geometry generation frameworks. PG/SG: plane/solid geometry; **Level:** K=K-12, C=Competition. SG FL: solid-geometry formal language; SG Diagram: solid-geometry diagram generation; Verified Steps: generated traces with symbolic/theorem-based step verification.

- **Benchmark and Data Resources.** Based on SOLIDGEOGEN, we construct **S^{2}BENCH**, a 2,000-problem benchmark with 10 task categories spanning solid geometry and spatial reasoning, together with **S^{2}TRAIN**, a multimodal training corpus with structured visual supervision and symbolic reasoning traces. Extensive evaluation reveals substantial limitations of current MLLMs.
- **Training Framework and Models.** Building on S^{2}TRAIN, we develop VISTA, a symbol-guided training framework, and train models that substantially improve on S^{2}BENCH while generalizing to external mathematical and spatial reasoning benchmarks.

2 RELATED WORK

Geometry Problem Solving. Existing benchmarks such as GeoQA, UniGeo, MathVerse, MathVision, We-Math, and GGBench evaluate MLLMs on geometric problem understanding and reasoning (Chen et al., 2021; 2022; Zhang et al., 2024b; Wang et al., 2024; Qiao et al., 2025; Wei et al., 2025). Building on these benchmarks, geometry-specialized methods such as Inter-GPS, G-LLaVA, GeoX, and GNS improve geometric problem solving through instruction tuning or neural-symbolic reasoning (Lu et al., 2021; Gao et al.; Xia et al.; Ning et al., 2025). However, both evaluation and modeling efforts have focused predominantly on plane geometry. Extending geometric reasoning to solid geometry introduces additional challenges, as models must reason over 3D structures, projections, unfolding, and spatial angles. Recent studies such as SolidGeo and HilbertGeo further reveal that current MLLMs remain substantially weaker on solid geometry (Wang et al., 2025b; Xu et al., 2026), highlighting a persistent gap in solid geometry and spatial reasoning.

Geometry Problem Generation. GeomVerse, AutoGeo, and MAVIS synthesize geometry data through procedural templates or rule-based engines (Kazemi et al., 2023; Huang et al., 2025; Zhang et al., 2025a). LLM-assisted methods, including GeoGPT4V, MagicGeo, and Socratic-Geo, further improve flexibility through instruction augmentation or plotting-code supervision (Cai et al., 2024; Wang et al., 2025a; Jiao et al., 2026). More recently, symbolic approaches such as GeoGen, GeoFM, TrustGeoGen, and NeSyGeo introduce formal representations and theorem-guided reasoning to improve generation correctness (Pan et al., 2025; Zhang et al., 2025b; Fu et al., 2025; Wu et al., 2025b; Lin et al., 2026). However, these methods remain predominantly focused on plane geometry, with formal systems and rendering pipelines designed around 2D primitives and planar theorems. DynaSolidGeo extends generation to solid geometry through numerical rewriting of seed templates, but remains limited in diversity and automation (Wu et al., 2025a). In contrast, SOLIDGEOGEN provides a theorem-driven symbolic framework that extends geometry generation beyond plane geometry to solid geometry and spatial reasoning, while jointly producing diverse aligned problems, diagrams, formal representations, and verified reasoning traces. Table 1 compares these frameworks.

3 SOLIDGEOGEN: VERIFIABLE SYMBOLIC GENERATION

We present **SOLIDGEOGEN**, a theorem-grounded framework for generating multimodal solid-geometry problems with executable formal semantics. Unlike pipelines that independently synthesize a question, an answer, and a diagram, SOLIDGEOGEN constructs all modalities from shared geometric parameters. Each generated instance is represented as

$$\mathcal{P} = (q, \mathcal{C}, g, a, \tau, I), \tag{1}$$

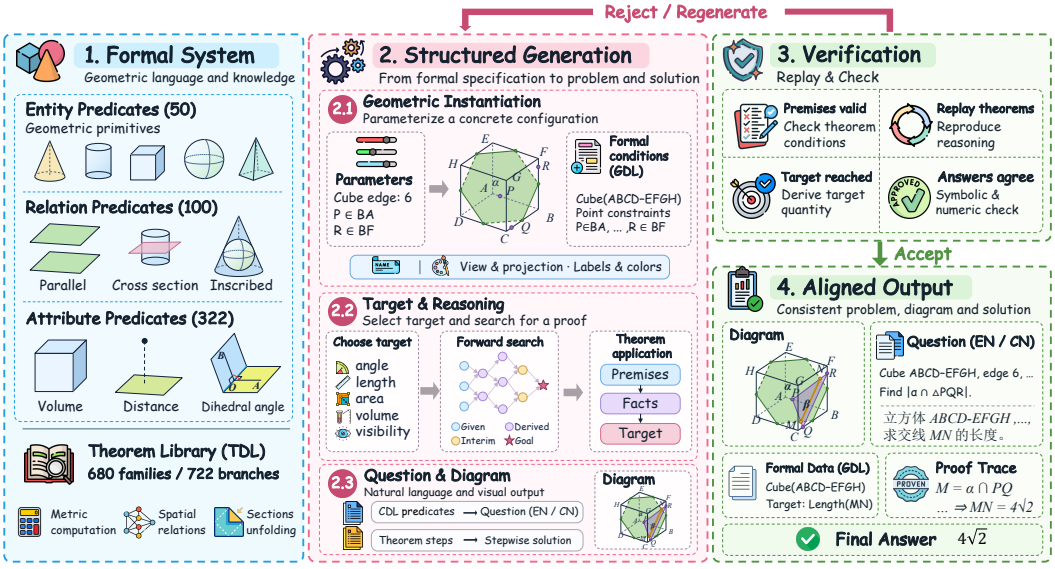


Figure 2: Overview of the SOLIDGEOGEN framework. Built on a solid-geometry formal system and executable theorem library, SOLIDGEOGEN performs geometric structure instantiation, theorem-driven symbolic reasoning, diagram-text rendering and validation to produce aligned solid geometry data with formal representations, final answers, and reasoning traces.

where q is the natural-language question, $\mathcal{C}$ is a set of formal geometric conditions, g is the target, a is the reference answer, τ is the structured reasoning trace, and I is the rendered diagram. SOLIDGEOGEN consists of three components: a formal system for solid geometry, a structured generation engine, and a symbolic replay and validation procedure. This design makes correctness and cross-modal alignment explicit constraints of generation, yielding high-quality multimodal instances.

3.1 SOLID-GEOMETRY FORMAL SYSTEM

Formal Representation. We represent a solid-geometry problem using a typed geometry description language (GDL) comprising entity predicates, relation predicates, and attribute functions. Entity predicates declare geometric objects such as spheres, cuboids, prisms, and pyramids. Relation predicates encode incidence, coplanarity, parallelism, perpendicularity, tangency, and inscribed or circumscribed configurations. Attribute functions associate geometric objects with symbolic quantities, including lengths, areas, volumes, distances, angles, and ratios. Using these symbols, the condition set $\mathcal{C}$ contains either geometric predicates, such as $\mathrm{Perpendicular}(AB, CD)$, or symbolic equalities over geometric attributes, such as $\mathrm{Length}(AB) = \ell$. A formal problem is defined as

$$\mathcal{F} = (\mathcal{C}, g), \qquad \mathcal{C} = \mathcal{C}_{\mathrm{con}} \cup \mathcal{C}_{\mathrm{text}} \cup \mathcal{C}_{\mathrm{img}}, \tag{2}$$

where $\mathcal{C}_{\mathrm{con}}$, $\mathcal{C}_{\mathrm{text}}$, and $\mathcal{C}_{\mathrm{img}}$ encode construction constraints, textual givens, and image-grounded facts, respectively. The target g specifies a geometric relation or the value of an attribute expression.

Executable Theorem System. Let $\mathcal{T}$ denote the theorem system. Each $r \in \mathcal{T}$ is represented as

$$r: \quad \phi_1 \wedge \cdots \wedge \phi_m \Rightarrow \{\psi_1, \ldots, \psi_n\}, \tag{3}$$

where each ϕ_i is a premise and each ψ_j is a conclusion, expressed as either a typed geometric predicate or an algebraic constraint. Given a grounding under which all premises are satisfied by the symbolic state, the conclusions are inserted together with their premises and theorem provenance.

The theorem system combines generic Euclidean rules for metric computation, similarity, and coordinate-vector reasoning with specific knowledge of spatial relations, cross-sections, composite solids, and folding/unfolding. A unified rule interface enables geometric deduction and symbolic equation solving within a single provenance-preserving derivation. See Appendix B for more details.

3.2 STRUCTURED SOLID-GEOMETRY GENERATION

Geometric Structure Instantiation. SOLIDGEOGEN instantiates the underlying geometry before realizing its textual and visual modalities. A generation schema G samples shared geometric

parameters θ and jointly instantiates a formal condition set $\mathcal{C}$ and a rendering specification ρ:

$$(\mathcal{C}, \rho) = G(\theta), \qquad \theta \in \Theta_G, \tag{4}$$

where Θ_G is the admissible parameter domain and ρ specifies the geometry to be rendered. For example, a cylinder schema samples $\theta = (r, h)$ and records the resulting radius and height consistently in both $\mathcal{C}$ and ρ. The domain Θ_G excludes invalid or degenerate configurations before construction. For discrete spatial tasks, θ may instead specify a voxel configuration or orientation state.

Target and Reasoning Construction. Given $\mathcal{C}$, the generator selects a target g that is well-defined and solvable under the instantiated conditions. We denote candidate solution construction by $\mathrm{Construct}_{\mathcal{T}}$, which grounds and applies a sequence of theorem rules from $\mathcal{T}$ to the entities in $\mathcal{C}$, records the derived conclusions, and evaluates the resulting symbolic expressions:

$$(\widetilde{a}, \tau) = \mathrm{Construct}_{\mathcal{T}}(\mathcal{C}, g), \tag{5}$$

where $\widetilde{a}$ is the candidate answer, τ is the reasoning trace, and $\tau_{\mathcal{T}} = (r_1, \ldots, r_K)$ denotes its theorem reasoning steps. For closed-form problems, the rules in $\tau_{\mathcal{T}}$ introduce equations from which $\widetilde{a}$ is evaluated symbolically. Enumeration and state-transition tasks instead invoke a deterministic executor, whose operations are recorded in τ and linked to $\mathcal{C}$ by a typed task predicate. The theorem subtrace is later replayed by the validator in Section 3.3.

Question and Diagram Generation. We denote controlled text realization by Text and geometric diagram generation by Render. They produce the question and diagram as

$$q = \mathrm{Text}(\mathcal{C}, g), \qquad I = \mathrm{Render}(\rho; \nu), \tag{6}$$

where Text selects conditions from $\mathcal{C}$ and expresses them together with the target g using controlled language templates, while Render projects ρ under viewpoint and layout parameters ν. The specification ρ maps the image-grounded conditions in $\mathcal{C}_{\mathrm{img}}$ to visual elements such as points, cross-sections, colors, tick marks, and arrows. Because $\mathcal{C}$ and ρ are jointly instantiated from the same parameters θ, the question, diagram, and formal representation remain mutually consistent. They form the candidate instance $\widetilde{\mathcal{P}} = (q, \mathcal{C}, g, \widetilde{a}, \tau, I)$ passed to symbolic replay and validation. Further details are provided in Appendix C.

3.3 Symbolic Replay and Validation

The generation schema records formal conditions and a rule sequence, while separately evaluating a candidate answer. Validation completes this process by executing the recorded rules in the symbolic solver and comparing the resulting target value with the candidate answer. This checks whether the generated solution agrees with its formal representation. Further details and worked examples are provided in Appendix D.

Starting from $\mathcal{C}_0 = \mathcal{C}$, the solver replays the instantiated theorem steps $\tau_{\mathcal{T}} = (r_1, \ldots, r_K)$:

$$\mathcal{C}_i = \mathcal{C}_{i-1} \cup \mathrm{Apply}_{\mathcal{T}}(r_i, \mathcal{C}_{i-1}), \qquad i = 1, \ldots, K, \tag{7}$$

where $\mathrm{Apply}_{\mathcal{T}}$ adds a rule's conclusions only when its premises hold. An inapplicable rule contributes no conclusions, and replay continues. After replay, the solver obtains $a_{\mathrm{sym}} = \mathrm{Solve}(g \mid \mathcal{C}_K)$. The candidate passes the answer check only if

$$a_{\mathrm{sym}} \text{ is defined}, \qquad a_{\mathrm{sym}} \simeq \widetilde{a}, \tag{8}$$

where $\simeq$ denotes exact symbolic equality or numerical agreement within tolerance. Candidates failing either condition are discarded. For an accepted candidate, we retain $a = \widetilde{a}$ as the reference answer in $\mathcal{P}$. For executor-based solutions, solver replay checks the formal connection between encoded computational results and the target g. The underlying computations and theorem definitions are assessed through separate audits using coordinate geometry, vector algebra, or exact enumeration. Shared generation parameters and batch-level visual inspection support diagram consistency.

3.4 Constructing S^{2}Bench and S^{2}Train

Using validated instances, we construct **S^{2}Bench**, a balanced benchmark comprising 2,000 problems across ten task categories: *Solid-Geometry Metric Reasoning* (SGMR), *Voxel Counting* (VC), *Dice-Rolling Orientation Tracking* (DROT), *Fine-Grained Visual-Geometric Reasoning* (FGVGR),

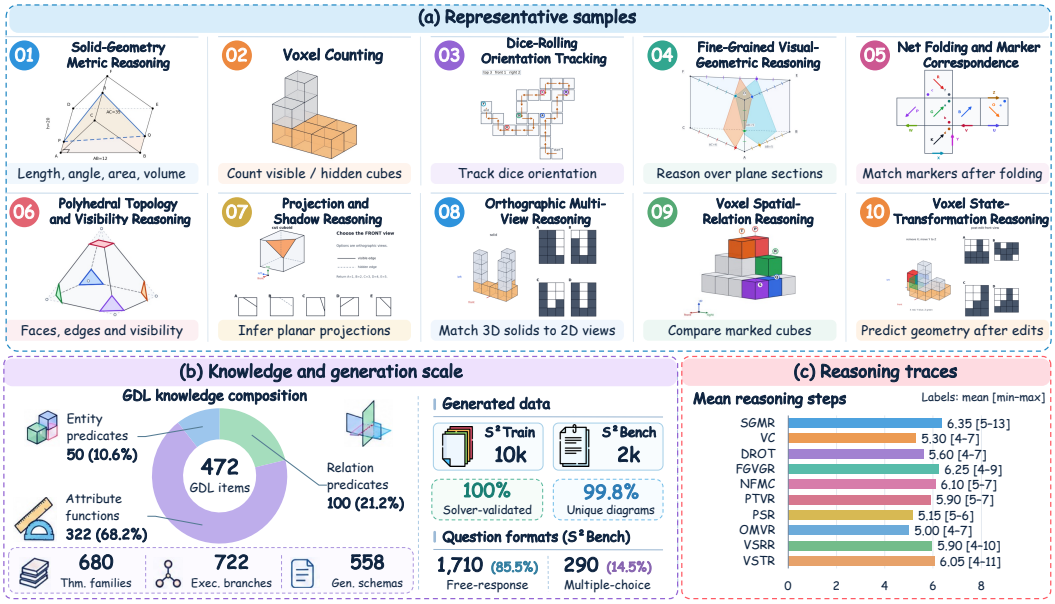


Figure 3: Overview of SOLIDGEOGEN: (a) representative samples across ten task categories, (b) symbolic knowledge and dataset statistics, and (c) benchmark reasoning-trace lengths.

Net Folding and Marker Correspondence (NFMC), *Polyhedral Topology and Visibility Reasoning* (PTVR), *Projection and Shadow Reasoning* (PSR), *Orthographic Multi-View Reasoning* (OMVR), *Voxel Spatial-Relation Reasoning* (VSRR), and *Voxel State-Transformation Reasoning* (VSTR). During evaluation, models receive only the original question and diagram. We separately construct S^{2}TRAIN, a training corpus comprising 10,000 examples augmented with structured visual descriptions and theorem-grounded reasoning traces. Figure 3 presents representative examples and summarizes key dataset statistics. All examples pass solver replay and task-specific deterministic checks, ensuring instance correctness; see Appendix F for illustrative cases.

We promote diversity at the level of geometric structure rather than surface wording. Samples are stratified by task family and generation schema, with variation in their parameters, answers, and reasoning paths. To prevent train–test leakage, each problem is mapped to a canonical record containing its task family, geometric structure, measurements, and queried target. Object names and task-preserving symmetries are normalized, so rotated voxel assemblies or vertex-renamed cuboids are treated as the same problem. Formal conditions and reasoning traces are compared to detect duplicates across representations. Any training candidate matching a S^{2}BENCH example under these checks is removed. Further details are provided in Appendix E.

4 THE VISTA TRAINING FRAMEWORK

We present **VISTA**, a **V**isual-structure-**I**nformed and **S**ymbolic-**T**race-**A**ssisted training framework shown in Figure 4. It uses privileged visual-structure descriptions and symbolic traces to synthesize hint-free supervised fine-tuning (SFT) data, then improves the resulting models through reinforcement learning with verifiable rewards (RLVR). We additionally explore reverse-KL on-policy distillation (OPD) to transfer the learned policy.

4.1 SYMBOL-GUIDED TRAINING DATA SYNTHESIS

For each generated instance $\mathcal{P}$, we take $x = (q, I)$ as the multimodal input and derive two privileged hints: $h_v = \mathrm{Hint}_v(\mathcal{C})$ from the formal conditions and $h_r = \mathrm{Hint}_r(\tau)$ from the symbolic trace. The former describes geometric entities, relations, and constraints, while the latter expresses the reasoning steps encoded in τ. A teacher model samples candidate solutions using this privileged guidance, and we construct the SFT dataset by retaining evaluator-approved solutions:

$$\mathcal{D}_{\mathrm{SFT}} = \{(x, y) \mid y \sim \pi_{\mathrm{teach}}(\cdot \mid x, h_v, h_r),\ \mathrm{Eval}(y, a) = 1\}, \tag{9}$$

where $\mathrm{Eval}(y, a) = 1$ indicates that the extracted final answer from y matches the reference answer a. Since only (x, y) is retained, trained models receive the original question and image without privileged guidance during training and inference.

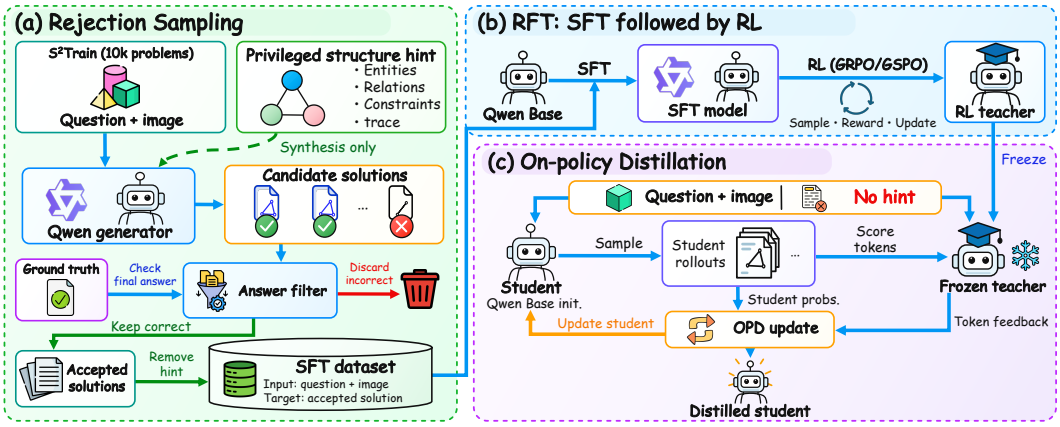


Figure 4: Overview of the VISTA training framework: (a) privileged visual-structure and symbolic-trace guidance synthesizes hint-free SFT data through rejection sampling, (b) SFT followed by RL optimizes the policy, and (c) reverse-KL OPD is additionally explored for policy transfer.

4.2 Model Post-Training

Supervised Fine-Tuning. We optimize the standard language-modeling objective

$$\mathcal{L}_{\mathrm{SFT}} = -\mathbb{E}_{(x,y)\sim\mathcal{D}_{\mathrm{SFT}}}\left[\frac{1}{|y|}\sum_{t=1}^{|y|}\log \pi_\theta(y_t \mid x, y_{<t})\right]. \tag{10}$$

Reinforcement Learning with Verifiable Rewards. Starting from the SFT policy, we use the binary final-answer reward $r(x, y) = \mathbf{1}[\mathrm{Eval}(y, a) = 1]$. For each prompt, rewards from G sampled responses are normalized within the group to obtain $\widehat{A}_i$. GRPO (Shao et al., 2024) then optimizes

$$\mathcal{L}_{\mathrm{GRPO}} = -\mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t}\min\Big(\rho_{i,t}\widehat{A}_i, \mathrm{clip}(\rho_{i,t}, 1-\varepsilon, 1+\varepsilon)\widehat{A}_i\Big)\right], \tag{11}$$

where $\rho_{i,t} = \pi_\theta(y_{i,t} \mid x, y_{i,<t})/\pi_{\mathrm{old}}(y_{i,t} \mid x, y_{i,<t})$. We also study GSPO (Zheng et al., 2025), which performs importance weighting and clipping at the sequence level.

Reverse-KL On-Policy Distillation. We also study OPD (Agarwal et al., 2024), which minimizes reverse KL to a fixed teacher along student-sampled trajectories, with $s_t = (x, y_{<t})$:

$$\mathcal{L}_{\mathrm{OPD}} = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\left[\frac{1}{|y|}\sum_{t} D_{\mathrm{KL}}(\pi_\theta(\cdot \mid s_t)\,\|\,\pi_T(\cdot \mid s_t))\right]. \tag{12}$$

5 Experiments

5.1 Experimental Setup

We evaluate 21 representative LLMs and MLLMs on S^{2}Bench, reporting per-task accuracy, overall accuracy, and average token usage. We further assess out-of-domain generalization on seven benchmarks spanning mathematical reasoning and spatial intelligence. For post-training, we adopt Qwen3.5-9B and Qwen3.5-27B (Qwen Team, 2026) as base models. For SFT, both are trained on 32 GPUs with a maximum sequence length of 32,768 and a global batch size of 64, using learning rates of 5×10^{-5} and 2×10^{-5}, respectively. We implement RLVR with GRPO and GSPO using ROLL (Wang et al., 2025c), and OPD with the k1 reverse-KL estimator.

