

Geometrically-Constrained Agent for Spatial Reasoning

Zeren Chen^{1,2*}, Xiaoya Lu^{2,3*}, Zhijie Zheng^{1,2}, Pengrui Li¹, Lehan He^{1,4}, Yijin Zhou^{2,3,4}
 Jing Shao², Bohan Zhuang^{5†}, Lu Sheng^{1†}

¹School of Software, Beihang University ²Shanghai AI Laboratory

³Shanghai Jiao Tong University ⁴Shanghai Innovation Institute ⁵ZIP Lab, Zhejiang University

{czr1604, lsheng}@buaa.edu.cn, {luxiaoya, shaojing}@pjlab.org.cn

Homepage: <https://gca-spatial-reasoning.github.io>

Abstract

*Vision Language Models (VLMs) exhibit a fundamental semantic-to-geometric gap in spatial reasoning: they excel at qualitative semantic inference but their reasoning operates within a lossy semantic space, misaligned with high-fidelity geometry. Current paradigms fail to bridge this gap. Training-based methods suffer from an “oracle paradox,” learning flawed spatial logic from imperfect oracles. Tool-integrated methods constrain the final computation but critically leave the VLM’s planning process unconstrained, resulting in geometrically flawed plans. In this work, we propose **Geometrically-Constrained Agent (GCA)**, a training-free agentic paradigm that resolves this gap by introducing a formal task constraint. Specifically, we strategically decouples the VLM’s role into two stages. First, acting as a semantic analyst, the VLM translates the user’s ambiguous query into the formal, verifiable task constraint, which defines the reference frame and objective. Second, acting as a task solver, the VLM generates and executes tool calls strictly within the deterministic bounds defined by the constraint. This geometrically-constrained reasoning strategy successfully resolve the semantic-to-geometric gap, yielding a robust and verifiable reasoning pathway for spatial reasoning. Comprehensive experiments demonstrate that GCA achieves SOTA performance on multiple spatial reasoning benchmarks, surpassing existing training-based and tool-integrated methods by ~27%.*

1. Introduction

Intelligent agents operating in real-world applications, such as robotics [41, 54, 62], AR/VR [2, 11, 31], and autonomous driving [9, 42, 51], demand a perceptual understanding of the world akin to humans. Humans intuitively comprehend their surroundings as a cohesive 3D environment, effort-

lessly discerning object orientations and complex spatial relationships. However, equipping Vision Language Models (VLMs) into agents with this holistic **spatial reasoning** capability remains a critical challenge [18, 53, 55, 59, 60].

As shown in Figure 1 (a), current VLMs lossily translate rich visual information into a textual semantic space, leading fine-grained geometric details to be omitted or distorted [24, 50]. This creates a fundamental semantic-to-geometric gap: *VLMs excel at probabilistic, qualitative semantic inference, but their lossy semantic space required for spatial reasoning fails to ground high-fidelity geometry*. For example, a VLM may possess the spatial common-sense (e.g., intuitively knowing that “sitting on a sofa” implies a viewpoint aligned with the sofa’s orientation), yet critically fail at high-precision geometric computation (e.g., determining the sofa’s orientation) and robust spatial imagination (e.g., imagining the user’s egocentric perspective). To reconcile this gap, robust **constraints** must be imposed, guiding the VLM’s reasoning onto a geometrically sound and verifiable pathway.

However, effectively applying these constraints remains a formidable challenge. Recent approaches that apply implicit constraints via end-to-end training [8, 22, 30, 33, 41, 48, 50, 52] attempt to embed geometric logic by fine-tuning on massive datasets. These methods, however, face an “oracle paradox”: their data generation relies on oracles like GPT-4o [17] which themselves struggle with spatial reasoning [18, 53, 55, 59, 60]. Consequently, the VLM is often trained on flawed spatial logic rather than sound geometric principles. An alternative paradigm, tool integration [12, 49, 62], attempts to bridge this gap by adopting an iterative plan-then-execute strategy, which offloads high-precision geometric computation to deterministic external tools. While this constrains the final computation process, the VLM’s planning process remains unconstrained. To plan next step, the VLM must still perform spatial imagination and further decision-making within its lossy semantic space, inevitably producing geometrically flawed plans.

*Equal contribution. †Corresponding author.

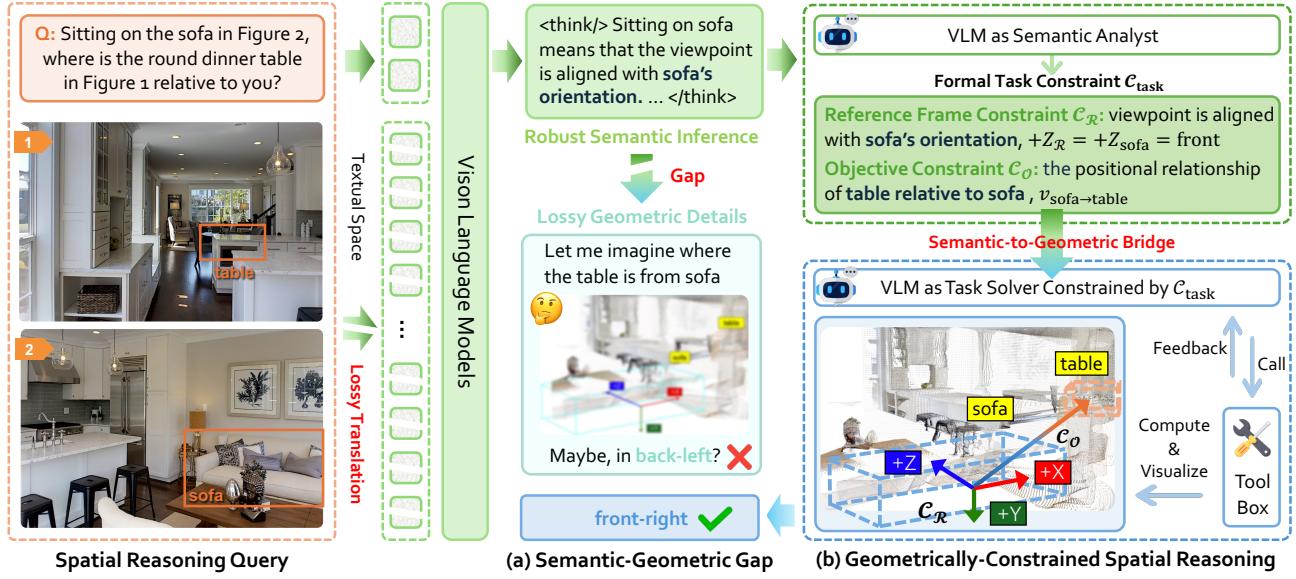


Figure 1. **Overview.** (a) **Semantic-Geometric Gap.** The geometric details required for spatial reasoning are lost when translating visual information into textual space, leading to VLM’s flawed reasoning or unconstrained planning. (b) **Geometrically-Constrained Spatial Reasoning.** We propose a formal task constraint that serves as a deterministic bridge between semantics and geometry in spatial reasoning.

