

🌐 SATURN: Symbolic Spatial Reasoning for Multi-Perspective Grounding

Danial Kamali¹, Tanawan Preamsri¹, Shreya Rajpal¹,
Amir Zadeh², Chuan Li², Parisa Kordjamshidi¹

¹Michigan State University, ²Lambda Labs

{kamalida, premsrit, rajpalsh, kordjams}@msu.edu

Abstract

Vision-Language Models (VLMs) remain unreliable when spatial reasoning requires composing relations whose meanings depend on frames of reference. Existing tool-augmented spatial reasoning methods make reasoning more explicit, but often rely on low-level geometric procedures and hard binary decisions over noisy perception. We propose SATURN, a neuro-symbolic framework for perspective-aware compositional spatial reasoning. SATURN reconstructs an approximate 3D scene, derives soft perspective-aware spatial predicates, and composes them with a training-free Pythonic symbolic executor, separating perception from reasoning while preserving uncertainty through multi-hop inference. We also introduce 3D FORCE, a diagnostic benchmark that controls reasoning depth, view, and perspective composition for spatial arrangement grounding (SAG) and referring expression grounding (REF). On 3D FORCE, VLMs and spatially trained models degrade sharply as depth and perspective complexity increase, whereas SATURN degrades the least and outperforms every baseline at each depth. On the real-world MindCube benchmark, SATURN achieves 78.06% overall accuracy, outperforming the strongest baseline by 14 percentage points.

Code: <https://github.com/HLR/SATURN>

1 Introduction

Many visually grounded instructions require resolving spatial relations in different frames of reference (FoRs), such as the camera’s view, an agent’s view, or an object’s intrinsic orientation as shown in Figure 1. A relation such as “left of the truck” can change depending on whether the relation is interpreted from the camera, the agent, or the truck itself. We define an FoR as the perspective induced by an anchor’s 3D position and orientation; cameras and oriented objects can therefore induce distinct FoRs.

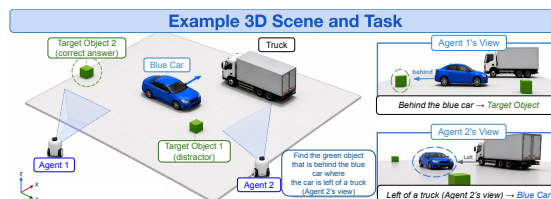


Figure 1: An example illustrating 3D scene understanding with multi-perspective reasoning.

Existing spatial benchmarks have demonstrated VLMs’ failure in spatial reasoning (Yang et al., 2026; Tong et al., 2024). Pure neural VLMs implicitly encode spatial relations, making it difficult to ensure that the same relation is applied consistently across different objects, views, and FoRs (Zhang et al., 2026). Explicit reasoning approaches, including neuro-symbolic pipelines and tool-augmented agents, make spatial computation more exact. However, in practice, these approaches often require low-level geometric programming and rely on pipelines of crisp decisions during multi-hop reasoning, making them less robust when perceptual outputs are noisy. To overcome these limitations, we introduce **SATURN**, a neuro-symbolic framework for 3D perspective-aware compositional reasoning. Rather than asking VLMs to infer multi-hop spatial relations implicitly or generate low-level geometric code, SATURN elevates the reasoning from raw 3D states to soft FoR-aware predicates. This abstraction helps downstream programs compose relations without re-implementing geometry or thresholding perceptual evidence. SATURN first approximates a coarse 3D scene using neural perception and derives predicates such as left and front under different FoRs. A Pythonic symbolic executor then composes these predicates to answer multi-hop spatial queries.

While we target perspective-aware compositional spatial grounding, existing benchmarks do not cleanly isolate this capability. Current spatial reasoning benchmarks largely emphasize ei-

ther primitive spatial judgments, such as relative position, orientation, depth, and perspective-taking (Zhang et al., 2026; Wang et al., 2025c; Li et al., 2025a), or broad embodied, navigational, and long-horizon tasks (Li et al., 2025c; Sohn et al., 2025; Zhao et al., 2026; Ma et al., 2026). Between these settings lies an intermediate capability: *perspective-aware compositional spatial grounding*, where a model must ground objects while resolving the frame of reference associated with each spatial relation and composing multiple relations into a single decision. For example, the spatial component of an instruction such as “Pick up the cup and place it on the desk on your left and in front of me” requires resolving *left* in the agent’s FoR, *front* in the user’s FoR, and composing these relations over the relevant objects before any action can be taken.

To evaluate this intermediate reasoning layer, we introduce 3D Frames Of Reference Composition Evaluation (**3D FORCE**), a diagnostic benchmark that controls relation depth, viewpoint, and FoR composition. 3D FORCE contains two complementary subsets: SAG, which evaluates whether a multi-hop spatial arrangement is satisfied, and REF, which evaluates whether a model can ground the target object described by a perspective-aware compositional referring expression. On 3D FORCE, SATURN consistently outperforms general-purpose VLMs, spatially trained VLMs, and tool-augmented baselines across both SAG and REF. This gain is most pronounced in settings that require composing relations across multiple objects, views, and FoRs. While spatial-focused tuned VLMs improve local spatial judgments, their performance degrades as reasoning depth and FoR composition increase. This suggests that current spatial supervision improves relation recognition more than systematic relation composition across FoRs. We further evaluate SATURN on real-world multi-view benchmarks MMSI (Yang et al., 2026) and MindCube (Yin et al., 2025), showing its gains beyond the controlled setting of 3D FORCE.

In summary, we make three contributions. 1) We introduce SATURN, a neuro-symbolic framework that converts noisy 3D visual perception into reusable soft spatial predicates defined under camera-centric, object-centric, and virtual-viewer FoRs, then composes these predicates through Pythonic symbolic execution. 2) We introduce 3D FORCE, a diagnostic benchmark that isolates perspective-aware relation composition across spa-

tial arrangement and referring-expression grounding while controlling relation depth, topology, view count, and FoR assignment. 3) We show that current VLMs become less reliable as relation depth and FoR composition increase. SATURN degrades the least while outperforming VLMs, spatially trained VLMs, and tool-augmented baselines on 3D FORCE, and these gains transfer to the real-world MindCube and MMSI benchmarks.

2 Related Work

Spatial Reasoning in Vision-Language Models.

Recent studies show that VLMs struggle with spatial reasoning, including relative position, depth, orientation, object localization, FoR ambiguity, and multi-view reasoning (Kamath et al., 2023; Wang et al., 2024; Liu et al., 2025; Daxberger et al., 2025; Wang et al., 2025c; Zhang et al., 2025c; Yu et al., 2025; Yang et al., 2026; Yin et al., 2025). A line of work addresses these limitations by fine-tuning models on spatial-focused data, incorporating geometric priors such as depth and 3D structure during training, or injecting linguistic structure for grounded compositional generalization (Chen et al., 2024; Cheng et al., 2024; Wu et al., 2025a; Li et al., 2025b; Wu et al., 2026; Ma et al., 2025c,b; Fan et al., 2025; Kamali and Kordjamshidi, 2023). While these methods improve performance on targeted benchmarks, they still encode spatial relations as learned model priors rather than deriving them explicitly from scene geometry, which can limit generalization beyond the training distribution (Chen et al., 2025a; Mirzaee and Kordjamshidi, 2023). SATURN instead externalizes spatial reasoning by reconstructing a coarse 3D scene, deriving FoR-aware predicates, and composing them through symbolic execution.

Tool-Augmented Spatial Reasoning. Neuro-symbolic frameworks vary in how they integrate neural and symbolic components (Sinha et al., 2026; Faghihi et al., 2021; Rajpal et al., 2026); recent work also uses agents to generate neuro-symbolic programs from task descriptions (Nafar et al., 2026). Tool-augmented neuro-symbolic methods translate visual queries into executable programs or tool calls over objects and image regions (Gupta and Kembhavi, 2023; Surís et al., 2023). Recent methods extend visual programming toward 3D reasoning, but differ from SATURN in the level of abstraction exposed to the reasoner: VADAR (Marsili et al., 2025) and TVP (Wu et al.,

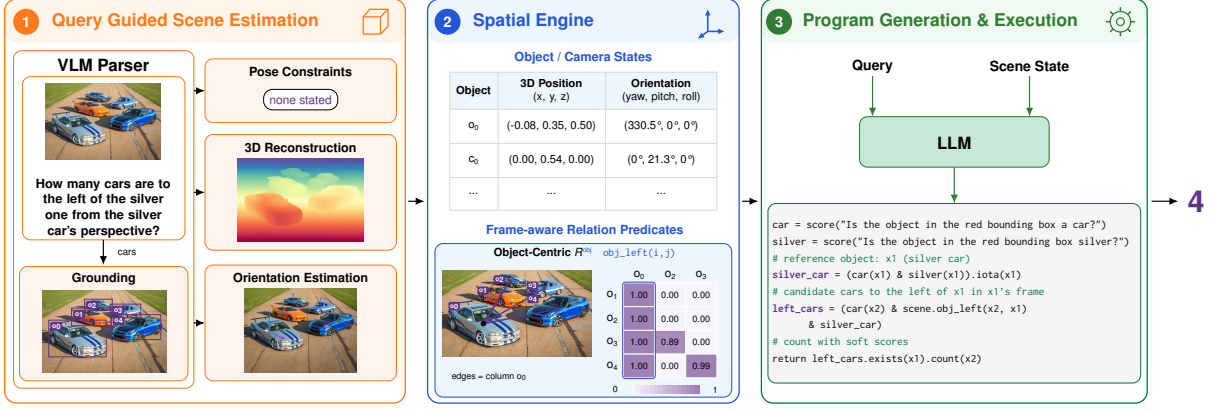


Figure 2: SATURN overview. The framework combines scene estimation, perceptual grounding, geometry-based computation of spatial predicates, and symbolic program execution for FoR-aware compositional reasoning.

2025b) reason over 3D APIs, GCA (Chen et al., 2025a) uses agentic tool calls to extract 3D object properties and invoke Python tools, pySpatial (Luo et al., 2026) combines 3D reconstruction with Python-based novel-view synthesis for perspective-aware reasoning, and TIGer (Han et al., 2026) fine-tunes a VLM to generate geometric tool calls rather than prompting a frozen model. In contrast, SATURN does not generate programs to perform low-level geometric calculations. It first derives soft FoR-aware spatial predicates from estimated scene state (objects and cameras), such as *left* and *behind*, under camera-centric, object-centric, or custom FoRs. The generated program then composes these soft predicate scores in Python, following NePTune (Kamali and Kordjamshidi, 2026). FoR-SALE (Premisri and Kordjamshidi, 2026) also treats FoR explicitly, but for image editing: it converts intrinsic spatial expressions to the camera perspective and uses layout mismatches to guide diffusion-based corrections.

Spatial Reasoning Benchmarks. Recent spatial reasoning benchmarks evaluate different levels of spatial understanding. Atomic benchmarks typically focus on primitive spatial abilities, such as spatial relation recognition, egocentric direction, relative position, depth estimation, and object orientation (Yang et al., 2019; Kamath et al., 2023; Rajabi and Kosecka, 2024; Du et al., 2024; Xu et al., 2025; Zhang et al., 2025c; Mirzaee et al., 2021; Premisri and Kordjamshidi, 2025; Li et al., 2025a; Zhang et al., 2026; Yang et al., 2026; Yin et al., 2025; Yeh et al., 2025; Gholami et al., 2025). Recent benchmarks evaluate broader embodied or long-horizon spatial skills, such as 3D awareness, 6D spatial relations, object localization, counting, functional knowledge, multi-view fusion, dynamic

scene understanding, and perspective-aware reasoning (Ma et al., 2025a; Wang et al., 2025c; Zhang et al., 2025b; Gong et al., 2025; Jia et al., 2025; Ray et al., 2025; Tang et al., 2025; Zhu et al., 2026; Wei et al., 2026). 3D FORCE fills a gap between primitive spatial tests and broad embodied tasks by evaluating compositional object-level spatial reasoning under camera-centric and object-centric FoRs. It explicitly controls hop count, relation topology, number of views, and the FoR associated with each relation. This benchmark consists of two complementary subsets: REF for frame-aware referring expressions and SAG for multi-view spatial arrangement grounding. Together, they assess whether models can compose mixed-frame spatial relations within a single reasoning problem.

3 Method

We introduce SATURN, a neuro-symbolic framework for answering perspective-aware spatial questions. As shown in Figure 2, given a natural-language query Q and a visual context C , SATURN decomposes the reasoning into three stages. First, query-guided scene estimation identifies the objects required for the query and estimates their 3D positions and orientations. In addition, when the query states facts about the camera setup, such as two views sharing a position or being separated by a rotation, SATURN uses these facts to refine the estimated camera states. Second, the spatial engine converts the resulting object and camera states into soft spatial predicates (i.e., probability scores) under camera-centric, object-centric, or virtual-viewer frames of reference. Finally, a symbolic executor runs an LLM-generated Python program over semantic and spatial predicates.

3.1 Query-Guided Scene Estimation

The first stage estimates the query-relevant geometric state of the scene. A VLM-based parser reads the query, given the context, and identifies the candidate objects, their descriptions, and the visible view needed for reasoning. The object descriptions are passed to an object grounder, which detects candidate object instances in the visual context.

Let N be the number of candidate object instances kept after grounding, and let V be the number of images (e.g., cameras). For each candidate object i , SATURN estimates a 3D position $\mathbf{x}_i$ and orientation $\mathbf{R}_i$. For each selected camera view v , SATURN estimates an initial camera position $\mathbf{x}_v^0$ and orientation $\mathbf{R}_v^0$ from the images. We denote these object and camera states as

$$P = \{(\mathbf{x}_i, \mathbf{R}_i)\}_{i=1}^N, \quad \Gamma^0 = \{(\mathbf{x}_v^0, \mathbf{R}_v^0)\}_{v=1}^V.$$

Here, P is the set of candidate object states, while Γ^0 is the set of initial image-derived camera states before any text-stated pose constraints are applied. Together, P and Γ^0 form the geometric scene state passed to the spatial engine.

Text-Stated Pose Constraints Some multi-view queries explicitly describe how the camera views are arranged. For example, a query may state that two images are taken from the same position, or that one view is rotated by a fixed angle relative to another. These statements provide constraints on the camera poses that can improve the image-derived camera estimates. SATURN extracts these statements as pose constraints and uses them to refine the initial camera states Γ^0 before computing spatial predicates. We consider two types of constraints: rotation constraints, which specify relative yaw (horizontal orientation) between camera views, and same-position constraints, which specify that multiple views share a camera position.

For each rotation constraint, SATURN refines the orientation of the constrained view by setting its yaw to the stated offset relative to its anchor view. For each same-position group, SATURN moves all cameras in the group to the position of the group’s lowest-indexed view. Camera views without extracted constraints keep their image-derived poses. The refined camera states Γ , together with the object states P , are then passed to the spatial engine. Additional details on constraint extraction and anchor-based propagation are provided in Appendix F.

3.2 Spatial Engine

The spatial engine converts the geometric scene state (P, Γ) into soft qualitative spatial predicates. The key challenge is that spatial relations depend on the FoR: object X may be to the left of object Y from one viewpoint but to the right from another. SATURN handles this by representing each frame of reference as a local coordinate system attached to an anchor $a = (\mathbf{x}_a, \mathbf{R}_a)$, where $\mathbf{x}_a$ is the anchor’s 3D position and $\mathbf{R}_a$ is its orientation. The anchor can be a camera, an object, or a virtual viewer. As a result, camera-centric relations use a camera anchor, object-centric relations use the reference object as the anchor, and virtual-viewer relations use the virtual viewpoint as the anchor.

To define predicates uniformly across these cases, let $\mathcal{E} = \{e_1, \dots, e_M\}$ denote the entities that can participate in spatial predicates, including candidate objects, camera views, and optional virtual viewers. Each entity e_i has an associated state $(\mathbf{x}_i, \mathbf{R}_i)$, where $\mathbf{x}_i$ is its 3D position and $\mathbf{R}_i$ is its orientation. For a target entity e_i , a reference entity e_j , and an anchor a , the spatial engine first expresses the displacement from e_j to e_i in the local coordinate system of a :

$$\Delta_{ij}^a = \mathbf{R}_a^\top (\mathbf{x}_i - \mathbf{x}_j).$$

Here, Δ_{ij}^a is the target-reference displacement expressed from the perspective of anchor a . Its signed coordinates provide score functions for directional relations such as LEFT, FRONT, and ABOVE. Some predicates also depend on orientation. For these predicates, we express the orientations of the target and reference entities in the same anchor frame:

$$\mathbf{R}_i^a = \mathbf{R}_a^\top \mathbf{R}_i, \quad \mathbf{R}_j^a = \mathbf{R}_a^\top \mathbf{R}_j.$$

Thus, each relation can use both relative position and relative orientation when computing the score.