5.2 Evaluation of Existing Models

Table 2 evaluates existing models on S^{2}Bench. The results reveal three key findings. First, the benchmark remains highly challenging and relies strongly on visual reasoning: text-only models achieve below 10% average accuracy, all evaluated open-source MLLMs remain below 30%, and only GPT-5.6-Sol and GPT-6-Sol exceed 60%. Second, models struggle most on tasks requiring the construction or transformation of spatial states. Most open-source models score near zero on

Table 2: Performance of existing models on S^{2}BENCH. We report accuracy (%) across ten reasoning tasks, together with overall accuracy and average token usage. The best and second-best accuracy results are shown in bold and underlined, respectively.

Model	SGMR	VC	DROT	FGVGR	NFMC	PTVR	PSR	OMVR	VSRR	VSTR	Overall	Avg.tokens
LLMs (Text-only)												
DeepSeek-V4-Flash	19.5	0.0	0.0	0.0	0.0	0.5	28.0	17.5	0.0	2.0	6.8	27,482
GLM-5.2	27.5	0.5	0.5	0.0	0.0	0.0	25.5	19.0	0.5	1.5	7.5	13,152
Frontier Proprietary MLLMs (Text + Image)												
Claude-Opus-4.8	**99.0**	25.0	17.0	52.0	34.5	19.5	40.0	44.0	10.0	9.0	35.0	9,297
Claude-Sonnet-5	97.0	30.0	29.0	**91.5**	5.0	43.5	45.0	47.0	7.0	3.0	39.8	16,644
Gemini-3.5-Flash	96.0	43.5	19.0	18.5	53.5	51.0	49.0	59.0	16.5	8.0	41.4	23,652
Gemini-3.7-Flash	94.0	54.0	33.0	18.0	73.5	54.5	57.5	60.0	15.0	14.5	47.4	8,979
GPT-5.6-Terra	79.5	49.0	32.0	44.0	23.0	34.5	55.5	52.5	29.0	11.0	41.0	4,663
GPT-5.5	97.0	43.5	61.0	65.0	**80.0**	53.0	70.0	**74.5**	13.5	13.0	57.0	7,013
GPT-6-Sol	96.0	**57.0**	**71.0**	81.5	35.0	**58.0**	**77.0**	66.5	69.5	**54.0**	66.5	3,497
GPT-5.6-Sol	96.0	52.5	63.5	78.0	64.0	**58.0**	76.0	65.5	**74.0**	44.5	**67.2**	4,023
Open-source MLLMs (Text + Image)												
Qwen2.5-VL-72B	3.5	9.0	0.0	0.0	0.0	0.0	14.0	23.5	3.0	0.0	5.3	1,688
InternVL3.5-38B	2.0	31.0	0.0	0.0	0.0	1.0	12.5	10.5	1.5	0.0	5.8	7,338
Qwen3-VL-32B	29.0	31.5	2.0	0.0	0.0	3.5	18.5	29.0	4.5	1.5	11.9	8,198
Gemma4-26B-A4B	42.0	36.5	1.0	0.0	0.5	3.0	20.5	23.5	2.5	1.0	13.1	4,664
Qwen3.5-9B	48.5	28.5	0.5	0.0	0.0	6.0	22.0	25.0	3.5	6.0	14.0	12,313
Gemma4-31B	29.0	36.0	3.5	1.0	1.5	4.0	23.5	34.5	6.5	3.5	14.3	2,127
Qwen3.5-27B	59.5	31.5	5.5	0.0	0.0	9.0	41.5	32.0	5.0	10.5	19.4	11,826
Qwen3.5-397B-A17B	66.0	45.0	1.5	0.0	0.0	7.5	32.5	34.0	5.0	3.5	19.5	16,646
Minimax M3	96.0	23.0	4.0	0.0	0.5	13.0	32.5	34.0	5.5	4.0	21.3	75,286
Qwen3.8-27B	67.5	49.5	13.5	25.5	5.0	21.5	38.0	49.5	9.0	5.0	28.4	8,343
Kimi-K2.6	98.5	34.5	10.5	1.0	4.5	16.5	54.5	53.5	6.0	4.5	28.4	48,105

NFMC and VSTR and remain weak on VSRR and VC, while frontier models also vary substantially. Third, models perform considerably better on calculation-intensive tasks such as SGMR, where some open-source models exceed 90%. This contrast suggests the primary bottleneck is not geometric calculation, but fine-grained visual perception and spatial reasoning over 3D structures.

5.3 Training Results and Analysis

Motivating analysis: privileged hints. We conduct a pilot study by providing both the visual hint h_v and reasoning hint h_r alongside the original input. Figure 5 shows that the combined hints raise nine-task average accuracy from 12.4% to 42.2% at 9B and from 18.1% to 67.6% at 27B. The decomposition in Appendix G shows that h_v accounts for most gains, whereas h_r alone yields only marginal improvements. These results identify fine-grained visual perception and 3D structural interpretation as major bottlenecks. Because the hints contain privileged cues, VISTA uses them only for supervision synthesis; trained models receive only the original question and image.

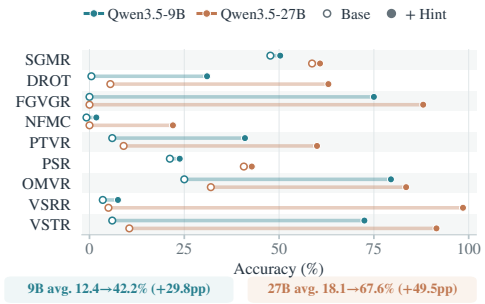


Figure 5: Effect of combined privileged hints across nine tasks, excluding VC.

Training results. Panel (a) of Table 3 summarizes our training results. VISTA-SFT raises average accuracy from 14.0% to 37.6% at 9B and from 19.4% to 43.7% at 27B, with particularly large gains on FGVGR, OMVR, VSRR, and VSTR. This indicates that privileged symbolic guidance is especially effective for reconstructing or transforming 3D structure. Starting from the SFT checkpoints, answer-based GRPO adds 4.8 and 7.2 points, reaching 42.4% and 50.9%, respectively, while slightly reducing average token usage. We also study OPD using the corresponding VISTA-RL models as teachers and base models as students. OPD raises average accuracy from 14.0% to 28.7% at 9B and from 19.4% to 35.2% at 27B. OPD transfers useful reasoning capabilities but remains less effective than staged SFT-to-RL training. More details are provided in Appendix H.

Ablation: role of SFT initialization. Panel (b) of Table 3 isolates the effect of SFT initialization. Direct GRPO/GSPO improves the base models only modestly, reaching 16.9%/16.6% at 9B and

Table 3: Training and ablation results on S^{2}BENCH. Panel (a) compares the base, VISTA-SFT, VISTA-RL, and VISTA-OPD models; Panel (b) evaluates direct GRPO/GSPO without SFT initialization. Bold marks the best accuracy per model scale in Panel (a).

Scale	Training Recipe	SGMR	VC	DROT	FGVGR	NFMC	PTVR	PSR	OMVR	VSRR	VSTR	Overall	Avg.Tokens
(a) Main Training Results													
9B	Qwen3.5-9B (Base)	48.5	28.5	0.5	0.0	0.0	6.0	22.0	25.0	3.5	6.0	14.0	12,313
9B	VISTA-SFT-9B	62.5	38.0	12.0	51.5	**4.0**	**15.5**	35.5	55.0	37.5	64.0	37.6	11,415
9B	VISTA-RL-9B	**66.5**	48.5	**26.5**	**59.5**	3.5	**15.5**	**38.0**	**57.5**	**40.5**	**67.5**	**42.4**	11,276
9B	VISTA-OPD-9B	51.0	**53.5**	3.5	19.5	0.0	5.5	24.0	49.0	35.0	46.0	28.7	10,934
27B	Qwen3.5-27B (Base)	59.5	31.5	5.5	0.0	0.0	9.0	41.5	32.0	5.0	10.5	19.4	11,826
27B	VISTA-SFT-27B	61.0	58.5	21.0	61.0	5.5	**24.0**	32.5	61.0	48.0	64.0	43.7	11,556
27B	VISTA-RL-27B	**75.5**	**66.0**	**37.0**	**75.5**	**7.5**	18.0	**44.5**	**64.0**	**50.0**	**71.0**	**50.9**	10,940
27B	VISTA-OPD-27B	63.5	61.0	10.5	24.5	1.0	9.0	44.0	59.0	38.5	40.5	35.2	10,420
(b) Direct-RL Controls (Without SFT Initialization)													
9B	Direct GRPO	64.0	29.5	1.5	0.0	0.0	5.0	29.5	30.5	4.5	4.5	16.9	16,021
9B	Direct GSPO	67.0	29.0	1.0	0.0	0.0	5.0	27.5	29.5	5.0	2.0	16.6	11,297
27B	Direct GRPO	81.5	33.0	4.5	0.0	0.0	9.5	46.5	31.5	4.0	11.5	22.2	13,613
27B	Direct GSPO	81.5	30.0	6.0	0.0	0.5	8.5	46.5	34.5	6.5	6.5	22.1	13,873

Table 4: Out-of-domain accuracy (%) on several external multimodal math reasoning and spatial-intelligence benchmarks. Bold indicates the best result within each model scale.

Scale	Model	Multimodal Math Reasoning				Spatial Perception & Intelligence			Overall
		MathVision	DynaSolidGeo	SolidFGeo2K	SolidGeo	MindCube	MMSI	3DSRBench	Avg.Acc.
9B	Qwen3.5-9B (Base)	65.1	54.4	40.9	38.7	42.4	32.6	65.1	48.5
9B	VISTA-SFT-9B	66.0	55.8	41.5	38.9	54.1	37.7	67.6	51.7
9B	VISTA-RL-9B	**67.3**	**57.2**	**42.6**	39.5	**56.6**	**39.8**	**68.7**	**53.1**
9B	VISTA-OPD-9B	66.2	56.3	42.2	**40.2**	48.7	35.1	67.8	50.9
27B	Qwen3.5-27B (Base)	68.1	60.6	39.9	40.2	52.5	35.7	67.0	52.0
27B	VISTA-SFT-27B	69.2	62.3	41.7	40.8	64.8	40.3	69.8	55.6
27B	VISTA-RL-27B	**70.6**	**64.1**	**42.6**	**41.6**	**67.1**	**42.2**	**70.7**	**57.0**
27B	VISTA-OPD-27B	70.0	63.6	42.0	**41.6**	60.2	38.4	68.9	55.0

22.2%/22.1% at 27B, whereas the staged models in Panel (a) achieve 42.4% and 50.9%. Category-level results further show that direct RL mainly improves tasks already accessible to the base models, while SFT-then-RL produces broader gains. This confirms that symbol-guided SFT provides an effective initialization for RL on S^{2}BENCH.

5.4 Out-of-Domain Generalization

We evaluate transfer beyond S^{2}BENCH on seven external benchmarks covering multimodal mathematical reasoning (Wang et al., 2024; Wu et al., 2025a; Xu et al., 2026; Wang et al., 2025b) and spatial intelligence (Yin et al., 2025; Yang et al., 2026; Ma et al., 2025). As shown in Table 4, VISTA-RL performs best overall, improving average accuracy over the corresponding base models by 4.6 and 5.0 points at 9B and 27B, respectively. Gains on mathematical reasoning are consistent but moderate (0.8–3.5 points), whereas improvements on spatial intelligence are more substantial: +14.2/+14.6 points on MindCube, +7.2/+6.5 on MMSI, and +3.6/+3.7 on 3DSRBench. These results demonstrate effective transfer to broader spatial-reasoning tasks.

6 Conclusion

We presented SolidGeoGen, a theorem-driven symbolic framework for generating aligned solid-geometry problems, diagrams, formal representations, and verifiable reasoning traces. Using this framework, we constructed S^{2}BENCH and S^{2}TRAIN to evaluate and improve multimodal solid-geometry and spatial reasoning. Our evaluation shows that current MLLMs struggle more with fine-grained visual perception and 3D structural reasoning than with geometric calculation. VISTA converts privileged visual-structure descriptions and symbolic traces into training supervision, substantially improving in-domain performance while generalizing effectively to external multimodal mathematical and spatial-reasoning benchmarks. These results demonstrate that formal systems can provide scalable and reliable supervision for multimodal reasoning. Future work will extend theorem coverage and diagram diversity toward more complex real-world 3D settings.

AI USE STATEMENT

Generative AI tools were used to assist with language editing, and the preparation of figures and tables. All AI-assisted text was reviewed and revised by the authors, and all AI-assisted part was inspected and tested against the intended behavior. Experimental results, numerical values, citations, and scientific claims were checked by the authors against the underlying implementations and source materials. The authors take full responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects or private personal data. The proposed framework generates synthetic solid-geometry problems and diagrams. Any public release of models, data, or code will follow the licenses and usage conditions of the underlying resources.

REPRODUCIBILITY STATEMENT

We provide the formal representation, executable theorem formulation, generation and solver-validation procedures, and training objectives in the main paper. The experimental section reports the evaluated models, metrics, training recipes, and principal optimization settings. The appendix further documents the formal system, structured generation process, validation cases, data diversity, privileged-hint analysis, SFT data construction, and qualitative model responses. Together, these materials specify the components needed to reproduce the datasets, training pipeline, and evaluations; code, generated data, and detailed configurations will be released with the project.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024.

Andrew Arana and Paolo Mancosu. On the relationship between plane and solid geometry. *The Review of Symbolic Logic*, 5(2):294–353, 2012.

Shihao Cai, Keqin Bao, Hangyu Guo, Jizhi Zhang, Jun Song, and Bo Zheng. Geogpt4v: Towards geometric multi-modal large language models with geometric image generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 750–766, 2024.

Jiaqi Chen, Jianheng Tang, Jinghui Qin, Xiaodan Liang, Lingbo Liu, Eric Xing, and Liang Lin. Geoqa: A geometric question answering benchmark towards multimodal numerical reasoning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 513–523, 2021.

Jiaqi Chen, Tong Li, Jinghui Qin, Pan Lu, Liang Lin, Chongyu Chen, and Xiaodan Liang. Unigeo: Unifying geometry logical reasoning via reformulating mathematical expression. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3313–3323, 2022.

Daocheng Fu, Jianlong Chen, Renqiu Xia, Zijun Chen, Qi Liu, Yuan Feng, Hongbin Zhou, Renrui Zhang, Shiyang Feng, Peng Gao, et al. Trustgeogen: Formal-verified data engine for trustworthy multi-modal geometric problem solving. *arXiv preprint arXiv:2504.15780*, 2025.

Jiahui Gao, Renjie Pi, Jipeng Zhang, Jiacheng Ye, Wanjun Zhong, Yufei Wang, Lanqing HONG, Jianhua Han, Hang Xu, Zhenguo Li, et al. G-llava: Solving geometric problem with multi-modal large language model. In *The Thirteenth International Conference on Learning Representations*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd*

Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 3828–3850, 2024.

Zihan Huang, Tao Wu, Wang Lin, Shengyu Zhang, Jingyuan Chen, and Fei Wu. Autogeo: Automating geometric image dataset creation for enhanced geometry understanding. *IEEE Transactions on Multimedia*, 27:3105–3116, 2025.

Zhengbo Jiao, Shaobo Wang, Zifan Zhang, Wei Wang, Bing Zhao, Hu Wei, and Linfeng Zhang. Socratic-geo: Synthetic data generation and geometric reasoning via multi-agent interaction. *arXiv preprint arXiv:2602.03414*, 2026.

Mehran Kazemi, Hamidreza Alvari, Ankit Anand, Jialin Wu, Xi Chen, and Radu Soricut. Geomverse: A systematic evaluation of large models for geometric reasoning. *arXiv preprint arXiv:2312.12241*, 2023.

Haobo Lin, Tianyi Bai, Chen Chen, Jiajun Zhang, Bohan Zeng, Wentao Zhang, and Binhang Yuan. Synthesizing multimodal geometry datasets from scratch and enabling visual alignment via plotting code. *arXiv preprint arXiv:2602.18745*, 2026.

Deng Linger, Linghao Zhu, Yuliang Liu, Yu Wang, Qunyi Xie, Jingjing Wu, Gang Zhang, Yingying Zhu, and Xiang Bai. Theorem-validated reverse chain-of-thought problem generation for geometric reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 718–735, 2025.

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

Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-chun Zhu. Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 6774–6786, 2021.

Wufei Ma, Haoyu Chen, Guofeng Zhang, Yu-Cheng Chou, Jieneng Chen, Celso De Melo, and Alan Yuille. 3dsrbench: A comprehensive 3d spatial reasoning benchmark. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6924–6934. IEEE, 2025.

Maizhen Ning, Zihao Zhou, Qiufeng Wang, Xiaowei Huang, and Kaizhu Huang. Gns: Solving plane geometry problems by neural-symbolic reasoning with multi-modal llms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24957–24965, 2025.

Yicheng Pan, Zhenrong Zhang, Pengfei Hu, Jiefeng Ma, Jun Du, Jianshu Zhang, Quan Liu, Jianqing Gao, and Feng Ma. Enhancing the geometric problem-solving ability of multimodal llms via symbolic-neural integration. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 5394–5403, 2025.

Peter Petersen. *Riemannian geometry*, volume 171. Springer, 2006.

Runqi Qiao, Qiuna Tan, Guanting Dong, MinhuiWu MinhuiWu, Chong Sun, Xiaoshuai Song, Jiapeng Wang, Zhuoma GongQue, Shanglin Lei, Yifan Zhang, et al. We-math: Does your large multimodal model achieve human-like mathematical reasoning? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 20023–20070, 2025.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

Junxiao Wang, Ting Zhang, Heng Yu, Jingdong Wang, and Hua Huang. Magicgeo: Training-free text-guided geometric diagram generation. *arXiv preprint arXiv:2502.13855*, 2025a.

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. *Advances in Neural Information Processing Systems*, 37:95095–95169, 2024.

Peijie Wang, Chao Yang, Zhong-Zhi Li, Fei Yin, Dekang Ran, Mi Tian, Zhilong Ji, Jinfeng Bai, and Cheng-Lin Liu. Solidgeo: Measuring multimodal spatial math reasoning in solid geometry. *arXiv preprint arXiv:2505.21177*, 2025b.

Weixun Wang, Shaopan Xiong, Gengru Chen, Wei Gao, Sheng Guo, Yancheng He, Ju Huang, Jiaheng Liu, Zhendong Li, Xiaoyang Li, et al. Reinforcement learning optimization for large-scale learning: An efficient and user-friendly scaling library. *arXiv preprint arXiv:2506.06122*, 2025c.

Jingxuan Wei, Caijun Jia, Xi Bai, Xinglong Xu, Siyuan Li, Linzhuang Sun, Bihui Yu, Conghui He, Lijun Wu, and Cheng Tan. Ggbench: A geometric generative reasoning benchmark for unified multimodal models. *arXiv preprint arXiv:2511.11134*, 2025.

Changti Wu, Shijie Lian, Zihao Liu, Lei Zhang, Laurence Tianruo Yang, and Kai Chen. Dynasolidgeo: A dynamic benchmark for genuine spatial mathematical reasoning of vlms in solid geometry. *arXiv preprint arXiv:2510.22340*, 2025a.

Weiming Wu, Jin Ye, Zi-kang Wang, Zhi Zhou, Yu-Feng Li, and Lan-Zhe Guo. Nesygeo: A neuro-symbolic framework for multimodal geometric reasoning data generation. *arXiv preprint arXiv:2505.17121*, 2025b.

Renqiu Xia, Mingsheng Li, Hancheng Ye, Wenjie Wu, Hongbin Zhou, Jiakang Yuan, Tianshuo Peng, Xinyu Cai, Xiangchao Yan, Bin Wang, et al. Geox: Geometric problem solving through unified formalized vision-language pre-training. In *The Thirteenth International Conference on Learning Representations*.

Liangyu Xu, Yingxiu Zhao, Jingyun Wang, Yingyao Wang, Bu Pi, Chen Wang, Mingliang Zhang, Jihao Gu, Xiang Li, Xiaoyong Zhu, et al. Geosense: Evaluating identification and application of geometric principles in multimodal reasoning. *arXiv preprint arXiv:2504.12597*, 2025.

Ruoran Xu, Haoyu Cheng, Bin Dong, and Qiufeng Wang. Hilbert-geo: Solving solid geometric problems by neural-symbolic reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9658–9667, June 2026.

Sihan Yang, Runsen Xu, Yiman Xie, Sizhe Yang, Mo Li, Jingli Lin, Chenming Zhu, Xiaochen Chen, Haodong Duan, Xiangyu Yue, et al. Mmsi-bench: A benchmark for multi-image spatial intelligence. In *International Conference on Learning Representations*, volume 2026, pp. 157051–157088, 2026.

Baiqiao Yin, Qineng Wang, Pingyue Zhang, Jianshu Zhang, Kangrui Wang, Zihan Wang, Jieyu Zhang, Keshigeyan Chandrasegaran, Han Liu, Ranjay Krishna, et al. Spatial mental modeling from limited views. In *Structural Priors for Vision Workshop at ICCV'25*, 2025.

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*, 2024.

Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Botao Yu, Ge Zhang, Huan Sun, et al. Mmmu-pro: A more robust multi-discipline multimodal understanding benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025.

Jiaxin Zhang, Zhong-Zhi Li, Ming-Liang Zhang, Fei Yin, Cheng-Lin Liu, and Yashar Moshfeghi. Geoeval: Benchmark for evaluating llms and multi-modal models on geometry problem-solving. In *Findings of the Association for Computational Linguistics ACL 2024*, pp. 1258–1276, 2024a.

Ming-Liang Zhang, Fei Yin, and Cheng-Lin Liu. A multi-modal neural geometric solver with textual clauses parsed from diagram. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pp. 3374–3382, 2023.

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

Renrui Zhang, Xinyu Wei, Dongzhi Jiang, Ziyu Guo, Yichi Zhang, Chengzhuo Tong, Jiaming Liu, Aojun Zhou, Shanghang Zhang, Gao Peng, et al. Mavis: Mathematical visual instruction tuning with an automatic data engine. In *International Conference on Learning Representations*, volume 2025, pp. 87955–87989, 2025a.