For instance, when asked to reason from the perspective of a user “sitting on the sofa” (see Figure 1), its unconstrained plan may default to the camera’s viewpoint, compromising the problem definition before any tool is even called.

These challenges reveal the critical research question: *How do we bridge the VLM’s semantic-to-geometric gap?* We argue the solution is not to force the VLM to reason about lossy geometric details directly, but to reframe the problem into a task that leverages its inherent strengths: using its spatial commonsense to define a **formal task constraint** C_{task} for subsequent computation. Specifically, this C_{task} must be (1) grammatically rich enough to define complex spatial concepts, such as viewpoints, which elude traditional state-based formalisms (2) semantically clear enough for a VLM to generate using its qualitative strengths, and (3) geometrically sound enough to provide a deterministic, verifiable constraint for subsequent computation.

To this end, we introduce **Geometrically-Constrained Agent (GCA)**, a training-free agentic paradigm for geometrically-constrained spatial reasoning. As shown in Figure 1 (b), this strategy leverages a formal task constraint, C_{task} , to decouple the reasoning process into two stages: (1) *Task Formalization*. The VLM, acting as a semantic analyst, translates the ambiguous query and visual data into the formal, verifiable task constraint C_{task} . This stage defines what to solve, establishing immutable sub-constraints: a **reference frame constraint** and an **objective constraint**. (2) *Constrained Geometric Computation*. The VLM then, acting as a task solver, generates and executes tool calls to compute the final answer, operat-

ing strictly within the deterministic bounds defined by C_{task} . This two-stage decoupling directly bridges the semantic-to-geometric gap. Through formulating a geometrically sound constraint, we force the VLM to solve deterministic mathematical problems, thereby avoiding the demands for directly computing or imagining about high-fidelity geometric details that are lost in its semantic space. Extensive experiments demonstrate the effectiveness and generalizability of GCA paradigm. GCA yields substantial performance gains when applied to several foundation VLMs (by an average of $\sim 37\%$), establishing a new state-of-the-art across a diverse suite of challenging spatial reasoning benchmarks.

2. Related Work

Spatial Reasoning in Vision Language Models. Spatial reasoning, including comprehension and mental manipulation of 3D spatial relationships [18, 53, 55, 59, 60], remains a foundational challenge for Vision Language Models (VLMs) [17, 21, 36, 43]. To address this deficit, recent research [4, 8, 22, 30, 33, 39, 48, 50, 52] focuses on large-scale, end-to-end training on specialized spatial datasets. These methods attempt to bridge the 2D-3D cognitive gap by incorporating geometric priors, such as explicit 3D structural features [48], or depth maps [4], directly into the VLM’s architecture, but they are often hindered by the reliance on high-quality datasets generated by flawed oracle. Another line of research introduce tool-integrated reasoning [12, 25, 29, 49, 62] to offloads deterministic geometric computation to external modules. For example, Spa-

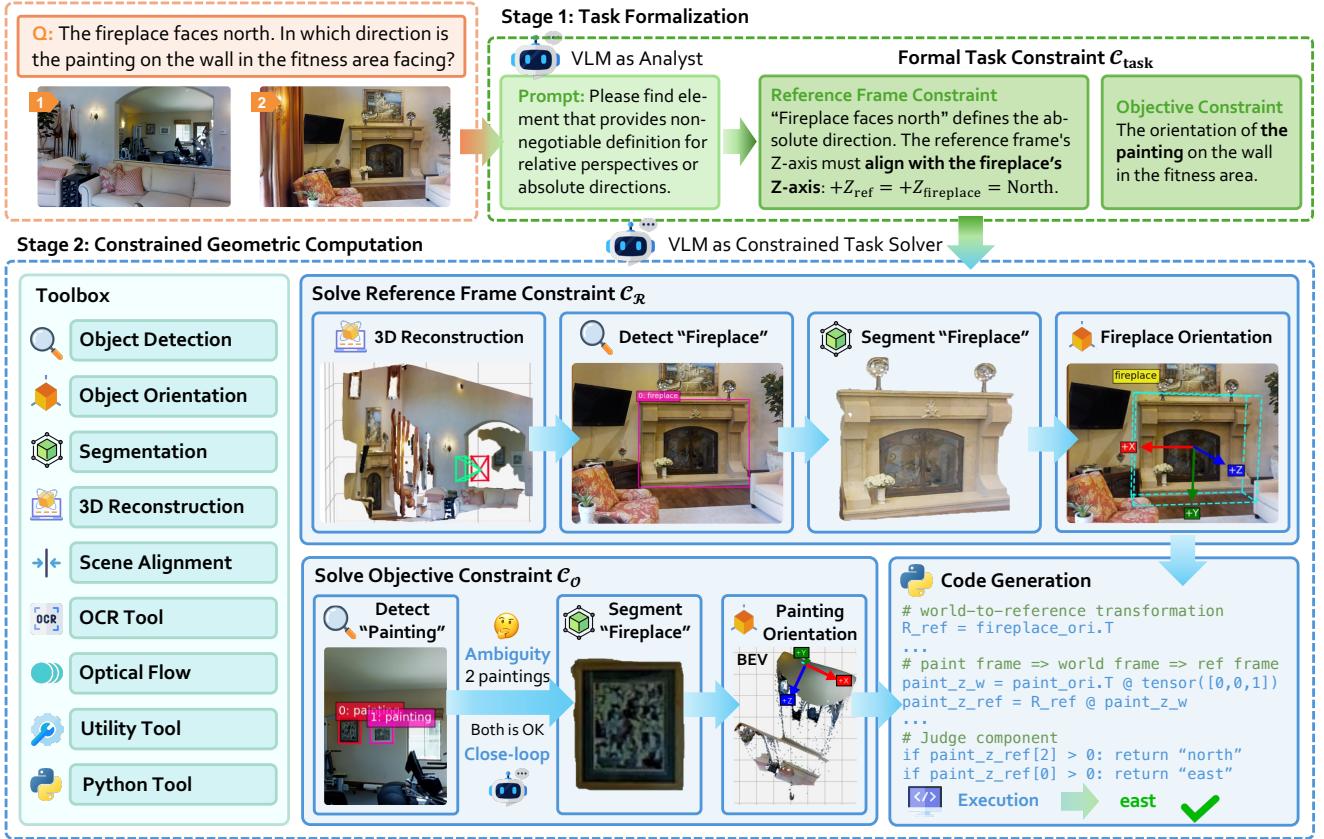


Figure 2. **Overall Paradigm of GCA.** Given a spatial reasoning query, our GCA leverages a geometrically-constrained reasoning strategy centered on the formal task constraint (C_{task}). The VLM first translates the ambiguous query into this explicit C_{task} , establishing a non-negotiable reference frame (C_R) and objective (C_O). Strictly constrained by C_{task} , the VLM then orchestrates a toolbox to perform deterministic geometric computation and derive the final answer.

tialAgent [49] and TIGeR [12] focus on translating the input query directly into an iterative sequence of tool executions. However, unconstrained planning process could lead to geometrically-flawed results, causing the agent to conflate “what to solve” with “how to solve it”.