Each relation r has a scoring function h_r that maps these local geometric quantities to a scalar value. The evidence is then converted into a predicate score:

$$S_r^a[i, j] = \sigma \left(\frac{h_r(\Delta_{ij}^a, \mathbf{R}_i^a, \mathbf{R}_j^a) - m_r}{\tau_r} \right),$$

where $S_r^a[i, j] \in [0, 1]$ measures how strongly entity e_i stands in relation r to entity e_j from anchor a ’s perspective. The margin m_r controls how much evidence is needed for relation r , and the temperature τ_r controls how sharply the score changes near

that margin. This formulation supports directional predicates such as LEFT, FRONT, and ABOVE; directional combinations such as FRONT-LEFT; and orientation-based predicates such as PERPENDICULAR and fixed-angle relations. Detailed predicate definitions are provided in Appendix B.

3.3 Program Generation and Soft Execution

Given the query Q , this module produces an executable Python program that specifies how the query should be answered. We use an LLM with few-shot examples and API documentation to generate the program. The generated code does not perform low-level geometry. Instead, it calls three interfaces exposed by SATURN: semantic scoring, open-ended querying, and spatial predicates from the spatial engine.

For semantic predicates, SATURN follows NeP-Tune and provides a score interface. Given a natural-language predicate and the entity set $\mathcal{E}$, score queries the backbone VLM over candidate objects and returns scores aligned with $\mathcal{E}$. For unary predicates ($n = 1$), it returns a score vector; for binary predicates ($n = 2$), it returns a pairwise score matrix aligned with $\mathcal{E} \times \mathcal{E}$. Entries involving invalid entity types, such as cameras for object-only predicates, are set to zero.

For example, `score("Is the object in the red box a car?", n=1)` scores each candidate object, while a binary predicate can ask whether two objects have the same color or shape. When the program needs an open-ended value rather than a probabilistic score, it calls `query`, which returns a textual answer such as an object’s color or category.

The generated program is executed directly in Python. During execution, it binds variables to candidate entities, retrieves semantic evidence through score or query, retrieves FoR-aware spatial predicates from the spatial engine, and composes these values to produce the final answer. This explicit execution separates reasoning from direct VLM answer prediction: the VLM provides the atomic semantic concepts, while the program specifies how predicates should be combined. SATURN preserves uncertainty by keeping semantic and spatial predicate values as continuous scores in $[0, 1]$, rather than thresholding them into hard predicates. The executor composes these scores with soft logical operators (Zadeh, 1965), such as $A \wedge B = \min(A, B)$. This allows multi-hop programs to propagate uncertainty from object grounding, 3D estimation, and spatial predicates. Additional com-

position functions are provided in Appendix C.

4 Benchmark

To isolate perspective-aware spatial composition, we introduce 3D FORCE, a diagnostic benchmark that keeps perception and object grounding simple while controlling reasoning depth, relation topology, view count, and FoR assignment. It combines high-resolution scenes with controlled multi-hop relation structures over objects, views, and FoRs. Each instance is paired with a formal logical representation following prior work (Hsu et al., 2024), making relation depth and composition structure explicit. Appendix D provides detailed comparison with existing benchmarks. 3D FORCE contains two complementary subsets: REF for perspective-aware compositional referring expression grounding and SAG for perspective-aware spatial arrangement grounding. Figure 3 illustrates examples of both subsets. Answers are derived deterministically from the scene graphs used to render each scene. The full image-rendering setup, instance-generation and verification procedures, and dataset statistics (hop count, perspective settings, and view count) are provided in Appendix E.

4.1 3D FORCE-SAG

The SAG subset evaluates VLM spatial reasoning by introducing a visual-spatial arrangement grounding task. Specifically, given a spatial configuration $S = \langle T, R \rangle$, where T includes object specifications (e.g., *object 1 is a blue minivan*) and R describes a set of spatial relations between pairs of objects (e.g., *object 1 is left of object 2*) in natural language text, the task is to determine whether the spatial configuration S is satisfied given the visual context C . The output is a binary answer $A \in \{\text{Yes}, \text{No}\}$.

4.2 3D FORCE-REF

The REF subset evaluates spatial reasoning ability in terms of compositionality and multi-hop reasoning using a referring-expression grounding task. Specifically, given a spatial referring expression S_q and a visual context(s) C , the task is to determine a bounding box B for the target object in C . The referring expression consists of a sequence of spatial relations that require multi-hop reasoning.

5 Experiments

We organize our experiments around five research questions. **RQ1:** How well do current VLMs and

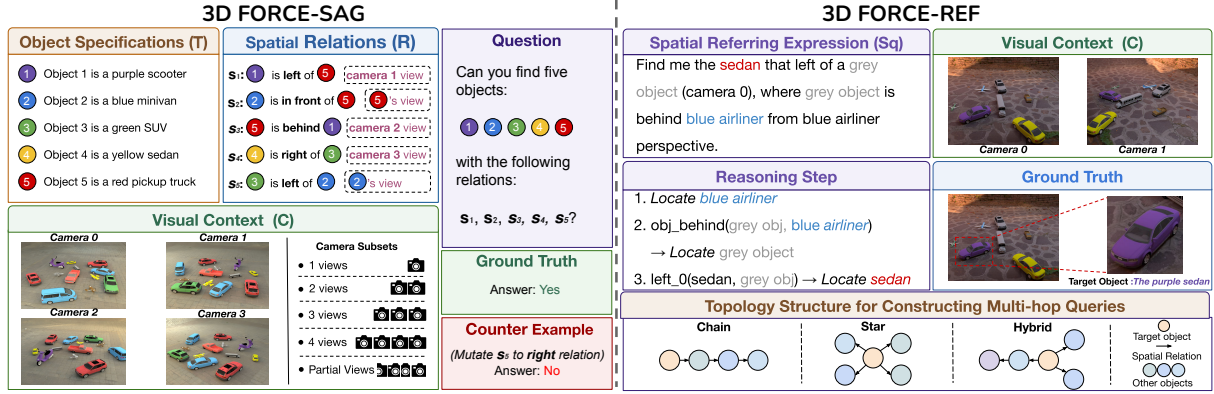


Figure 3: Overview of 3D FORCE benchmark containing two subsets: SAG, evaluates multi-hop spatial arrangement grounding, and REF, evaluates perspective-aware compositional referring expressions.

tool-augmented methods perform on composing spatial relations with various FoRs? (§5.2) **RQ2**: How does performance vary across camera-centric, object-centric, and mixed-FoR settings? (§5.3) **RQ3**: How does performance vary with relation depth and topology? (§5.4) **RQ4**: What is the effect of soft symbolic reasoning on performance? (§5.5) **RQ5**: Do improvements on our synthetic benchmark transfer to real-world benchmarks? (§5.6)

5.1 Experimental Setting

We use 3D FORCE for RQ1–RQ4 because it provides controlled annotations for task type, FoR, reasoning depth, and relation topology while keeping perception and object grounding simple. These controlled axes let us isolate the effects of perspective and compositional complexity. For RQ5, we evaluate SATURN on the real-world multi-view benchmarks MindCube-1K (Yin et al., 2025) and MMSI (Yang et al., 2026) to test whether the findings extend beyond synthetic scenes. Appendix G describes these benchmarks and provides implementation details.

Baselines. We compare SATURN against general-purpose VLMs, spatially trained VLMs, and tool-augmented spatial reasoning systems. The general-purpose VLMs include Qwen3-VL (Qwen Team, 2025), Qwen3.5 (Qwen Team, 2026), InternVL3.5 (Chen et al., 2025b), Molmo (Clark et al., 2026), Gemma 4 (Team, 2026), GLM-4.5V (Team et al., 2026), GPT (OpenAI Team, 2026), and Gemini (Comanici et al., 2025; Google DeepMind, 2026). The spatially trained VLMs include SpaceQwen2.5-VL (Remyx AI, 2025b), SpaceOm (Remyx AI, 2025a), SpaceThinker (Remyx AI, 2025c), and Cosmos-Reason (NVIDIA et al., 2025). The tool-augmented group in-

cludes GCA, pySpatial, and TIGeR. Unless otherwise stated, we use Qwen3-VL-8B-Instruct as the VLM backbone for parsing and semantic scoring and DeepSeek-V4-Flash (DeepSeek-AI, 2026) for program generation. Grounding, reconstruction, and orientation use SAM3 (Carion et al., 2026), VGGT (Wang et al., 2025a), and Orient Anything V2 (OAv2) (Wang et al., 2026). GCA and pySpatial run on the same VLM, code generator, and perception modules, which controls for differences in the principal neural components and leaves the reasoning procedure as the main distinction. For TIGeR, we use its own fine-tuned weights. Appendix I details each tool-augmented method’s configuration. All SATURN numbers are the mean and standard deviation over three runs.

Evaluation Metrics. For SAG, MMSI, and MindCube, we report exact-match MCQ accuracy. For REF, we evaluate grounding using IoU between the predicted and ground-truth bounding boxes, counting a prediction as correct when $\text{IoU} > 0.5$. For REF baselines, all models are prompted to output a bounding box in the same format; malformed or missing boxes are counted as incorrect.

5.2 Perspective-Aware Reasoning

We evaluate how well current models compose spatial relations across frames of reference (RQ1). The three strongest VLMs cluster at about 57.7% overall (Table 1): Qwen3.5-9B, Qwen3.5-35B, and Gemini-3.1-Pro fall within 0.12 percentage points of one another, spanning a small open model, a mid-sized one, and a frontier proprietary system. Within Qwen3-VL, 8B, 32B, and 235B reach 24.31%, 28.81%, and 26.44%, whereas Qwen3.5-9B reaches 57.75%. Similarly, the evaluated spatially trained VLMs are the weakest group and fall

Method	Micro Avg.	SAG	REF	BBox
Random choice	18.80	50.17	1.53	—
Qwen3-VL-8B	24.31	65.91	1.39	99.28
Qwen3-VL-32B	28.81	72.26	4.89	99.81
Qwen3-VL-235B	26.44	73.22	0.67	99.81
Qwen3.5-9B	57.75	74.87	48.32	98.75
Qwen3.5-35B	<u>57.63</u>	76.35	47.32	99.76
InternVL3.5-8B	24.46	53.30	8.57	99.04
InternVL3.5-14B	29.09	52.35	16.28	99.62
InternVL3.5-38B	30.82	58.26	15.71	99.81
Molmo2-4B	21.09	57.65	0.96	88.07
Molmo2-8B	19.89	54.61	0.77	98.61
Gemma4-2B	20.69	52.78	3.02	98.23
Gemma4-4B	21.62	55.74	2.83	98.95
Gemma4-31B	26.25	64.35	5.27	99.66
GPT-5.1	29.80	71.39	6.90	99.43
Gemini-3.1-Pro	57.63	<u>75.30</u>	<u>47.89</u>	91.19
SpaceQwen-3B	13.96	37.74	0.86	72.22
SpaceOm-3B	17.79	49.39	0.38	95.35
SpaceThinker-3B	17.51	48.00	0.72	97.13
Cosmos-Reason1-7B	13.47	37.04	0.48	78.74
GCA	24.24	51.00	9.50	—
pySpatial	<u>44.84</u>	48.35	<u>42.91</u>	—
TIGer	20.38	30.43	14.85	—
SATURN (ours)	82.89$\pm$0.40	85.88$\pm$0.13	81.24$\pm$0.68	—
SATURN _{GT-BBox}	88.85 $\pm$ 0.55	87.57 $\pm$ 0.97	89.56 $\pm$ 0.67	—
SATURN _{Oracle 3D}	95.94 $\pm$ 0.35	94.40 $\pm$ 0.68	96.79 $\pm$ 0.39	—

Table 1: Overall results on 3D FORCE. BBox is the parsing success rate of output boxes. Oracle 3D uses ground-truth geometry. SATURN rows report the mean and standard deviation over three runs.

below random choice on SAG, suggesting that neither model scale nor spatial supervision alone is sufficient for FoR composition.

Tool augmentation does not close the gap either: despite using explicit 3D geometry, the strongest of these systems, pySpatial, reaches only 44.84%. By comparison, SATURN reaches 82.89%, exceeding pySpatial by 38.1 percentage points and the strongest VLM by 25.1 percentage points.

5.3 Frame of Reference Analysis

The aggregate results in Table 1 do not show how performance changes across FoRs. To answer RQ2, we group SAG and REF by the perspective required for reasoning and report the strongest baselines in Figure 4. Every VLM shown performs best in the single-camera setting and declines when reasoning involves an object frame or a second camera frame. Across the five settings, each VLM varies by at least 22 percentage points between its best and worst result, while SATURN varies by about 7 percentage points on SAG and 13 percentage points on REF. For example, Qwen3.5-35B falls from 62% in the single-camera setting to 33% in the object-centric setting, while SATURN falls from 86% to 74%.

The breakdown also distinguishes models with

nearly identical aggregate performance. Qwen3.5-35B and Gemini-3.1-Pro both achieve about 57.6% overall, yet Gemini is 8 percentage points weaker on object-only REF and 12 percentage points stronger on multi-camera REF. Similar overall performance can therefore mask different sensitivities to perspective.

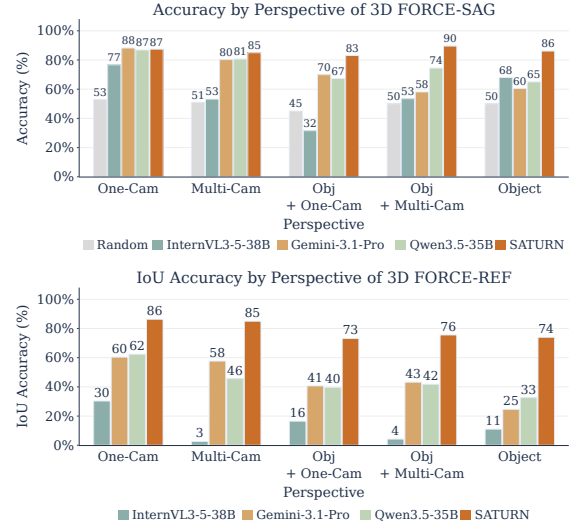


Figure 4: Accuracy on SAG (top) and REF (bottom) across perspective settings for top-performing models.

5.4 Reasoning Complexity Analysis

In addition to FoR, compositional depth and relation topology also affect task difficulty. To answer RQ3, we group REF by the number of reasoning hops. Figure 5 reports performance by hop count, pooled over topologies, while Appendix H.3 separates the results by topology. Every model performs substantially worse at six hops than at zero hops: SATURN falls from 94% to 67%, Qwen3.5-35B from 74% to 29%, Gemini-3.1-Pro from 80% to 33%, and InternVL3.5-38B from 42% to 9%. SATURN shows the smallest decline and outperforms every baseline at each hop count.

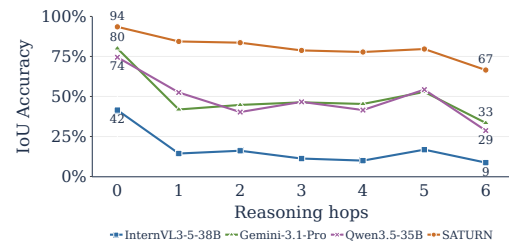


Figure 5: Accuracy on REF by number of reasoning hops, pooled over topologies.

The pooled results are not monotonic because the available topologies change with hop count.

Method	MindCube				MMSI				
	Overall	Rotation	Among	Around	Overall	Positional	Motion	Attribute	MSR
Qwen3-VL-8B-Instruct	38.86	43.50	32.83	49.60	29.40	32.18	27.33	31.54	22.22
Qwen3-VL-235B-Thinking	47.30	87.00	35.00	47.30	32.60	33.70	23.30	40.00	31.80
GLM-4.5V	39.60	60.00	42.20	25.50	33.80	35.60	29.30	36.90	30.30
Gemini-2.5-Pro	57.50	89.50	48.80	54.50	36.90	39.00	33.30	36.20	<u>34.30</u>
SpaceQwen2.5-VL-3B	34.95	25.50	40.20	30.00	27.10	28.93	20.00	30.00	25.76
SpaceOm	41.90	30.50	41.50	52.00	26.60	28.35	20.67	26.15	26.77
SpaceThinker-3B	35.14	30.00	40.00	27.60	27.60	27.78	24.67	30.77	27.27
Cosmos-Reason1-7B	36.67	39.50	34.20	40.40	26.90	28.54	22.67	28.46	24.75
TIGeR	26.95	18.50	27.50	32.40	25.50	28.35	22.00	23.85	21.72
GCA _{235B-Think}	<u>64.20</u>	82.00	59.80	61.80	41.90	<u>46.74</u>	<u>44.00</u>	36.92	30.81
GCA _{8B}	32.19	34.00	34.00	26.40	37.69	40.88	42.18	30.23	30.81
pySpatial	53.14	38.50	50.30	<u>71.60</u>	28.20	30.27	35.33	17.69	24.24
Qwen3VL + scene state	60.86	49.50	<u>60.50</u>	70.80	39.40	41.38	42.67	<u>42.31</u>	29.80
SATURN (ours)	78.06$\pm$1.05	85.67 $\pm$ 2.93	77.00$\pm$1.02	74.53$\pm$2.31	48.77$\pm$0.61	53.19$\pm$0.73	50.44$\pm$1.39	45.38$\pm$2.04	38.05$\pm$0.29

Table 2: Results on MindCube and MMSI. Refer to Appendix K for paired confidence intervals.