Yuhao Zhang, Dingxin Hu, Tinghao Yu, Hao Liu, and Yiting Liu. Geofm: Enhancing geometric reasoning of mllms via synthetic data generation through formal language. *arXiv preprint arXiv:2510.27448*, 2025b.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Chengke Zou, Xingang Guo, Rui Yang, Junyu Zhang, Bin Hu, and Huan Zhang. Dynamath: A dynamic visual benchmark for evaluating mathematical reasoning robustness of vision language models. In *International Conference on Learning Representations*, 2025.

APPENDIX

A BENCHMARK DIFFICULTY IN CONTEXT

Several established plane-geometry and multimodal mathematical-reasoning benchmarks now yield high accuracy for strong MLLMs, reducing their ability to differentiate progress near the upper end. Figure 6 contextualizes S^{2}BENCH using the same Qwen3.5-27B model across general multimodal mathematical-reasoning benchmarks (Lu et al.; Wang et al., 2024; Zou et al., 2025; Yue et al., 2024; 2025; He et al., 2024; Qiao et al., 2025; Zhang et al., 2024b) and geometry-focused benchmarks (Zhang et al., 2024a; Chen et al., 2021; Zhang et al., 2023; Lu et al., 2021). The model achieves 82.3–95.7% on nine established benchmarks and 68.1–75.0% on three more challenging benchmarks, but only 19.4% on S^{2}BENCH. Although scores across benchmarks are not strictly comparable because their tasks and evaluation protocols differ, this contrast shows that S^{2}BENCH probes a substantially more challenging regime of fine-grained visual interpretation and complex mathematical and spatial reasoning. This underscores the value of S^{2}BENCH for tracking progress in complex multimodal reasoning.

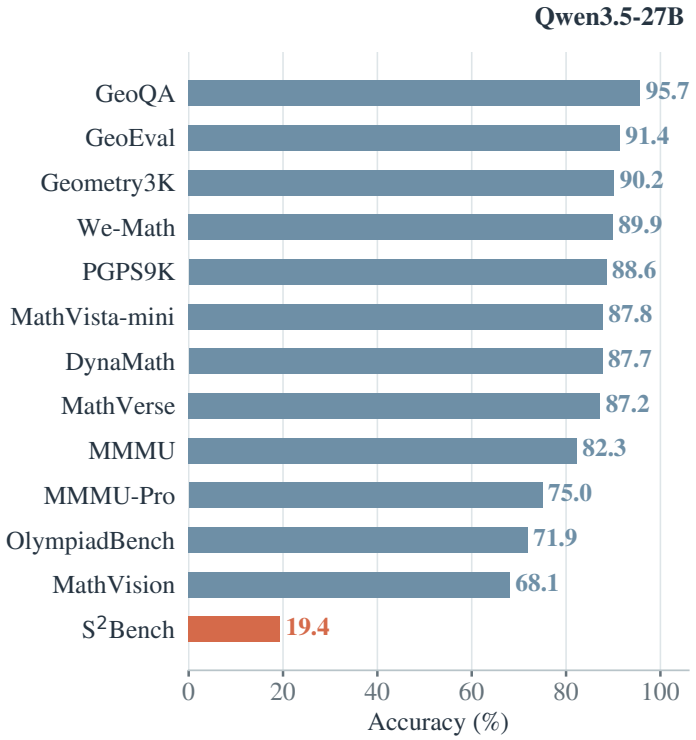


Figure 6: Accuracy of Qwen3.5-27B across established multimodal mathematical reasoning benchmarks and S^{2}BENCH. The substantially lower S^{2}BENCH score highlights the demands of fine-grained visual perception and complex 3D spatial reasoning.

B ADDITIONAL DETAILS OF THE SOLID-GEOMETRY FORMAL SYSTEM

This appendix details the formal representation and executable rule system introduced in Section 3.1. We first describe how geometric objects, relations, and quantities are represented, and then illustrate how executable theorem rules transform these conditions into traceable solutions.

B.1 GEOMETRY DESCRIPTION LANGUAGE

Typed geometric symbols. The GDL contains three complementary types of symbols. Entity predicates declare geometric objects or structured spatial configurations. Relation predicates de-

Table 5: Representative symbols in the solid-geometry GDL.

Type	Example	Semantics
Entity	$\mathrm{Cube}(ABCDEFGH)$	A cube with labeled vertices
	$\mathrm{Cylinder}(P, Q)$	A cylinder with two circular bases
	$\mathrm{VoxelAssembly}(S)$	A structured voxel configuration
	$\mathrm{TriangularPrism}(ABC, DEF)$	A triangular prism with two labeled bases
	$\mathrm{QuadrangularPyramid}(ABCD, O)$	A pyramid with base $ABCD$ and apex O
	$\mathrm{RegularOctahedron}(ABCD, E, F)$	A regular octahedron with labeled vertices
	$\mathrm{DieRollPath}(D)$	A die-orientation sequence
Relation	$\mathrm{Midpoint}(M, AB)$	M is the midpoint of AB
	$\mathrm{Perpendicular}(AB, U)$	Line AB is perpendicular to plane U
	$\mathrm{InscribedSphere}(O, ABCDEFGH)$	Sphere O is inscribed in the cube
	$\mathrm{Parallel}(U, V)$	Planes U and V are parallel
	$\mathrm{Projection}(H, P, U)$	H is the projection of P onto plane U
	$\mathrm{TangentPlane}(U, O)$	Plane U is tangent to sphere O
	$\mathrm{SectionCircle}(P, O, U)$	Circle P is the section of sphere O by plane U
Attribute	$\mathrm{Length}(AB)$	Length of segment AB
	$\mathrm{Volume}(P, Q)$	Volume of cylinder (P, Q)
	$\mathrm{Distance}(P, U)$	Distance from point P to plane U
	$\mathrm{Angle}(AB, U)$	Angle between line AB and plane U
	$\mathrm{Area}(ABCD)$	Area of quadrilateral $ABCD$
	$\mathrm{Radius}(O)$	Radius of sphere or circle O
	$\mathrm{Height}(P, Q)$	Height of cylinder (P, Q)
	$\mathrm{Ratio}(P, AB)$	Position ratio of point P on line AB

scribe how these objects interact, including incidence, parallelism, perpendicularity, projection, tangency, and inscribed or circumscribed configurations. Attribute functions map objects to symbolic quantities such as length, area, volume, distance, angle, and ratio. Table 5 gives representative examples.

Compositional problem representation. Predicate applications form logical conditions, while attribute functions can be combined into algebraic constraints and numerical targets. As defined in Equation 2, the condition set is partitioned by information source: $\mathcal{C}_{\mathrm{con}}$ specifies the geometric construction, $\mathcal{C}_{\mathrm{text}}$ records givens stated in the question, and $\mathcal{C}_{\mathrm{img}}$ records facts grounded in the diagram. A fact may appear in more than one subset when it is conveyed by multiple modalities. Their union, together with the target g, defines a single formal problem shared by text generation, rendering, and reasoning.

Consider a cylinder with radius 3 and height 5. Using abbreviated attribute names, its essential formal representation is

$$\begin{aligned} \mathcal{C} &= \{\mathrm{Cylinder}(P, Q), \mathrm{Radius}(P) = 3, \mathrm{Height}(P, Q) = 5\}, \\ g &= \mathrm{Volume}(P, Q). \end{aligned} \tag{13}$$

The entity predicate fixes the solid type, the attribute equalities provide its measurements, and the target identifies the requested quantity. The same representation can support both a natural-language question and a diagram in which the radius and height are visually labeled.

Representation statistics. The formal representation comprises 50 entity predicates, 100 relation predicates, and 322 attribute functions, for a total of 472 GDL symbols. These symbols cover conventional solid geometry as well as structured spatial tasks involving voxels, projections, folding, and orientation.

Table 6: Scale of the solid-geometry formal representation.

Symbol type	Count
Entity predicates	50
Relation predicates	100
Attribute functions	322
Total	472

Table 7: Representative capabilities of the executable theorem library.

Capability	Representative conclusion
Solid metrics	Cylinder volume, sphere volume, and surface area
Spatial relations	Transitivity of parallel and perpendicular planes
Projection and distance	Point–plane distance from an orthogonal projection
Cross-sections	Shape and area of axial or oblique sections
Inscribed/circumscribed solids	Radius relations between a sphere and its host solid
Similarity and composite solids	Scale relations and additive/subtractive volumes
Unfolding and coordinates	Surface-path lengths and vector-based distances or angles

B.2 Executable Theorem Library

Rule organization. The theorem library represents geometric knowledge as executable rules. A *theorem family* groups rules with a common geometric meaning and argument signature, while a *branch* specifies one admissible premise–conclusion pattern. Given a grounding of the rule variables to objects in a problem, a branch can be applied when all its premises are satisfied. Its conclusions are then added to the symbolic state together with their theorem provenance.

For example, plane transitivity contains two branches:

$$U \parallel V \ \wedge \ V \parallel W \ \Rightarrow \ U \parallel W, \tag{14}$$

$$U \parallel V \ \wedge \ V \perp W \ \Rightarrow \ U \perp W. \tag{15}$$

Both branches share the same three plane variables but encode different consequences. A grounded application replaces (U, V, W) with the planes of a particular problem, allowing the solver to check the premises and record the resulting relation explicitly.

From geometric rules to symbolic solutions. Rules may conclude geometric predicates, algebraic constraints, or both. For a cylinder with base P and top Q, the volume rule is

$$\mathrm{Cylinder}(P, Q) \quad \Rightarrow \quad V(P, Q) = A(P)H(P, Q), \tag{16}$$

where V, A, and H denote cylinder volume, base area, and height. Combining this rule with $A(P) = \pi r^2$ and the instantiated measurements yields the target value. For the example in Appendix B.1, $r = 3$ and $h = 5$ give $V = 45\pi$. The same execution mechanism supports rules with several conclusions; an axial-section rule, for instance, identifies the section as a rectangle and derives its side-length and area relations.

A multi-step solution alternates between rule application and symbolic equation solving. Each derived condition retains the premises and theorem that produced it, so the final answer can be traced to the instantiated givens and rules. This provenance forms the formal basis of the reasoning trace $\tau_{\mathcal{T}}$ used in Section 3.3.

Knowledge coverage. The library combines general Euclidean knowledge with solid-specific rules covering metric computation, spatial relations, cross-sections, similarity, inscribed and circumscribed configurations, composite solids, unfolding, and coordinate–vector reasoning. Table 7 summarizes representative capabilities and conclusions.

Table 8: Representative generation schemas. Constraints define the admissible parameter domain before the formal problem is constructed.

Problem family	Sampled parameters	Representative constraints	Possible targets
Cylinder	Radius and height	Positive dimensions	Volume or surface area
Polyhedral section	Edge lengths and section ratios	Nondegenerate intersection	Section area or cut volume
Voxel assembly	Occupied cells and marked cubes	Valid, connected structure	Count or spatial relation
Die rolling	Initial orientation and move sequence	Valid orientation transitions	Final face or orientation

Theorem-based and executor-based reasoning. For theorem-based problems, the target follows from geometric conditions and equations introduced by the theorem library. Representative examples include volume, similarity, and cross-section derivations in SGMR. Some tasks instead use a deterministic executor to compute a state transition, enumeration, or combinatorial quantity. Examples include voxel counting and state changes (VC and VSTR), and voxel relations (VSRR); exact-coordinate section instances in FGVGR use the same executor interface. The distinction is therefore made at the instance level rather than imposed on an entire task category. An interface rule then connects the executor result to the formal target, allowing both solution types to share the same target representation and validation interface. Such an interface rule certifies the formal connection to the target; the underlying executor computation is checked separately, as detailed in the validation example in Appendix D.

Library statistics. The complete formal system contains 680 theorem families with 722 branches, covering both general Euclidean and solid-geometry reasoning. The theorem sequences in S^2BENCH instantiate 76 distinct families. These statistics count reusable rule definitions rather than their grounded applications across individual problems.

C Additional Details of Structured Solid-Geometry Generation

This appendix expands the generation process in Section 3.2. We first describe how a generation schema defines a family of valid geometric problems, and then follow one released example from parameter instantiation to its aligned question, diagram, target, and reasoning trace.

C.1 Generation Schemas and Parameter Sampling

A generation schema specifies a geometric structure, its admissible parameter domain, a set of possible targets, and the information required to render the structure. Sampling $\theta \in \Theta_G$ instantiates the formal conditions $\mathcal{C}$ and rendering specification ρ jointly, as defined in Equation 4. The admissible domain constrains both individual parameters and their geometric interactions. It therefore excludes configurations such as zero-length edges, degenerate sections, disconnected voxel assemblies, and transformations with ambiguous states before the question is realized.

Table 8 illustrates this design across representative families. A single schema can support several targets over the same structure, while different parameter samples produce distinct numerical instances and diagrams.

The sampled parameters represent the semantic state of a problem. Rendering parameters such as viewpoint, scale, and label placement are introduced only after this state is fixed. They may change how a diagram is presented, but do not change its formal conditions or target.

C.2 A Worked Generation Example

Figure 7 shows a released S^2BENCH instance generated from a cone–sphere schema. A right circular cone with base O and vertex D is inscribed in sphere S. A plane parallel to the base cuts the cone

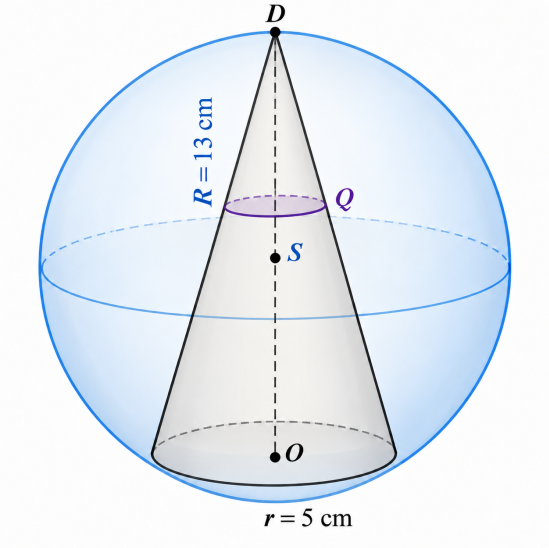


Figure 7: Generated diagram for the worked example. The circumsphere, cone, and parallel section are instantiated from the same parameters as the formal conditions and question.

Table 9: Condensed theorem trace for the worked generation example.

Deduction	Derived relation or value
Recover cone height	$h = R + \sqrt{R^2 - r^2} = 25$
Infer the similarity ratio	$V_f = \frac{1}{3}\pi r^2 h(1 - \lambda^3)$, hence $\lambda = 2/5$
Instantiate the upper cone	$r_Q = \lambda r = 2$ and $h_Q = \lambda h = 10$
Compute its total surface area	$L(Q, D) + A(Q) = 4\pi(1 + \sqrt{26})$
Compute its volume	$V(Q, D) = 40\pi/3$
Evaluate the target	$g = \frac{3}{10}(1 + \sqrt{26})$

in circle Q, forming a smaller similar cone above the section. The instantiated parameters are

$$\theta = (R, r, V_f) = (13, 5, 195\pi), \tag{17}$$

where R and r are the sphere and base radii, and V_f is the frustum volume.

Using abbreviated notation, the essential formal conditions and target are

$$\begin{aligned} \mathcal{C} = \{&\text{Cone}(O, D), \text{Circumsphere}(S, O, D), \text{ParallelSection}(Q, O, D), \\ &R(S) = 13, r(O) = 5, V_f(Q, O, D) = 195\pi\}, \end{aligned} \tag{18}$$

$$g = \frac{L(Q, D) + A(Q)}{V(Q, D)}, \tag{19}$$

where $L(Q, D)$ is the lateral area of the smaller cone, $A(Q)$ is its base area, and $V(Q, D)$ is its volume. The rendering specification binds the same objects to the sphere outline, cone boundary, section ellipse, and dimension labels in Figure 7. Thus, the symbols S, O, D, and Q have a shared identity across the formal representation, question, and diagram.

The schema selects a target supported by the instantiated configuration and constructs a theorem trace. The principal deductions are summarized in Table 9. Because the question specifies the taller cone, the circumsphere relation selects $h = R + \sqrt{R^2 - r^2} = 25$. The known frustum volume then determines the similarity ratio of the upper cone. Similarity yields its radius, height, lateral area, and volume, from which the target follows.

Finally, the text realization describes the same cone, measurements, parallel section, frustum volume, and requested ratio in natural language. The resulting candidate therefore contains an aligned question, formal problem, diagram, answer, and theorem-grounded trace. Solver-grounded validation is applied only after this candidate has been assembled.

Table 10: Two candidate-solution construction paths in SolidGeoGen.

Path	Typical tasks	Trace content
Theorem-based	Metric geometry, sections, similarity, spatial angles	Grounded theorem applications and symbolic calculations
Executor-based	Counting, folding, orientation, visibility, and state transformation	Deterministic state updates, enumeration, and derived quantities

C.3 Target and Reasoning Construction

Targets are selected after the geometric structure has been instantiated, so every queried quantity is defined by the sampled configuration. Depending on the schema, the target may request a geometric relation, a metric quantity, a ratio of derived quantities, a discrete count, or a final spatial state. Changing the target can yield a new problem over the same underlying scene, provided that an executable derivation is available.

SolidGeoGen constructs candidate solutions through two complementary paths. Theorem-based construction instantiates rules from $\mathcal{T}$ and combines their conclusions through symbolic equation solving, as in the worked example. Executor-based construction applies deterministic operations to structured states, such as voxel sets, face adjacencies, folding configurations, or die orientations. Both paths produce a structured trace τ and a candidate answer $\widetilde{a}$; their different validation scopes are discussed in Appendix D.

In either path, the full reasoning trace can include given conditions, geometric or state-transition steps, symbolic calculations, and the final evaluation. The theorem subtrace $\tau_{\mathcal{T}}$ records the executable theorem applications used by the symbolic solver.

C.4 Question Generation and Diagram Rendering

Question generation. Question and diagram generation operate on the same instantiated geometric state. The text generator selects the conditions that should be exposed to the solver and expresses them together with the target through controlled language patterns. Object names and numerical values are filled from the shared parameters θ, ensuring that the question refers to the same objects and measurements as the formal representation.

Geometry-to-diagram mapping. The rendering specification ρ describes the visual geometry associated with the instantiated problem. For polyhedral scenes, it contains vertices, edges, faces, and constructed objects such as section points or planes. For round solids, it specifies axes, boundary curves, and circular or elliptical sections. Discrete spatial tasks use corresponding structures such as voxel cells, face adjacencies, nets, or orthographic views. Because these elements are derived from the same $heta$ used to construct $\mathcal{C}$, changing a dimension, section ratio, or spatial state updates both the formal problem and its diagram.

Projection and visibility. For a three-dimensional point $\mathbf{X}_i$, an orthographic view can be written as

$$\mathbf{p}_i = \Pi_\nu(\mathbf{X}_i) = s_\nu \begin{bmatrix} \mathbf{u}_\nu^\top \\ \mathbf{v}_\nu^\top \end{bmatrix} \mathbf{X}_i + \mathbf{t}_\nu, \tag{20}$$

where $\mathbf{u}_\nu$ and $\mathbf{v}_\nu$ span the image plane, and s_ν and $\mathbf{t}_\nu$ control scale and placement. The renderer uses the selected view to order surfaces and distinguish visible from hidden structure. Visible boundaries are drawn as solid lines, hidden edges as dashed lines, and relevant sections or faces as translucent regions. Circular bases and spherical sections are represented by projected curves, while nets and orthographic multi-view tasks are constructed directly in their canonical two-dimensional layouts.

Semantic annotations. Image-grounded conditions in $\mathcal{C}_{\mathrm{img}}$ are bound to visual elements in ho. These bindings determine point labels, dimension labels, colored objects, equal-length ticks, right-angle marks, arrows, and highlighted sections. An annotation is introduced only when it belongs to the information available in the problem; derived quantities and the target answer are not rendered.

Table 11: Rendering strategies for representative solid-geometry and spatial reasoning tasks.

Task family	Geometric representation	Rendered evidence
Metric solid geometry	Vertices, edges, axes, and curved boundaries	Dimensions, labeled objects, and visible/hidden edges
Cross-section reasoning	Solid boundary and plane–solid intersections	Section polygon or ellipse, intersection points, and plane cues
Voxel and polyhedral reasoning	Occupied cells, faces, and adjacency structure	Visible faces, marked elements, colors, and occlusion cues
Folding and orientation	Face layout or discrete orientation state	Nets, markers, fold relations, and directional arrows
Projection and multi-view	Three-dimensional structure and view directions	Orthographic panels, shadows, and candidate views

In the worked example, the labels S, O, D, and Q, the two radii, and the parallel section are all attached to their corresponding geometric objects rather than placed as independent text.

Controlled visual variation. The nuisance parameters ν vary viewpoint, scale, spacing, and label placement without changing the underlying problem. Admissible views avoid degenerate projections and preserve the visibility of task-relevant objects. For tasks in which color, orientation, or viewpoint carries semantic meaning, that variable is included in the geometric state rather than treated as nuisance variation. This separation allows SolidGeoGen to diversify diagram appearance while keeping the formal conditions, question, and answer fixed.

Table 11 summarizes how the same rendering principles are adapted to different task families.

Manual visual audit. To complement the shared-parameter construction, we manually audited 100 released instances sampled across the ten task categories. Each diagram was checked against its question and formal representation for semantic consistency, visibility of task-relevant elements, correctness of labels and visual cues, and the absence of clipping, ambiguous overlap, or other rendering artifacts. All 100 instances passed these checks, providing direct evidence of diagram quality beyond the automatic alignment mechanism.

The resulting alignment is structural: q verbalizes selected conditions, I visualizes the objects bound by ρ, and both originate from the same instantiated state. The symbolic validator checks the formal derivation and answer, while diagram fidelity is supported by this shared construction and the manual visual audit above.

C.5 Scalability of Schema-Guided Symbolic Generation

SolidGeoGen operates over a large global space induced jointly by 558 parameterized generation schemas, 680 theorem families with 722 branches, continuous or combinatorial scene parameters, target choices, and rendering variation. A schema represents a reusable problem family rather than an individual question: changing its geometric structure, parameters, or target produces a new formal problem and reasoning trace. The resulting instance space is therefore substantially larger than the number of schemas alone.

Exhaustively enumerating theorem sequences over the complete library would be inefficient. Instead, schemas provide semantic constraints that restrict theorem selection and grounding to the rules relevant to an instantiated scene. The global symbolic space is thus expressive, while execution for an individual instance remains sparse: the solver checks a compact grounded trace rather than branching over every theorem family.