Constrained-Guided Reasoning. Constraint-guided reasoning, which involves restricting a search space by defining variables and the constraints governing them [38], has been adapted to manage the probabilistic nature of LLMs and VLMs. A primary application is in neuro-symbolic reasoning [1, 13, 14, 34, 40, 58], where the LLM is constrained to act as a translator, converting ambiguous natural language into a formal, verifiable representation. For example, LogicLM [34] leverage LLMs to translate NL problems into task-specific formalisms like formal logic. This constraint-guided reasoning can also be extended to planning [3, 16, 26, 35, 56]. Frameworks like LLM+P [26] uses an LLM to translate an NL problem into a formal PDDL format and then applies an optimal planner to generate the plan. Similarly, ReKep [16] employs a VLM to translate a

free-form language into relational keypoint constraints and solves the constraints to generate final robot actions.

3. Methodology

As illustrated in Figure 2, we propose Geometrically-Constrained Agent (GCA), a training-free agentic paradigm designed for geometrically-constrained spatial reasoning. The core of GCA is the introduction of a formal task constraint C_{task} that serves as a deterministic bridge between semantics and geometry. Section 3.1 defines this geometrically-constrained paradigm. Section 3.2 details the formal task constraint C_{task} and its automated generation. Section 3.3 describes the subsequent constrained computation stage, which is strictly governed by this constraint. Finally, Section 3.4 discusses how GCA resolves the VLM’s semantic-to-geometric gap in spatial reasoning.

3.1. Geometrically-Constrained Spatial Reasoning

Contemporary agentic frameworks often model reasoning as a generic, iterative policy. Those based on the ReAct

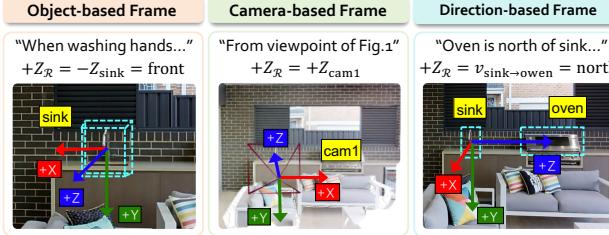


Figure 3. **Reference Frame.** Here, $v_{\text{sink} \rightarrow \text{oven}}$ denotes a vector calculated by “normalize (Centroid(oven) – Centroid(sink))”.

framework [57], for example, can be defined by:

$$r_t = \mathcal{A}(q, v, \mathcal{T}, r_{t-1}). \quad (1)$$

In this framework, an agent \mathcal{A} produces a response r_t based on a query q , visual information v , a set of tools \mathcal{T} , and its past history r_{t-1} . This generic policy \mathcal{A} is unconstrained, making it unreliable for high-stakes, deterministic domains like spatial reasoning. Recent work [12, 49] attempts to mitigate this by using external tools to constrain the final computation. However, they fail to constrain the VLM’s planning process. The VLM may still rely on its flawed spatial imagination in lossy semantic space to formulate plan, conflating “what to solve” with “how to solve it”.

We solve this by replacing the generic policy \mathcal{A} with a two-stage process. This paradigm is built on the formal task constraint $\mathcal{C}_{\text{task}}$, which functions as the architectural scaffolding to align the VLM’s asymmetric capabilities:

$$\begin{aligned} \mathcal{C}_{\text{task}} &\leftarrow \mathcal{F}_{\text{formalize}}(q, v), \\ r_t &= \mathcal{F}_{\text{compute}}(\mathcal{C}_{\text{task}}, \mathcal{T}, r_{t-1}). \end{aligned} \quad (2)$$

In the $\mathcal{F}_{\text{formalize}}$ stage, the VLM acts as a semantic analyst, translating the ambiguous query q and visual information v into a formal, verifiable task constraint $\mathcal{C}_{\text{task}}$. This stage defines what to solve by establishing the necessary geometric scaffolding (e.g., the reference frame and target subjects). In the $\mathcal{F}_{\text{compute}}$ stage, the VLM’s role shifts to a task solver. Governed by the constraint $\mathcal{C}_{\text{task}}$ established in the $\mathcal{F}_{\text{formalize}}$ stage, the VLM iteratively executes tool calls to acquire necessary geometric data and perform final computations.

3.2. Task Constraint Formalization

3.2.1. Constraint for Spatial Reasoning

While existing constraint-guided reasoning paradigms leverage formalisms such as PDDL [26] or relational key-point constraints [16], these constraints are insufficient for spatial reasoning. PDDL, for instance, excels at describing discrete, symbolic object states (e.g., “is_on (A, B)”) but fundamentally lacks the geometric grammar to express the continuous, relative, and perspective-dependent nature of spatial queries (e.g., egocentric vs. allocentric viewpoints).

This gap necessitates a new formalism. We thus propose a novel **formal task constraint** $\mathcal{C}_{\text{task}}$ specifically designed to capture the geometric nature of spatial reasoning. We define $\mathcal{C}_{\text{task}}$ as a tuple containing two key sub-constraints: a single, non-negotiable **Reference Frame Constraint** ($\mathcal{C}_{\mathcal{R}}$) that defines the coordinate system for answering the query, and an **Objective Constraint** ($\mathcal{C}_{\mathcal{O}}$) that specify the objective to be measured within that frame.

Reference Frame Constraint. Humans intuitively understand spatial language (e.g., “north of”) by grounding it within a specific coordinate system, namely a reference frame (\mathcal{R}). In contrast, VLM failures often stem from ambiguity in this crucial grounding step, causing them to adopt flawed geometrically flawed plans [55] (e.g., defaulting to the camera’s viewpoint). The $\mathcal{F}_{\text{formalize}}$ stage addresses this ambiguity by requiring the VLM to first formally anchor \mathcal{R} to the scene’s geometry.

We model all spatial queries as requiring a 3D cartesian coordinate system defined by an origin $O_{\mathcal{R}}$ and three orthogonal basic vectors ($x_{\mathcal{R}}, y_{\mathcal{R}}, z_{\mathcal{R}}$). This system follows the OpenCV convention, where $+z_{\mathcal{R}}$ points forward, $+y_{\mathcal{R}}$ points down and $+x_{\mathcal{R}}$ follows the right-hand rule. The agent’s task is to anchor \mathcal{R} to one of three geometric primitives (see Figure 3) derived from the visual information:

- **Object-based Frame.** \mathcal{R} is defined by an object’s intrinsic coordinate system. For example, the query “when the user is washing hand” implies a reference frame defined by $+z_{\mathcal{R}} = -z_{\text{sink}}$ (one must face a sink to wash hand).
- **Camera-based Frame.** \mathcal{R} is defined by a specific camera’s viewpoint. For “from the viewpoint of Figure 1”, the reference frame is defined by $+z_{\mathcal{R}} = +z_{\text{cam1}}$.
- **Direction-based Frame.** \mathcal{R} is defined by a vector connecting two locations. For “Owen is north of sink”, the reference frame is defined by $+z_{\mathcal{R}} = \text{normalize}(\text{Centroid}(\text{oven}) - \text{Centroid}(\text{sink}))$ = north.

The output of this step is a human-readable and machine-parsable definition of \mathcal{R} , which becomes a non-negotiable constraint $\mathcal{C}_{\mathcal{R}}$ for all subsequent computation.