Five-hop queries use only the *chain* topology, while six-hop queries use only *hybrid*; the increase at five hops therefore reflects a shift to an easier topology rather than improved performance at greater depth. The topology-specific results in Appendix H.3 support this interpretation: performance remains relatively stable across chain depth, while the VLMs decline as star depth increases. We also vary the number of images and camera frames composed by each query to test other dimensions of input complexity. Neither produces a systematic trend for SATURN, whose performance remains between 76% and 93% on REF as shown in Appendix H.2.

5.5 Soft Symbolic Reasoning Effect

To isolate the contributions of the declarative predicate interface and continuous predicate scores (RQ4), we compare three variants that use the same scenes built from the benchmark’s ground-truth bounding boxes. *Geometric Programming* has no declarative API and asks the backbone LLM to write low-level Python code. *Crisp Predicates* provides the declarative interface but with binary 0 or 1 predicate scores. *Soft Predicates* uses the same symbolic reasoning predicates but with continuous scores. The last two variants therefore differ only in thresholding. *Geometric Programming* reaches 75.43% on REF (Figure 6). Adding the declarative predicate interface raises performance to 79.07%, a gain of 3.6 pp, while retaining continuous scores further raises it to 89.6%, a gain of 10.5 pp. Both components contribute to performance, with the larger gain coming from preserving uncertainty.

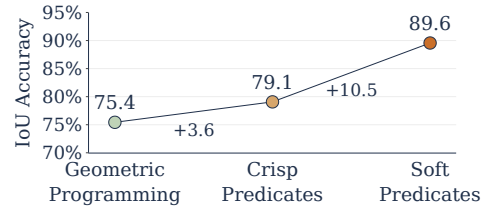


Figure 6: Ablation of symbolic reasoning variants.

5.6 Real-World Multi-View Reasoning

To test whether the gains on 3D FORCE extend to real images (RQ5), we evaluate on the multi-view benchmarks MindCube and MMSI. As shown in Table 2, SATURN achieves the best overall performance on both benchmarks, reaching 78.06% on MindCube and 48.77% on MMSI. On MindCube, SATURN outperforms GCA by 13.9 percentage points in its released configuration and by 45.9 percentage points when GCA uses the same backbone as SATURN. SATURN also achieves the best performance on the *Among* and *Around* categories. On *Rotation*, however, Gemini-2.5-Pro and Qwen3-VL-235B remain stronger. SATURN incorporates textual rotation cues through its pose-constraint module, but this category remains sensitive to errors in constraint extraction and view alignment.

On MMSI, SATURN outperforms all baselines overall and achieves the best performance in the positional, motion, attribute, and multi-step reasoning categories. The scene-state control helps separate the contributions of scene estimation and symbolic composition. Providing SATURN’s estimated scene as text to Qwen3-VL-8B raises performance from 38.86% to 60.86% on MindCube and from 29.40% to 39.40% on MMSI. Compos-

Model	Overall	Rotation	Around	Among
VGGT	79.14 (+1.71)	84.50 (+6.00)	77.20 (+2.40)	78.17 (+0.00)
DA3	75.05 (+0.76)	81.50 (+3.50)	70.40 (+1.20)	74.83 (-0.33)
CUT3R	76.25 (+2.53)	85.00 (+9.00)	72.00 (-0.40)	75.10 (+1.60)

Table 3: Per-backbone single runs on MindCube. Parentheses show the gain from pose constraints.

ing predicates over the same scene further increases performance by 17.2 and 9.4 percentage points, respectively. Thus, scene estimation and symbolic composition each account for roughly half of the improvement over direct VLM inference. These results show that the gains on 3D FORCE extend to real images, while performance on rotation-heavy queries remains sensitive to view alignment.

6 Analysis

6.1 3D Backbone and Pose Constraints

To test whether SATURN’s performance depends on the 3D reconstruction backbone, we evaluate it with VGGT, DA3 (Lin et al., 2025), and CUT3R (Wang et al., 2025b) on MindCube. As shown in Table 3, overall accuracy varies by at most 4.1 percentage points, so its performance is not tied to a single reconstruction backbone. Pose constraints also improve overall performance for every backbone, with the largest gains on Rotation, where accurate view alignment is crucial.

6.2 Error Sources

To identify the main sources of error in SATURN’s end-to-end pipeline, we manually trace errors from one MindCube run to their earliest identifiable source. As shown in Table 4, nearly half occur while connecting the question to the images: the query parser selects the relevant objects and views, while the grounder locates those objects; together, these steps account for 46.6% of failures. Another 23.7% arise from 3D reconstruction estimating object and camera positions and orientations, while VLM-based semantic scoring contributes 18.6%. Only 6.8% come from program generation, indicating that the main bottleneck is supplying the program with the correct objects and geometry. We provide qualitative examples in Appendix L.

6.3 Cost

Table 5 reports end-to-end per-query latency on MindCube. SATURN is slower than pySpatial, direct inference, and TIGeR, but faster than GCA.

Earliest identified source	Share
Visual grounding / detection	33.9%
Reconstruction / camera pose	22.9%
Semantic scoring	18.6%
Query parsing / object selection	12.7%
Program generation	6.8%
Ambiguous reference	4.2%
Object orientation	0.8%

Table 4: Earliest identified source among manually categorized MindCube failures.

Most of SATURN’s cost is due to visual processing and can be amortized across queries related to the same scene; Appendix H.4 breaks the total down by stage. However, additional computation from the visual processing alone is not the only reason for the SATURN’s performance gain. The VLM + scene state baseline using chain of thought prompting is slower end-to-end yet 17.2 percentage points less accurate than SATURN. Moreover, all methods in §5.5 share the same perception tools, yet their accuracy ranges from 75.4% to 89.6%.

Method	Median (s)	Mean (s)
pySpatial	7	7
Qwen3-VL-8B (direct)	12	15
TIGeR	25	28
SATURN	91	105
VLM + scene state	114	174
GCA _{8B}	123	205

Table 5: Per-query end-to-end latency on MindCube.

7 Conclusion

We introduced SATURN, a neuro-symbolic framework for FoR-aware spatial composition, and 3D Frames Of Reference Composition Evaluation, a diagnostic benchmark for this capability. 3D FORCE shows VLMs remain unreliable when composing relations across objects, views, and FoRs. SATURN addresses this limitation by composing soft camera- and object-centric predicates over an estimated 3D scene. It outperforms general-purpose and spatially trained VLMs and tool-augmented systems on 3D FORCE and performs best overall on the real-world multi-view benchmarks MindCube and MMSI. Together, these results identify FoR-aware composition as a distinct spatial reasoning challenge and show the value of explicit spatial predicates with soft truth values.

Limitations

Although SATURN achieves robust performance, it remains a neuro-symbolic framework whose effectiveness is bounded by the capabilities of the underlying neural backbone and the reliability of external tools, which may introduce noisy intermediate predictions. SATURN depends on accurate camera and object orientation estimates; errors in view alignment can propagate directly into derived predicates. 3D FORCE is a diagnostic benchmark for compositional grounding across multiple frames of reference. It is rendered from a controlled simulator, so results on it measure how well a system composes relations under multiple frames of reference. It does not establish generality to open embodied environments. In general, constructing benchmarks that fully capture the intricacies of real-world environments remains a fundamental limitation of this line of work and embodied AI applications. Furthermore, our framework assumes access to substantial GPU resources to support VLM backbones of varying sizes. These computational demands may limit accessibility for researchers with constrained resources.

Acknowledgements

This project is partially supported by the Office of Naval Research (ONR) grant N00014-23-1-2417. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Office of Naval Research.

References

- Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baisihan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, and 19 others. 2026. [SAM 3: Segment anything with concepts](#). In *The Fourteenth International Conference on Learning Representations*.
- Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. 2024. [Spatialvlm: Endowing vision-language models with spatial reasoning capabilities](#). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14455–14465.
- Zeren Chen, Xiaoya Lu, Zhijie Zheng, Pengrui Li, Lehan He, Yijin Zhou, Jing Shao, Bohan Zhuang, and Lu Sheng. 2025a. [Geometrically-constrained agent for spatial reasoning](#). *Preprint*, arXiv:2511.22659.
- Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, Lixin Gu, Xuehui Wang, Qingyun Li, Yimin Ren, Zixuan Chen, Jiapeng Luo, Jiahao Wang, Tan Jiang, Bo Wang, and 23 others. 2025b. [Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling](#). *Preprint*, arXiv:2412.05271.
- An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. 2024. [Spatialrgpt: Grounded spatial reasoning in vision-language models](#). *Advances in Neural Information Processing Systems*, 37:135062–135093.
- Christopher Clark, Jieyu Zhang, Zixian Ma, Jae Sung Park, Mohammadreza Salehi, Rohun Tripathi, Sangho Lee, Zhongzheng Ren, Chris Dongjoo Kim, Yinyu Yang, Vincent Shao, Yue Yang, Weikai Huang, Ziqi Gao, Taira Anderson, Jianrui Zhang, Jitesh Jain, George Stoica, Winson Han, and 2 others. 2026. [Molmo2: Open weights and data for vision-language models with video understanding and grounding](#). *Preprint*, arXiv:2601.10611.
- Gheorghe Comanici, Eric Bieber, Mike Schaeckermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, and Gemini Team. 2025. [Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities](#). *Preprint*, arXiv:2507.06261.
- Erik Daxberger, Nina Wenzel, David Griffiths, Haiming Gang, Justin Lazarow, Gefen Kohavi, Kai Kang, Marcin Eichner, Yinfei Yang, Afshin Dehghan, and Peter Gräsch. 2025. [Mm-spatial: Exploring 3d spatial understanding in multimodal llms](#). *Preprint*, arXiv:2503.13111.
- DeepSeek-AI. 2026. [Deepseek-v4: Towards highly efficient million-token context intelligence](#). *Preprint*, arXiv:2606.19348.
- Mengfei Du, Binhao Wu, Zejun Li, Xuanjing Huang, and Zhongyu Wei. 2024. [Embspatial-bench: Benchmarking spatial understanding for embodied tasks with large vision-language models](#). *Preprint*, arXiv:2406.05756.
- Hossein Rajaby Faghihi, Quan Guo, Andrzej Uszok, Aliakbar Nafar, Elaheh Raisi, and Parisa Kordjamshidi. 2021. [DomiKnowS: a library for integration of symbolic domain knowledge in deep learning](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP), Demo Track*, Online. Association for Computational Linguistics.
- Zhiwen Fan, Jian Zhang, Renjie Li, Junge Zhang, Runjin Chen, Hezhen Hu, Kevin Wang, Huaizhi Qu, Dilin Wang, Zhicheng Yan, and 1 others. 2025.

- Vlm-3r: Vision-language models augmented with instruction-aligned 3d reconstruction. *arXiv preprint arXiv:2505.20279*.
- Zhiyuan Feng, Zhaolu Kang, Qijie Wang, Zhiying Du, Jiongrui Yan, Shubin Shi, Chengbo Yuan, Huizhi Liang, Yu Deng, Qixiu Li, Rushuai Yang, Arctanx An, Leqi Zheng, Weijie Wang, Shawn Chen, Sicheng Xu, Yaobo Liang, Jiaolong Yang, and Baining Guo. 2026. [Seeing across views: Benchmarking spatial reasoning of vision-language models in robotic scenes](#). *Preprint*, arXiv:2510.19400.
- Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A. Smith, Wei-Chiu Ma, and Ranjay Krishna. 2024. [Blink: Multimodal large language models can see but not perceive](#). *Preprint*, arXiv:2404.12390.
- Mohsen Gholami, Ahmad Rezaei, Zhou Weimin, Sitong Mao, Shunbo Zhou, Yong Zhang, and Mohammad Akbari. 2025. [Spatial reasoning with vision-language models in ego-centric multi-view scenes](#). *Preprint*, arXiv:2509.06266.
- Ziyang Gong, Wenhao Li, Oliver Ma, Songyuan Li, Zhaokai Wang, Songyuan Li, Jiayi Ji, Xue Yang, Gen Luo, Junchi Yan, and Rongrong Ji. 2025. [Space-10: A comprehensive benchmark for multimodal large language models in compositional spatial intelligence](#). *Preprint*, arXiv:2506.07966.
- Google DeepMind. 2026. Gemini 3.1 pro model card. <https://deepmind.google/models/model-cards/gemini-3-1-pro/>. Published February 2026.
- Tanmay Gupta and Aniruddha Kembhavi. 2023. [Visual programming: Compositional visual reasoning without training](#). In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14953–14962.
- Yi Han, Enshen Zhou, Shanyu Rong, Jingkun An, Pengwei Wang, Zhongyuan Wang, Cheng Chi, Lu Sheng, and Shanghang Zhang. 2026. [Tiger: Tool-integrated geometric reasoning in vision-language models for robotics](#). *Preprint*, arXiv:2510.07181.
- Joy Hsu, Jiayuan Mao, Josh Tenenbaum, and Jiajun Wu. 2024. What’s left? concept grounding with logic-enhanced foundation models. *Advances in Neural Information Processing Systems*, 36.
- Mengdi Jia, Zekun Qi, Shaochen Zhang, Wenyao Zhang, Xinqiang Yu, Jiawei He, He Wang, and Li Yi. 2025. Omnispacial: Towards comprehensive spatial reasoning benchmark for vision language models. *arXiv preprint arXiv:2506.03135*.
- Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*.
- Danial Kamali, Elham J. Barezi, and Parisa Kordjamshidi. 2025. [Nesycoco: A neuro-symbolic concept composer for compositional generalization](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(4):4184–4193.
- Danial Kamali and Parisa Kordjamshidi. 2023. [Syntax-guided transformers: Elevating compositional generalization and grounding in multimodal environments](#). In *Proceedings of the 1st GenBench Workshop on (Benchmarking) Generalisation in NLP*, pages 130–142, Singapore. Association for Computational Linguistics.
- Danial Kamali and Parisa Kordjamshidi. 2026. [NeP-Tune: A neuro-pythonic framework for tunable compositional reasoning on vision-language](#). In *The Fourteenth International Conference on Learning Representations*.
- Amita Kamath, Jack Hessel, and Kai-Wei Chang. 2023. What’s “up” with vision-language models? investigating their struggle with spatial reasoning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 9161–9175.
- Dingming Li, Hongxing Li, Zixuan Wang, Yuchen Yan, Hang Zhang, Siqi Chen, Guiyang Hou, Shengpei Jiang, Wenqi Zhang, Yongliang Shen, Weiming Lu, and Yueting Zhuang. 2025a. [Viewspatial-bench: Evaluating multi-perspective spatial localization in vision-language models](#). *Preprint*, arXiv:2505.21500.
- Hongxing Li, Dingming Li, Zixuan Wang, Yuchen Yan, Hang Wu, Wenqi Zhang, Yongliang Shen, Weiming Lu, Jun Xiao, and Yueting Zhuang. 2025b. Spatialladder: Progressive training for spatial reasoning in vision-language models. *arXiv preprint arXiv:2510.08531*.
- Yifan Li, Lichi Li, Anh Dao, Xinyu Zhou, Yicheng Qiao, Zheda Mai, Daeun Lee, Zichen Chen, Zhen Tan, Mohit Bansal, and Yu Kong. 2025c. [Industrynav: Exploring spatial reasoning of embodied agents in dynamic industrial navigation](#). *Preprint*, arXiv:2511.17384.
- Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. 2025. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*.
- Jingping Liu, Ziyang Liu, Zhedong Cen, Yan Zhou, Yinan Zou, Weiyan Zhang, Haiyun Jiang, and Tong Ruan. 2025. [Can multimodal large language models understand spatial relations?](#) In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 620–632, Vienna, Austria. Association for Computational Linguistics.
- Zhanpeng Luo, Ce Zhang, Silong Yong, Cunxi Dai, Qianwei Wang, Haoxi Ran, Guanya Shi, Katia Sycara, and Yaqi Xie. 2026. [pyspatial: Generating 3d visual programs for zero-shot spatial reasoning](#). *Preprint*, arXiv:2603.00905.