We measure this behavior in a single-process CPU evaluation on an Intel Xeon Platinum 8369B. First, we generate one instance from every schema using a fixed random seed (2027). Symbolic construction and validation, excluding diagram rendering, require 0.133 seconds per problem on average, and all 558 schemas produce a validated instance within the standard attempt budget. Second, we replay the formal solutions of all 2,000 released S^2BENCH instances. Table 12 reports the resulting statistics.

Table 12: Scale and efficiency of schema-guided symbolic generation. Runtime is measured with a single process; symbolic generation excludes diagram rendering.

Statistic	Result
Generation schemas	558
Theorem families / branches	680 / 722
Validated schemas in the coverage audit	558 / 558
Structured trace length (mean / median / P_{95} / max)	5.76 / 6 / 7 / 13
Symbolic generation time (mean / median / P_{95})	0.133 / 0.071 / 0.412 s
Solver replay time (mean / median / P_{95})	11.3 / 4.4 / 12.9 ms
Successful replay on S^2BENCH	2,000 / 2,000

Table 13: Validation of the two solution-construction paths.

Path	Candidate source	Solver-grounded check	Complementary check
Theorem-based	Geometric rules and symbolic evaluation	Target derivability and answer agreement	Independent audit of theorem semantics
Executor-based	State transitions or exact enumeration	Formal connection between the computed result and target	Independent re-execution from the initial state

These results show that library-wide expressivity does not require library-wide search for each problem. Schema-guided grounding converts the global combinatorial space into short executable traces, enabling efficient generation while retaining solver-grounded verification.

D Additional Details of Symbolic Replay and Validation

This appendix expands the validation procedure in Section 3.3. Its purpose is to check an instantiated candidate, rather than to compensate for an unreliable symbolic system. A generation schema jointly constructs formal conditions, a reasoning trace, and a candidate answer; however, these outputs are assembled by several operations, including parameter substitution, theorem grounding, and symbolic or discrete evaluation. Validation checks that the assembled instance remains consistent with the formal state from which it was generated.

D.1 Validation Protocol and Scope

SolidGeoGen uses two solution paths. In the theorem-based path, the candidate answer is obtained from geometric rules and symbolic equations. The validator replays the theorem subtrace $\tau_{\mathcal{T}}$ from $\mathcal{C}$ and solves the resulting formal state for the target g. In the executor-based path, the answer is produced by deterministic state transitions or exact enumeration, and a bridge rule connects the computed quantity to g. Table 13 summarizes the corresponding checks.

For theorem-based candidates, the solver starts from $\mathcal{C}_0 = \mathcal{C}$ and applies the recorded rules in order. A rule adds its conclusions only if its premises hold; an inapplicable rule makes no change, and replay continues. The final decision is made from the target: the candidate is accepted only when the replayed state determines $a_{\mathrm{sym}} = \mathrm{Solve}(g \mid \mathcal{C}_K)$ and $a_{\mathrm{sym}} \simeq \widetilde{a}$. Thus, an omitted or incorrectly grounded step is rejected whenever it leaves the target underived or produces a different answer. The candidate answer participates only in this final comparison; it is not used as a premise from which the target is derived.

This protocol establishes instance-level agreement along the chain

$$\mathcal{C} \xrightarrow{\tau_{\mathcal{T}}} g = a_{\mathrm{sym}} \simeq \widetilde{a}. \tag{21}$$

It does not suggest that theorem application is intrinsically error-prone. Rather, it provides an executable check at the boundary between the formal system and the schema that instantiates it.

Table 14: Validation trace for a rejected theorem-based candidate.

Stage	Result
Formal state	Cube side $s = 6$ and a midpoint corner cut
Grounded theorem	$V = s^3/48$
Candidate evaluation	$\widetilde{a} = 4$
Solver replay	$a_{\mathrm{sym}} = 9/2$
Decision	Reject because $a_{\mathrm{sym}} \not\approx \widetilde{a}$

D.2 A Rejected Theorem-Based Candidate

We illustrate this boundary with a candidate that is rejected before dataset inclusion. Consider a cube of side length $s = 6$. The midpoints I, J, K of three mutually perpendicular edges incident to a vertex A cut off the tetrahedron A–IJK. The corresponding theorem states

$$\mathrm{CornerCut}(IJK, A, ABCDEFGH) \;\Rightarrow\; V(A\text{–}IJK) = \frac{s^3}{48}. \tag{22}$$

The geometric conditions and grounded theorem are valid, but a defect in the schema-level numerical evaluation rounded $6^3/48$ down and produced the candidate answer $\widetilde{a} = 4$. Replaying the theorem instead yields

$$a_{\mathrm{sym}} = \frac{6^3}{48} = \frac{9}{2} \neq 4, \tag{23}$$

so the candidate is rejected. Table 14 separates the valid geometric construction from the inconsistent answer.

The same result can be checked independently of the specialized theorem. With $A = (0, 0, 0)$, $I = (3, 0, 0)$, $J = (0, 3, 0)$, and $K = (0, 0, 3)$,

$$V = \frac{1}{6}\,|\det[I - A,\; J - A,\; K - A]| = \frac{27}{6} = \frac{9}{2}. \tag{24}$$

This independent calculation audits the mathematical meaning of the encoded rule, whereas solver replay checks whether a particular candidate executes consistently under that rule. The two checks therefore address different failure modes.

D.3 Validation of Executor-Based Solutions

Some spatial tasks are more naturally expressed as finite computations than as chains of Euclidean theorems. Examples include rolling a die, folding a net, updating a voxel assembly, and enumerating visible or adjacent elements. For such a task, let z_0 be the initial structured state and $o_1, \ldots, o_L$ the operations specified by the instance. A deterministic executor computes

$$z_t = E_{o_t}(z_{t-1}), \quad t = 1, \ldots, L, \qquad b = h(z_L), \tag{25}$$

where h extracts the requested quantity. A bridge rule then associates the computed value b with the formal target g. Symbolic replay verifies this association, while an independent executor audit recomputes Equation 25 from z_0 and the operation sequence rather than trusting the stored result.

For die rolling, write an orientation as $z = (t, b, f, k, \ell, r)$ for the top, bottom, front, back, left, and right faces. The four one-square transitions are exact permutations:

$$\begin{aligned} E_{\mathtt{R}}(z) &= (\ell, r, f, k, b, t), \quad E_{\mathtt{L}}(z) = (r, \ell, f, k, t, b), \\ E_{\mathtt{U}}(z) &= (f, k, b, t, \ell, r), \quad E_{\mathtt{D}}(z) = (k, f, t, b, \ell, r). \end{aligned} \tag{26}$$

Figure 8 shows an instance from the die rolling category of S^{2}BENCH. Its initial top, bottom, front, back, left, and right faces are $(5, 2, 6, 1, 3, 4)$, and its move sequence is `LLLDRRRRRDDRD`. Each move applies a fixed permutation of the six faces. Exact replay gives the successive top faces

$$(4, 2, 3, 1, 2, 6, 5, 1, 2, 4, 5, 6, 3) \tag{27}$$

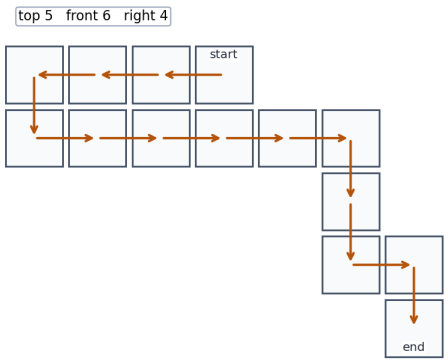


Figure 8: Executor-based validation example from S^{2}BENCH. The initial die orientation and arrow path define a finite sequence of exact orientation transitions.

Table 15: Scope of the checks used in SolidGeoGen.

Property	Validation mechanism
Rule application	Conclusions are added only when their premises hold
Target and answer consistency	Recomputed a_{sym} compared with $\widetilde{a}$
Executor correctness	Exact state replay or enumeration from instance inputs
Theorem semantics	Independent coordinate, vector, algebraic, or enumerative audit
Diagram alignment	Shared generation parameters and batch-level visual inspection

and the final visible orientation $(T, F, R) = (3, 6, 5)$. Hence the requested score is

$$b = 20T + 5F + R = 20 \cdot 3 + 5 \cdot 6 + 5 = 95. \tag{28}$$

For this instance, the formal state contains an encoded value of 95, and the bridge theorem derives the queried orientation score from that value. Solver replay therefore confirms that the formal target and candidate answer agree. Separately replaying the face permutations confirms that the encoded value itself follows from the initial orientation and path. This distinction is important: a bridge theorem alone certifies the formal connection to the target, but not the correctness of the preceding discrete computation.

The same principle applies across executor families. Voxel tasks replay set updates or enumerate occupied cells; folding tasks propagate face frames across net adjacencies; and graph-based tasks recompute connectivity, visibility, or shortest paths from the instantiated structure. These audits operate on exact discrete states and are therefore not subject to geometric rounding.

D.4 VALIDATION GUARANTEES

Table 15 summarizes the property checked by each mechanism. The separation is deliberate: no single check establishes formal executability, mathematical correctness, and visual fidelity simultaneously.

Solver replay is a certificate relative to the theorem library $\mathcal{T}$; it cannot detect an incorrect theorem if the generator and solver share the same erroneous definition. We therefore audit new theorem families against independent mathematical formulations. Likewise, the solver does not inspect rendered pixels. Diagram fidelity is supported by constructing $\mathcal{C}$ and the rendering specification from shared parameters, followed by visual inspection of generated batches. Only candidates satisfying the applicable formal answer check are released, and executor and visual audits provide complementary evidence for properties outside the scope of symbolic replay.

Table 16: Sources of variation in generated problems. Semantic variation changes the problem itself; rendering variation changes only its presentation.

Variation	Representative examples	Type
Structure	Solid type, section topology, voxel occupancy, or net layout	Semantic
Parameters	Lengths, ratios, marked elements, or initial orientation	Semantic
Target	Volume, spatial relation, final state, or discrete count	Semantic
Reasoning	Theorem composition, state transitions, or enumeration	Semantic
Rendering	Viewpoint, scale, layout, and label placement	Presentation

Table 17: Complementary train–test decontamination checks.

Check	Compared content
Exact instance	Question–diagram pair and associated answer
Formal problem	Task family, normalized formal conditions, and queried target; answers and traces are checked separately
Structural equivalence	Canonicalized objects and task-preserving geometric symmetries

E Dataset Diversity and Train–Test Decontamination

This appendix describes how we distinguish genuine problem variation from surface variation and how we prevent evaluation instances from entering the training corpus. The released S^2Bench contains 2,000 problems, balanced at 200 instances for each of the ten task categories. S^2Train contains 10,000 instances, with 1,000 per category. The benchmark is fixed before training examples are screened, and all retained instances pass the validation procedure described in Appendix D.

E.1 Diversity Beyond Surface Form

We treat two instances as meaningfully different only when their underlying problem states differ. Diversity therefore arises from five sources: geometric structure, numerical or discrete parameters, queried targets, reasoning paths, and visual realization. Table 16 summarizes their roles. Changes that affect only object names or linguistic wording are not counted as new geometric problems.

Generation schemas vary these factors under task-specific admissibility constraints. For metric geometry, this includes changing dimensions, constructed points, and the requested quantity. For discrete spatial tasks, the sampled state may instead be a voxel arrangement, face-adjacency graph, folding configuration, or orientation trajectory. Consequently, diversity is measured from the instantiated geometric or discrete state rather than from the number of language templates. Rendering parameters are varied only after the semantic state is fixed and therefore provide visual robustness without being treated as additional problems.

E.2 Train–Test Decontamination

Each candidate for S^2Train is compared against the fixed S^2Bench at the levels shown in Table 17. The first check detects identical multimodal instances. The second compares the complete problem specification, including its formal conditions and target, so that superficial differences in wording do not hide the same problem. The final check uses category-specific structural normalization for task-preserving symmetries.

Object identifiers are canonicalized before the formal comparison. Thus, renaming the vertices of a cuboid does not create a new problem. For voxel and multi-view tasks, equivalent grid representations under admissible rotations or reflections are compared in a common orientation. Similar category-specific representations encode a path by its relative transitions, a net by face adjacency

Table 18: Dataset construction and decontamination summary.

Statistic	Result
S^2BENCH instances	2,000
S^2TRAIN instances	10,000
Task categories	10
Instances per category (benchmark / training)	200 / 1,000
Validated instances (benchmark / training)	2,000 / 10,000
Exact train–test problem matches after screening	0
Normalized structural matches after screening	0

and marker bindings, and a graph task by its labeled adjacency structure. A candidate is removed if it matches any benchmark instance under an applicable check.

Image equality is used as an auxiliary diagnostic rather than as the sole definition of leakage. A schematic diagram may legitimately be shared by different formal problems when parameter values or targets are stated only in the question. Conversely, two renderings of the same voxel structure may have different pixels because of viewpoint or layout. The decontamination decision therefore follows the joint semantic record, while image hashes provide an additional exact-match check.

E.3 AUDIT SUMMARY

Table 18 summarizes the released data snapshot. All 2,000 benchmark instances and 10,000 training instances pass their applicable formal or deterministic validation checks. After decontamination, no training instance matches a benchmark instance by its complete problem record, and no collision remains under the applicable category-specific structural checks.

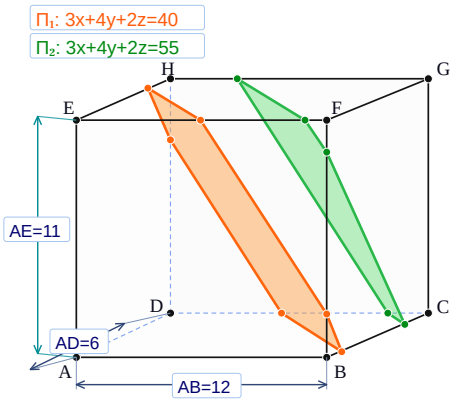
These checks are stricter than lexical deduplication: changing a diagram layout, paraphrasing a question, or renaming geometric objects cannot move an evaluation problem into the training split. They also preserve legitimate diversity, since problems that share a schema but instantiate different structures, measurements, targets, or transition sequences remain distinct.

F ILLUSTRATIVE CASES FROM S^{2}BENCH

Figures 9–29 present 21 representative cases from the ten task categories of S^{2}BENCH. Each case contains the original question–diagram pair, structured visual information, structured reasoning steps, and the reference answer. The structured annotations are shown only to illustrate the symbolic supervision provided by SOLIDGEOGEN; they are not available to models evaluated on S^{2}BENCH. Together, these cases demonstrate the benchmark's coverage of solid-geometry calculation, fine-grained visual interpretation, folding, projection, multi-view reasoning, and voxel-based spatial transformations.

Solid-Geometry Metric Reasoning (SGMR) **Case 0001**

Problem



The figure shows transparent rectangular prism *ABCD-EFGH*. The orange and green polygons are cross-sections made by two parallel oblique planes Π_1 and Π_2. Use the coordinate system with *A* as the origin, *AB* as the *x*-axis, *AD* as the *y*-axis, and *AE* as the *z*-axis; the prism dimensions and the two plane equations are marked in the figure. Let *V* be the volume of the part of the prism between Π_1 and Π_2, *X* be the area of the orange section, *Y* be the area of the green section, and Δ be the distance between the two planes. Let *m* and *n* be the numbers of sides of the orange and green section polygons. Write $V = a/b$, $X^2 = c/d$, $Y^2 = e/f$, and $\Delta^2 = g/h$ in lowest terms. Find $13a+ 17b+ 17c+ 23d+ 31e+ 29f+ 31g+ 41h+ 43m+ 59n$. Return only the integer.

Structured Annotations

Structured Visual Information


```
{
  "coordinate_convention": "A is origin; AB is +x, AD is +y, AE is +z",
  "dimensions": {"AB": 12, "AD": 6, "AE": 11},
  "parallel_planes": [
    {
      "plane": "Pi1", "drawn_color": "orange",
      "normal_components": [3, 4, 2], "constant": 40
    },
    {
      "plane": "Pi2", "drawn_color": "green",
      "normal_components": [3, 4, 2], "constant": 55
    }
  ],
  ...
}
```


Structured Reasoning Steps

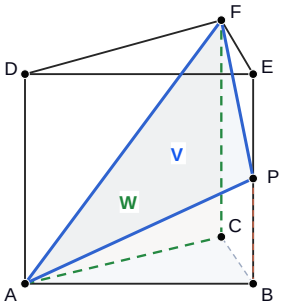

```
{
  "steps": [
    { "step_id": 1, "type": "given",
      "natural": "The diagram gives a rectangular prism and two parallel oblique section polygons."
    },
    { "step_id": 2, "type": "calculation",
      "natural": "Use the displayed coordinate equations to locate where each plane cuts the prism edges."
    },
    { "step_id": 3, "type": "calculation",
      "natural": "Compute V, X², Y², Δ², m, and n. Use clipped-box volume and section-area computations
                  for the two parallel planes."
    },
    { "step_id": 4, "type": "calculation",
      "natural": "Evaluate the weighted expression requested in the question using the reduced rational
                  pairs and the section side counts."}
  ]
}
```


Reference answer: 465436598

Figure 9: Representative S^{2}BENCH case for Solid-Geometry Metric Reasoning (SGMR), example 1 of 2.

Solid-Geometry Metric Reasoning (SGMR) — Case 0086

Problem



In a regular triangular prism *ABC-DEF*, all edges are equal but their common length is unknown. Point *P* moves on *BE*. Plane *APF* is denoted by *V*, and plane *ACF* is denoted by *W*. When the cosine of the dihedral angle between *V* and *W* along *AF* is minimal, the volume of tetrahedron *P-ACF* is $\frac{1125\sqrt{3}}{4}$ cubic centimeters. Find the ratio of the total surface area of prism *ABC-DEF* to its volume.

Structured Annotations

Structured Visual Information


```
{
  "shape": "regular triangular prism ABC-DEF with every edge equal to unknown s",
  "symbolic_coordinates": {
    "A": ["0", "0", "0"], "B": ["s", "0", "0"],
    "C": ["s/2", "sqrt(3)*s/2", "0"], "D": ["0", "0", "s"],
    "E": ["s", "0", "s"], "F": ["s/2", "sqrt(3)*s/2", "s"],
    "P": "B + t*(E-B), with 0<=t<=1"
  },
  "visual_bindings": {
    "moving_point": "P on BE", "plane_V": "APF",
    "plane_W": "ACF", "dihedral_edge": "AF"
  },
  "given_measurements": [
    "Volume of P-ACF at the minimizing position = 1125*sqrt(3)/4"
  ],
  "unknown_parameters": ["s", "t"],
  ...
}
```


Structured Reasoning Steps


```
{
  "steps": [
    {
      "step_id": 1, "type": "given",
      "natural": "Identify the regular triangular prism,
                  the moving point P on BE, and the planes
                  V=APF and W=ACF."
    },
    {
      "step_id": 2, "type": "given",
      "natural": "Use the equality of the prism height
                  and common edge length, together with the
                  given tetrahedron volume at the minimizing
                  position."
    },
    {
      "step_id": 3, "type": "theorem",
      "natural": "Express the AF-dihedral cosine through
                  BP/BE."
    },
    {
      "step_id": 4, "type": "theorem",
      "natural": "Determine the position of P on BE that
                  minimizes the dihedral cosine."
    },
    {
      "step_id": 5, "type": "theorem",
      "natural": "Determine the minimal cosine value."
    },
    {
      "step_id": 6, "type": "theorem",
      "natural": "Invert the extremal tetrahedron volume
                  to recover AB."
    },
    {
      "step_id": 7, "type": "theorem",
      "natural": "Compute triangle ACF as an auxiliary section."
    },
    {
      "step_id": 8, "type": "theorem",
      "natural": "Check the tetrahedron volume at the
                  extremal point."
    },
    {
      "step_id": 9, "type": "theorem",
      "natural": "Compute the prism lateral area after
                  recovering the common edge."
    },
    {
      "step_id": 10, "type": "theorem",
      "natural": "Add the two equilateral bases to get
                  the full surface area."
    },
    {
      "step_id": 11, "type": "calculation",
      "natural": "Evaluate the total surface area using
                  the recovered common edge."
    },
    {
      "step_id": 12, "type": "theorem",
      "natural": "Compute the full prism volume after
                  the common edge is recovered."
    },
    {
      "step_id": 13, "type": "calculation",
      "natural": "Divide the total surface area by the
                  prism volume and simplify the exact ratio."
    }
  ]
}
```

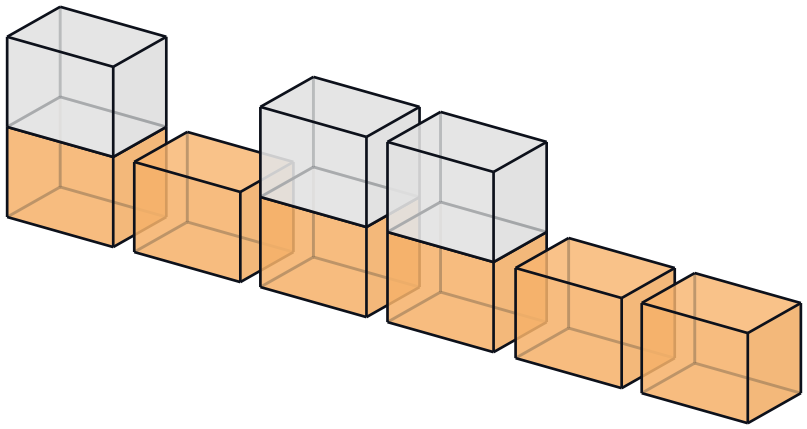

Reference answer: $\frac{2+4\sqrt{3}}{15}$

Figure 10: Representative S^{2}BENCH case for Solid-Geometry Metric Reasoning (SGMR), example 2 of 2.

Voxel Counting (VC)

Problem

Please count how many cubes are in the figure.