Objective Constraint. Concurrently, the agent identifies the objective \mathcal{O} from the query. This constraint $\mathcal{C}_{\mathcal{O}}$ defines *what* must be measured relative to the established \mathcal{R} . For the query, “Is chair to the west of toaster?”, the toaster defines $\mathcal{C}_{\mathcal{R}}$, while the positional relationship between toaster and chair is the objective constraint $\mathcal{C}_{\mathcal{O}}$.

3.2.2. Automated Formalization via VLM

We exploit the VLM’s innate strength in semantic interpretation to generate $\mathcal{C}_{\text{task}}$ automatically. Acting as a semantic analyst, the VLM performs qualitative interpretation, guided by the formal definitions of $\mathcal{C}_{\text{task}}$, to generate the $\mathcal{C}_{\text{task}} = (\mathcal{C}_{\mathcal{R}}, \mathcal{C}_{\mathcal{O}})$. This formal task constraint, generated by the VLM but grounded in geometry, serves as the geometrically sound contract for the $\mathcal{F}_{\text{compute}}$ stage. In our

implementation, we enforce this architectural decoupling procedurally. The VLM executes the $\mathcal{F}_{\text{formalize}}$ stage and formalizes the $\mathcal{C}_{\text{task}}$ before any computation begins.

3.3. Constrained Geometric Computation

3.3.1. Tool Integration and Code Generation

Once the formal task constraint $\mathcal{C}_{\text{task}} = (\mathcal{C}_{\mathcal{R}}, \mathcal{C}_{\mathcal{O}})$ is established, the VLM’s role shifts to a constrained task solver. This $\mathcal{F}_{\text{compute}}$ stage then operates as a ReAct-style framework, consuming the $\mathcal{C}_{\text{task}}$ as an immutable constraint. This execution is not a one-shot generation but an iterative, closed-loop process involving data acquisition, ambiguity resolution, and augmented computation.

Data Acquisition. $\mathcal{C}_{\text{task}}$ dictates a set of geometric ingredients that the agent must acquire. For instance, as shown in Figure 2, to instantiate an object-based frame \mathcal{R} defined by a sink, the agent must acquire the orientation of that sink. The $\mathcal{F}_{\text{compute}}$ stage begins by generating a sequence of tool calls to parameterize the geometry, and acquire all variables necessary to instantiate $\mathcal{C}_{\text{task}}$.

Tool Orchestration and Ambiguity Resolution. The VLM is responsible for managing tool feedback and resolving ambiguity, ensuring the data acquired from tools correctly binds to the symbols in $\mathcal{C}_{\text{task}}$. For example, considering $\mathcal{C}_{\mathcal{O}}$ involves an object like “leftmost chair”, the perception tool returns several “chair” detections. The VLM analyzes this feedback (*e.g.*, visualizing bounding boxes) and resolves the ambiguity by determining which object index correctly corresponds to the context (“leftmost”) specified. This closed-loop mechanism allows the agent to handle noisy tool outputs while ensuring the final computation remains strictly grounded in the intent of $\mathcal{C}_{\text{task}}$.

Knowledge-Augmented Code Generation. Once all variables in $\mathcal{C}_{\text{task}}$ are bound to concrete geometric data, the agent invokes a code generator for the final computation. To prevent the coder from hallucinating incorrect formulas, we leverage a knowledge-augmented strategy, which functions analogously to a static Retrieval-Augmented Generation (RAG) [10, 20] system. Specifically, when invoking the code generator, the VLM specifies a high-level requirement and the necessary bound variables (*e.g.*, object’s orientation). Instead of expecting the coder to generate complex geometric formulas from memory, our framework maintains a pre-prepared, fixed library of basic, verified geometric formulas. Based on the data types of the bound variables, the system automatically retrieves the relevant, fixed set of formulas (*e.g.*, object’s local-to-world transformation formula) and injects them directly into the code generator’s context. This ensures the computation steps do not produce black-box guesses, but rather deterministic results, derived from a formally structured task and sound geometric principles. More details are provided in Section E.

3.3.2. Toolbox

We equips the agent with perceptual and computation capabilities required to execute its geometrically-constrained reasoning flow in $\mathcal{F}_{\text{compute}}$, as shown in Figure 2. Detailed APIs for all tools are provided in Section C.2.

Geometry and Perception Tools. These tools are responsible for parameterizing the visual world. “3D Reconstruction” tool leverages foundational models like VGGT [44] to build a unified, high-fidelity 3D representation of the scene. This provides the geometric context required for complex scenarios. This category also contains a suite of 2D perception tools, such as “Object Detection” for open-vocabulary object detection, “Segmentation” for instance segmentation.

Computation and Utility Tools. These tools operate on the data extracted by the perception tools and executes the final deterministic geometric computation. “Python Tool” is the core computational engine, which prompts the VLM to generate and execute Python code in a sandbox environment, using the knowledge-augmented strategy. This category also includes essential utilities (“Utility Tool”). For example, “project_box_to_points” bridges 2D perception to 3D computation by converting 2D bounding boxes into corresponding 3D point clouds.

3.4. Discussion

Our GCA decouples VLM’s spatial reasoning through the formal constraint $\mathcal{C}_{\text{task}}$, jointly addressing two core deficiencies in spatial reasoning.

$\mathcal{F}_{\text{formalize}}$ Solves Flawed Planning and Imagination. Directly solving an ambiguous query forces the VLM to plan and perform spatial imagination within its native lossy semantic space. This is a primary failure mode, as unconstrained planning can lead to geometrically flawed assumptions before any computation even begins. Our paradigm resolves this by reframing the problem. Leveraging VLM’s strength in qualitative semantic interpretation, the $\mathcal{F}_{\text{formalize}}$ stage transform the original spatial query into a deterministic mathematical problem with constraint, preventing the VLM to solve the query in its lossy semantic space directly.

$\mathcal{F}_{\text{compute}}$ Solves Flawed Execution and Computation. In this stage, the VLM acting as the task solver, orchestrating external tools to execute the plan. Crucially, its entire reasoning and execution process is bound by the formal task constraint $\mathcal{C}_{\text{task}}$ generated in $\mathcal{F}_{\text{formalize}}$. This ensures that all subsequent high-precision computations are executed strictly within the deterministic, geometrically sound constraint, effectively bridging the semantic-to-geometric gap.