- Tianyi Ma, Yue Zhang, Zehao Wang, and Parisa Kordjamshidi. 2026. [Breaking down and building up: Mixture of skill-based vision-and-language navigation agents](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13035–13065, San Diego, California, United States. Association for Computational Linguistics.
- Wufei Ma, Haoyu Chen, Guofeng Zhang, Yu-Cheng Chou, Jieneng Chen, Celso M de Melo, and Alan Yuille. 2025a. [3dsrbench: A comprehensive 3d spatial reasoning benchmark](#). *Preprint*, arXiv:2412.07825.
- Wufei Ma, Yu-Cheng Chou, Qihao Liu, Xingrui Wang, Celso M de Melo, Jianwen Xie, and Alan Yuille. 2025b. [Spatialreasoner: Towards explicit and generalizable 3d spatial reasoning](#). In *Advances in Neural Information Processing Systems*.
- Wufei Ma, Luoxin Ye, Celso M de Melo, Alan Yuille, and Jieneng Chen. 2025c. [Spatialllm: A compound 3d-informed design towards spatially-intelligent large multimodal models](#). In *Proceedings of the Computer Vision and Pattern Recognition Conference*.
- Damiano Marsili, Rohun Agrawal, Yisong Yue, and Georgia Gkioxari. 2025. [Visual agentic ai for spatial reasoning with a dynamic api](#). In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19446–19455.
- Roshanak Mirzaee and Parisa Kordjamshidi. 2023. [Disentangling extraction and reasoning in multi-hop spatial reasoning](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3379–3397, Singapore. Association for Computational Linguistics.
- Roshanak Mirzaee, Hossein Rajaby Faghihi, Qiang Ning, and Parisa Kordjamshidi. 2021. [SPARTQA: A textual question answering benchmark for spatial reasoning](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4582–4598, Online. Association for Computational Linguistics.
- Aliakbar Nafar, Chetan Chigurupati, Danial Kamali, Hamid Karimian, and Parisa Kordjamshidi. 2026. [An agentic framework for neuro-symbolic programming](#). *Preprint*, arXiv:2601.00743.
- NVIDIA, Alisson Azzolini, Junjie Bai, Hannah Brandon, Jiaxin Cao, Prithvijit Chattopadhyay, Huayu Chen, Jinju Chu, Yin Cui, Jenna Diamond, Yifan Ding, Liang Feng, Francesco Ferroni, Rama Govindaraju, Jinwei Gu, Siddharth Gururani, Imad El Hanafi, Zekun Hao, Jacob Huffman, and 34 others. 2025. [Cosmos-reason1: From physical common sense to embodied reasoning](#). *Preprint*, arXiv:2503.15558.
- OpenAI Team. 2026. [Openai gpt-5 system card](#). *Preprint*, arXiv:2601.03267.
- Tanawan Premisri and Parisa Kordjamshidi. 2025. [FoR-EST: Frame of reference evaluation in spatial reasoning tasks](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 34977–35003, Suzhou, China. Association for Computational Linguistics.
- Tanawan Premisri and Parisa Kordjamshidi. 2026. [FoR-SALE: Frame of reference-guided spatial adjustment in LLM-based diffusion editing](#). In *Conference on Language Modeling*.
- Qwen Team. 2025. [Qwen3-VL: Sharper vision, deeper thought, broader action](#).
- Qwen Team. 2026. [Qwen3.5: Towards native multi-modal agents](#).
- Navid Rajabi and Jana Kosecka. 2024. [Gsr-bench: A benchmark for grounded spatial reasoning evaluation via multimodal llms](#). *Preprint*, arXiv:2406.13246.
- Shreya Rajpal, Tanawan Premisri, and Parisa Kordjamshidi. 2026. [Spatial reasoning via modality switching between language and symbolic representation](#). *Preprint*, arXiv:2606.31285.
- Arijit Ray, Jiafei Duan, Ellis Brown, Reuben Tan, Dina Bashkirova, Rose Hendrix, Kiana Ehsani, Anirudha Kembhavi, Bryan A. Plummer, Ranjay Krishna, Kuo-Hao Zeng, and Kate Saenko. 2025. [Sat: Dynamic spatial aptitude training for multimodal language models](#). *Preprint*, arXiv:2412.07755.
- Remyx AI. 2025a. SpaceOm. <https://huggingface.co/remyxai/SpaceOm>. Hugging Face model repository; accessed 2026-05-07.
- Remyx AI. 2025b. SpaceQwen2.5-VL-3B-Instruct. <https://huggingface.co/remyxai/SpaceQwen2.5-VL-3B-Instruct>. Hugging Face model repository; accessed 2026-05-07.
- Remyx AI. 2025c. SpaceThinker-Qwen2.5VL-3B. <https://huggingface.co/remyxai/SpaceThinker-Qwen2.5VL-3B>. Hugging Face model repository; accessed 2026-05-07.
- Fatemeh Shiri, Xiao-Yu Guo, Mona Golestan Far, Xin Yu, Gholamreza Haffari, and Yuan-Fang Li. 2024. [An empirical analysis on spatial reasoning capabilities of large multimodal models](#). *Preprint*, arXiv:2411.06048.
- Sanja Sinha, Tanawan Premisri, Danial Kamali, and Parisa Kordjamshidi. 2026. [Neuro-symbolic frameworks: Conceptual characterization and empirical comparative analysis](#). *Neurosymbolic Artificial Intelligence*.
- Tin Stribor Sohn, Maximilian Dillitzer, Jason J. Corso, and Eric Sax. 2025. [Embodied4c: Measuring what matters for embodied vision-language navigation](#). *Preprint*, arXiv:2512.18028.

- Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. Vipergpt: Visual inference via python execution for reasoning.
- Kexian Tang, Junyao Gao, Yanhong Zeng, Haodong Duan, Yanan Sun, Zhening Xing, Wenran Liu, Kaifeng Lyu, and Kai Chen. 2025. Lego-puzzles: How good are mllms at multi-step spatial reasoning? *Preprint*, arXiv:2503.19990.
- Gemma Team. 2026. *Gemma 4 technical report*. *Preprint*, arXiv:2607.02770.
- V Team, Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, Shuaiqi Duan, Weihang Wang, Yan Wang, Yean Cheng, Zehai He, Zhe Su, Zhen Yang, Ziyang Pan, and 74 others. 2026. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *Preprint*, arXiv:2507.01006.
- Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, Ziteng Wang, Rob Fergus, Yann LeCun, and Saining Xie. 2024. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *Preprint*, arXiv:2406.16860.
- Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. 2025a. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306.
- Jiayu Wang, Yifei Ming, Zhenmei Shi, Vibhav Vineet, Xin Wang, Sharon Li, and Neel Joshi. 2024. Is a picture worth a thousand words? delving into spatial reasoning for vision language models. *Advances in Neural Information Processing Systems*, 37:75392–75421.
- Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A. Efros, and Angjoo Kanazawa. 2025b. Continuous 3d perception model with persistent state. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10510–10522.
- Xingrui Wang, Wufei Ma, Tiezheng Zhang, Celso M de Melo, Jieneng Chen, and Alan Yuille. 2025c. Spatial457: A diagnostic benchmark for 6d spatial reasoning of large multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24669–24679.
- Zehan Wang, Ziang Zhang, Jiayang Xu, Jialei Wang, Tianyu Pang, Chao Du, HengShuang Zhao, and Zhou Zhao. 2026. Orient anything v2: Unifying orientation and rotation understanding. *Preprint*, arXiv:2601.05573.
- Yuxi Wei, Wei Huang, Qirui Chen, Lu Hou, and Xiaojuan Qi. 2026. See, remember, explore: A benchmark and baselines for streaming spatial reasoning. *Preprint*, arXiv:2603.23864.
- Diankun Wu, Fangfu Liu, Yi-Hsin Hung, and Yueqi Duan. 2025a. Spatial-MLLM: Boosting MLLM capabilities in visual-based spatial intelligence. In *Advances in Neural Information Processing Systems*.
- Junfei Wu, Jian Guan, Kaituo Feng, Qiang Liu, Shu Wu, Liang Wang, Wei Wu, and Tieniu Tan. 2026. Reinforcing spatial reasoning in vision-language models with interwoven thinking and visual drawing. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Shengguang Wu, Xiaohan Wang, Yuhui Zhang, Hao Zhu, and Serena Yeung-Levy. 2025b. Transductive visual programming: Evolving tool libraries from experience for spatial reasoning. *Preprint*, arXiv:2512.20934.
- Wenrui Xu, Dalin Lyu, Weihang Wang, Jie Feng, Chen Gao, and Yong Li. 2025. Defining and evaluating visual language models’ basic spatial abilities: A perspective from psychometrics. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 11571–11590. Association for Computational Linguistics.
- Kaiyu Yang, Olga Russakovsky, and Jia Deng. 2019. SpatialSense: An adversarially crowdsourced benchmark for spatial relation recognition. *Preprint*, arXiv:1908.02660.
- Sihan Yang, Runsen Xu, Yiman Xie, Sizhe Yang, Mo Li, Jingli Lin, Chenming Zhu, Xiaochen Chen, Haodong Duan, Xiangyu Yue, Dahua Lin, Tai Wang, and Jiangmiao Pang. 2026. Mmsi-bench: A benchmark for multi-image spatial intelligence. In *The Fourteenth International Conference on Learning Representations (ICLR)*.
- Chun-Hsiao Yeh, Chenyu Wang, Shengbang Tong, Ta-Ying Cheng, Ruoyu Wang, Tianzhe Chu, Yuexiang Zhai, Yubei Chen, Shenghua Gao, and Yi Ma. 2025. Seeing from another perspective: Evaluating multi-view understanding in mllms. *Preprint*, arXiv:2504.15280.
- Baiqiao Yin, Qineng Wang, Pingyue Zhang, Jianshu Zhang, Kangrui Wang, Zihan Wang, Jieyu Zhang, Keshigeyan Chandrasegaran, Han Liu, Ranjay Krishna, and et al. 2025. Spatial mental modeling from limited views. In *Structural Priors for Vision Workshop at ICCV’25*.
- Songsong Yu, Yuxin Chen, Hao Ju, Lianjie Jia, Fuxi Zhang, Shaofei Huang, Yuhang Wu, Rundui Cui, Binghao Ran, Zaibin Zhang, and et al. 2025. How far are vlms from visual spatial intelligence? a benchmark-driven perspective. *arXiv preprint arXiv:2509.18905*.

Lotfi A. Zadeh. 1965. Fuzzy sets. *Information and Control*, 8(3):338–353.

Jiahui Zhang, Yurui Chen, Yanpeng Zhou, Yueming Xu, Ze Huang, Jilin Mei, Junhui Chen, Yu-Jie Yuan, Xinyue Cai, Guowei Huang, Xingyue Quan, Hang Xu, and Li Zhang. 2025a. *From flatland to space: Teaching vision-language models to perceive and reason in 3d*. *Preprint*, arXiv:2503.22976.

Wenyu Zhang, Wei En Ng, Lixin Ma, Yuwen Wang, Junqi Zhao, Allison Koenecke, Boyang Li, and Lu Wang. 2025b. *Sphere: Unveiling spatial blind spots in vision-language models through hierarchical evaluation*. *Preprint*, arXiv:2412.12693.

Yuyou Zhang, Radu Corcodel, Chiori Hori, Anoop Cherian, and Ding Zhao. 2026. *Spinbench: Perspective and rotation as a lens on spatial reasoning in vlms*. *Preprint*, arXiv:2509.25390.

Zheyuan Zhang, Fengyuan Hu, Jayjun Lee, Freda Shi, Parisa Kordjamshidi, Joyce Chai, and Ziqiao Ma. 2025c. *Do vision-language models represent space and how? evaluating spatial frame of reference under ambiguities*. In *The Thirteenth International Conference on Learning Representations*.

Yanpeng Zhao, Wentao Ding, Hongtao Li, Baoxiong Jia, and Zilong Zheng. 2026. *Espire: A diagnostic benchmark for embodied spatial reasoning of vision-language models*. *Preprint*, arXiv:2603.13033.

Rui Zhu, Xin Shen, Shuchen Wu, Chenxi Miao, Xin Yu, Yang Li, Weikang Li, Deguo Xia, and Jizhou Huang. 2026. *Video-msr: Benchmarking multi-hop spatial reasoning capabilities of mllms*. *Preprint*, arXiv:2601.09430.

A Ethical Considerations and Artifacts

This work uses and creates scientific artifacts, including vision-language models, 3D perception models, existing spatial reasoning benchmarks, and the proposed 3D FORCE benchmark. Existing artifacts are used for research evaluation and are cited in the relevant sections. 3D FORCE is a synthetic diagnostic benchmark built from rendered scenes, scene graphs, and template-based spatial expressions. It does not contain human subjects, personally identifying information, or real-world images of people. Generated instances were spot-checked for labeling errors and inappropriate content.

3D FORCE is intended for research on compositional spatial reasoning and should not be used as evidence that a system is safe for real-world robotic, navigation, or surveillance deployment. We will release the 3D FORCE dataset for non-commercial research use under the Creative Commons Attribution-NonCommercial 4.0 International license (CC BY-NC 4.0), and the SATURN

code under the Apache License 2.0. The release will include only artifacts for which we have redistribution rights; third-party source assets, models, and benchmarks remain subject to their original licenses and terms of use.

B Details of Anchor-Aware Spatial Predicate Computation

Anchor representation. The spatial engine represents objects, cameras, agents, and virtual viewers through a unified anchor abstraction. Let

$$\mathcal{E} = \{e_1, \dots, e_M\}$$

denote the set of entities available to the symbolic executor. Each entity e_i is associated with an anchor state

$$a_i = (\mathbf{x}_i, \mathbf{R}_i),$$

where $\mathbf{x}_i \in \mathbb{R}^3$ is a 3D position and $\mathbf{R}_i \in \mathbb{R}^{3 \times 3}$ is an orientation in the global coordinate frame. Each anchor state induces a frame of reference (FoR). The selected FoR anchor is denoted by

$$a = (\mathbf{x}_a, \mathbf{R}_a),$$

which may correspond to an object, camera, agent, or virtual viewer constructed during program execution.

For implementation, we use the semantic axes induced by each entity anchor:

$$\mathbf{r}_i = \mathbf{R}_i \mathbf{e}_{\text{right}}, \quad \mathbf{u}_i = \mathbf{R}_i \mathbf{e}_{\text{up}}, \quad \mathbf{q}_i = \mathbf{R}_i \mathbf{e}_{\text{front}},$$

where $\mathbf{r}_i$, $\mathbf{u}_i$, and $\mathbf{q}_i$ denote the right, up, and forward directions of entity e_i ’s anchor state. All object, camera, agent, and virtual-viewer anchors are converted into this semantic-axis representation before predicate computation, avoiding ambiguity from different native camera or object coordinate conventions.

Anchor-conditioned predicates. For a selected FoR anchor a and spatial relation r , the spatial engine produces an anchor-conditioned predicate matrix

$$S_r^a \in [0, 1]^{|\mathcal{E}| \times |\mathcal{E}|}.$$

Each entry $S_r^a[i, j]$ scores whether entity e_i is in relation r to entity e_j , evaluated in the FoR induced by anchor a .

Object-centric predicates are a special case in which the reference object supplies the FoR anchor. When e_j is an oriented object, we define

$$S_r^{\text{obj}}[i, j] \equiv S_r^{a_j}[i, j].$$

Thus, $S_r^{\text{obj}}[i, j]$ evaluates the relation between e_i and e_j from e_j 's own perspective.

General soft predicate scoring. For an anchor-conditioned relation, the target-reference displacement between entities e_i and e_j is expressed in the FoR induced by anchor a . Assuming $\mathbf{R}_a$ maps coordinates from anchor a 's local frame to the global frame, we compute

$$\Delta_{ij}^a = \mathbf{R}_a^\top (\mathbf{x}_i - \mathbf{x}_j).$$

We also express the local orientations of the two entity anchors in the same FoR:

$$\mathbf{R}_i^a = \mathbf{R}_a^\top \mathbf{R}_i, \quad \mathbf{R}_j^a = \mathbf{R}_a^\top \mathbf{R}_j.$$

Here, Δ_{ij}^a is the displacement from reference entity e_j to target entity e_i , expressed in anchor a 's local coordinate frame. The matrices $\mathbf{R}_i^a$ and $\mathbf{R}_j^a$ are the orientations of entities e_i and e_j , also expressed in anchor a 's FoR.