Structured Annotations

Structured Visual Information


```
{
  "rendering_legend": {
    "orange_cube": "a physical unit cube at ground level z=0",
    "gray_cube": "a physical upper unit cube; transparency is drawing style, not empty
                  space",
    "black_outline": "cube-face boundaries; count physical cubes rather than visible
                      faces"
  },
  "layout_family": "single_row",
  ...
}
```


Structured Reasoning Steps


```
{
  "steps": [
    { "step_id": 1, "type": "given",
      "natural": "The diagram shows a low-occlusion assembly of unit cubes."
    },
    { "step_id": 2, "type": "calculation",
      "natural": "Count each physical cube once."
    },
    { "step_id": 3, "type": "theorem",
      "natural": "The visual enumeration gives the requested cube count."
    },
      ......
    { "step_id": 5, "type": "calculation",
      "natural": "Return the total number of cubes as one integer."
    }
  ]
}
```

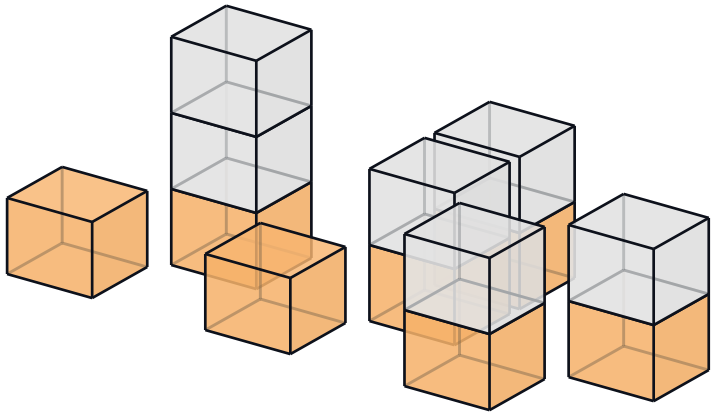

Reference answer: 9

Figure 11: Representative S^{2}BENCH case for Voxel Counting (VC), example 1 of 2.

Voxel Counting (VC)

Problem

Please count how many cubes are in the figure.



Structured Annotations

Structured Visual Information


```
{
  "rendering_legend": {
    "orange_cube": "a physical unit cube at ground level z=0",
    "gray_cube": "a physical upper unit cube; transparency is drawing style, not empty
                  space",
    "black_outline": "cube-face boundaries; count physical cubes rather than visible
                      faces"
  },
  "layout_family": "staggered_two_row",
  ...
}
```


Structured Reasoning Steps

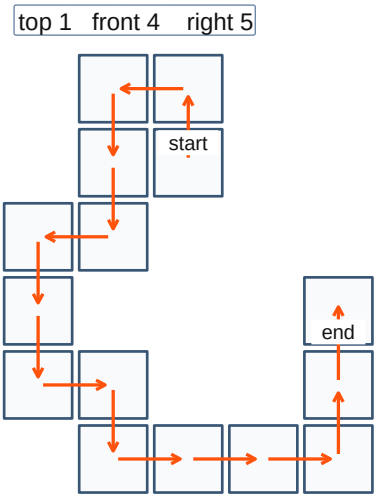

```
{
  "steps": [
    { "step_id": 1, "type": "given",
      "natural": "The diagram shows a low-occlusion assembly of unit cubes."
    },
    { "step_id": 2, "type": "calculation",
      "natural": "Count each physical cube once."
    },
    { "step_id": 3, "type": "theorem",
      "natural": "The visual enumeration gives the requested cube count."
    },
      ......
    { "step_id": 5, "type": "calculation",
      "natural": "Return the total number of cubes as one integer."
    }
  ]
}
```


Reference answer: 13

Figure 12: Representative S^{2}BENCH case for Voxel Counting (VC), example 2 of 2.

Dice-Rolling Orientation Tracking (DROT)

Problem



The figure shows a standard die rolling without slipping along the arrow path. Opposite faces sum to 7. At the start, the front face points toward the bottom of the page and the right face points toward the right side of the page.
Let T, F, and R be the numbers on the final top, front, and right faces.
Find $30T + 3F + 2R$.
Return only the integer.

Structured Annotations

Structured Visual Information


```
{
  "coordinate_frame": {
    "U": "page_up", "D": "page_down",
    "L": "page_left", "R": "page_right",
    "front_face_direction": "page_down"
  },
  "die": {
    "initial_faces_printed": {
      "top": 1, "front": 4, "right": 5
    },
    "move_sequence":
      "U L D D L D D R D R R R U U"
  },
  ...
}
```


Roll transitions


```
{
  "state": ["T", "F", "R"],
  "opposite_rule": "opp(x)=7-x",
  "R": "(T,F,R)->(7-R,F,T)",
  "L": "(T,F,R)->(R,F,7-T)",
  "U": "(T,F,R)->(F,7-T,R)",
  "D": "(T,F,R)->(7-F,T,R)",
  "page_direction_note":
    "U/D/L/R are page directions,
     not viewer-relative rotations."
}
```


Structured Reasoning Steps

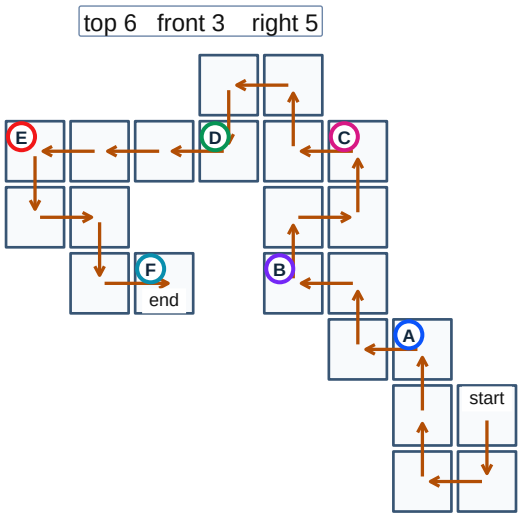

```
{
   "steps": [
      { "step_id": 1, "type": "given",
        "natural": "Initialize the die from the printed top, front, and right
                    faces and the opposite-face rule." },
      { "step_id": 2, "type": "calculation",
        "natural": "Apply one roll transition per arrow, ... in order." },
      { "step_id": 3, "type": "calculation",
        "natural": "After the last roll, read the final top, ... right faces." },
       ......
      { "step_id": 5, "type": "calculation",
        "natural": "Evaluate 30T+3F+2R." }
   ]
}
```


Reference answer: 117

Figure 13: Representative S^{2}BENCH case for Dice-Rolling Orientation Tracking (DROT), example 1 of 2.

Dice-Rolling Orientation Tracking (DROT)

Problem



The figure shows a standard die rolling without slipping along the arrow path. Opposite faces sum to 7. At the start, the front face points toward the bottom of the page and the right face points toward the right side of the page.
The path contains six marked checkpoint squares *A*-*F*. After the die reaches checkpoint *A*, let *A* be the top face.
After checkpoint *B*, let *B* be the front face.
After checkpoint *C*, let *C* be the right face.
After checkpoint *D*, let *D* be the top face.
After checkpoint *E*, let *E* be the front face.
After checkpoint *F*, let *F* be the right face.
Find $50A + 19B + 13C + 8D + 5E + 2F$.
Return only the integer.

Structured Annotations

Structured Visual Information


```
{
  "coordinate_frame": {
    "U": "page_up", "D": "page_down",
    "L": "page_left", "R": "page_right",
    "front_face_direction": "page_down"
  },
  "initial_faces_printed": {"top": 6, "front": 3, "right": 5},
  "checkpoint_segments": [
    {"to": "A", "moves": "DLUU", "requested_face": "top"},
    {"to": "B", "moves": "LUL", "requested_face": "front"},
    {"to": "C", "moves": "URU", "requested_face": "right"},
    {"to": "D", "moves": "LULD", "requested_face": "top"},
    {"to": "E", "moves": "LLL", "requested_face": "front"},
    {"to": "F", "moves": "DRDR", "requested_face": "right"}
  ],
  "step_indexing_rule":
    "Read each requested face after entering its checkpoint cell.",
  ...
}
```


Roll transitions


```
{
  "state": ["T", "F", "R"],
  "state_meaning": "T=top; F=front; R=right",
  "opposite_rule": "opp(x)=7-x",
  "R": "(T,F,R)->(7-R,F,T)",
  "L": "(T,F,R)->(R,F,7-T)",
  "U": "(T,F,R)->(F,7-T,R)",
  "D": "(T,F,R)->(7-F,T,R)",
  "page_direction_note":
    "U/D/L/R are page directions,
     not viewer-relative rotations."
}
```


Structured Reasoning Steps

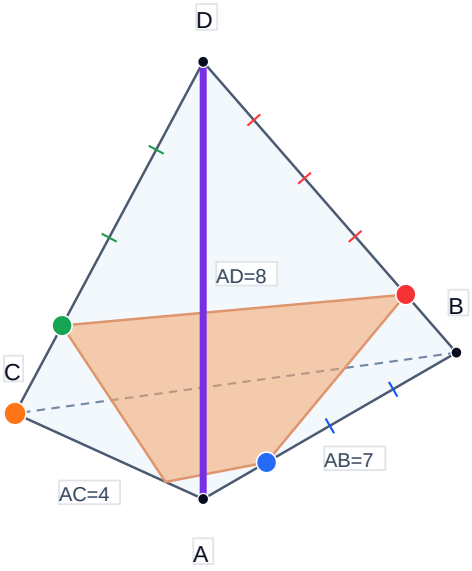

```
{
   "steps": [
      { "step_id": 1, "type": "given",
         "natural": "Initialize the die from the printed faces and the opposite-face rule." },
      { "step_id": 2, "type": "calculation",
         "natural": "Process the segments in order, rolling once per arrow without resetting the
                     orientation." },
      { "step_id": 3, "type": "calculation",
         "natural": "At each checkpoint, record the requested face after entering its cell." },
      ......
      { "step_id": 7, "type": "calculation",
         "natural": "Evaluate 50A+19B+13C+8D+5E+2F using the recorded values." }
   ]
}
```


Reference answer: 287

Figure 14: Representative S^{2}BENCH case for Dice-Rolling Orientation Tracking (DROT), example 2 of 2.

Fine-Grained Visual-Geometric Reasoning (FGVGR)

Problem



The figure shows a tri-rectangular tetrahedron *A-BCD*, where *AB*, *AC*, and *AD* are pairwise perpendicular at *A*. The edge lengths *AB*, *AC*, and *AD* are printed in the diagram.

The red, green, and blue points each lie on one tetrahedron edge, but the edge names are not written beside the colored points. For the edge carrying a colored point, small ticks of the same color divide the whole edge into equal parts, and the colored point is one of those division ticks. Read the division from the alphabetically earlier endpoint of that edge to the alphabetically later endpoint. For example, if the point is the kth same-color division tick from the earlier endpoint on an edge divided into N equal parts, then its directed ratio is $k : (N - k)$.

Let α be the plane through the red, green, and blue points, and let S be the orange section polygon where α cuts the tetrahedron. The orange dot marks one tetrahedron vertex, and the purple segment marks one tetrahedron edge.

Let q be the distance from the orange vertex to α, and let φ be the angle between the purple segment and α. Write $Area(S)^2 = \frac{a}{b}$, $q^2 = \frac{c}{d}$, and $\sin^2(\varphi) = \frac{e}{f}$, in lowest terms. Let g be the number of sides of S. Find $11a+11b+7c+11d+17e+17f+23g$. Return only the integer.

Structured Annotations

Structured Visual Information


```
{
 "dimensions_printed_in_diagram": {"AB":7,"AC":4,"AD":8},
 "marker_evidence": [
  {"color":"red","edge":"BD","read_from":"B","fraction_from_read_from":"1/5"},
  {"color":"green","edge":"CD","read_from":"C","fraction_from_read_from":"1/4"},
  {"color":"blue","edge":"AB","read_from":"A","fraction_from_read_from":"1/4"}
 ],
 "query_objects": {"query_vertex":"C","highlight_edge":"AD"},
 "tick_reading_rule":
  "Read each colored division from the alphabetically earlier endpoint ... to the alphabetically later endpoint.",
 ...
}
```


Structured Reasoning Steps

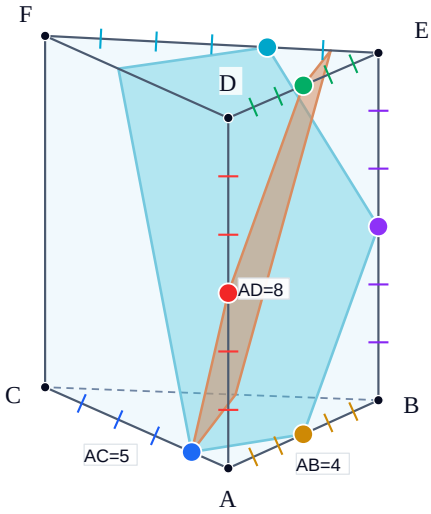

```
{
 "steps": [
  { "step_id": 1, "type": "given",
   "natural": "The diagram gives a tri-rectangular tetrahedron with colored division ticks and highlighted query objects." },
  { "step_id": 2, "type": "calculation",
   "natural": "Recover the colored points from their edge fractions and construct plane alpha." },
  { "step_id": 3, "type": "calculation",
   "natural": "Intersect alpha with the tetrahedron; compute the section area, ,..., and side count." },
  ....
  { "step_id": 6, "type": "calculation",
   "natural": "Reduce the required fractions and evaluate the expression in the question." }
 ]
}
```


Reference answer: 915484

Figure 15: Representative S^{2}BENCH case for Fine-Grained Visual-Geometric Reasoning (FGVGR), example 1 of 2.

Fine-Grained Visual-Geometric Reasoning (FGVGR)

Problem



The figure shows a right triangular prism ABC-DEF. The base triangle ABC is right-angled at A, and AD, BE, and CF are perpendicular to plane ABC. Visible prism edges are solid and hidden prism edges are dashed. The edge lengths AB, AC, and AD are printed in the diagram.

The red, green, and blue points lie on visible prism edges and determine plane α; the orange polygon is the section S_α where α cuts the prism. Then cyan, purple, and gold points lie on visible prism edges and determine plane β; the teal polygon is the section S_β where β cuts the prism.

For every colored point, the edge name is not written beside it. Small ticks of the same color divide the whole edge into equal parts, and the colored point is one of those division ticks. Read each division from the alphabetically earlier endpoint of its edge to the alphabetically later endpoint.

Write $Area(S_\alpha)^2 = \frac{a}{b}$, $Area(S_\beta)^2 = \frac{c}{d}$, and $\cos^2(\theta) = \frac{e}{f}$ in lowest terms, where θ is the angle between planes α and β. Let g and h be the numbers of sides of S_α and S_β, respectively. Each section meets exactly one vertical prism edge; let u and v be the corresponding codes for S_α and S_β, using $AD = 1$, $BE = 2$, and $CF = 3$.

Find $5a+7b+13c+23d+29e+23f+29g+37h+31u+41v$. Return only the integer.

Structured Annotations

Structured Visual Information


```
{
 "dimensions_printed_in_diagram": {"AB":4,"AC":5,"AD":8},
 "marker_evidence": [
  {"color":"red","edge":"AD","read_from":"A","fraction_from_read_from":"3/6","plane":"alpha"},
  {"color":"green","edge":"DE","read_from":"D","fraction_from_read_from":"3/6","plane":"alpha"},
  {"color":"blue","edge":"AC","read_from":"A","fraction_from_read_from":"1/5","plane":"alpha"},
  {"color":"cyan","edge":"EF","read_from":"E","fraction_from_read_from":"2/6","plane":"beta"},
  {"color":"purple","edge":"BE","read_from":"B","fraction_from_read_from":"3/6","plane":"beta"},
  {"color":"gold","edge":"AB","read_from":"A","fraction_from_read_from":"3/6","plane":"beta"}
 ],
 "tick_reading_rule":
  "Read each edge from its alphabetically earlier endpoint to its later endpoint.",
 ...
}
```


Structured Reasoning Steps

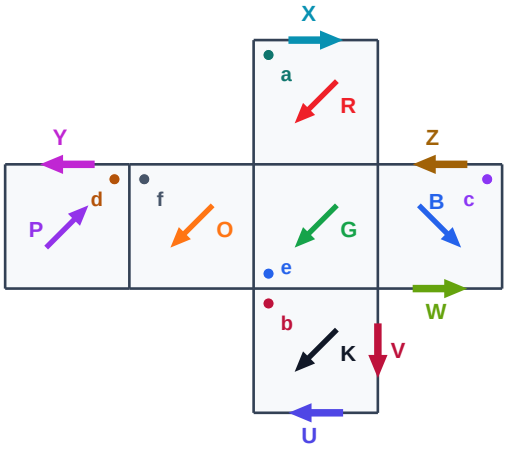

```
{
 "steps": [
  { "step_id": 1, "type": "given",
    "natural": "The diagram gives a right triangular prism with two colored triples of section points." },
  { "step_id": 2, "type": "calculation",
    "natural": "Recover all six edge points; construct alpha from red, green, blue and beta from ... gold." },
  { "step_id": 3, "type": "calculation",
    "natural": "Intersect each plane with the prism; compute both section areas, plane angle, side counts, and
               vertical-edge codes." },
  .....
  { "step_id": 8, "type": "calculation",
    "natural": "Reduce the required fractions and evaluate the expression in the question." }
 ]
}
```


Reference answer: 63399

Figure 16: Representative S^{2}BENCH case for Fine-Grained Visual-Geometric Reasoning (FGVGR), example 2 of 2.

Net Folding and Marker Correspondence (NFMC)

Problem



The figure is a planar net of a cube. Diagonal colored arrows lie on faces; directed colored arrows X, Y, Z, W, U, V lie along boundary face edges; and small lowercase corner markers lie at face corners. All markers fold with their faces.

For two diagonal face arrows m, n, Dir(m, n) = 1 if their folded arrows point in the same 3D direction, 2 if opposite, 3 if perpendicular, 4 if they make an acute 60-degree angle, and 5 if they make an obtuse 120-degree angle.

For two directed edge arrows s, t, EdgeDir(s, t) = 1 if they fold onto the same cube edge and point in the same direction, 2 if they fold onto the same cube edge and point in opposite directions, 3 if they fold onto distinct parallel cube edges and point in the same 3D direction, 4 if they fold onto distinct parallel cube edges and point in opposite 3D directions, and 5 otherwise.

For two lowercase corner markers p, q, Vertex(p, q) = 1 if they fold to the same cube vertex, 2 if to adjacent cube vertices, 3 if to opposite cube vertices, and 4 otherwise.

Let A = Dir(purple, orange); B = Dir(orange, black); C = Dir(red, black); D = EdgeDir(Z, V); E = EdgeDir(X, W); F = EdgeDir(X, Y); G = Vertex(a, d); H = Vertex(a, b); I = Vertex(b, c); J be the number of unordered pairs of face arrows with Dir = 4; K be the number of unordered pairs of directed edge arrows with EdgeDir equal to 1 or 2; L be the number of unordered pairs of corner markers that fold to the same cube vertex.

Find $17A + 19B + 23C + 29D + 31E + 37F + 41G + 43H + 47I + 53J + 59K + 61L$.
Return only the integer.

Structured Annotations

Structured Visual Information


```
{
 "coordinate_frame": {"E": "page_right",
  "W": "page_left", "N": "page_up", "S": "page_down"},
 "net": {
  "cells": [
   [−1, 0], [0, 0], [1, −1], [1, 0], [1, 1],
   [2, 0]
  ]
 },
 "face_arrows": {
  "purple": {"cell": [−1, 0], "drawn_dir": "NE"},
  "orange": {"cell": [0, 0], "drawn_dir": "SW"}
 },
 "directed_edge_arrows": {
  "Z": {"cell": [2, 0],
        "boundary_side": "N", "along": "W"},
  "V": {"cell": [1, −1],
        "boundary_side": "E", "along": "S"}
 },
 "corner_markers": {
  "a": {"cell": [1, 1], "local_corner": "NW"},
  "d": {"cell": [−1, 0], "local_corner": "NE"}
 },
 "folding_guidance": {
  "folding_rule": "Apply a quarter turn across
    every shared panel edge."
 }
}
```


Structured Reasoning Steps

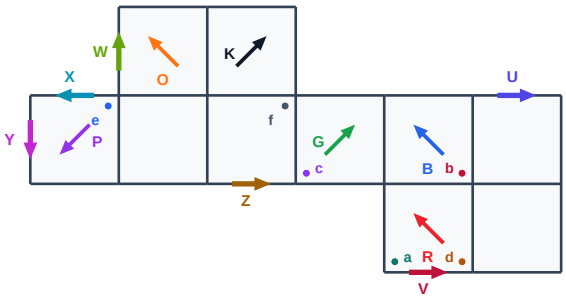

```
{
 "steps": [
  {"step_id": 1, "type": "given",
   "natural": "Identify the cube net and its face arrows,
     edge arrows, and corner markers."},
  {"step_id": 2, "type": "calculation",
   "natural": "Fold adjacent panels by quarter turns and
     transport each diagonal arrow with its panel."},
  {"step_id": 3, "type": "calculation",
   "natural": "Transport directed edge arrows, retaining
     their edge endpoints and directions."},
  {"step_id": 4, "type": "calculation",
   "natural": "Fold corner markers and determine their 3D
     vertex positions."},
  {"step_id": 5, "type": "calculation",
   "natural": "Classify the requested pairs; use all markers
     and count every unordered pair once for J, K, L."},
  {"step_id": 6, "type": "calculation",
   "natural": "Evaluate the weighted expression using the
     computed relations and counts."}
 ]
}
```


Reference answer: 1678

Figure 17: Representative S^{2}BENCH case for Net Folding and Marker Correspondence (NFMC), example 1 of 2.

Net Folding and Marker Correspondence (NFMC)

Problem



The figure is a unit-square surface net of a 2 by 1 by 1 rectangular box made from two unit cubes. Diagonal colored arrows lie on six square panels; directed colored arrows X, Y, Z, W, U, V lie along boundary panel edges; and small lowercase corner markers lie at panel corners. All markers fold with their panels.

For two diagonal face arrows m, n, Dir(m, n) = 1 if their folded arrows point in the same 3D direction, 2 if opposite, 3 if perpendicular, 4 if they make an acute 60-degree angle, and 5 if they make an obtuse 120-degree angle.

For two directed edge arrows s, t, EdgeDir(s, t) = 1 if they fold onto the same surface-grid edge and point in the same direction, 2 if they fold onto the same surface-grid edge and point in opposite directions, 3 if they fold onto distinct parallel surface-grid edges and point in the same 3D direction, 4 if they fold onto distinct parallel surface-grid edges and point in opposite 3D directions, and 5 otherwise.

For two lowercase corner markers p, q, Vertex(p, q) = 1 if they fold to the same surface-grid vertex, 2 if the two vertices are joined by one unit surface-grid edge, 3 if they lie two unit steps apart on the same straight long edge of the box, and 4 otherwise.