4. Experiments

4.1. Experimental Setup

Implementation Details. GCA is implemented as a training-free agentic paradigm, requiring no model fine-

Table 1. **Experimental Results on Several Spatial Reasoning Benchmarks.** The best and second best results are shown in **bold** and underlined, respectively. “Avg.” denotes the average of overall accuracy across all benchmarks. More details about these benchmarks’ subcategory (*e.g.*, ‘PR.’) are provided in Appendix.

| | MMSI-Bench | | | | | MindCube-tiny | | | | OmniSpatial | | | SPBench | | | CV-Bench | | | Avg. |
|---------------------------------------|-------------|-------------|-------------|-------------|-------------|---------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| | PR. | Attr. | Mot. | MSR | All | Rot. | Ard. | Amg. | All | Dyn. | Pers. | All | SI | MV | All | 2D | 3D | All | |
| Baseline Foundation VLMs | | | | | | | | | | | | | | | | | | | |
| Qwen3-VL-Thinking [36] | 33.7 | 40.0 | 23.3 | 31.8 | 32.6 | 87.0 | 47.3 | 35.0 | 47.3 | 60.5 | 43.9 | 51.0 | 51.9 | 61.2 | 54.1 | 81.9 | <u>92.6</u> | 86.8 | 54.4 |
| GLM-4.5V [15] | 35.6 | 36.9 | 29.3 | 30.3 | 33.8 | 60.0 | 25.5 | 42.2 | 39.6 | 58.6 | 47.2 | 52.1 | 50.0 | 55.1 | 51.3 | 80.7 | 91.6 | 85.6 | 52.5 |
| GPT-4o [17] | 28.0 | 32.3 | <u>36.0</u> | 30.8 | 30.3 | 33.5 | 35.0 | 37.2 | 35.8 | 58.7 | 46.2 | 51.5 | 42.4 | 48.3 | 43.8 | 69.4 | 84.9 | 76.5 | 47.6 |
| Gemini-2.5-Pro [6] | <u>39.0</u> | 36.2 | 33.3 | 34.3 | <u>36.9</u> | 89.5 | <u>54.5</u> | <u>48.8</u> | <u>57.5</u> | <u>70.7</u> | 44.6 | <u>55.8</u> | 55.6 | 58.3 | 56.3 | 81.2 | 92.5 | 86.3 | <u>58.5</u> |
| Training-based Spatial VLMs | | | | | | | | | | | | | | | | | | | |
| SpatialLLM [30] | 24.5 | 23.1 | 22.7 | 30.8 | 25.3 | 34.0 | 26.8 | 33.0 | 31.1 | 59.6 | 42.9 | 49.5 | 32.2 | 26.4 | 30.7 | 51.3 | 78.6 | 64.5 | 40.2 |
| Spatial-MLLM [48] | 28.5 | 25.4 | 18.0 | 26.3 | 26.1 | 33.8 | 34.5 | 28.3 | 32.1 | 37.2 | 42.1 | 40.0 | 52.0 | 52.0 | 52.0 | 59.5 | 63.3 | 61.2 | 42.3 |
| SpatialLadder [22] | 30.3 | 23.3 | 16.0 | 21.2 | 25.4 | 30.5 | 39.8 | 47.8 | 42.3 | 46.5 | 43.1 | 44.5 | 70.2 | 70.9 | 70.3 | 72.4 | 74.9 | 73.7 | 51.2 |
| Spacer [33] | 29.1 | 29.4 | 21.9 | 22.5 | 26.9 | 29.8 | 30.0 | 26.8 | 28.3 | 53.5 | 40.5 | 46.0 | 48.6 | 59.4 | 51.1 | 74.1 | 77.4 | 75.6 | 45.7 |
| Video-R1 [8] | 30.5 | 25.4 | 22.0 | 26.8 | 27.8 | 30.0 | 30.5 | 41.3 | 35.8 | 50.0 | 44.2 | 46.7 | 44.8 | 40.7 | 43.8 | 73.5 | 74.7 | 74.0 | 45.6 |
| RoboBrain-2.0 [41] | 28.9 | 28.8 | 22.5 | 28.0 | 28.9 | 29.7 | 35.8 | 45.2 | 39.6 | 49.4 | 42.2 | 45.2 | 49.1 | 46.8 | 48.5 | 77.1 | 90.7 | 83.4 | 49.1 |
| VILASR [50] | 35.9 | 26.0 | 21.0 | 23.2 | 29.8 | 34.4 | 25.7 | 29.4 | 29.1 | 37.5 | 42.2 | 40.2 | 50.2 | 57.6 | 51.9 | 75.7 | 77.7 | 76.6 | 45.5 |
| VLaser [52] | 29.8 | 26.9 | 26.0 | 18.9 | 27.3 | 31.5 | 24.8 | 38.2 | 32.6 | 39.1 | 42.6 | 41.1 | 53.2 | <u>69.2</u> | 56.9 | 79.9 | 87.8 | 83.6 | 48.3 |
| Tool-Integrated Spatial Agents | | | | | | | | | | | | | | | | | | | |
| TIGeR [12] | 29.1 | 27.7 | 26.0 | 25.8 | 27.8 | 33.0 | 28.3 | 26.7 | 28.3 | 52.9 | 45.7 | 49.8 | 48.7 | 38.8 | 46.3 | 75.2 | 95.7 | 84.5 | 47.3 |
| GCA (ours) | 52.8 | 45.0 | 44.7 | 38.0 | 47.6 | 82.0 | 61.8 | 59.8 | 64.2 | 73.6 | 58.6 | 65.1 | <u>61.7</u> | 61.9 | <u>61.8</u> | 83.6 | 90.8 | 86.9 | 65.1 |

tuning. It centers on a VLM responsible for both stages of our paradigm: acting as the semantic analyst to generate the $\mathcal{C}_{\text{task}}$ in the $\mathcal{F}_{\text{formalize}}$ stage, and as the task solver to manage a suite of off-the-shelf foundation models for perception and computation [27, 37, 44, 45, 47]. For our primary experiments, we utilize Qwen3-VL-Thinking [36] as the central VLM. To assess the paradigm’s generalizability, we also evaluate other leading VLMs in our ablation studies, including GLM-4.5V [15], GPT-4o [17], and *etc.* All open-source VLMs are deployed using the vLLM inference engine [19] for efficiency. The agent’s architecture is built using Ray [32] for concurrent tool execution and LangGraph for robust state management.

Evaluation Benchmarks and Counterparts. We conduct comprehensive experiments on several spatial reasoning benchmarks. As our current toolbox is primarily designed for image-based inputs, we focus on evaluations that test complex spatial logic from single and multiple images, including MMSI-Bench [55], MindCube-tiny [59], OmniSpatial (Perspective Taking + Dynamic Reasoning) [18], SPBench [22] and CV-Bench [43]. For all benchmarks, we report both overall accuracy (%) and subcategory accuracy (%). We compare our paradigm against several counterparts, including baseline foundation VLMs [6, 15, 17, 36], training-based methods [8, 22, 30, 33, 41, 48, 50, 52] and tool-integrated agents [12].

4.2. Main Results

SOTA Performance. As shown in Table 1, GCA establishes a new state-of-the-art across a wide range of spatial reasoning benchmarks, achieving an average accuracy of 64.8%. Our geometrically-constrained paradigm sur-

passes the strongest foundation VLM baseline (Gemini-2.5-Pro [6] by 12%) and demonstrates a massive lead over other training-based (*e.g.*, SpatialLadder [22] by 27%) or agentic approaches (*e.g.*, TIGeR [12] by 38%). These results strongly validate that our strategy, centered on the $\mathcal{C}_{\text{task}}$, successfully bridges the VLM’s semantic-to-geometric gap.