Each spatial relation is implemented through a relation-specific evidence function

$$h_r(\Delta_{ij}^a, \mathbf{R}_i^a, \mathbf{R}_j^a),$$

which is mapped to a soft predicate score:

$$S_r^a[i, j] = \sigma \left(\frac{h_r(\Delta_{ij}^a, \mathbf{R}_i^a, \mathbf{R}_j^a) - m_r}{\tau_r} \right).$$

Here, m_r is a relation-specific margin and τ_r controls the softness of the predicate. This formulation covers directional and orientation-based predicates using the same interface.

Scene-scale normalization. To make positional margins comparable across scenes, we normalize spatial evidence by a robust scene scale. Let

$$s_{\text{scene}} = \text{quantile}_{0.9} \{ \|\mathbf{x}_p - \mathbf{x}_q\|_2 : p < q \},$$

with a small constant ε for numerical stability. The normalized displacement is

$$\bar{\Delta}_{ij}^a = \frac{\Delta_{ij}^a}{s_{\text{scene}} + \varepsilon}.$$

Pairwise directional predicates use $\bar{\Delta}_{ij}^a$; egocentric direction labels use bearing evidence that depends only on the direction of the displacement and is therefore scale-invariant by construction; orientation predicates use dot-product evidence between anchor forward axes and therefore do not require scene-scale normalization.

Predicate type	Margin	Temperature
Directional	$m_{\text{dir}} = 0.0$	$\tau_{\text{dir}} \approx 0.0714$
Directional combination	$m_{\text{comb}} = 0.0$	$\tau_{\text{comb}} \approx 0.0714$
Direction label	$m_d = 0.0$	$\tau_d = 1$
Parallel	$m_{\parallel} = 0.70$	$\tau_{\text{ori}} \approx 0.0714$
Perpendicular	$m_{\perp} = 0.30$	$\tau_{\text{ori}} \approx 0.0714$

Table 6: Default hyperparameters for soft spatial predicates. Directional predicates project the local displacement onto the anchor's semantic axes; orientation predicates use dot-product evidence between anchor forward vectors. Direction labels use bearing-alignment evidence $h_d = 2 \ln \cot(\delta_d/2)$.

Default hyperparameters. We use a small set of fixed hyperparameters across all experiments. These values are not tuned per benchmark or per task. For directional predicates, we use

$$m_{\text{dir}} = 0.0, \quad \tau_{\text{dir}} \approx 0.0714.$$

For directional-combination predicates, we use

$$m_{\text{comb}} = 0.0, \quad \tau_{\text{comb}} \approx 0.0714.$$

For egocentric direction labels, $m_d = 0$ and $\tau_d = 1$; the bearing evidence carries its own scale. For orientation-based predicates, we use

$$m_{\parallel} = 0.70, \quad m_{\perp} = 0.30, \quad \tau_{\text{ori}} \approx 0.0714.$$

These defaults are summarized in Table 6 and are fixed across all experiments; we do not tune them per benchmark.

Directional predicates. Directional relations are computed from signed projections of the normalized displacement $\bar{\Delta}_{ij}^a$ onto the local axes of the selected FoR anchor a . Because $\bar{\Delta}_{ij}^a$ is already expressed in anchor a 's local coordinate frame, we use the canonical semantic axes. In a Y-up right-handed coordinate system, we define

$$h_{\text{right}}(\bar{\Delta}_{ij}^a) = \langle \bar{\Delta}_{ij}^a, \mathbf{e}_{\text{right}} \rangle,$$

$$h_{\text{left}}(\bar{\Delta}_{ij}^a) = -\langle \bar{\Delta}_{ij}^a, \mathbf{e}_{\text{right}} \rangle,$$

$$h_{\text{above}}(\bar{\Delta}_{ij}^a) = \langle \bar{\Delta}_{ij}^a, \mathbf{e}_{\text{up}} \rangle,$$

$$h_{\text{below}}(\bar{\Delta}_{ij}^a) = -\langle \bar{\Delta}_{ij}^a, \mathbf{e}_{\text{up}} \rangle,$$

$$h_{\text{front}}(\bar{\Delta}_{ij}^a) = -\langle \bar{\Delta}_{ij}^a, \mathbf{e}_{\text{front}} \rangle,$$

$$h_{\text{behind}}(\bar{\Delta}_{ij}^a) = \langle \bar{\Delta}_{ij}^a, \mathbf{e}_{\text{front}} \rangle.$$

Here FRONT follows occluder semantics: “ e_i is in front of e_j ” means e_i lies nearer the observer along

the anchor’s viewing axis. The corresponding soft predicate is

$$S_r^a[i, j] = \sigma \left(\frac{h_r(\bar{\Delta}_{ij}^a) - m_{\text{dir}}}{\tau_{\text{dir}}} \right).$$

Directional combinations such as FRONT-LEFT and BEHIND-RIGHT are computed from the same local displacement vector. For a relation r composed of component directions $\mathcal{C}(r)$, we define

$$h_r(\bar{\Delta}_{ij}^a) = \min_{c \in \mathcal{C}(r)} h_c(\bar{\Delta}_{ij}^a),$$

and score it as

$$S_r^a[i, j] = \sigma \left(\frac{h_r(\bar{\Delta}_{ij}^a) - m_{\text{comb}}}{\tau_{\text{comb}}} \right).$$

This implements combined directions as soft geometric conjunctions without requiring additional tool calls or query-specific geometric programs.

Egocentric direction labels. In addition to the pairwise axial predicates above, SATURN exposes egocentric direction *labels* that classify where a target lies relative to an anchor: eight horizontal directions with target bearings $\theta_d \in \{0^\circ, \pm 45^\circ, \pm 90^\circ, \pm 135^\circ, 180^\circ\}$, plus ABOVE and BELOW. Their evidence is bearing alignment: let δ_d be the angle between the displacement Δ_{ij}^a — projected onto the anchor’s horizontal plane for horizontal labels — and the label’s target direction. The evidence and hyperparameters are

$$h_d = 2 \ln \cot(\delta_d/2), \quad m_d = 0, \quad \tau_d = 1,$$

which under the general scoring interface reduces to the closed form

$$S_d^a = \sigma(h_d) = \cos^2(\delta_d/2) = \frac{1}{2}(1 + \cos \delta_d);$$

the implementation computes this closed form directly. Because δ_d depends only on the direction of the displacement, label scores are scale-invariant and require no scene-scale normalization, and diagonal labels are ordinary labels at $\pm 45^\circ$ bearings rather than conjunctions. $\arg \max_d S_d^a$ is exactly nearest-direction classification, so the label family acts as a soft direction classifier.

Orientation-based predicates. Orientation predicates compare local orientation axes of entity anchors. Let

$$\mathbf{q}_i^a = \mathbf{R}_a^\top \mathbf{R}_i \mathbf{e}_{\text{front}}, \quad \mathbf{q}_j^a = \mathbf{R}_a^\top \mathbf{R}_j \mathbf{e}_{\text{front}}$$

denote the forward axes of entities e_i and e_j , expressed in the FoR induced by anchor a .

A parallel predicate is scored as

$$S_{\text{parallel}}^a[i, j] = \sigma \left(\frac{|\langle \mathbf{q}_i^a, \mathbf{q}_j^a \rangle| - m_{\parallel}}{\tau_{\text{ori}}} \right).$$

A perpendicular predicate is scored as

$$S_{\text{perpendicular}}^a[i, j] = \sigma \left(\frac{m_{\perp} - |\langle \mathbf{q}_i^a, \mathbf{q}_j^a \rangle|}{\tau_{\text{ori}}} \right).$$

A fixed-angle predicate with target angle θ^* is scored using dot products:

$$S_{\theta^*}^a[i, j] = \sigma \left(\frac{m_{\theta} - |(\mathbf{q}_i^a)^\top \mathbf{q}_j^a - \cos \theta^*|}{\tau_{\text{ori}}} \right).$$

This implements relations such as PARALLEL, PERPENDICULAR, and fixed-angle predicates such as 60° , without explicitly computing angles; m_{θ} is the margin of the instantiated relation, $m_{\parallel}$ or $m_{\perp}$ in Table 6. If signed orientation is needed, the predicate can instead use cross-product evidence.

Constructed anchors. In addition to object- and camera-induced anchors, SATURN supports virtual anchors instantiated at arbitrary 3D positions and orientations. Existing anchors can also be transformed through rotation and translation. A constructed anchor supplies a frame of reference to the same predicate interface without enlarging the entity set $\mathcal{E}$, allowing the symbolic program to evaluate spatial relations from arbitrary viewpoints.

Overall, the spatial engine is predicate-extensible: any relation that can be expressed as a soft evidence function over estimated anchor states can be added as an h_r . Our current instantiation covers directional, directional-combination, and orientation-based predicates.

C Symbolic Composition Functions

A component of SATURN is the symbolic executor, which runs the LLM-generated program. SATURN uses the hybrid soft-execution model from prior Pythonic neuro-symbolic reasoning, adapting it to compose both semantic grounding scores and geometry-derived spatial predicate scores.

Soft Compositional Reasoning. To reason about visual concepts themselves, we employ a set of soft logical operations based on fuzzy logic principles similar to (Hsu et al., 2024; Kamali et al., 2025; Kamali and Kordjamshidi, 2026). Instead of operating

on binary true/false values, these operations work directly on the uncertainty scores obtained from the grounding interface and the spatial engine. This is implemented through custom data structures that encapsulate scores for a given predicate and overload standard Python operators, such as $\&$ for AND, $|$ for OR, and not for negation, to perform the corresponding logical operations. Details of these operations are shown in Table 7. The table lists both a differentiable form and the form the executor actually evaluates; since SATURN performs inference only, we use the non-differentiable executed forms. For example, when the program executes $\alpha_x \& \alpha_y$, our framework takes the element-wise minimum of the two corresponding score vectors, implementing the fuzzy t-norm for conjunction. Similarly, relational composition between an object-centric score vector α_x and a relational score matrix β_{xy} preserves uncertainty while propagating object-level evidence through spatial relations.

Imperative Reasoning. Our symbolic executor leverages a standard Python interpreter to handle the program’s overall structure and control flow. This is possible because we define iteration and Boolean operations on concept objects, allowing the generated programs to express complex procedural logic, including conditionals, loops, and variable assignments. As a result, the framework retains the expressive power of a general-purpose programming language while allowing the underlying predicates to remain uncertainty-aware. Some operations, such as counting over string sets, may break the computation graph, but this hybrid design enables the system to reason flexibly under uncertainty while still supporting the program structures needed for compositional spatial reasoning.

D Comparison to Existing Spatial Reasoning Benchmarks

Existing spatial reasoning benchmarks vary along several orthogonal axes: whether they evaluate multi-view inputs, whether they support both camera-centric and object-centric frames of reference (FoRs), whether they allow a single query to mix FoRs, the type of compositionality they evaluate, and whether they control multi-hop depth and topology. Table 8 summarizes these axes for the most closely related benchmarks. 3D FORCE is the only benchmark in this set that explicitly controls all four axes simultaneously: multi-view, FoR mixing within a single query, relation-level compo-

sitionality, and controlled multi-hop depth.

E Benchmark Construction and Validation

E.1 Image Rendering Setup

Our benchmark generation pipeline uses Blender3D as the simulation environment for image rendering. We include 50 diverse backgrounds to increase visual variety across scenes. The scene generation process contains 12 object categories: airliner, biplane, double bus, fighter plane, minivan, school bus, scooter motorcycle, sedan, SUV, truck, horse, and tank. Each object is assigned one of five colors—blue, brown, green, red, and yellow—with assignments chosen to preserve visual realism. For example, horses are always assigned the color brown rather than arbitrary colors from the palette.

E.2 Image and Scene Graph Generation

To render the underlying scenes, we use the image-generation pipeline of Spatial457 (Wang et al., 2025c) with several modifications, including higher-resolution images, less ambiguous object categories and names, and a revised color palette.

For the standard multi-view setting, we render additional views by placing cameras at approximately uniform azimuth intervals around the scene. For two, three, and four views, cameras are separated by roughly 180° , 120° , and 90° , respectively. This produces controlled viewpoint variation while keeping the query-relevant objects visible across the views. For the partial-view setting, camera positions and orientations are randomized to create incomplete individual observations. We constrain this randomization so that every pair of views shares at least one visible object, and the union of all views contains all objects required by the query. Thus, partial-view examples require integrating information across images rather than solving the query from a single complete view. Each scene contains 6-12 objects.

Scene graphs are then constructed using each object’s 3D world coordinates (X, Y, Z), 2D image coordinates (X', Y'), and pose direction derived from 3D orientation. Based on the resulting scene graph, we render additional views to create multi-image subsets of 2, 3, and 4 images, with the visibility constraints of the standard and partial-view settings described above.

Syntax	Logical Form	Description	Differentiable Form	Inference
$\alpha_x.\text{exists}()$	$\exists x \alpha_x$	Existential quantification	$\max(\alpha_x)$	$\max(\alpha_x)$
$\alpha_x.\text{forall}()$	$\forall x \alpha_x$	Universal quantification	$\min(\alpha_x)$	$\mathbf{1}[\min_x \alpha_x > 0.5]$
$\alpha_x \& \alpha_y$	$\alpha_x \wedge \alpha_y$	Logical conjunction	$\min(\alpha_x, \alpha_y)$	$\min(\alpha_x, \alpha_y)$
$\alpha_x \& \beta_{xy}$	$\alpha_x \wedge \beta_{xy}$	Relational conjunction	$\min(\alpha_x, \beta_{xy})$	$\min(\alpha_x, \beta_{xy})$
$\alpha_x \alpha_y$	$\alpha_x \vee \alpha_y$	Logical disjunction	$\max(\alpha_x, \alpha_y)$	$\max(\alpha_x, \alpha_y)$
$\alpha_x.\text{implies}(\alpha_y)$	$\alpha_x \rightarrow \alpha_y$	Logical implication	$\max(1 - \alpha_x, \alpha_y)$	$\max(1 - \alpha_x, \alpha_y)$
$\text{not } \alpha_x$	$\neg \alpha_x$	Logical negation	$1 - \alpha_x$	$1 - \alpha_x$
$\alpha_x.\text{iota}(\text{var})$	$\iota(\text{var}, \alpha)$	Best-match	$\frac{\alpha - \min \alpha}{\max \alpha - \min \alpha + \varepsilon}$	$\frac{\alpha - \min \alpha}{\max \alpha - \min \alpha + \varepsilon}$
$\alpha_x.\text{count}()$	$\text{count}(\alpha_x)$	Count	$\sum_x \alpha_x$	$\sum_x \mathbf{1}[\alpha_x \geq 0.5]$
$s_1 == s_2$	$s_1 = s_2$	Equality	$\sigma\left(\frac{2\gamma - s_1 - s_2 }{2\gamma\tau}\right)$	$\sigma\left(\frac{2\gamma - s_1 - s_2 }{2\gamma\tau}\right)$
$s_1 > s_2$	$s_1 > s_2$	Inequality	$\sigma\left(\frac{s_1 - s_2 + \gamma}{\tau}\right)$	$\mathbf{1}[s_1 > s_2]$

Table 7: Soft compositional operators used in symbolic execution. Here, α represents an object-centric 1D score vector, β relational kD ($k \geq 2$) score matrix, $\tau = 0.25$ is a temperature parameter, $\gamma = 0.25$ is a margin.

E.3 Benchmark Instance Generation

3D FORCE-SAG To generate 3D FORCE-SAG, we use a simulation environment in which a set of images, along with their corresponding scene graphs, is rendered. For each image, we randomly select k objects from the scene graph to generate object descriptions, T , and sample l object pairs from k to generate the spatial relation descriptions R . The object and relation descriptions are generated based on a set of sentence templates. The number of hops of reasoning is limited to a maximum of five. These instances constitute the set of positive examples in our benchmark. To generate negative examples, we mutate one or more relations in the relation set by reversing their directions (e.g., "A to the left of B" is changed to "B is to the left of A"). For each image, the descriptions are generated multiple times under different camera perspectives to support multi-view evaluation. Before generation, each instance is randomly assigned to either the positive or negative set, which determines whether the generated spatial configuration is satisfiable. For each instance, the maximum number of objects and spatial relations is 5. In total, we generate 1,150 questions, including 972 standard multi-view questions and 178 partial-view questions. Detailed statistics are provided in Table 9 and 10.