Let A = Dir(blue, purple); B = Dir(red, black); C = Dir(green, black); D = EdgeDir(U, V); E = EdgeDir(X, Y); F = EdgeDir(Z, V); G = Vertex(b, c); H = Vertex(e, f); I = Vertex(a, c); J be the number of unordered pairs of face arrows with Dir = 4; K be the number of unordered pairs of directed edge arrows with EdgeDir equal to 1 or 2; L be the number of unordered pairs of corner markers that fold to the same surface-grid vertex.

Find $17A + 19B + 23C + 29D + 31E + 37F + 41G + 43H + 47I + 53J + 59K + 61L$.
Return only the integer.

Structured Annotations

Structured Visual Information


```
{
 "coordinate_frame": {"E": "page_right",
  "W": "page_left", "N": "page_up", "S": "page_down"},
 "net": {
  "box_dimensions": [2, 1, 1],
  "cells": [
   [0, 1], [1, 1], [1, 2], [2, 1], [2, 2],
   [3, 1], [4, 0], [4, 1], [5, 0], [5, 1]
  ]
 },
 "face_arrows": {
  "blue": {"cell": [4, 1], "drawn_dir": "NW"},
  "purple": {"cell": [0, 1], "drawn_dir": "SW"}
 },
 "directed_edge_arrows": {
  "U": {"cell": [5, 1],
        "boundary_side": "N", "along": "E"},
  "V": {"cell": [4, 0],
        "boundary_side": "S", "along": "E"}
 },
 "corner_markers": {
  "b": {"cell": [4, 1], "local_corner": "SE"},
  "c": {"cell": [3, 1], "local_corner": "SW"}
 },
 "folding_guidance": {
  "flat_internal_seams": [
   [[1, 1], [2, 1]], [[1, 2], [2, 2]],
   [[4, 0], [5, 0]], [[4, 1], [5, 1]]],
  "folding_rule": "Keep flat seams coplanar; apply
   quarter turns at the remaining shared edges."}
}
```


Structured Reasoning Steps

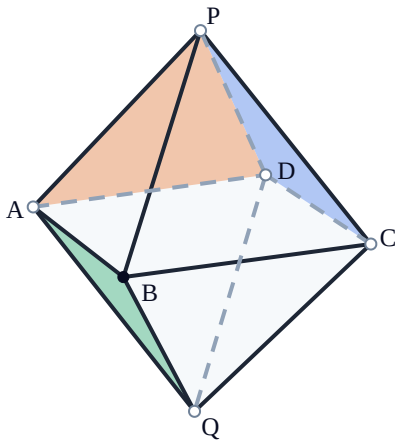

```
{
 "steps": [
  {"step_id": 1, "type": "given",
   "natural": "Identify the 2 by 1 by 1 box net and its
    face arrows, edge arrows, and corner markers."},
  {"step_id": 2, "type": "calculation",
   "natural": "Keep flat seams coplanar, fold the other
    joints by quarter turns, and transport each face arrow."},
  {"step_id": 3, "type": "calculation",
   "natural": "Transport directed edge arrows, retaining
    their edge endpoints and directions."},
  {"step_id": 4, "type": "calculation",
   "natural": "Fold corner markers and determine their
    3D vertex positions."},
  {"step_id": 5, "type": "calculation",
   "natural": "Classify the queries using all markers.
    Count each unordered pair once for J, K and L."},
  ......
  {"step_id": 7, "type": "calculation",
   "natural": "Evaluate the weighted expression using the
    computed relations and counts."}
 ]
}
```


Reference answer: 1508

Figure 18: Representative S^{2}BENCH case for Net Folding and Marker Correspondence (NFMC), example 2 of 2.

Polyhedral Topology and Visibility Reasoning (PTVR)

Problem



The figure shows a convex octahedron. Solid segments are visible edges and dashed segments are hidden edges. Three faces are shaded blue, green, and orange. A face path moves from one face to an adjacent face across their shared edge, and it may not repeat a face.

Consider only face paths from the blue face to the orange face that pass through the green face. Let L be the minimum possible number of face-to-face moves among such paths. Among all such paths of length L, let H be the largest number of dashed shared edges crossed, and let N be the number of length-L paths that cross exactly H dashed shared edges.

Find $37L + 19H + 9N$.

Return only the integer.

Structured Annotations

Structured Visual Information


```
{
  "shape": "octahedron",
  "line_style_legend": {
    "solid": "visible edge",
    "dashed": "hidden edge; still usable"
  },
  "original_face_boundaries": {
    "F1": "P-A-B", "F2": "P-B-C", "F3": "P-C-D", "F4": "P-D-A",
    "F5": "Q-A-B", "F6": "Q-B-C", "F7": "Q-C-D", "F8": "Q-D-A"
  },
  "original_edge_style_ledger": {
    "solid": ["A-B", "A-P", "A-Q", "B-C", "B-P", "B-Q", "C-P", "C-Q"],
    "dashed": ["A-D", "C-D", "D-P", "D-Q"]
  },
  "colored_marker_bindings": {
    "blue_start_face": "F3",
    "required_green_face": "F5",
    "orange_end_face": "F4"
  },
  ...
}
```


Structured Reasoning Steps

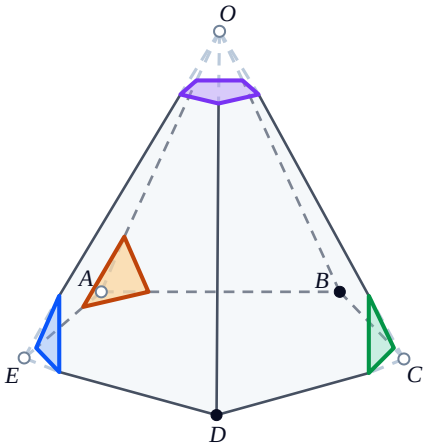

```
{
  "steps": [
    {"step_id": 1, "type": "given",
     "natural": "Read the colored faces, edge styles, and face boundaries from the diagram."},
    {"step_id": 2, "type": "calculation",
     "natural": "Join faces sharing an edge. Enumerate blue-to-orange paths through green without repeating a face;
                 find the minimum length L."},
    {"step_id": 3, "type": "calculation",
     "natural": "Among length-L paths, maximize dashed shared-edge crossings to obtain H; count the paths attaining H
                 to obtain N."},
    {"step_id": 4, "type": "calculation",
     "natural": "Evaluate 37L + 19H + 9N and return the integer."}
  ]
}
```


Reference answer: 241

Figure 19: Representative S^{2}BENCH case for Polyhedral Topology and Visibility Reasoning (PTVR), example 1 of 2.

Polyhedral Topology and Visibility Reasoning (PTVR)

Problem



The figure shows a convex polyhedron after several small vertex truncations. Each colored face is one new cut face; pale dashed stubs show removed original corner references only and are not final faces. Consider the final truncated solid.

A face path moves from one final face to an adjacent final face across their shared final edge, and it may not repeat a face. Use the blue, orange, and green colored cut faces shown in the diagram. Among face paths from the blue cut face to the orange cut face that pass through the green cut face, let L be the minimum possible number of face-to-face moves. Among all such paths of length L, let H be the largest number of dashed remaining body edges crossed, and let N be the number of length-L paths that cross exactly H dashed remaining body edges.

Let B be the largest number of remaining body edges crossed among all such paths of length L. Call a remaining body edge a connector if both of its endpoints lie on colored cut faces; let C be the number of connector edges and D be the number of dashed connector edges.

Find $L(17H + N) + 19C + 23D + 5B$.

Return only the integer.

Structured Annotations

Structured Visual Information


```
{
 "shape": "pentagonal pyramid",
 "line_style_legend": {
  "solid": "visible remaining body edge",
  "dashed": "hidden remaining body edge",
  "pale_dashed_stub": "removed corner reference; not a final edge"
 },
 "original_edge_style_ledger": {
  "solid": ["C-D", "C-O", "D-E", "D-O", "E-O"],
  "dashed": ["A-B", "A-E", "A-O", "B-C", "B-O"]
 },
 "colored_cut_face_bindings": [
  {"color": "blue", "original_cut_vertex": "E", "cut_face_node": "K_E"},
  {"color": "orange", "original_cut_vertex": "A", "cut_face_node": "K_A"},
  {"color": "purple", "original_cut_vertex": "O", "cut_face_node": "K_O"},
  {"color": "green", "original_cut_vertex": "C", "cut_face_node": "K_C"}
 ],
 ...
}
```


Structured Reasoning Steps

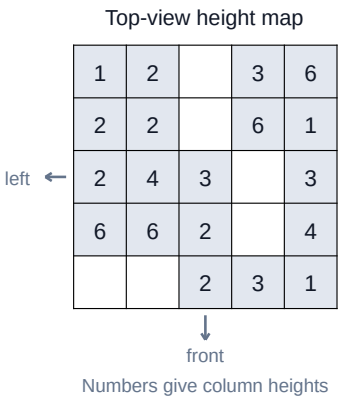

```
{
 "steps": [
  {"step_id": 1, "type": "given",
   "natural": "The diagram shows a polyhedron with small, non-overlapping vertex cuts."},
  {"step_id": 2, "type": "calculation",
   "natural": "Identify the source solid and all colored cut faces; use blue as the start, green as the checkpoint,
              and orange as the end."},
  {"step_id": 3, "type": "calculation",
   "natural": "A remaining body edge is a connector when both original endpoints were cut. Count all connectors as C
              and dashed connectors as D."},
  {"step_id": 4, "type": "calculation",
   "natural": "Build the final face-adjacency graph, excluding pale stubs. Find simple checkpoint paths of minimum
              length L. Among them, maximize dashed body-edge crossings for H and count those attaining H as N.
              Independently maximize body-edge crossings over all length-L paths for B."},
  {"step_id": 5, "type": "calculation",
   "natural": "Evaluate L(17H + N) + 19C + 23D + 5B and return the integer."}
 ]
}
```


Reference answer: 217

Figure 20: Representative S^{2}BENCH case for Polyhedral Topology and Visibility Reasoning (PTVR), example 2 of 2.

Projection and Shadow Reasoning (PSR)

Problem



A unit-cube stack has no gaps inside any vertical column. The figure is its top-view height map, where each number is the column height. The red and blue arrows show two horizontal shadow-shift directions for parallel light on the ground grid. For one light, a cube in layer 1 projects to its own footprint cell, a cube in layer 2 projects one cell in the arrow direction, and in general a cube in layer k projects $k-1$ cells in that arrow direction. Ignore all cells in the original footprint. Place 1 marker on each remaining ground cell covered only by the red shadow, 2 markers on each covered only by the blue shadow, and 5 markers on each covered by both shadows. How many markers are placed? Return only the integer.

Structured Annotations

Structured Visual Information


```
{
  "coordinate_convention": {
    "height_map_entry": "height_map[y][x]",
    "row_order": "front_to_back (bottom_to_top in the image)",
    "x_axis": "+x points right",
    "y_axis": "+y points up in the image",
    "layer_numbering": "layer 1 touches the ground"
  },
  "height_map": [
    [0, 0, 2, 3, 1],
    [6, 6, 2, 0, 4],
    [2, 4, 3, 0, 3],
    [2, 2, 0, 6, 1],
    [1, 2, 0, 3, 6]
  ],
  "colored_arrow_vectors": {
    "red":  {"dx": 1, "dy": 1},
    "blue": {"dx": −1, "dy": 1}
  }
}
```


Structured Reasoning Steps

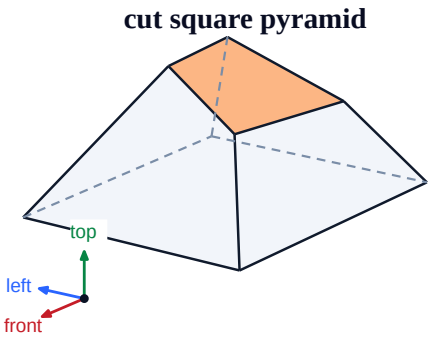

```
{
  "steps": [
    {"step_id": 1, "type": "given",
     "natural": "The figure gives a no-gap unit-cube stack as a top-view height map."},
    {"step_id": 2, "type": "calculation",
     "natural": "Treat each nonzero entry h as a full column with layers k=1,...,h."},
    {"step_id": 3, "type": "calculation",
     "natural": "For each color, project each layer to (x+(k-1)dx, y+(k-1)dy). Take the union of all
       projected cells."},
    {"step_id": 4, "type": "calculation",
     "natural": "Remove every original-footprint cell from both unions, then partition into red-only,
       blue-only, and overlap cells."},
    {"step_id": 5, "type": "calculation",
     "natural": "Apply weights 1, 2, and 5 to the three category counts and sum."}
  ]}
```


Reference answer: 77

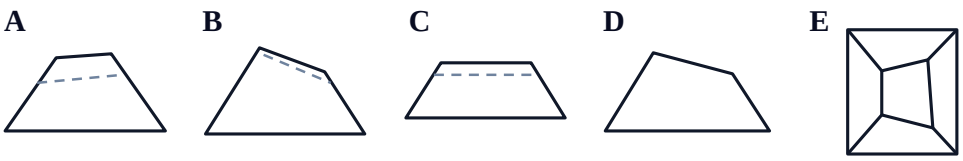
Figure 21: Representative S^{2}BENCH case for Projection and Shadow Reasoning (PSR), example 1 of 3.

Projection and Shadow Reasoning (PSR)

Problem



The figure shows a pyramid cut by one oblique plane. The red, blue, and green arrows label the front, left, and top sides of the solid; each named orthographic view is taken from that labeled side toward the solid. The five diagrams A-E are candidate orthographic views of the cut solid. In the candidate views, solid segments are visible edges and dashed segments are hidden edges. Which option is the front view? Return the option index with A=1, B=2, C=3, D=4, and E=5.



Structured Annotations

Structured Visual Information


```
{
  "coordinate_axes": {
    "right": "+x", "front_side": "+y", "up": "+z"
  },
  "direction_arrow_binding": {
    "front_red": "camera is on the +y side and looks toward -y",
    "left_blue": "camera is on the -x side and looks toward +x",
    "top_green": "camera is on the +z side and looks toward -z"
  },
  "line_style_convention": {
    "visible": "solid",
    "hidden": "dashed"},
  ...
}
```


Structured Reasoning Steps

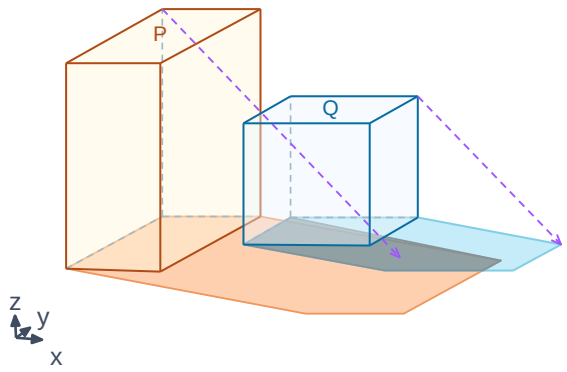

```
{
  "steps": [
    {"step_id": 1, "type": "given",
      "natural": "The diagram gives a cut polyhedron, three viewing directions, and five ... views."},
    {"step_id": 2, "type": "calculation",
      "natural": "Use the requested arrow-defined direction. Project every real edge, including the cut
                  boundary; resolve overlapping segments with solid-line precedence."},
    {"step_id": 3, "type": "calculation",
      "natural": "Compare the silhouette, junctions, and solid/dashed edge styles with all five options."},
    {"step_id": 4, "type": "calculation",
      "natural": "Return the matching option index using A=1, B=2, C=3, D=4, E=5."}
  ]
}
```


Reference answer: 2

Figure 22: Representative S^{2}BENCH case for Projection and Shadow Reasoning (PSR), example 2 of 3.

Projection and Shadow Reasoning (PSR)

Problem



The figure shows two opaque rectangular blocks P and Q standing on the ground plane $z = 0$. Block P occupies $0 \le x \le 6$, $0 \le y \le 10$, $0 \le z \le 12$. Block Q occupies $8 \le x \le 16$, $5 \le y \le 10$, $0 \le z \le 7$. All light rays have common direction $(5, -2, -3)$.

The amber and blue regions are the individual shadows on the ground, and the darker region is their overlap. Let U be the area of the union of the two shadows, O be the area of their overlap, and k be the number of sides of the overlap polygon. Write $U = a/b$ and $O = c/d$ in lowest terms. Find $31a + 19b + 37c + 31d + 53k$.

Return only the integer.

Structured Annotations

Structured Visual Information


```
{
 "coordinate_convention": {
  "ground_plane": "z=0", "up": "+z",
  "ray_parameterization": "point + t * light_direction for t >= 0"
 },
 "blocks": [
  {"block": "P", "axis_aligned_box":
   {"x0": 0, "y0": 0, "width": 6, "depth": 10, "height": 12}},
  {"block": "Q", "axis_aligned_box":
   {"x0": 8, "y0": 5, "width": 8, "depth": 5, "height": 7}}
 ],
 "parallel_light_direction": {"x": 5, "y": −2, "z": −3},
 "diagram_region_binding": {
  "amber": "individual ground shadow cast by block P",
  "blue": "individual ground shadow cast by block Q",
  "darker_overlap": "ground points covered by both individual shadows"},
 ...
}
```


Structured Reasoning Steps


```
{
 "steps": [
  {"step_id": 1, "type": "given",
   "natural": "The diagram gives two opaque blocks and their overlapping parallel-light shadows."},
  {"step_id": 2, "type": "calculation",
   "natural": "Project each top vertex to z=0 along the light direction using t=-z/dz. Each shadow
        is the convex hull of the footprint and projected top footprint."},
  {"step_id": 3, "type": "calculation",
   "natural": "Intersect the shadow polygons. Compute O by the shoelace formula and U by inclusion-
        exclusion. Remove duplicate and collinear boundary vertices before counting k."},
  {"step_id": 4, "type": "calculation",
   "natural": "Reduce U=a/b and O=c/d to lowest terms. Evaluate 31a+19b+37c+31d+53k and return the
        integer."}
 ]
}
```

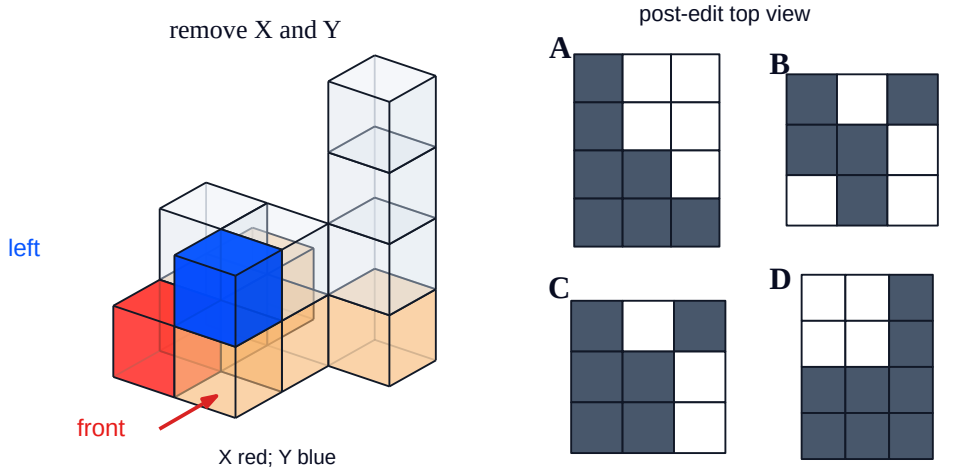

Reference answer: 639676

Figure 23: Representative S^{2}BENCH case for Projection and Shadow Reasoning (PSR), example 3 of 3.

Orthographic Multi-View Reasoning (OMVR)

Problem

The figure shows the original solid made of unit cubes and four candidate orthographic views. The front and left directions are fixed in the figure. Remove both marked cubes X and Y at the same time. Which option is the top view after the edit? Return only the option letter A, B, C, or D.



Structured Annotations

Structured Visual Information


```
{
  "coordinate_axes": {"right": "+x", "behind": "+y", "up": "+z",
                      "front_row": "y=0", "bottom_layer": "z=0"},

  "scene_state_shown_in_solid_panel":
    {"semantic_height_grid": [[1,2,0],[2,2,0],[1,0,4]]},
  "height_grid_semantics":
    "Before edit: height_grid[y][x]; rows front-to-back, columns left-to-right.",
  "image_bound_edit_markers": {"operation":"remove_two_cubes","marked_cubes":[[0,0,0],[1,0,1]]},
  "edit_marker_legend": {"X":"red cube to remove","Y":"blue cube to remove"},
  ...
}
```


Structured Reasoning Steps


```
{
  "steps": [
    {"step_id": 1, "type": "given",
      "natural": "Read the original unit-cube assembly,fixed directions, and marked edit."},
    {"step_id": 2, "type": "calculation",
      "natural": "Remove both marked cubes simultaneously. Project the remaining ...from
                  above; display rows back-to-front, with y=0 at the bottom."},
    {"step_id": 3, "type": "calculation",
      "natural": "Compare the derived top view with every option, checking ... orientation,
                  and each occupied cell."},
    {"step_id": 4, "type": "calculation",
      "natural": "Return only the matching option letter A, B, C, or D."}
  ]
}
```

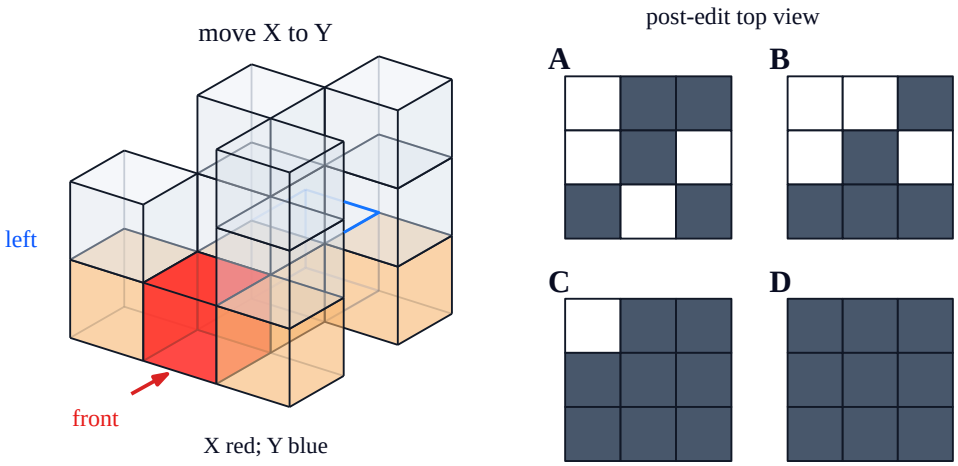

Reference answer: B

Figure 24: Representative S^{2}BENCH case for Orthographic Multi-View Reasoning (OMVR), example 1 of 2.