Effectiveness on Challenging Benchmarks. The advantage of our constrained paradigm is most pronounced on complex, multi-step spatial reasoning benchmarks. For example, on MMSI-Bench, the performance of even SOTA foundation VLMs remain severely limited. Considering its 4-choice questions, most counterparts perform near the 25% random-guess threshold. In contrast, GCA achieves an overall accuracy of 47.6%, surpassing the strongest VLM baseline (Gemini-2.5-Pro) by a 28% relative improvement. A similar trend is evident on other challenging benchmarks like MindCube-tiny, where GCA (64.2%) also significantly outperforms the top baselines. This superior performance stems directly from our paradigm. The introduction of $\mathcal{C}_{\text{task}}$ prevents the VLM from defaulting to flawed semantic shortcuts or falling into a lossy spatial imagination.

Generalizability Across Benchmarks. Our training-free paradigm also demonstrates superior generalizability compared to training-based specialists, which often suffer from biases inherent to their training data. For example, SpatialLadder [22] is fine-tuned on data originating from the same source as the SPBench, leading to a high in-domain score of 70.3%. However, its performance on out-of-domain benchmarks is suboptimal, where GCA consistently outperforms it, often by a margin of \sim 20 points. A similar bias affects TIGeR [12]. While its tools theoretically support multi-view processing, the model is primarily trained on single-

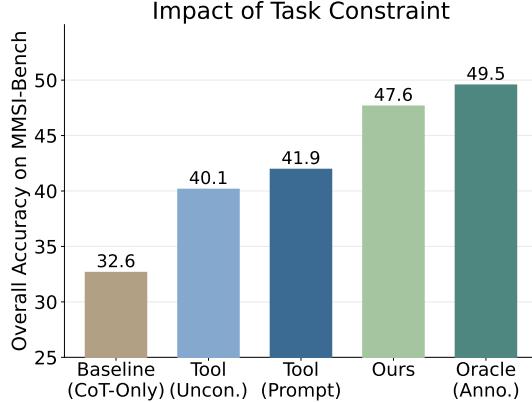


Figure 4. Ablation Study on Formalization. We compare our method in against several baselines: (1) no tool integration (“Baseline (CoT-Only)”), (2) unconstrained tool integration with (“Tool (Prompt)”) or without (“Tool (Uncon.)”) hints, (3) using a human-annotated $\mathcal{C}_{\text{task}}$ (“Oracle (Anno.)”).

image tasks. Consequently, it performs well on single-image benchmarks like CV-Bench but fails on multi-view benchmarks such as MMSI-Bench and MindCube. GCA, in contrast, is not compromised by these training priors and leverages its multi-view tools as dictated by the problem. This demonstrates that our GCA, which forces the VLM to derive a geometrically sound task constraint for each new problem, provides a more robust and generalizable pathway to spatial reasoning.

4.3. Ablation Study

In this section, we conduct extensive ablation studies to dissect the GCA paradigm and validate its core design. Our analysis aims to answer four critical questions. (1) How essential is the formal task constraint $\mathcal{C}_{\text{task}}$? (2) How generalizable is the GCA paradigm across different VLMs? (3) What is the contribution of each system component?

4.3.1. Formalization Analysis

We first investigate the necessity and impact of our core contribution, the $\mathcal{C}_{\text{task}}$ constraint, by comparing our method against different reasoning strategies in Figure 4. The results strongly confirm our central hypothesis. Simply prompting the VLM to “pay attention to the reference frame and objective in the query” (“Tool (Prompt)”) only yields a negligible improvement on unconstrained tool integration. This empirically suggests that the VLM’s unconstrained planning process remains fundamentally flawed and unreliable, even when weakly guided by hints. In comparison, the introduction of our formal $\mathcal{C}_{\text{task}}$ constraint (“Ours”) delivers a substantial performance boost, far surpassing all unconstrained methods. This demonstrates that a deterministic and verifiable constraint is essential for bridging the VLM’s semantic-to-geometric gap, as it forces the VLM

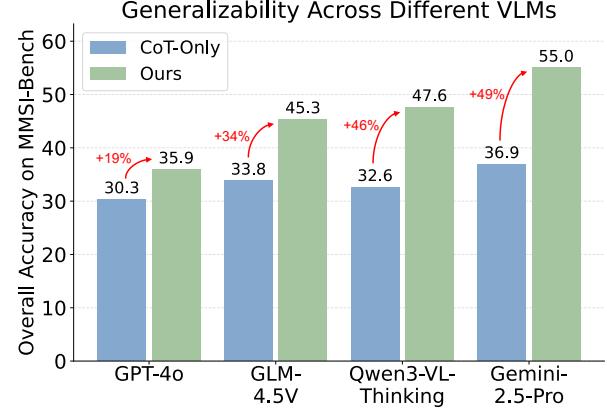


Figure 5. Ablation Study on Generalizability across Different VLMs. Our GCA achieves an average of 37% relative performance improvement across all tested foundation VLMs.

to first establish what to solve before determining how to solve it. Furthermore, we explore the theoretical upper bound using a human-annotated oracle formalization (“Oracle (Anno.)”). The gap between our method (47.6%) and oracle (49.5%) is relative small. As revealed in Section 4.4, the $\mathcal{F}_{\text{formalize}}$ stage achieves $\sim 70\%$ accuracy, confirming the formalization task is well within the VLM’s capabilities.

4.3.2. Generalizability Across VLMs

We assess the generalizability of our GCA paradigm by applying it to several leading foundation VLMs, including GLM-4.5V [15], GPT-4o [17], and Gemini-2.5-Pro [6]. As shown in Figure 5, GCA proves to be a highly generalizable architectural solution, substantially enhancing the spatial reasoning capabilities of every VLM tested compared to their CoT-only baselines. We observe that the magnitude of this enhancement appears to correlate strongly with the VLM’s inherent agentic proficiency and their baseline spatial reasoning capability. It is most evident that Gemini-2.5-Pro, which holds the strongest CoT-only baseline on MMSI-Bench (36.9%), also achieves the most dramatic gain (+49%), rising to 55.0%. On the other hand, the improvement on GPT-4o, while significant, is more modest (+19%). We attribute it to its suboptimal agentic reasoning capability and coding skills. Through introduction of formal task constraint $\mathcal{C}_{\text{task}}$, our paradigm serves as a catalyst, successfully unlocking and guiding the VLM’s powerful execution engine towards the robust spatial reasoning across a diverse set of SOTA models.