3D FORCE-REF To generate 3D FORCE-REF instances, we first construct a random topology graph for each instance to encode multi-hop reasoning, where nodes represent entities and edges represent spatial relations. We consider three topological structures, chain, star, and hybrid, as illustrated in Figure 3. Next, we identify objects and spatial relations that align with the constructed

topology using a brute-force search over all feasible object–relation combinations in the rendered scene graph. We enforce that the full relational chain is necessary to identify the target, such that any partial subset of relations is insufficient, and ensure that each instance contains a unique solution. The number of hops of reasoning is limited to a maximum of six. After selecting the objects and relations, we generate referring expressions using predefined spatial-relation templates conditioned on the topology graph. Similar to 3D FORCE-SAG, the descriptions are generated multiple times under different camera perspectives to support multi-view evaluation. Separate parameter configurations are defined for each topology structure used to encode multi-hop reasoning. For the chain structure, we allow up to 5 reasoning hops. For the star structure, the number of hops is limited to 2–4 to ensure that the complete query is required to identify the target object, while any partial query remains insufficient. For the hybrid structure, we generate only 4-hop and 6-hop instances, as the hybrid topology requires an even number of reasoning hops for valid construction. In total, we generate 2,088 questions for 3D FORCE-REF. Detailed statistics are provided in Table 9 and 10. We do not define train/dev/test splits because 3D FORCE is intended as an evaluation-only diagnostic benchmark for zero-shot compositional spatial reasoning.

E.4 Examples

We provide the examples of 3D FORCE, including 3D FORCE-SAG and 3D FORCE-REF in Figure 7.

Benchmark	Multi-view	FoR		FoR Mixing	Compositionality	Multi-hop	
		Camera	Object			Chains	Controlled depth
CLEVR (Johnson et al., 2017)	✗	✓	✗	✗	Relation	✓	✓
BLINK (Fu et al., 2024)	✓	✓	✗	✗	Perception	✗	✗
SpaCE-10 (Gong et al., 2025)	✗	✓	✗	✗	Skill	✗	✗
3DSRBench (Ma et al., 2025a)	✗	✓	✓	✗	3D skill	✗	✗
SPHERE (Zhang et al., 2025b)	✗	✓	✓	✗	Skill	✗	✗
ViewSpatial (Li et al., 2025a)	✗	✓	✓	✗	Perspective	✗	✗
Spatial457 (Wang et al., 2025c)	✗	✓	✓	✗	6D relation	✗	✗
FoREST [†] (Premisri and Kordjamshidi, 2025)	✗	✓	✓	✗	Perspective	✗	✗
MMSI-Bench (Yang et al., 2026)	✓	✓	✓	✗	Perspective	✓	✗
OmniSpatial (Jia et al., 2025)	✗	✓	✓	✗	Skill	✗	✗
MindCube (Yin et al., 2025)	✓	✓	✓	✗	Mental-model	✗	✗
SPINBENCH (Zhang et al., 2026)	✓	✓	✓	✗	Perspective	✗	✗
COMFORT (Zhang et al., 2025c)	✗	✓	✓	✗	Perspective / FoR	✗	✗
Spatial-MM (Shiri et al., 2024)	✗	✓	✓	✗	Relation	✓	✗
SPAR-Bench (Zhang et al., 2025a)	✓	✓	✗	✗	Skill	✗	✗
Ego3D-Bench (Gholami et al., 2025)	✓	✓	✗	✗	3D skill	✗	✗
MV-RoboBench (Feng et al., 2026)	✓	✓	✗	✗	Skill	✗	✗
3D FORCE (ours)	✓	✓	✓	✓	Relation	✓	✓

Table 8: Comparison of spatial reasoning benchmarks across evaluation capabilities. ✓ marks a capability as an explicitly evaluated axis; ✗ marks absence or only incidental support. **Multi-view**: whether the benchmark evaluates multiple images, views, or viewpoint changes together as part of the task design. **FoR**: whether questions are anchored to the camera/viewer frame or to an oriented object’s intrinsic frame. **FoR Mixing**: whether a single query composes relations stated in different reference frames. **Compositionality Type**: *Relation* chains atomic spatial relations between specific objects; *Skill* composes heterogeneous capabilities such as recognition, localization, counting, or functional reasoning; *3D skill* evaluates 3D spatial capabilities such as height, location, orientation, and multi-object reasoning; *6D relation* evaluates spatial relations involving 3D position and orientation; *Mental-model* requires constructing an internal scene representation across partial views; *Perspective* evaluates viewpoint or frame-of-reference transformations; *Perception* evaluates low-level visual perception skills. **Multi-hop / Chains**: questions chain ≥ 2 atomic spatial relations between specific objects. **Multi-hop / Controlled depth**: hop count and topology are explicit, controlled axes. [†]FoREST is primarily a text/LLM frame-of-reference benchmark.

Benchmark	0-Hop	1-Hop	2-Hop	3-Hop	4-Hop	5-Hop	6-Hop
3D FORCE-SAG	150	158	199	223	204	216	-
3D FORCE-REF	200	320	360	356	342	280	230

Table 9: Distribution of reasoning hops in 3D FORCE-SAG and 3D FORCE-REF. 3D FORCE-REF additionally includes 6-hop reasoning instances.

F Text-Stated Camera-Pose Constraint Extraction

Some multi-view inputs contain explicit natural-language facts about the camera setup. For example, a question may state that one image was captured after rotating the camera 90° from another image, or that several images were captured from the same physical position. These statements are not target spatial relations to be inferred from the scene; they are auxiliary geometric facts about the camera poses. We therefore extract them before code generation and apply them to the reconstructed camera states used by the spatial engine.

Constraint extraction. Given the query Q , the extractor returns a set of structured pose constraints. We use a lightweight keyword filter to avoid calling the extractor when Q does not mention camera-

pose information. When the filter fires, a VLM-based extractor classifies candidate clauses as either stated camera facts or hypothetical observer motions. Only stated camera facts are extracted. For instance, “image 2 was taken after rotating the camera 90° from image 1” is extracted, while “if I stand at image 2 and turn right” is treated as an imagined observer motion and ignored.

The current implementation supports two constraint types:

$$\mathcal{K} = \mathcal{K}_r \cup \mathcal{K}_p.$$

A rotation constraint $(u, v, \theta) \in \mathcal{K}_r$ states that camera v ’s orientation is related to camera u ’s orientation by a yaw angle θ . A same-position constraint $S \in \mathcal{K}_p$ states that all cameras in S share the same world-frame translation. Extracted constraints are schema-validated before application; records with

Setting	3D FORCE-SAG	3D FORCE-REF
Single-image	313	524
Multi-image	285	631
Object-centric	200	435
Object-centric + Single-image	228	154
Object-centric + Multi-image	124	344
1 View	232	436
2 Views	228	463
3 Views	267	548
4 Views	423	641

Table 10: Distribution of perspective settings and number of views in 3D FORCE-SAG and 3D FORCE-REF.

invalid camera indices or malformed yaw values are discarded.

Rotation propagation. Let $\Gamma^0 = \{(\mathbf{x}_v^0, \mathbf{R}_v^0)\}_{v=1}^V$ be the camera states produced by image-based reconstruction. SATURN selects a single anchor camera a : the lowest-indexed camera appearing in any rotation constraint. Yaw offsets are composed along the constraint graph by a breadth-first traversal from a . The anchor rotation is initialized from $\mathbf{R}_a^0$.

When multiple constrained cameras provide evidence for the anchor, SATURN can refine the anchor by back-projecting each camera’s image-derived rotation to an implied anchor rotation. For a camera v connected to a , let $\theta_{a \rightarrow v}$ be the composed yaw along the path from a to v . The implied anchor rotation from camera v is

$$\hat{\mathbf{R}}_a^{(v)} = \mathbf{R}_{\text{yaw}}(-\theta_{a \rightarrow v})\mathbf{R}_v^0.$$

The anchor rotation $\tilde{\mathbf{R}}_a$ is obtained by averaging these implied rotations on $\text{SO}(3)$ with uniform weights. If this averaging step is disabled, $\tilde{\mathbf{R}}_a = \mathbf{R}_a^0$.

The remaining cameras reachable from the anchor are then assigned by forward propagation:

$$\tilde{\mathbf{R}}_v = \mathbf{R}_{\text{yaw}}(\theta_{a \rightarrow v})\tilde{\mathbf{R}}_a.$$

Cameras not reachable from any rotation constraint keep their original image-derived orientations.

Same-position constraints. Same-position constraints modify translations but not rotations. For each group $S \in \mathcal{K}_p$, SATURN moves every camera in the group onto the position of its lowest-indexed member. Since spatial predicates depend only on relative positions, the choice of the shared location does not change predicate scores; using the lowest-indexed member keeps camera 0 at the origin of the reconstruction frame. Cameras not included in

any same-position group keep their original translations.

Output. The output is a refined camera-state set

$$\tilde{\Gamma} = \{(\tilde{\mathbf{x}}_v, \tilde{\mathbf{R}}_v)\}_{v=1}^V.$$

These refined camera states replace the initial camera states in the scene representation, so all downstream spatial predicates and generated programs operate on the corrected camera poses. This procedure applies text-stated constraints as hard geometric facts rather than as soft penalties; there is no joint optimization objective or loss-weight tradeoff.

G Experimental Details

G.1 External Benchmarks

G.1.1 MindCube

Throughout the main paper we refer to this evaluation set simply as MindCube. The specific subset is the 1K split of Yin et al. (2025)’s benchmark (also referred to as MindCube-tiny in the original release), which evaluates spatial mental modeling from limited visual observations. The benchmark tests whether a model can construct and manipulate a 3D representation of a scene from a small set of 2D images. We use MindCube as an external multi-view benchmark to evaluate whether SATURN transfers beyond our controlled 3D FORCE-REF and 3D FORCE-SAG benchmarks.

MindCube contains three primary sub-tasks. **Rotation** evaluates whether a model can infer the complete environment from partial visual information and maintain consistency across sequential views. **Around** evaluates whether a model can infer the scene from a novel viewpoint, requiring interpolation and extrapolation over an implicit 3D scene representation. **Among** evaluates whether a model can reason about 3D spatial arrangements

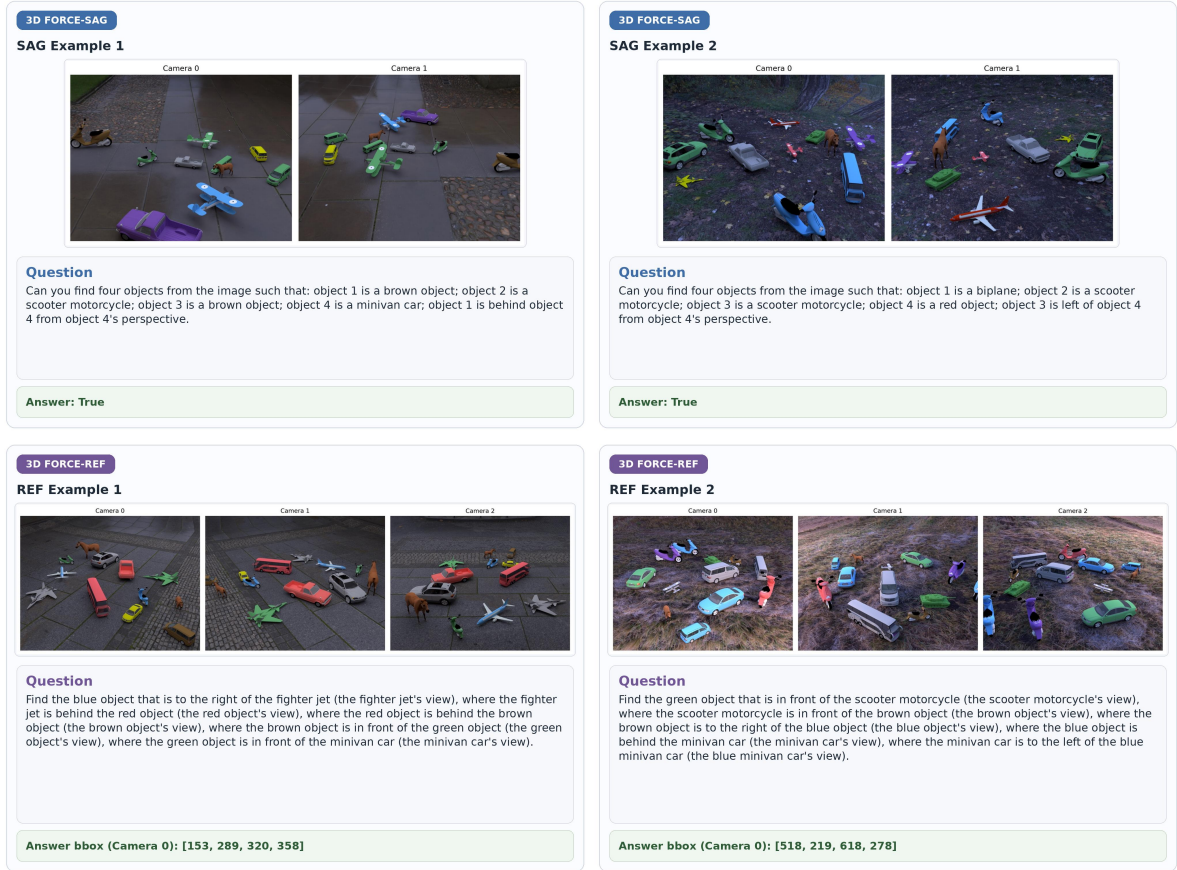


Figure 7: Examples from 3D FORCE-SAG (top) and 3D FORCE-REF (bottom), showing two examples from each benchmark together with the corresponding ground-truth answers.

from four orthogonal views with substantial occlusion, requiring consistency across perspectives and relative-position reasoning over partially hidden objects.

G.1.2 MMSI-Bench

MMSI-Bench (Yang et al., 2026) evaluates multi-image spatial reasoning, where models must integrate information across multiple views rather than answer from a single image. We use MMSI as a real-world transfer benchmark to test whether SATURN’s explicit spatial predicates and symbolic composition remain useful outside our controlled 3D FORCE setting. Following the original benchmark, we report performance on four subcategories: *Positional Relationships*, which evaluates relative positions among objects, cameras, and semantic regions across views; *Motion*, which tests reasoning about object or camera movement; *Attribution*, which evaluates spatially relevant object properties such as shape, size, or length; and *Multi-Step Reasoning*, which requires chaining multiple spatial cues to answer a query. Together, these categories

provide a complementary real-world evaluation of multi-view spatial understanding and compositional reasoning.

G.2 Implementation Details

SATURN is evaluated without task-specific training or fine-tuning. The program generator is provided with API documentation and in-context examples; therefore, the setting is zero-shot with respect to benchmark supervision but not prompt-free. We use Qwen3-VL-8B-Instruct (Qwen Team, 2025) for visual interpretation and semantic scoring, and DeepSeek-V4-Flash (DeepSeek-AI, 2026) in non-thinking mode for Python program generation. Unless otherwise stated, program generation uses greedy decoding with temperature 0 and a maximum output length of 16,000 tokens.

For perception, we use SAM3 (Carion et al., 2026) for segmentation, VGGT (Wang et al., 2025a) as the default depth and 3D reconstruction module, and OAv2 (Wang et al., 2026) for object orientation estimation. VGGT is used as the default geometry backbone to keep the comparison

with prior 3D visual-programming methods fair. The spatial engine converts the resulting object and camera states into soft perspective-aware predicate tensors, which are composed by the Pythonic symbolic executor.

All SATURN variants use the same perception modules, program-generation setup, and symbolic executor unless explicitly stated. Spatial predicate margins, temperatures, and symbolic composition operators are fixed across benchmarks and are not tuned per dataset. For hard symbolic variants, predicate scores are thresholded at 0.5. For experiments reporting mean and standard deviation, we run SATURN three times and report mean $\pm$ standard deviation. For paired comparisons between two variants evaluated on the same items, we report the difference in accuracy with a 95% confidence interval from a bootstrap over items with 10,000 resamples, and a p -value from an exact McNemar test on the discordant pairs. Pairing on items removes between-item variance, so these intervals are tighter than the run-to-run standard deviations above and answer a different question: whether one variant beats the other on the same questions, not how much either varies across runs. Variation across runs comes from nondeterminism in perception and model-serving backends; decoding settings are kept fixed.