Orthographic Multi-View Reasoning (OMVR)

Problem

The figure shows the original solid made of unit cubes and four candidate orthographic views. The front and left directions are fixed in the figure. Move cube X to the blue outlined position Y. Which option is the top view after the edit? Return only the option letter A, B, C, or D.



Structured Annotations

Structured Visual Information


```
{
 "coordinate_axes": {"right": "+x", "behind": "+y", "up": "+z",
   "front_row": "y=0", "bottom_layer": "z=0"},
 "scene_state_shown_in_solid_panel":
   {"semantic_height_grid": [[2,1,3],[0,3,0],[0,0,3]]},
 "height_grid_semantics":
   "Before edit: height_grid[y][x]; rows front-to-back, columns left-to-right.",
 "image_bound_edit_markers": {"operation":"move_one_cube","source_cube":[1,0,0],
   "destination_position":[1,2,0]},
 "edit_marker_legend": {"X":"red source cube","Y":"blue outlined empty destination"},
 ...
}
```


Structured Reasoning Steps

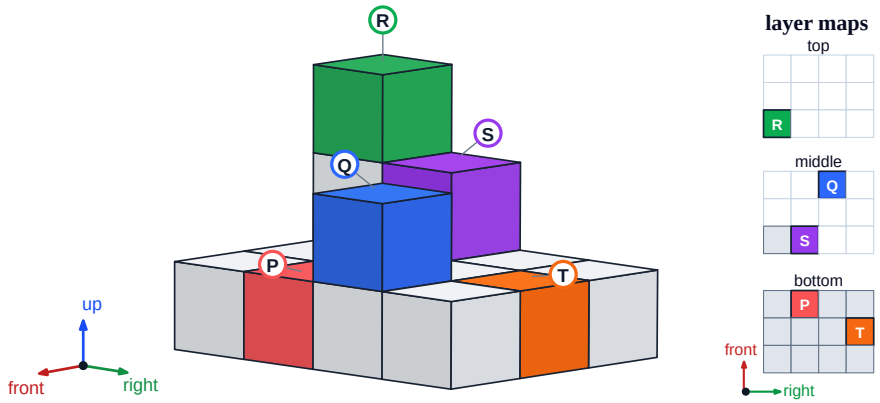

```
{
 "steps": [
   {"step_id": 1, "type": "given",
     "natural": "Read the original unit-cube assembly, fixed directions, and marked edit."},
   {"step_id": 2, "type": "calculation",
     "natural": "Remove cube X from its source and place it at Y. Project ... occupied
           columns from above; display rows back-to-front, with y=0 at the bottom."},
   {"step_id": 3, "type": "calculation",
     "natural": "Compare the derived top view with every option, checking matrix dimensions,
           orientation, and each occupied cell."},
   ......
   {"step_id": 5, "type": "calculation",
     "natural": "Return only the matching option letter A, B, C, or D."}
 ]
}
```


Reference answer: A

Figure 25: Representative S^{2}BENCH case for Orthographic Multi-View Reasoning (OMVR), example 2 of 2.

Voxel Spatial-Relation Reasoning (VSRR)

Problem



The figure shows a solid made of face-to-face unit cubes. The left drawing shows the 3D solid, and the layer maps on the right show exactly which grid cells are occupied in the bottom, middle, and top layers. In every layer map, right points to the right and front points upward; the layers are stacked upward from bottom to top. Five surface cubes are marked *P*, *Q*, *R*, *S*, and *T*, and the same letters mark their cells in the layer maps. No numeric coordinates are given.

For marked cubes *X* and *Y*, *Rel(X,Y)* is used only when *Y* is strictly to one side, strictly front or behind, and strictly above or below *X*. Use the code 1=right-front-above, 2=right-front-below, 3=right-behind-above, 4=right-behind-below, 5=left-front-above, 6=left-front-below, 7=left-behind-above, and 8=left-behind-below. Let $A = Rel(R,T)$, $B = Rel(S,T)$, $C = Rel(R,Q)$, $D = Rel(P,R)$, $E = Rel(Q,T)$, $F = Rel(T,Q)$.

A face-step path moves from one cube to a cube sharing a square face and stays inside the solid. Let *M* be the number among all ten unordered pairs of marked cubes that share a square face. Let *N* be the number among all ten unordered pairs of marked cubes whose shortest face-step path has exactly three moves. Let *O* be the largest shortest face-step distance among all unordered pairs of marked cubes.

Let *V* be the number among all twenty ordered pairs of distinct marked cubes for which the second cube is both in front of and above the first cube. Let *W* be the number among all twenty ordered pairs of distinct marked cubes for which the second cube is both to the right of and below the first cube.

For *K*, project the marked cubes onto the layer maps and ignore height: *K* is the number among all ten unordered pairs whose projected cells are in the same front-back row or the same right-left column.

Find $23A + 23B + 29C + 37D + 29E + 43F + 53M + 67N + 61O + 67V + 83W + 79K$. Return only the integer.

Structured Annotations

Structured Visual Information


```
{
 "coordinate_frame": {"right": "page right in layer maps",
   "front": "page up in layer maps", "up": "bottom -> middle -> top"},
 "layer_map_transcription": {
   "cell_legend": "0=empty; 1=occupied; letters=marked occupied cells",
   "row_order": "page top to bottom; front decreases",
   "column_order": "page left to right; right increases",
   "bottom": ["1P11", "111T", "1111"],
   "middle": ["00Q0", "0000", "1S00"],
   "top": ["0000", "0000", "R000"]},
   ...
}
```


Structured Reasoning Steps

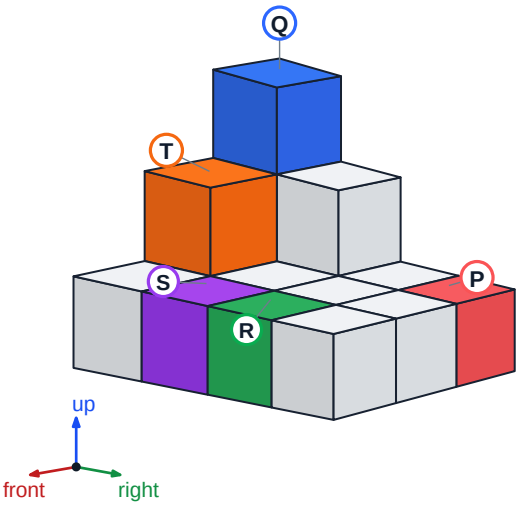

```
{
 "steps": [
   {"step_id": 1, "type": "given",
    "natural": "The diagram gives a unit-cube solid, its layer maps, and five marked cubes."},
   {"step_id": 2, "type": "calculation",
    "natural": "Read every occupied cell and marker from the layers. Fix right/front/up before ... cube coordinates."},
   {"step_id": 3, "type": "calculation",
    "natural": "For Rel, subtract source from target and map the signs to codes. Use breadth-first search on the occupied-cube
               face-neighbor graph for M, N, O. Count V, W over ordered pairs and K over unordered projected pairs."},
   ......
   {"step_id": 6, "type": "calculation",
    "natural": "Evaluate the weighted expression in the question after computing all requested variables."}
 ]
}
```


Reference answer: 2170

Figure 26: Representative S^{2}BENCH case for Voxel Spatial-Relation Reasoning (VSRR), example 1 of 2.

Voxel Spatial-Relation Reasoning (VSRR)

Problem



The figure shows a solid made of face-to-face unit cubes. Five surface cubes are marked *P*, *Q*, *R*, *S*, and *T*. The small arrows show the fixed front, right, and up directions; no coordinates are given.

For marked cubes *X* and *Y*, Rel(*X*, *Y*) is used only when *Y* is strictly to one side, strictly front or behind, and strictly above or below *X*. Use the code
1 = right-front-above, 2 = right-front-below,
3 = right-behind-above, 4 = right-behind-below,
5 = left-front-above, 6 = left-front-below,
7 = left-behind-above, and 8 = left-behind-below.

Face(*X*, *Y*) is the direction from *X* to *Y* if the two marked cubes share a square face, using 1 = right, 2 = left, 3 = front, 4 = behind, 5 = above, and 6 = below; use 0 if they do not share a square face.

Dist(*X*, *Y*) is the minimum number of face-step moves from *X* to *Y* while staying inside the solid.

Let *A* = Rel(*Q*, *S*), *B* = Rel(*T*, *P*); let *C* = Face(*S*, *R*), *D* = Face(*T*, *R*); and let *E* = Dist(*P*, *S*).

Find *23A + 19B + 37C + 43D + 47E*.

Return only the integer.

Structured Annotations

Structured Visual Information


```
{
  "coordinate_source": "teacher-only scene reconstruction",
  "coordinate_order": ["right", "front", "up"],
  "semantic_cube_coordinates": [
    [0,0,0], [1,0,0], [2,0,0], [3,0,0],
    [0,1,0], [1,1,0], [2,1,0], [3,1,0],
    [0,2,0], [1,2,0], [2,2,0], [3,2,0],
    [0,0,1], [1,0,1], [0,1,1], [0,0,2]],
  "semantic_marked_cube_coordinates": {
    "P": [3,0,0], "Q": [0,0,2], "R": [2,2,0],
    "S": [1,2,0], "T": [0,1,1]},
  ...
}
```


Structured Reasoning Steps

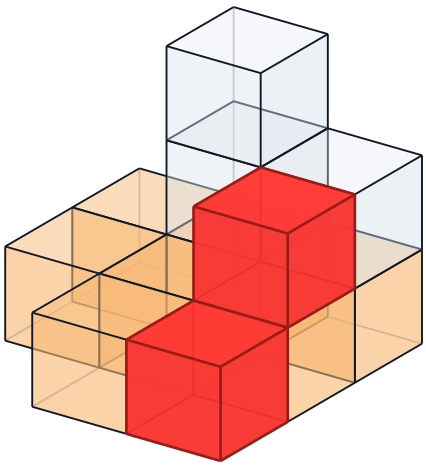

```
{
  "steps": [
    {"step_id": 1, "type": "given",
      "natural": "The diagram gives a unit-cube solid with ... and fixed direction arrows."},
    {"step_id": 2, "type": "calculation",
      "natural": "Bind each marker to its right/front/up position. Include every occupied cube
                  when constructing the face-neighbor graph."},
    {"step_id": 3, "type": "calculation",
      "natural": "Use target-minus-source signs for Rel. For Face, match an exact unit-axis
                  difference; otherwise use 0. Use ... search inside the solid for Dist."},
    ......
    {"step_id": 6, "type": "calculation",
      "natural": "Evaluate 23A+19B+37C+43D+47E and return the integer."}
  ]
}
```


Reference answer: 347

Figure 27: Representative S^{2}BENCH case for Voxel Spatial-Relation Reasoning (VSRR), example 2 of 2.

Voxel State-Transformation Reasoning (VSTR)

Problem



The figure shows the original solid, assembled from face-to-face unit cubes. The gray cubes are drawn transparent so that all red marked cubes can be seen. It contains 12 unit cubes, of which 2 red cubes are marked for removal.

First paint every exterior unit-square face of the original solid. Then remove all red cubes; newly exposed cut faces are not painted.

Let A be the number of remaining cubes with exactly 2 painted faces from the original painting and exactly 4 exposed faces after the removal. Let N be the number of newly exposed unpainted cut faces, G be the number of shared square faces between pairs of remaining cubes, and R be the number of red cubes removed.

Find $23A + 7N + 7G + 17R$.

Return only the integer.

Structured Annotations

Structured Visual Information


```
{
  "coordinate_source": "teacher-only diagram transcription",
  "coordinate_order": ["right", "front", "up"],
  "semantic_original_cube_coordinates": [
    [1,0,0], [2,0,0], [0,1,0], [1,1,0],
    [2,1,0], [0,2,0], [1,2,0], [2,2,0],
    [2,1,1], [1,2,1], [2,2,1], [1,2,2]],
  "semantic_red_cube_coordinates": [[2,0,0], [2,1,1]],
  "color_legend": {"red": "remove after original exterior painting",
    "gray_or_amber": "retained cube; transparency is a drawing style"},
  ...
}
```


Structured Reasoning Steps


```
{
  "steps": [
    {"step_id": 1, "type": "given",
      "natural": "Read the original cube assembly and the red cubes marked for removal."},
    {"step_id": 2, "type": "calculation",
      "natural": "Record the original exterior faces before changing the solid. Paint only these faces,
        then remove all red cubes without repainting."},
    {"step_id": 3, "type": "calculation",
      "natural": "For each remaining cube, painted faces equal 6 minus its original face-neighbor degree;
        final exposed faces equal 6 minus its remaining face-neighbor degree."},
    {"step_id": 4, "type": "calculation",
      "natural": "Count A using both required face counts. Count red-to-remaining contacts for N and
        unordered remaining-to-remaining contacts for G. Count removed cubes for R."},
    {"step_id": 5, "type": "calculation",
      "natural": "Evaluate 23A+7N+7G+17R and return the integer."}
  ]
}
```

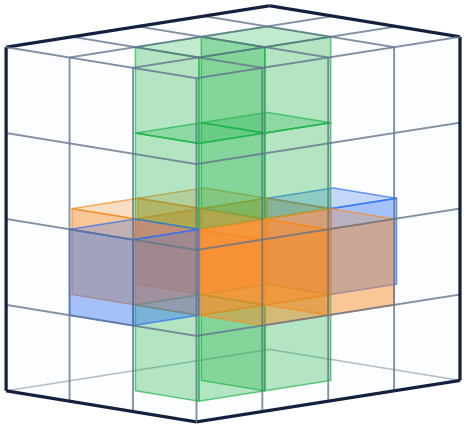

Reference answer: 169

Figure 28: Representative S^{2}BENCH case for Voxel State-Transformation Reasoning (VSTR), example 1 of 2.

Voxel State-Transformation Reasoning (VSTR)

Problem

3 x 4 x 4 unit block; colored cells are drilled out



orange: left-right | blue: front-back | green: vertical

The figure shows a transparent 3 by 4 by 4 rectangular block made from unit cubes. Colored one-cell-wide tunnels are drilled straight through the block: orange tunnels go left-right, blue tunnels go front-back, and green tunnels go vertically from top to bottom.

If tunnels meet, the common unit cube is removed only once.

After all drilling, let S be the total number of exposed unit-square faces of the remaining solid, W be the number of exposed faces on tunnel walls, O be the number of outer opening unit-square faces, and I be the number of removed unit cubes that lie in at least two tunnel directions.

Find $S + 3W + 5O + 11I$.

Return only the integer.

Structured Annotations

Structured Visual Information


```
{
 "block_dimensions": {"right": 3, "front": 4, "up": 4},
 "coordinate_indexing": "zero-based unit-cube cells",
 "axis_color_legend": {"right": "orange", "front": "blue", "up": "green"},
 "tunnel_specs": [
  {"tunnel_id": 1, "axis": "right", "fixed_coordinates": {"front": 1, "up": 1}},
  {"tunnel_id": 2, "axis": "front", "fixed_coordinates": {"right": 1, "up": 1}},
  {"tunnel_id": 3, "axis": "up", "fixed_coordinates": {"right": 1, "front": 1}},
  {"tunnel_id": 4, "axis": "up", "fixed_coordinates": {"right": 1, "front": 2}},
  {"tunnel_id": 5, "axis": "right", "fixed_coordinates": {"front": 2, "up": 1}}],
 ...
}
```


Structured Reasoning Steps


```
{
 "steps": [
  {"step_id": 1, "type": "given",
   "natural": "Read the block dimensions and the colored through-tunnel specifications."},
  {"step_id": 2, "type": "calculation",
   "natural": "Expand each tunnel across the full block along its axis, keeping the other two
        coordinates fixed."},
  {"step_id": 3, "type": "calculation",
   "natural": "Form the union of drilled cells. Track distinct axis memberships per cell; count
        I using at least two different directions."},
  {"step_id": 4, "type": "calculation",
   "natural": "For each drilled-cell face, count an outer opening if it borders outside the
        block, or a tunnel-wall face if it borders a remaining cube. Ignore faces
        between drilled cells. Count exposed faces on the remaining solid."},
  {"step_id": 5, "type": "calculation",
   "natural": "Check the surface balance: original exterior minus openings plus tunnel walls.
        Then evaluate the weighted expression in the question."}
 ]
}
```


Reference answer: 294

Figure 29: Representative S^{2}BENCH case for Voxel State-Transformation Reasoning (VSTR), example 2 of 2.

G Privileged Structural Hint Analysis

Table 19 compares the visual hint h_v, reasoning hint h_r, and their combination across all ten tasks. The decomposition shows that h_v contributes most of the improvement, while h_r alone yields only small gains and no improvement on VC, DROT, FGVGR, NFMC, VSRR, or VSTR.

We exclude Voxel Counting (VC) from Figure 5 and its macro average because a useful visual-structure hint must enumerate the relevant objects or occupied voxels, which can reveal the answer. Projection and Shadow Reasoning (PSR) remains included; the combined hints yield modest gains from 22.0% to 23.0% at 9B and from 41.5% to 42.0% at 27B. Complete results, including VC, are reported below for transparency.

For clarity, the table uses the following task names: Solid-Geometry Metric Reasoning (SGMR), Voxel Counting (VC), Dice-Rolling Orientation Tracking (DROT), Fine-Grained Visual-Geometric Reasoning (FGVGR), Net Folding and Marker Correspondence (NFMC), Polyhedral Topology and Visibility Reasoning (PTVR), Projection and Shadow Reasoning (PSR), Orthographic Multi-View Reasoning (OMVR), Voxel Spatial-Relation Reasoning (VSRR), and Voxel State-Transformation Reasoning (VSTR).

Table 19: Ablation of privileged visual and reasoning hints on the ten S^{2}BENCH tasks. We compare the base input with the visual hint h_v, reasoning hint h_r, and their combination. Overall denotes the ten-task macro-average. VC results involving h_v are leakage-prone and are reported only for analysis. Bold indicates the best result within each model scale.

Scale	Prompt	SGMR	VC	DROT	FGVGR	NFMC	PTVR	PSR	OMVR	VSRR	VSTR	Overall
9B	Base	48.5	28.5	0.5	0.0	0.0	6.0	22.0	25.0	3.5	6.0	14.0
	$+h_v$	49.0	90.0	29.0	70.0	0.5	38.5	23.0	75.0	7.0	68.0	45.0
	$+h_r$	49.0	28.5	0.5	0.0	0.0	8.0	**23.5**	29.0	3.5	6.0	14.8
	$+h_v + h_r$	**49.5**	**99.5**	**31.0**	**75.0**	**1.0**	**41.0**	23.0	**79.5**	**7.5**	**72.5**	**48.0**
27B	Base	59.5	31.5	5.5	0.0	0.0	9.0	41.5	32.0	5.0	10.5	19.4
	$+h_v$	**60.0**	94.0	59.0	84.0	19.0	56.0	42.0	80.0	93.0	87.0	67.4
	$+h_r$	59.5	31.5	5.5	0.0	0.0	12.0	**43.5**	37.0	5.0	10.5	20.5
	$+h_v + h_r$	**60.0**	**100.0**	**63.0**	**88.0**	**22.0**	**60.0**	42.0	**83.5**	**98.5**	**91.5**	**70.9**

H SFT Data Construction Details

We use Qwen3.5-27B to generate candidate responses for all 10,000 instances in the balanced S^{2}TRAIN corpus. For nine categories, the model is conditioned on both h_v and h_r; 6,098 of the 9,000 responses pass the task-specific evaluators. Voxel Counting (VC) is handled separately because an informative visual hint must enumerate the relevant voxels and may reveal the target count. We therefore sample responses for the 1,000 VC instances directly from Qwen3.5-27B (Base), without privileged hints, and retain 324. The final supervised fine-tuning set thus contains $|\mathcal{D}_{\mathrm{SFT}}| = 6{,}422$ examples.

I Qualitative Analysis of Model Responses

Figures 30–36 present seven representative cases comparing Qwen3.5-27B with VISTA-RL-27B on S^{2}BENCH. Both models receive the same question and image. The base model often recognizes parts of the relevant geometry but fails to maintain a consistent 3D representation through the reasoning process. In contrast, VISTA-RL-27B more reliably extracts fine-grained visual information, tracks spatial states, and derives the verified answer. These examples complement the aggregate results in Table 3 with qualitative evidence.

Dice-Rolling Orientation Tracking (DROT)

Question

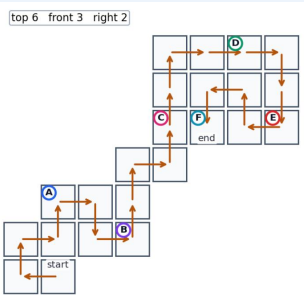
The figure shows a standard die rolling without slipping along the arrow path. Opposite faces sum to 7.

At the start, the front face points toward the bottom of the page and the right face points toward the right side of the page.

The path contains six marked checkpoint squares A-F. After the die reaches checkpoint A, let A be the top face. After checkpoint B, let B be the front face. After checkpoint C, let C be the right face. After checkpoint D, let D be the top face. After checkpoint E, let E be the front face. After checkpoint F, let F be the right face.

Find 60A+17B+11C+7D+5E+3F. Return only the integer.