4.3.3. Component Contribution

We quantify the importance of each component in the GCA, as presented in Table 2. This analysis reveals improvements in two distinct parts. First, building a standard tool-integrated agent by adding tool integration (+4.2 points), knowledge-augmented code generation (KACG, +1.9 points), and visual feedback (+1.4 points) provides

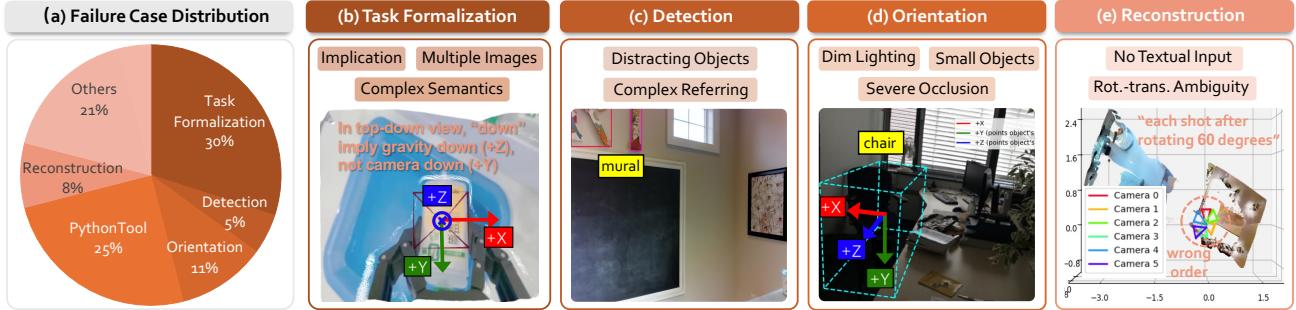


Figure 6. **Error Attribution and Failure Cases.** We provide a detailed error attribution analysis to identify the main failure modes within the VLM’s reasoning trajectory.

Table 2. **Ablation Study on Each Component in GCA.** Here, “KACG” denotes applying knowledge-augmented code generation, and “Feedback” denotes applying the VLM to manage tool feedback and resolve ambiguity.

| Tool Integration | KACG | Feedback | C_{task} | MMSI-Bench |
|------------------|--------------|--------------|-------------------|-------------|
| \times | \times | \times | \times | 32.6 |
| \checkmark | \times | \times | \times | 36.8 |
| \checkmark | \checkmark | \times | \times | 38.7 |
| \checkmark | \checkmark | \checkmark | \times | 40.1 |
| \checkmark | \checkmark | \checkmark | \checkmark | 47.6 |

a cumulative +7.5 points gain over the CoT-only baseline. The second part, the introduction of $\mathcal{F}_{\text{formalize}}$, brings an additional massive improvement, increasing the overall accuracy by +7.5 points. This result strongly validates that constraining the VLM’s planning via a formal C_{task} is essential to prevent flawed reasoning within its lossy semantic space.

4.4. Error Attribution and Failure Cases

A key advantage of GCA paradigm is its verifiable and interpretable nature, which allows us to trace the reasoning pathway and perform detailed error attribution. As shown in Figure 6 (a), this analysis pinpoints the current bottlenecks, attributing failures to either the VLM’s initial formalization or the subsequent tool orchestration.

Errors in $\mathcal{F}_{\text{formalize}}$. Failures in the initial $\mathcal{F}_{\text{formalize}}$ stage account for 30% of all errors. Given this is the first step of the paradigm, it indicates the VLM achieves approximately 70% accuracy in correctly formalizing the task constraint C_{task} . A deeper analysis reveals these failures primarily lie in challenging cases involving complex semantics, ambiguity in multiple images, or ignored implications. For instance, as shown in Figure 6 (b), when asked about a top-down view, the VLM fails to grasp the query’s implication that “down” referred to the direction of gravity, defaulting instead to “camera down” and establishing an incorrect reference frame.

Errors in $\mathcal{F}_{\text{compute}}$. The remaining 70% of errors occur during $\mathcal{F}_{\text{compute}}$ stage. Perception failures ($\sim 24\%$) are a major bottleneck, particularly in “Reconstruction” and “Orientation”. A typical reconstruction failure, shown in Figure 6 (e), is caused by the inability of the underlying VGTT [44] to accept textual input. The query’s textual input, “each shot after rotating 60 degrees” provides a deterministic rotational sequence. However, the VGTT model, which cannot accept this textual input, parameterize the scene incorrectly, resulting in the “wrong order” of cameras and a flawed geometric foundation. Errors from “Python Tool” (25%) are also significant, often stemming from forgotten coordinate transformations or lacking nuanced problem-solving logic, such as identifying a principal direction. Besides, “Other” (21%) errors capture issues like incorrect parameter passing between tools, exhausting the predefined budget (*e.g.*, a maximum of 15 turns), and *etc*.

5. Conclusion

In this work, we introduce GCA, a training-free agentic paradigm designed to bridge the VLM’s semantic-to-geometric gap in spatial reasoning. We address it through leveraging a formal task constraint, transforming the ambiguous spatial query into a deterministic mathematic problem with constraints, preventing the VLM reasoning about the geometric details within its lossy semantic space. As demonstrated experimentally, GCA establishes a new state-of-the-art on multiple challenging spatial reasoning benchmarks, showcasing an effective and generalizable pathway for robust spatial reasoning.

Limitations and Future Prospects. The GCA paradigm, involving iterative tool calls and VLM interactions, is computationally more costly than simple end-to-end CoT reasoning. However, this trade off yields a more robust and verifiable reasoning pathway. Furthermore, we believe the structured outputs from our $\mathcal{F}_{\text{formalize}}$ and $\mathcal{F}_{\text{compute}}$ stages can serve as a valuable source of supervision, such as a process reward in reinforcement learning, for training more ef-

ficient end-to-end spatial VLMs in the future. Besides, our current toolbox is primarily designed for image-based spatial reasoning. A key direction for future work is to extend this geometrically-constrained framework by incorporating tools for temporal reasoning and motion tracking, thereby addressing a broader range of spatial intelligence tasks.