We use vLLM 0.19.1 for VLM inference and Ray for scalable serving of tool calls. The main SATURN pipeline runs on four GPUs in our implementation: 2 GPUs for VLM inference and 2 GPUs for perception/tool execution. For throughput, the full experimental sweep was run on an 8-H100 node. The per-query cost in Table 5 was measured on A6000 GPUs. The total compute budget for each MindCube and MMSI is approximately 24 GPU-hours, including perception, VLM inference, and program generation. For proprietary or API-based baselines, model sizes and infrastructure are not disclosed by the providers; for open-weight models, we report the public model names and parameter scales in the main experiments.

H Additional Results

H.1 Backbone VLM and Reasoning Mode

Backbone VLM. Figure 8 compares three ways of using the same VLM families on MindCube: CoT prompting of instruction model, thinking-mode, and using the VLM as the perception backbone inside SATURN. Thinking mode provides

limited gains over instruction prompting: it improves the Qwen-based models, but gives no gain for InternVL3.5. In contrast, SATURN substantially improves all three backbones, with minimum +29 percentage points gain. These results suggest that the main improvement comes from explicit 3D predicate computation and symbolic composition, rather than from stronger language-side reasoning alone. At the same time, the remaining variation across backbones indicates that SATURN is still affected by the quality of the underlying VLM perception.

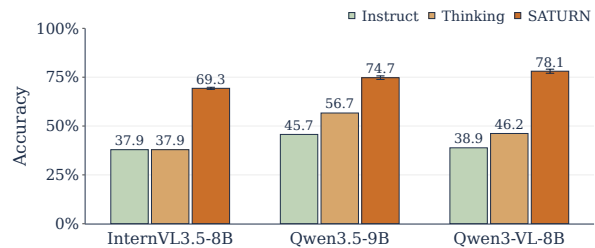


Figure 8: Effect of VLM backbone and reasoning mode on MindCube. Instruct and Thinking are single runs; the SATURN bar is the mean over three runs with the standard deviation as the error bar (0.54, 0.91, and 1.05 points for InternVL3.5-8B, Qwen3.5-9B, and Qwen3-VL-8B).

H.2 Images and Camera Frames

Two scale axes are separate from reasoning depth: the number of images a query spans, and the number of camera frames of reference it composes. They are coupled in the benchmark, since a query composes at most as many frames as it has images, so Figures 9 and 10 hold both visible. Both restrict to camera-perspective queries, so the column is an exact frame count; pooling all perspectives would make the axis ambiguous, because the zero-frame column holds both object-anchored and camera-anchored queries.

On REF, SATURN shows no systematic trend on either axis, staying between 76% and 93%. The VLMs share no pattern. InternVL3.5-38B falls from 57% with a single image to at most 15% with more, a step at the input boundary rather than a gradient. Qwen3.5-35B declines along both axes, from 65% to 33%. Gemini-3.1-Pro stays between 48% and 65%. The same holds on SAG.

H.3 Reasoning Topology

Figure 11 separates the topologies pooled in Figure 5. *Chain* is flat across hop counts, *star* degrades

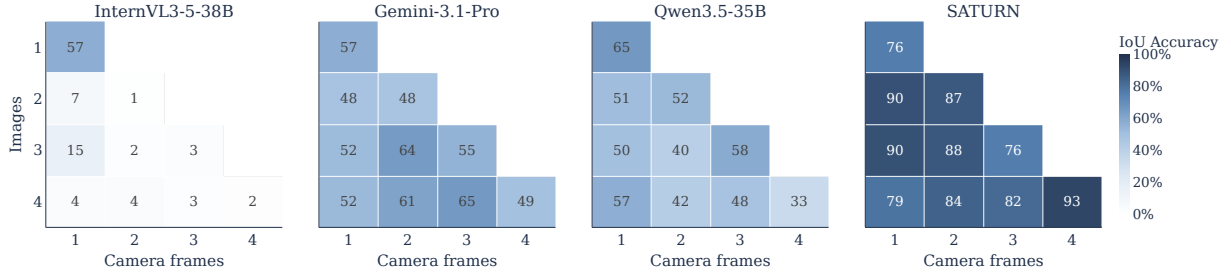


Figure 9: Accuracy on REF by number of images (rows) and number of camera frames of reference the query composes (columns), on camera-perspective queries. Cells above the diagonal do not exist.

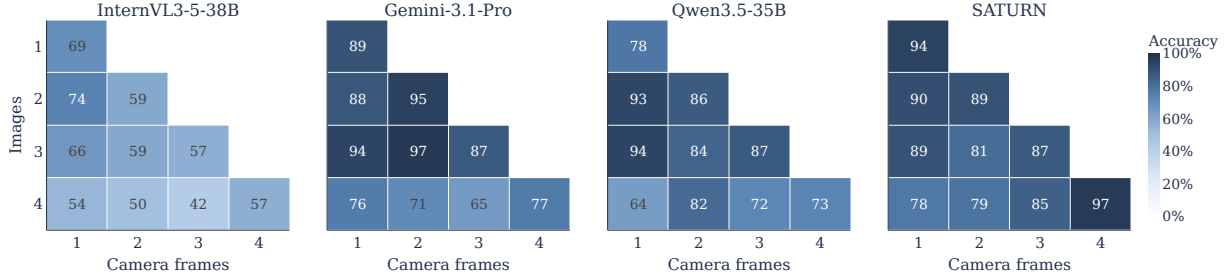


Figure 10: Accuracy on SAG by number of images (rows) and number of camera frames of reference composed (columns), on camera-perspective queries.

as hops grow for the VLMs, and *hybrid* is the weakest structure for every model. At a fixed four hops, every model scores lower on *hybrid* than on *chain*: SATURN falls from 81% to 68%, Qwen3.5-35B from 54% to 34%, Gemini-3.1-Pro from 53% to 41%, and InternVL3.5-38B from 16% to 5%.

H.4 Per-Query Cost by Stage

Table 11 breaks down the per-query cost reported in Table 5. Scene estimation dominates, and it can be amortized when several queries concern the same scene.

Stage	Mean (s)
Scene estimation	97.0
Program generation	5.1
Symbolic execution	2.1
Scene assembly	0.7
Total	105

Table 11: Per-query cost of SATURN on MindCube, by stage.

I Symbolic Baselines

We compare against three tool-augmented systems. Each was released with its own backbone and its own perception stack, so a direct comparison would confound the reasoning method with the models it happens to call. We therefore run GCA and pyS-

patial with the same perception backbone as SATURN, Qwen3-VL-8B-Instruct, and change nothing else in their reasoning. For TIGeR, we use its own fine-tuned weights. This appendix records what each system does, the configuration we ran it in, and what we could not match.

Object boxes on REF. On REF we provide all three tool-augmented baselines with the benchmark’s candidate object boxes. Each answers by selecting a candidate, and we score the selected box against the target.

I.1 GCA

GCA (Chen et al., 2025a) formalizes a query into a reference-frame constraint and then has a vision-language model emit tool calls and Python code against a fixed tool set: 3D reconstruction, open-vocabulary detection, box-to-3D projection, pose estimation, scale estimation, OCR, and motion analysis.

The released configuration uses Qwen3-VL-235B-A22B-Thinking for both the reasoner and the code generator. We run it with Qwen3-VL-8B-Instruct as the reasoner and DeepSeek-V4-Flash as the code generator, matching SATURN’s own backbone and generator, and route its perception calls to the same SAM3, VGGT, and OAv2 services, so the two systems see the same detections and the same reconstruction.

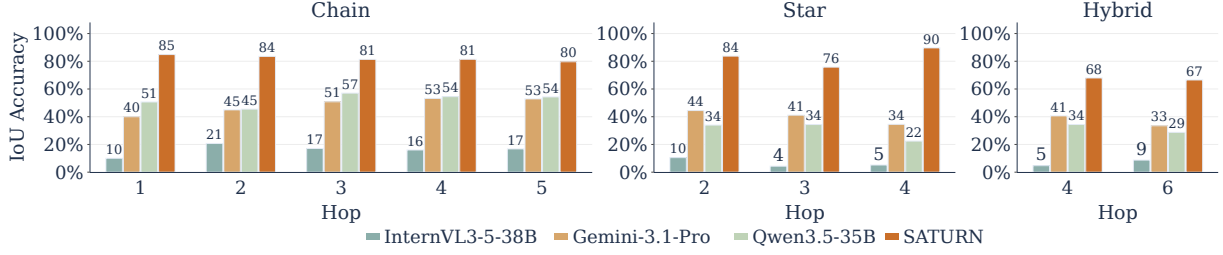


Figure 11: Accuracy on REF categorized by the topology structures used to encode multi-hop reasoning, chain (left), star (center), and hybrid (right), evaluated across different reasoning-hop lengths for top-performing models.

I.2 pySpatial

pySpatial (Luo et al., 2026) has a code agent compose calls to reconstruction, camera-pose recovery, and novel-view synthesis. The program does not compute spatial relations; it renders one or more new views, and a vision-language model answers from the original images together with the rendered output.

We run it with Qwen3-VL-8B-Instruct as the answering model and DeepSeek-V4-Flash as the code agent, with VGGT for reconstruction.

I.3 TIGeR

TIGeR (Han et al., 2026) is a model rather than a prompting method: GLM-4.1V-Thinking (Team et al., 2026) fine-tuned on 300K tool-invocation traces, so that it emits geometric code and calls external libraries for exact computation. Because the behavior is trained into the checkpoint, its backbone cannot be swapped the way the other two can. We serve the released checkpoint and route its perception calls to the shared VGGT and SAM3 services, which is the only part of its stack we can match to SATURN.

TIGeR is therefore the one baseline whose reasoning model differs from ours by design. We include it because it is the closest published system that computes geometry rather than estimating it, and we mark the mismatch wherever we report its numbers.

J Error Analysis

J.1 Object Orientation Error

Table 12 reports orientation accuracy under ground-truth boxes, isolating orientation estimation from detection. Errors are small in the median, and the mean is raised by a small share of front-back flips on categories that are close to front-back symmetric.

Orientation error (20,818 objects)	Value
Median yaw error	5.4°
Mean yaw error	11.4°
Mean excluding front-back flips	6.9°
Within 22.5°	95.4%
Front-back flips	2.7%

Table 12: Object orientation error on REF, measured under ground-truth boxes.

J.2 Program Reliability

Table 13 reports what happens to the 5,288 programs SATURN generated across the four benchmarks. *Parse*, *Run*, and *Answer* are measured after retries, so we also report first-attempt success, the share of queries that needed no regeneration. Program generation is not a bottleneck, because the program only composes predicates from a small API.

Benchmark	N	Parse	Run	Answer	First try
MindCube	1050	100.0	100.0	99.9	99.3
MMSI	1000	99.7	99.4	98.7	94.5
SAG	1150	100.0	100.0	100.0	100.0
REF	2088	100.0	99.8	99.8	99.2

Table 13: Program reliability (%). *Parse*, *Run*, and *Answer* are measured after retries; *First try* is the share of queries answered without regenerating the program.

K Paired Significance of the Baseline Gaps

Table 2 reports the mean and standard deviation over three SATURN runs. Those standard deviations describe run-to-run variation; they do not say whether SATURN beats a baseline on the same questions. For that we pair the two systems on the id-intersection of their predictions and test the difference directly.

We use one SATURN run per benchmark for the pairing and use the same run for every base-

line comparison within that benchmark. For each baseline we report the accuracy difference on the matched set, a 95% confidence interval from a paired case-resampling bootstrap with 10,000 resamples, and a two-sided exact McNemar test on the discordant pairs. Because the pairing removes between-question variance, these intervals are much tighter than the run-to-run standard deviations, and they answer a different question.

Baseline	MindCube		MMSI	
	Δ	95% CI	Δ	95% CI
GCA _{8B}	+46.9	[43.0, 50.6]	+11.4	[7.5, 15.4]
pySpatial	+26.0	[22.5, 29.5]	+21.1	[16.8, 25.3]
TIGeR	+52.2	[48.6, 55.7]	+23.4	[19.3, 27.5]
VLM + scene state	+18.3	[15.0, 21.5]	+9.5	[5.9, 13.1]
VLM (direct)	+40.3	[36.7, 43.9]	+19.5	[15.4, 23.6]

Table 14: Accuracy difference (SATURN – baseline) in percentage points on the matched set, with a 95% paired-bootstrap interval. Every difference is significant at $p < 0.001$ by an exact McNemar test. $n = 1050$ on MindCube and $n = 1000$ on MMSI, except pySpatial on MMSI ($n = 914$).

L Qualitative Examples

Here we show six representative samples drawn from 3D FORCE-REF, MindCube, and MMSI: three where SATURN produces the correct answer on a hard query and three where it fails, followed by a full end-to-end trace (Example G).

Patterns across the examples. Two observations stand out. *First*, because every step is explicit, the failures can be traced to one stage. In Example B the program is correct, but the estimated front of one anchor object is flipped by 173.5° . In Example F the scene is sound, but the code generator binds north to the wrong vector, unlike the structurally identical Example E, which asks about the same eight frames. *Second*, keeping scores soft lets a weak label still count. In Example G the VLM gives the grey-green wall a label score of only 0.438; without rescaling, option B would score 0.438 and the dressing table (0.700) would win. Because `.iota` rescales each label across the objects, the wall’s label counts fully and the wall wins (0.963 against 0.700).

M Prompt Templates

SATURN uses natural-language prompts to interface its symbolic executor with external language

and vision-language models. We describe the grounder and pose-constraint extractor prompts below; the full templates are released with the code at <https://github.com/HLR/SATURN>.

M.1 Grounder Prompt

The grounder prompt drives the VLM that parses the natural-language question and emits a structured noun-phrase plan: which object categories and descriptions to detect, and from which camera views. The prompt contains a compact specification of the expected output schema together with a small set of in-context examples illustrating how to extract role-indexed object phrases and assign cam-id cues.

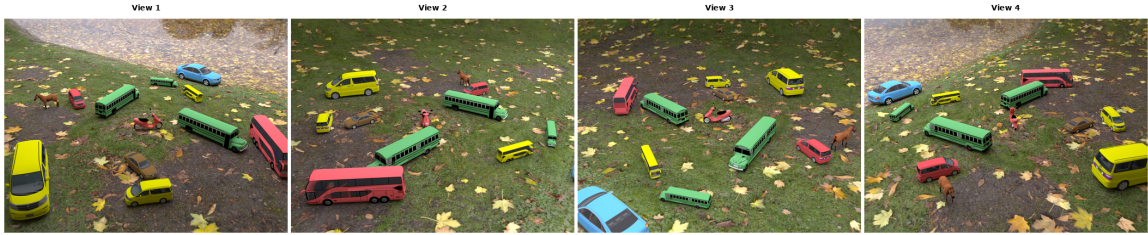
M.2 Pose Constraint Extractor

The pose-constraint extractor classifies clauses in the question into stated camera facts versus hypothetical observer motions, and emits schema-validated yaw rotations and same-position groups for the downstream geometric refinement step (Appendix F).

Example A: 3D FORCE-REF, 6-hop hybrid (success)

Setup. 4 cameras, 12 detected objects, 6-hop hybrid topology, every relation FoR-tagged. (benchmark: 3D FORCE-REF; sample id: hybrid_d6_multiview_4_multi_cam_4_with_bbox_q108)

Question. Find the yellow object that is behind the green object (camera 2 view) and to the left of the red object (the red object's view), where the green object is behind the brown object (camera 3 view) and the brown object is to the left of the red double-decker bus (camera 3 view) and the red object is in front of the sedan car (camera 0 view) and the sedan car is in front of the scooter motorcycle (camera 1 view).



Planner groundings.

Phrase	Description	Unique
objects in the image	-	X

```

is_yellow = score("is the object in the red bounding box a yellow object?")
is_green = score("is the object in the red bounding box a green object?")
is_red = score("is the object in the red bounding box a red object?")
is_brown = score("is the object in the red bounding box a brown object?")
is_bus = score("is the object in the red bounding box a double-decker bus?")
is_sedan = score("is the object in the red bounding box a sedan car?")
is_scooter = score("is the object in the red bounding box a scooter motorcycle?")

cam2_behind = scene.cameras[2].third_person.behind
cam3_behind = scene.cameras[3].third_person.behind
cam3_left = scene.cameras[3].third_person.left
cam0_front = scene.cameras[0].third_person.front
cam1_front = scene.cameras[1].third_person.front

scooter = is_scooter("x1").iota("x1")
sedan = (is_sedan("x2") & cam1_front("x2", "x1") & scooter("x1")).iota("x2")
red_obj = (is_red("x3") & cam0_front("x3", "x2") & sedan("x2")).iota("x3")
brown_obj = (is_brown("x4") & cam3_left("x4", "x5") & is_bus("x5") & cam3_behind("x6", "x4") & is_green("x6")).iota("x4")
green_obj = (is_green("x6") & cam3_behind("x6", "x4") & brown_obj("x4")).iota("x6")

target = (is_yellow("x7") & cam2_behind("x7", "x6") & green_obj("x6") & scene.obj_left("x7", "x3") & red_obj("x3")).iota("x7")
return int(target.argmax())

```

SATURN answer: 11 **Ground truth:** box [435, 573, 608, 675] **✓ Success**

Index to box. REF is scored by IoU against a box, while the program returns an entity index. In this setting the scene is seeded from the released boxes, so entity i is box i : index 11 resolves to [435, 573, 608, 675], which is the ground-truth box.