Input Figure



Same question and image for both models

Ground Truth: 477

Qwen3.5-27B

Output tokens: 11,231 — **Incorrect**

Let's try this path. Define the path step by step.
Start at (0,0): T=6, F=3, R=2
Move 1: right to (1,0) // +x, rolling right
[…]
Move 3: up to (1,2) = A // +y, rolling backward
So at A: T=2, F=4, R=6. Checkpoint A: **A = top face = 2**
[…]
So at B: T=6, F=2, R=3. Checkpoint B: **B = front face = F = 2**
[…]
So at C: T=1, F=3, R=2. Checkpoint C: **C = right face = R = 2**
[…]
In this path, we reached A at move 3, B at move 7, C at move 12, F at move 13,
E at move 15, D at move 17.
And the checkpoints are in order **A,B,C,F,E,D**, which matches the diagram where F is before E and D.
[…]
A = 2 B = 2 C = 2 D = 6 E = 3 F = 1

60 × 2 + 17 × 2 + 11 × 2 + 7 × 6 + 5 × 3 + 3 × 1 = 236 ✗

VISTA-RL-27B (Ours)

Output tokens: 4,111 — **Correct**

Initial state at start (step 0): top=6, front=3, right=2
Path moves: "**LURURDRUURUUURRRDDLULD**" — 22 steps,
but we only care about checkpoints at steps **4,7,11,15,18,22**.
[…]
Step 4: U → Checkpoint A
→ (5, 1, 3). Checkpoint A: after entering cell, A is the top face → **A = 5**
[…]
Step 7: R → Checkpoint B
→ (2, 4, 6). Checkpoint B: after entering, B is the front face → **B = 4**
[…]
Step 11: U → Checkpoint C
→ (3, 6, 5). Checkpoint C: after entering, C is the right face → **C = 5**
[…]
Now collect: A = 5 B = 4 C = 5 D = 3 E = 6 F = 1

60 × 5 + 17 × 4 + 11 × 5 + 7 × 3 + 5 × 6 + 3 × 1 = 477 ✓

Figure 30: Response comparison on Dice-Rolling Orientation Tracking (DROT). Qwen3.5-27B loses track of the die orientation and checkpoint sequence, whereas VISTA-RL-27B explicitly updates the orientation after each roll and obtains the verified answer.

Fine-Grained Visual-Geometric Reasoning (FGVGR)

Question

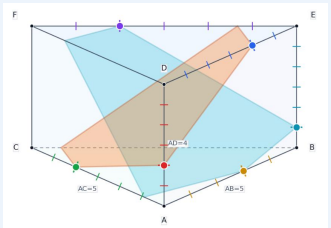
The figure shows a right triangular prism ABC-DEF. The base triangle ABC is right-angled at A, and AD, BE, and CF are perpendicular to plane ABC. Visible prism edges are solid and hidden prism edges are dashed. The edge lengths AB, AC, and AD are printed in the diagram.

The red, green, and blue points lie on visible prism edges and determine plane α; the orange polygon is the section Sα where alpha cuts the prism. The cyan, purple, and gold points lie on visible prism edges and determine plane β; the teal polygon is the section Sβ where beta cuts the prism.

For every colored point, the edge name is not written beside it. Small ticks of the same color divide that whole edge into equal parts, and the colored point is one of those division ticks. Read each division from the alphabetically earlier endpoint of that edge to the alphabetically later endpoint.

Write Area(Sα)²=a/b, Area(Sβ)²=c/d, and cos²(θ)=e/f in lowest terms, where θ is the angle between planes α and β. Let g and h be the numbers of sides of Sα and Sβ, respectively. Each section meets exactly one vertical prism edge; let u and v be the corresponding codes for Sα and Sβ, using AD=1, BE=2, and CF=3. **Find 13a+11b+11c+13d+17e+31f+29g+41h+37u+37v.** Return only the integer.

Input Figure



Same question and image for both models

Ground Truth: 340786

Qwen3.5-27B

Output tokens: 19,887 — **Incorrect**

In our assumption:
Red on AD: at (0,0,1) — so if AD=4, and divided into 4 parts, then at 1/4 from A, so first division.
Green on AC: at (0,2,0) — AC=5, so if divided into 5 parts, at 2/5 from A.
Blue on AB: at (3,0,0) — AB=5, at 3/5 from A.
[…]
So Area(Sα) = 7/2. Thus Area(Sα)² = (49)/4. So **a=49, b=4**
[…]
For Sβ, we assumed gold on AB at (2,0,0), **cyan on CF** at (0,5,3), purple on EF at (2,3,4)
[…]
So Area(Sβ) = 125 √221 / 88
Thus Area² = (125)² × 221 / (88)² = 15625 × 221 / 7744
[…]
For Sα, we have triangle with area 7/2, squared 49/4, good.
For Sβ, pentagon, area squared **3453125/7744**, good.
cos² θ = **100/10829**, good. **g=3, h=5, u=1, v=3**.

Final answer: 38423567 ✗

VISTA-RL-27B (Ours)

Output tokens: 12,894 — **Correct**

So plane α passes through: R = (0, 0, 4/3) G = (0, 10/3, 0) Bp = (10/3, 0, 4)
[…]
Plane alpha: 12x - 6y -15z + 20 = 0
[…]
So Area(Sα)² = **2000/9**
[…]
So plane β passes through:
Cy = (5, 0, 2/3) Pu = (5/3, 10/3, 4) Go = (3, 0, 0)
[…]
Plane beta: x + 4y - 3z = 3
[…]
Then Area(Sβ)² = [31 √26 / 9]² = (961 × 26) / 81
So **c=24986, d=81**. Then cos² θ = 121 / (9 × 130) = **121 / 1170**
[…]
Both sections are pentagons, as we found 5 intersection points each. So **g=5, h=5**.
Sα meets AD → u=1 **Sβ meets BE → v=2**

Final answer: 340786 ✓

Figure 31: Response comparison on Fine-Grained Visual-Geometric Reasoning (FGVGR). Qwen3.5-27B misinterprets the diagram's fine-grained geometric configuration, while VISTA-RL-27B reconstructs the relevant geometry and completes the calculation correctly.

Net Folding and Marker Correspondence (NFMC)

Question

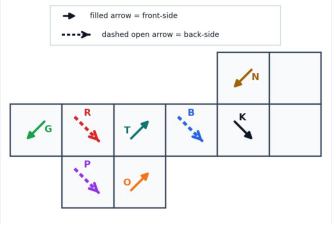
The figure is a unit-square surface net of a 2 by 1 by 1 rectangular box made from two unit cubes. Each colored diagonal arrow is printed either on the front side of the sheet or on the back side: a filled arrowhead means front-side printing, and an open arrowhead with a dashed shaft means back-side printing. The diagram is viewed from the front side. For an open-headed back-side arrow, the drawn direction is its front-side projection, so its actual printed direction is left-right mirrored inside that square before folding. No exterior-side visibility is used in this problem.

For two arrows m,n, Dir(m,n)=1 if their actual folded printed directions are the same, 2 if opposite, 3 if perpendicular, 4 if they make an acute 60-degree angle, and 5 if they make an obtuse 120-degree angle.

RawDir(m,n) is computed the same way but using the drawn directions exactly as seen, ignoring the back-side mirror. Change(m,n)=1 if Dir(m,n) and RawDir(m,n) are different, and 0 otherwise. SameSide(m,n)=1 if both arrows are printed on the same side of the sheet, and 0 otherwise.

Let A=Dir(green,black); B=Dir(orange,black); C=Dir(black,teal); D=Dir(blue,green); E=RawDir(green,teal); F=RawDir(orange,teal); G=Change(purple,orange); H=Change(green,teal); I=SameSide(orange,teal); J be the number of unordered arrow pairs with Dir=4; K be the number of unordered arrow pairs with Change=1; L be the number of unordered arrow pairs where exactly one arrow is open-headed and Dir=3. **Find 17A+19B+23C+29D+31E+37F+41G+43H+47I+53J+59K+61L.** Return only the integer.

Input Figure



Same question and image for both models

Ground Truth: 2100

Qwen3.5-27B

Output tokens: 11,744 — **Incorrect**

So let's proceed with that assumption.
So actual directions:
G: SW R: SW T: NE B: SW K: SE N: SW P: SW O: NE
[…]
A = Dir(G,K) = **Dir(SW, SE) = 3**
B = Dir(O,K) = Dir(NE, SE) = 3
C = Dir(K,T) = Dir(SE, NE) = 3
D = Dir(B,G) = **Dir(SW, SW) = 1**
E = RawDir(G,T) = RawDir(SW, NE) = 2 (opposite)
F = RawDir(O,T) = RawDir(NE, NE) = 1 (same)
[…]
J = number of pairs with Dir=4 = **0 (no 45° angles in 2D)**
K = number of pairs with Change=1 = **15 (front-back pairs)**
L = number of pairs with exactly one open-headed and **Dir=3 = 3** (K with each back-side)
[…]
G = Change(P,O) = 1 (as computed) H = Change(G,T) = 0 I = SameSide(O,T) = 1

Final answer: 1461 ✗

VISTA-RL-27B (Ours)

Output tokens: 18,412 — **Correct**

For back-side arrows, we mirror them horizontally before folding.
[…]
Let me assume that for the sake of proceeding.
Given two vectors v1, v2, both of magnitude √2, dot product d = v1·v2
If d = 2, Dir=1; if d = -2, Dir=2; if d = 0, Dir=3;
if d = 1, Dir=4 (acute 45°); if d = -1, Dir=5 (obtuse 135°).
[…]
G: v_dir = (-1,-1,0) K: v_dir = (0,-1,1)
O: v_dir = (1,0,-1) T: v_dir = (0,1,-1)
B: v_dir = (1,-1,0) R: v_dir = (0,-1,1)
P: v_dir = (-1,0,1) N: v_dir = (-1,0,-1)
[…]
A=4 B=5 C=2 D=3 E=5 F=4 G=1 H=0 I=1
[…]
So **J = 11** So **K = 12**
So (G,B) and (P,N) satisfy. (O,N) both front, so not. So **L=2**

Final answer: 2100 ✓

Figure 32: Response comparison on Net Folding and Marker Correspondence (NFMC). The base model mishandles the orientation changes induced by folding, whereas VISTA-RL-27B tracks the folded faces and marker correspondence consistently.

Polyhedral Topology and Visibility Reasoning (PTVR)

Question

The figure shows a convex octahedron. Solid segments are visible edges and dashed segments are hidden edges. Three faces are shaded blue, green, and orange.

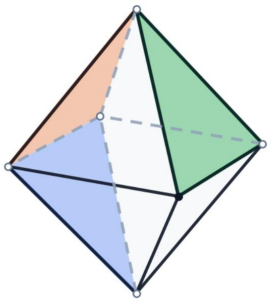
A face path moves from one face to an adjacent face across their shared edge, and it may not repeat a face.

Consider only face paths from the blue face to the orange face that pass through the green face. Let L be the minimum possible number of face-to-face moves among such paths.

Among all such paths of length L, let H be the largest number of dashed shared edges crossed, and let N be the number of length-L paths that cross exactly H dashed shared edges.

Find 19L+23H+6N. Return only the integer.

Input Figure



Same question and image for both models

Ground Truth: 153

Qwen3.5-27B

Output tokens: 14,332 — **Incorrect**

But for shading, suppose:
Blue face: F8 = V4,V1,V6 Green face: F2 = V2,V3,V5 Orange face: F3 = V3,V4,V5
[…]
Now, with V3 and V4 as back vertices, **assume edges incident to V3 or V4 are dashed**.
[…]
Path B: F8 - F7 - F6 - F2 - F3
1. F8 to F7: share V4-V6 — dashed (V4-V6)
2. F7 to F6: share V3-V6 — dashed (V3-V6)
3. F6 to F2: share V2-V3 — dashed (V2-V3)
4. F2 to F3: share V3-V5 — dashed (V3-V5)
All 4 edges are dashed.
[…]
So the number of dashed edges crossed: 1,2,2,4
Thus, the maximum **H = 4**
Number of paths that cross exactly H=4 dashed edges: only path B, so **N=1**
L=4 (minimal length)

19 × 4 + 23 × 4 + 6 × 1 = 174 ✗

VISTA-RL-27B (Ours)

Output tokens: 3,932 — **Correct**

Blue start face: F5 Required green face (checkpoint): F3 Orange end face: F1
[…]
Shortest from F5 to F3: 3 steps (e.g., F5-F6-F2-F3 or F5-F8-F7-F3)
Shortest from F3 to F1: F3 → F4 → F1: 2 steps; F3 → F2 → F1: 2 steps
So minimum total: **3 + 2 = 5 steps.**
[…]
So valid paths of length 5:
1. F5-F6-F2-F3-F4-F1 → 1 dashed
2. F5-F6-F7-F3-F2-F1 → 0 dashed
3. F5-F6-F7-F3-F4-F1 → 1 dashed
4. F5-F8-F7-F3-F2-F1 → 1 dashed
5. F5-F8-F7-F3-F4-F1 → 2 dashed
6. F5-F8-F4-F3-F2-F1 → 2 dashed
[…]
Now, among them, maximum dashed crossings **H = 2** (achieved by paths 5 and 6).
Number of such paths **N = 2**.

19 × 5 + 23 × 2 + 6 × 2 = 153 ✓

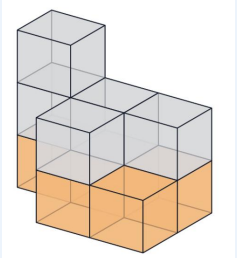
Figure 33: Response comparison on Polyhedral Topology and Visibility Reasoning (PTVR). Qwen3.5-27B makes an incorrect topological inference from the visible and hidden edges, while VISTA-RL-27B reasons over the face connectivity and reaches the correct result.

Voxel Counting (VC)

Question

Please count how many cubes are in the figure.

Input Figure



Ground Truth: 10

Qwen3.5-27B

Output tokens: 1,954 — **Incorrect**

There is a tall column on the left, which appears to be 3 cubes high.
[…]
Then to the right, we have a 2x2 area: positions (B,1), (C,1), (B,2), (C,2)
— where y=1 is front, y=2 is back.
At (B,1): orange bottom + gray top → 2 cubes
At (C,1): orange bottom + gray top → 2 cubes
At (B,2): ? — we see a gray cube on top, so must have orange below → 2 cubes
At (C,2): same — gray on top, so orange below → 2 cubes
[…]
So total cubes = **3 (left column) + 8 (2x2 block) = 11**.
[…]
Alternative way: count by layers.
Total layer 1: 5 Total layer 2: 5 Total layer 3: 1
Grand total: **5 + 5 + 1 = 11**

Final answer: 11 ✗

VISTA-RL-27B (Ours)

Output tokens: 576 — **Correct**

We are given a 3D isometric view of stacked cubes, with coordinates defined as:

x: right (+x)
y: behind (+y), so front row is y=0
z: up (+z), bottom layer z=0

The occupied columns (with their heights) are listed in the "occupied_column_records":

1. (x=1, y=0, height=2) → cubes at z=0 and z=1
2. **(x=2, y=0, height=1) → cube at z=0**
3. (x=0, y=1, height=3) → cubes at z=0, z=1, z=2
4. (x=1, y=1, height=2) → cubes at z=0 and z=1
5. (x=2, y=1, height=2) → cubes at z=0 and z=1

Let's list all individual cube positions to avoid double-counting:

From column (1,0): (1,0,0), (1,0,1)
From column (2,0): (2,0,0)
From column (0,1): (0,1,0), (0,1,1), (0,1,2)
From column (1,1): (1,1,0), (1,1,1)
From column (2,1): (2,1,0), (2,1,1)

Now count them:

From (1,0): 2 cubes
From (2,0): 1 cube
From (0,1): 3 cubes
From (1,1): 2 cubes
From (2,1): 2 cubes

Total = 2 + 1 + 3 + 2 + 2 = 10

Alternatively, we can sum the heights of each occupied column — since each column has no gaps (as per problem statement: "no-gap column"), the number of cubes in that column equals its height.

So:
Column (1,0): height 2 → 2 cubes
Column (2,0): height 1 → 1 cube
Column (0,1): height 3 → 3 cubes
Column (1,1): height 2 → 2 cubes
Column (2,1): height 2 → 2 cubes

Sum: 2+1+3+2+2 = 10
Thus, total cubes = 10.

Final answer: 10 ✓

Figure 34: Response comparison on Voxel Counting (VC). The base model miscounts the configuration, whereas VISTA-RL-27B systematically decomposes the voxel structure and returns the verified count.

Voxel Spatial-Relation Reasoning (VSRR)

Question

The figure shows a solid made of face-to-face unit cubes. Five surface cubes are marked P, Q, R, S, and T. The small arrows show the fixed front, right, and up directions; no coordinates are given.

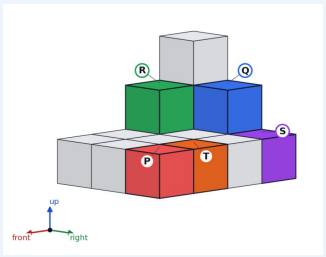
For marked cubes X and Y, Rel(X,Y) is used only when Y is strictly to one side, strictly front or behind, and strictly above or below X. Use the code 1=right-front-above, 2=right-front-below, 3=right-behind-above, 4=right-behind-below, 5=left-front-above, 6=left-front-below, 7=left-behind-above, and 8=left-behind-below.

Face(X,Y) is the direction from X to Y if the two marked cubes share a square face, using 1=right, 2=left, 3=front, 4=behind, 5=above, and 6=below; use 0 if they do not share a square face.

Dist(X,Y) is the minimum number of face-step moves from X to Y while staying inside the solid.

Let A=Rel(R,S), B=Rel(P,R); let C=Face(P,R), D=Face(T,P); and let E=Dist(Q,R). **Find 19A+23B+29C+31D+53E.** Return only the integer.

Input Figure



Same question and image for both models

Ground Truth: 436

Qwen3.5-27B

Output tokens: 4,275 — **Incorrect**

Set cube P at (0, 0, 0) for simplicity.
Cube **T is immediately to the right of P** → so T is at (1, 0, 0)
[…]
Assume that the cube under R is at (-1,-1,0), so R is at (-1,-1,1)
Then Q is at (0,-1,1) — to the right of R.
[…]
So relative to R, S is: **right, front, below.**
Code for right-front-below is 2.
[…]
From P to R: left (x decreases), behind (y decreases), above (z increases)
→ left-behind-above → code 7, correct.

Face(P,R): P(0,0,0), R(-1,-1,1) — not adjacent, so 0, correct.
Face(T,P): T(1,0,0), P(0,0,0) — adjacent, **from T to P is left → code 2**, correct.
Dist(Q,R): Q(0,-1,1), R(-1,-1,1) — **adjacent in x, so distance 1**, correct.

19 × 2 + 23 × 7 + 29 × 0 + 31 × 2 + 53 × 1 = 314 ✗

VISTA-RL-27B (Ours)

Output tokens: 2,635 — **Correct**

We are given the exact coordinates of each labeled cube in a (right, front, up) coordinate system:
P: (2, 3, 0) Q: (1, 0, 1) R: (0, 1, 1) S: (2, 0, 0) T: (2, 2, 0)
[…]
R = (0,1,1), S = (2,0,0)
Δ = S - R = (2-0, 0-1, 0-1) = (2, -1, -1)
Signs: right=+, front=-, up=- → corresponds to **code 4 (right-behind-below)**
[…]
Δ = R - P = (0-2, 1-3, 1-0) = (-2, -2, 1)
Signs: right=-, front=-, up=+ → corresponds to code 7 (left-behind-above)
[…]
Differences: Δr = -2, Δf = -2, Δu = +1 → not adjacent at all → no shared face.
Δ = P - T = (0, 1, 0) → only front changes by +1 → so **from T to P is front → code 3**
[…] Assuming (0,0,1) exists, then:
Q(1,0,1) → (0,0,1) → R(0,1,1) : **2 steps.**

19 × 4 + 23 × 7 + 29 × 0 + 31 × 3 + 53 × 2 = 436 ✓

Figure 35: Response comparison on Voxel Spatial-Relation Reasoning (VSRR). Qwen3.5-27B derives inconsistent spatial relations from the perspective view, while VISTA-RL-27B establishes a coherent representation of the voxel configuration and answers correctly.

Voxel State-Transformation Reasoning (VSTR)

Question

The figure shows the original solid, assembled from face-to-face unit cubes. The gray cubes are drawn transparent so that all red marked cubes can be seen. It contains 16 unit cubes, of which 3 red cubes are marked for removal.

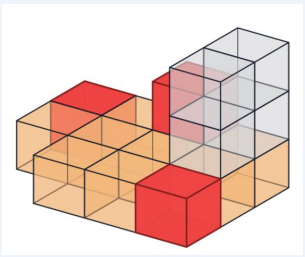
First paint every exterior unit-square face of the original solid. Then remove all red cubes; newly exposed cut faces are not painted.

Let A be the number of remaining cubes with exactly 2 painted faces from the original painting and exactly 3 exposed faces after the removal.

Let N be the number of newly exposed unpainted cut faces, G be the number of shared square faces between pairs of remaining cubes, and R be the number of red cubes removed.

Find 29A+17N+7G+11R. Return only the integer.

Input Figure



Same question and image for both models

Ground Truth: 341

Qwen3.5-27B

Output tokens: 8,786 — **Incorrect**

So final list:
z=0: (1,1,0), (2,1,0), (3,1,0), (1,2,0), (2,2,0), (1,3,0), (2,3,0), (3,3,0) → 8
z=1: (1,3,1), (2,3,1), (3,3,1), (3,2,1) → 4
z=2: (1,3,2), (2,3,2), (3,3,2) → 3
z=3: (3,3,3) → 1
Total: 8+4+3+1=16. Good.
[…]
Red cubes: C3=(3,1,0), C6=(1,3,0), C11=(3,3,1)
[…]
So only C7 and C10 satisfy deg_orig=4 and deg_rem=3. **Thus A=2.**
[…]
C3: 1 remaining neighbor C6: 3 remaining neighbors C11: 4 remaining neighbors
Total N = 1 + 3 + 4 = 8
[…]
G=13: total shared faces 21, minus 8 involving red cubes, so 13 within remaining.
R=3.

29 × 2 + 17 × 8 + 7 × 13 + 11 × 3 = 318 ✗

VISTA-RL-27B (Ours)

Output tokens: 5,277 — **Correct**

Original cubes (16 total):
(1,0,0), (2,0,0), (3,0,0) → row front=0
(0,1,0), (1,1,0), (2,1,0), (3,1,0) → front=1
(0,2,0), (1,2,0), (2,2,0), (3,2,0) → front=2
(3,1,1), (2,2,1), (3,2,1) → up=1 (3,1,2), (3,2,2) → up=2
[…]
Red_set = {(3,0,0), (0,2,0), (2,2,1)}
Remaining_set = Original_set - Red_set
[…]
So far, **A = 3. Cubes qualifying: (3,1,0), (2,2,0), (3,2,1)**
[…]
Total N = 2 + 2 + 2 = 6
[…]
Since each shared face is counted twice (once per cube), **G = 34 / 2 = 17**
R = 3 (given)

29 × 3 + 17 × 6 + 7 × 17 + 11 × 3 = 341 ✓

Figure 36: Response comparison on Voxel State-Transformation Reasoning (VSTR). The base model fails to propagate the transformation through the complete voxel state, whereas VISTA-RL-27B reconstructs the updated configuration and obtains the verified answer.