Example B: 3D FORCE-REF, 2-hop star (failure)

Setup. 2 cameras, 7 detected objects, 2-hop star topology, both anchors object-centric. (benchmark: 3D FORCE-REF; sample id: star_d2_multiview_2_multi_cam_0_q1247)

Question. Find the green object that is in front of the red sedan car (the red sedan car's view) and to the left of the tank (the tank's view).



```
is_green = score("is the object in the red bounding box a green object?")
is_red = score("is the object in the red bounding box a red object?")
is_sedan = score("is the object in the red bounding box a sedan car?")
is_tank = score("is the object in the red bounding box a tank?")

red_sedan = (is_red("x1") & is_sedan("x1")).iota("x1")
tank = is_tank("x1").iota("x1")

target = (is_green("x3") &
  scene.obj_front("x3", "x1") & red_sedan("x1") &
  scene.obj_left("x3", "x2") & tank("x2")).iota("x3")

return int(target.argmax())
```

SATURN answer: box [123, 193, 277, 316] **Ground truth:** box [798, 277, 877, 332]

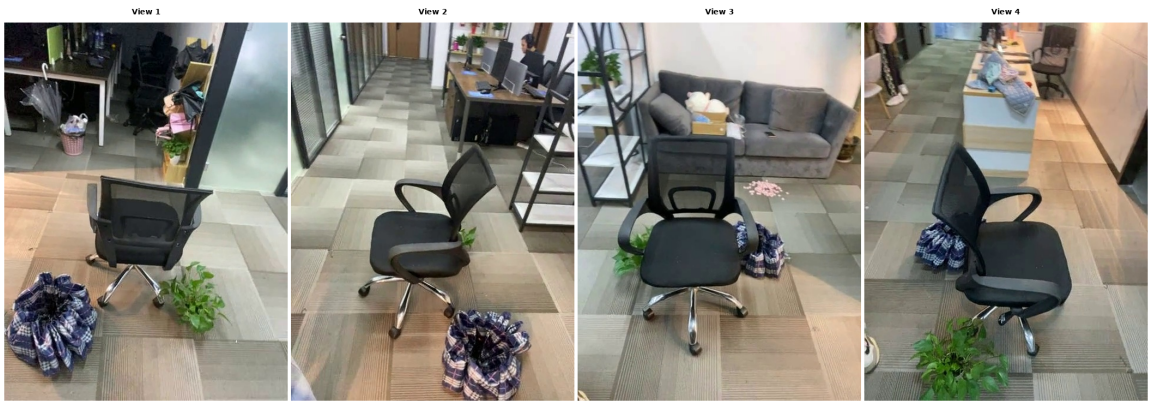
✗ Failure

Diagnosis. The program is correct; the scene it reads is not. One anchor is accurate and the other is inverted: the red sedan's estimated front azimuth is off by 7.7° , while the tank's is off by 173.5° (76.2° ground truth against 262.7° estimated). Every other object in the scene is within 8° . The accurate anchor does not discriminate, because both green candidates lie in front of the red sedan in the ground-truth and estimated scenes alike. The flipped anchor does: the target lies 5.6 m to the tank's left in the ground-truth scene but 4.5 m to its right in the estimated one, and the selected double-decker bus inverts the same way, so the program returns it. The tank's estimated front is nearly parallel to the optical axis in both views, the configuration in which front and back are least separable.

Example C: MindCube Among, 4-view perspective shift (success)

Setup. 4 real-world photos taken around the chair (front, left, back, right); query references image 2’s viewpoint explicitly. (benchmark: MindCube-Among; sample id: among_group525_q1_2_1)

Question. Based on these four images (image 1, 2, 3, and 4) showing the black chair from different viewpoints (front, left, back, and right), with each camera aligned with room walls and partially capturing the surroundings: If I am standing at the same spot and facing the same direction as shown in image 2, what is behind me? A. Grey sofa B. Office area C. Two chairs on the corridor D. Cabinet desk along a corridor



Planner groundings.

Phrase	Description	Unique
black chair	black mesh-back office chair with chrome base and wheels, viewed from the front	✓
grey sofa	large grey upholstered sofa with cushions, positioned against a wall	✓
office area	workspace with desks, computers, and office chairs in the background	✓
two chairs on the corridor	two office chairs side by side in a hallway, visible in the background	✓
cabinet desk along a corridor	long white cabinet desk with items on top, situated along a hallway	✓

```
# Allocentric anchor at camera 1 (image 2, 0-indexed)
anchor = scene.frame(position=scene.cameras[1].position,
                     orientation=scene.cameras[1].orientation)

# INVERSE Rule 1: geometric argmax FIRST - pick the scene object most behind me
target_idx = anchor.first_person.behind[:scene.objects_count].argmax()

# Score each option phrase against the chosen target's box (RAW score, target is fixed)
options = {"A": "grey sofa", "B": "office area", "C": "two chairs on the corridor", "D": "cabinet desk along a corridor"}
return max(options, key=lambda k:
           float(score(f"is the object in the red bounding box a {options[k]}?")[target_idx]))
```

SATURN answer: D Ground truth: D ✓ Success

Explanation. The question pins the reasoning anchor to image 2 (camera index 1). The generated program constructs the anchor as scene.frame(position=scene.cameras[1].position, orientation=scene.cameras[1].orientation) and then takes the geometric argmax along the anchor’s behind axis. The scene.frame call is the anchor-conditioned predicate interface at the core of SATURN: a single call constructs a virtual viewer, after which standard predicates are applied.

Example D: MindCube Rotation, 3-view 180° flip (failure)

Setup. 3 real-world bedroom photos at 0/90/180°; text-stated rotation constraints extracted. (benchmark: MindCube-Rotation; sample id: rotation_group005_q1_2)

Question. *These three images (image 1, 2, and 3) show the same scene from three different viewpoints. The image 2 was taken after turning the camera 90 degrees to the right (clockwise) from the position of image 1. For image 3, the camera was turned another 90 degrees right, so it's basically facing the opposite direction of image 1. Based on these three images: If I am standing at the same spot and facing the same direction as shown in image 1, what is to my behind? A. Study table and black chair B. Window C. Bed*



Planner groundings.

Phrase	Description	Unique
study table and black chair	light-wood desk with a black office chair tucked under it	✓
window	window with blinds casting bright light, located on the left wall	✓
bed	wooden bunk bed with a dark blue mattress, positioned against the right wall	✓

```
anchor = scene.frame(position=scene.cameras[0].position,
                    orientation=scene.cameras[0].orientation)
target_idx = anchor.first_person.behind[:scene.objects_count].argmax()

options = {"A": "study table and black chair", "B": "window", "C": "bed"}
return max(options, key=lambda k:
           float(score(f"is the object in the red bounding box a {options[k]}?")[target_idx]))
```

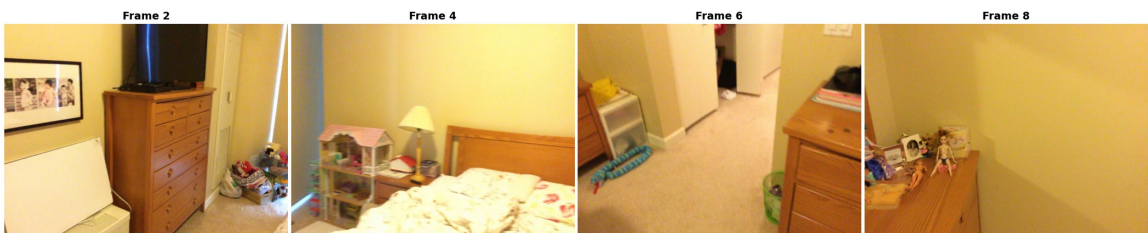
SATURN answer: B **Ground truth:** A **✗ Failure**

Diagnosis. The pose-constraint extractor correctly captured the rotations (yaw 90° from cam 0 to cam 1 and 180° from cam 0 to cam 2), and the program anchors at camera 0 (image 1). The failure is in grounding: the room has two desks with black chairs, one in view 1, in front of the viewer, and one in view 3, behind the viewer. The VLM grounded “study table and black chair” to the desk in view 1, so the object most behind the viewer is the window, and the program answers B rather than the ground-truth option A.

Example E: MMSI Multi-Step Reasoning, 8 cameras (success)

Setup. 8 consecutive first-person frames; north is set by the direction from the TV to the nightstand.
(benchmark: MMSI-MSR; sample id: 298)

Question. *Where is the trash can in the room if the nightstand near the window faces north relative to the TV? Options: A: Near the east wall, B: Southeast corner, C: Near the west wall, D: Northwest corner*



Planner groundings.

Phrase	Description	Unique
TV	black flat-screen television on top of a wooden dresser	✓
nightstand	wooden nightstand with a lamp and tissues, next to the bed	✓
window	large window with vertical blinds, letting in bright light	✓
trash can	green plastic trash can with a lid, on the floor	✓

```
# Find the TV and nightstand
tv_idx = score("is the object in the red bounding box a black flat-screen television on top of a wooden dresser?").iota("x1").argmax()
nightstand_idx = score("is the object in the red bounding box a wooden nightstand with a lamp and tissues, next to the bed?").iota("x1").argmax()
trash_can_idx = score("is the object in the red bounding box a green plastic trash can with a lid, on the floor?").iota("x1").argmax()

# The nightstand faces north relative to the TV
# This means the direction from TV to nightstand is north
north_vec = scene.objects[nightstand_idx].position - scene.objects[tv_idx].position
scene.set_cardinal_vector(north_vec)

# Build anchor at the TV with its intrinsic orientation
anchor = scene.frame(
    position=scene.objects[tv_idx].position,
    orientation=scene.objects[tv_idx].orientation
)

# Score each cardinal direction for the trash can
options = {"A": "east", "B": "south-east", "C": "west", "D": "north-west"}
return max(options, key=lambda k: float(anchor.first_person(options[k])[trash_can_idx]))
```

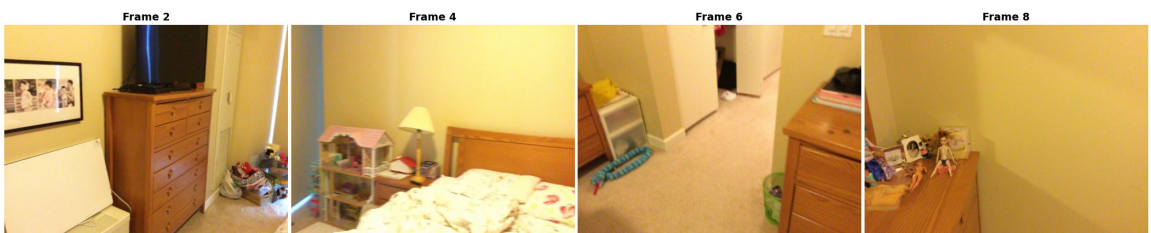
SATURN answer: A **Ground truth:** A ✓ **Success**

Explanation. The query states that “the nightstand near the window faces north relative to the TV”. The code generator interprets this as a position-difference cardinal binding, emitting $\text{north_vec} = \text{nightstand.position} - \text{tv.position}$ and binding the world-frame *north* axis to this vector. The TV-anchored frame is then constructed, and the trash-can’s position is evaluated against the cardinal axes derived from this north. This matches the MMSI ground-truth convention for object-anchored cardinal bindings, and is the interpretation that Example F fails to reproduce on a structurally identical query.

Example F: MMSI Multi-Step Reasoning, 8 cameras (failure)

Setup. 8 consecutive frames, 4 detected objects; cardinal direction is defined by the same template as Example E (*X near window faces north relative to Y*), so the contrast pair holds query structure fixed; both questions are asked about the same eight frames. (benchmark: MMSI-MSR; sample id: 223)

Question. *The bedside table near the window faces north relative to the television; where is the light switch in the room located?* Options: A: Southeast corner, B: Northwest corner, C: Near the east wall, D: Near the west wall



Planner groundings.

Phrase	Description	Unique
television	black flat-screen TV on top of a wooden dresser	✓
bedside table	wooden nightstand with a lamp and tissues, next to the bed	✓
window	large window with vertical blinds, letting in bright light	✓
light switch	white rectangular switch plate on a beige wall, near a doorway	✓

```
# Get indices for the objects
bedside_table_idx = score("is the object in the red bounding box a bedside table?").iota("x1").argmax()
television_idx = score("is the object in the red bounding box a television?").iota("x1").argmax()
light_switch_idx = score("is the object in the red bounding box a light switch?").iota("x1").argmax()

# The bedside table faces north, so its front vector is north
scene.set_cardinal_vector(scene.objects[bedside_table_idx].front_vec)

# Build anchor at the television with its intrinsic orientation
anchor = scene.frame(
    position=scene.objects[television_idx].position,
    orientation=scene.objects[television_idx].orientation
)

# Score each cardinal direction for the light switch
options = {"A": "south-east", "B": "north-west", "C": "east", "D": "west"}
return max(options, key=lambda k: float(anchor.first_person(options[k])[light_switch_idx]))
```

SATURN answer: D **Ground truth:** C **✗ Failure**

Diagnosis. The failure is at code generation. For the cardinal-anchor clause “the bedside table faces north relative to the television”, the code generator emitted `scene.set_cardinal_vector(bedside_table.front_vec)` — using the table’s own facing axis as north — rather than the position-difference interpretation `table.position - tv.position` that the structurally identical Example E correctly used. The two programs differ in the line that sets north, but the resulting world-frame north is rotated, and the program answers west where the ground truth is east.

Example G: MindCube Rotation, end-to-end trace (success)

Setup. 3 views, 3 detected objects, two stated rotations. (benchmark: MindCube-Rotation; sample id: rotation_group020_q2_6)

Question. The image 2 was taken after turning the camera 90° to the right from image 1, and image 3 another 90° right. If I am standing at the same spot and facing the same direction as shown in image 2, then I turn 90° to the right, what is behind me? **A.** Dressing table **B.** Grey-green decorative wall **C.** Window



(1) **Planner groundings.** The planner expands each question phrase into a description.

Phrase	Description	Unique
dressing table	wooden dressing table with a large mirror and drawers, cluttered with items	✓
grey-green decorative wall	wall painted in a grey-green gradient with a framed picture hanging on it	✓
window	large window with white frames and a view of trees outside	✓

(2) **Pose constraints and estimated scene.** The extractor reads the stated turns from the question and applies them to the cameras: rotation(cam1→2, yaw 90°) and rotation(cam1→3, yaw 180°). The table lists the estimated objects and, below, the camera headings before and after the constraints are applied.

Id	Label	Centre (x, y, z)	Azimuth	View
0	dressing table	(+0.01, +0.07, +0.28)	−78.9°	2
1	window	(−0.83, +0.33, −0.44)	−22.1°	3
2	grey-green decorative wall	(−0.30, +0.10, +0.76)	129.0°	1

camera headings, reconstruction only: c_1 0.0°, c_2 89.2°, c_3 154.0° (turns 89.2°, 64.8°)
after applying the constraints: c_1 −7.7°, c_2 82.3°, c_3 172.3° (turns 90.0°, 90.0°)

(3) **Soft scores.** One vector per call over the three objects. back is read from camera 2's frame turned 90° right; .iota rescales each label vector so that its best object scores 1.

	obj 0	obj 1	obj 2
first_person.back	0.700	0.171	0.963
"...a dressing table?"	1.000	0.000	0.000
"...a grey-green decorative wall?"	0.000	0.000	0.438 → 1.000
"...a window?"	0.000	1.000	0.002

(4) **Generated program.** Boilerplate wrappers elided.

```
anchor = scene.frame(position=camera(2).position, orientation=camera(2).orientation).rotate(yaw=90)
J = anchor.first_person.back('x1')
options = {'A': 'dressing table', 'B': 'grey-green decorative wall', 'C': 'window'}
return max(options, key=lambda k: float(
    (J & score(f'is the object in the red bounding box a {options[k]}?').iota('x1')).exists()))
```

(5) **SATURN answer:** B **Ground truth:** B **✓ Success** Option values: A 0.700, B 0.963, C 0.171.