References

- [1] Bikram Pratim Bhuyan, Amar Ramdane-Cherif, Ravi Tomar, and TP Singh. Neuro-symbolic artificial intelligence: a survey. *Neural Computing and Applications*, 36(21):12809–12844, 2024. [3](#)
- [2] Keshigeyan Chandrasegaran, Agrim Gupta, Lea M Hadzic, Taran Kota, Jimming He, Cristóbal Eyzaguirre, Zane Du-rante, Manling Li, Jiajun Wu, and Li Fei-Fei. Hourvideo: 1-hour video-language understanding. *Advances in Neural Information Processing Systems*, 37:53168–53197, 2024. [1](#)
- [3] Zeren Chen, Zhelun Shi, Xiaoya Lu, Lehan He, Sucheng Qian, Zhenfei Yin, Wanli Ouyang, Jing Shao, Yu Qiao, Cewu Lu, et al. Rh20t-p: A primitive-level robotic dataset towards composable generalization agents. *arXiv preprint arXiv:2403.19622*, 2024. [3](#)
- [4] An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Rui-han Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. Spatial-rgpt: Grounded spatial reasoning in vision-language models. *Advances in Neural Information Processing Systems*, 37:135062–135093, 2024. [2](#)
- [5] Jae-Woo Choi, Youngwoo Yoon, Hyobin Ong, Jaehong Kim, and Minsu Jang. Lota-bench: Benchmarking language-oriented task planners for embodied agents. In *The Twelfth International Conference on Learning Representations*, 2024. [12](#)
- [6] Gheorghe Comanici, Eric Bieber, Mike Schaeckermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blisstein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025. [6, 7, 14](#)
- [7] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer, 2003. [14](#)
- [8] Kaituo Feng, Kaixiong Gong, Bohao Li, Zonghao Guo, Yibing Wang, Tianshuo Peng, Junfei Wu, Xiaoying Zhang, Benyou Wang, and Xiangyu Yue. Video-r1: Reinforcing video reasoning in mllms. *Advances in Neural Information Processing Systems*, 2025. [1, 2, 6](#)
- [9] Daocheng Fu, Xin Li, Licheng Wen, Min Dou, Pinlong Cai, Botian Shi, and Yu Qiao. Drive like a human: Rethinking autonomous driving with large language models. In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, pages 910–919. IEEE, 2024. [1](#)
- [10] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jin-liu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1), 2023. [5](#)
- [11] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18995–19012, 2022. [1](#)
- [12] Yi Han, Cheng Chi, Enshen Zhou, Shanyu Rong, Jingkun An, Pengwei Wang, Zhongyuan Wang, Lu Sheng, and Shanghang Zhang. Tiger: Tool-integrated geometric reasoning in vision-language models for robotics. *arXiv preprint arXiv:2510.07181*, 2025. [1, 2, 3, 4, 6](#)
- [13] Yilun Hao, Yang Zhang, and Chuchu Fan. Planning anything with rigor: General-purpose zero-shot planning with llm-based formalized programming. In *The Thirteenth International Conference on Learning Representations*, 2024. [3](#)
- [14] Pascal Hitzler and Md Kamruzzaman Sarker. *Neuro-symbolic artificial intelligence: The state of the art*. IOS press, 2022. [3](#)
- [15] Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Li-hang Pan, et al. Glm-4.1 v-thinking: Towards versatile multi-modal reasoning with scalable reinforcement learning. *arXiv e-prints*, pages arXiv–2507, 2025. [6, 7](#)
- [16] Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *Conference on Robot Learning*, pages 4573–4602. PMLR, 2025. [3, 4, 12](#)
- [17] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. [1, 2, 6, 7](#)
- [18] Mengdi Jia, Zekun Qi, Shaochen Zhang, Wenyao Zhang, Xinqiang Yu, Jiawei He, He Wang, and Li Yi. Omnispatial: Towards comprehensive spatial reasoning benchmark for vision language models. *arXiv preprint arXiv:2506.03135*, 2025. [1, 2, 6, 11, 12, 15](#)
- [19] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626, 2023. [6, 14](#)
- [20] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020. [5](#)
- [21] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Zi-wei Liu, et al. Llava-onevision: Easy visual task transfer. *Transactions on Machine Learning Research*, 2025. [2](#)
- [22] Hongxing Li, Dingming Li, Zixuan Wang, Yuchen Yan, Hang Wu, Wenqi Zhang, Yongliang Shen, Weiming Lu, Jun Xiao, and Yueteng Zhuang. Spatialladder: Progressive train-

- ing for spatial reasoning in vision-language models. *arXiv preprint arXiv:2510.08531*, 2025. 1, 2, 6, 11, 12, 15
- [23] Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Eran Li Li, Ruohan Zhang, et al. Embodied agent interface: Benchmarking llms for embodied decision making. *Advances in Neural Information Processing Systems*, 37: 100428–100534, 2024. 12
- [24] Songtao Li and Hao Tang. Multimodal alignment and fusion: A survey. *arXiv preprint arXiv:2411.17040*, 2024. 1
- [25] Zefu Lin, Rongxu Cui, Chen Hanning, Xiangyu Wang, Jun-jia Xu, Xiaojuan Jin, Chen Wenbo, Hui Zhou, Lue Fan, Wending Li, et al. Embodiedcoder: Parameterized embodied mobile manipulation via modern coding model. *arXiv preprint arXiv:2510.06207*, 2025. 2, 12
- [26] Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023. 3, 4
- [27] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pages 38–55. Springer, 2024. 6, 13
- [28] Xiaoya Lu, Zeren Chen, Xuhao Hu, Yijin Zhou, Weichen Zhang, Dongrui Liu, Lu Sheng, and Jing Shao. Is-bench: Evaluating interactive safety of vlm-driven embodied agents in daily household tasks. *arXiv preprint arXiv:2506.16402*, 2025. 12
- [29] Chenyang Ma, Kai Lu, Ta-Ying Cheng, Niki Trigoni, and Andrew Markham. Spatialpin: Enhancing spatial reasoning capabilities of vision-language models through prompting and interacting 3d priors. *Advances in neural information processing systems*, 37:68803–68832, 2024. 2
- [30] Wufei Ma, Luoxin Ye, Celso M de Melo, Alan Yuille, and Jieneng Chen. Spatialllm: A compound 3d-informed design towards spatially-intelligent large multimodal models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 17249–17260, 2025. 1, 2, 6
- [31] Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. Egoschema: A diagnostic benchmark for very long-form video language understanding. *Advances in Neural Information Processing Systems*, 36:46212–46244, 2023. 1
- [32] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. Ray: A distributed framework for emerging {AI} applications. In *13th USENIX symposium on operating systems design and implementation (OSDI 18)*, pages 561–577, 2018. 6, 14
- [33] Kun Ouyang, Yuanxin Liu, Haoning Wu, Yi Liu, Hao Zhou, Jie Zhou, Fandong Meng, and Xu Sun. Spacer: Reinforcing mllms in video spatial reasoning. *arXiv preprint arXiv:2504.01805*, 2025. 1, 2, 6
- [34] Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. 3
- [35] Mingjie Pan, Jiyao Zhang, Tianshu Wu, Yinghao Zhao, Wenlong Gao, and Hao Dong. Omnimaniip: Towards general robotic manipulation via object-centric interaction primitives as spatial constraints. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 17359–17369, 2025. 3
- [36] QwenTeam. Qwen3-vl: Sharper vision, deeper thought, broader action. <https://qwen.ai/blog?id=99f0335c4ad9ff6153e517418d48535ab6d8afe&from=research.latest-advancements-list>, 2025. 2, 6, 13, 14
- [37] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. In *The Thirteenth International Conference on Learning Representations*, 2024. 6, 13
- [38] Stuart Russell, Peter Norvig, and Artificial Intelligence. A modern approach. *Artificial Intelligence. Prentice-Hall, Englewood Cliffs*, 25(27):79–80, 1995. 3
- [39] Chan Hee Song, Valts Blukis, Jonathan Tremblay, Stephen Tyree, Yu Su, and Stan Birchfield. Robospatial: Teaching spatial understanding to 2d and 3d vision-language models for robotics. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15768–15780, 2025. 2
- [40] Oren Sultan, Eitan Stern, and Dafna Shahaf. Towards reliable proof generation with llms: A neuro-symbolic approach. *arXiv preprint arXiv:2505.14479*, 2025. 3
- [41] BAAI RoboBrain Team, Mingyu Cao, Huajie Tan, Yuheng Ji, Xiansheng Chen, Minglan Lin, Zhiyu Li, Zhou Cao, Pengwei Wang, Enshen Zhou, et al. Robobrain 2.0 technical report. *arXiv preprint arXiv:2507.02029*, 2025. 1, 6
- [42] Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Yang Wang, Zhiyong Zhao, Kun Zhan, Peng Jia, XianPeng Lang, and Hang Zhao. Drivevlm: The convergence of autonomous driving and large vision-language models. In *Conference on Robot Learning*, pages 4698–4726. PMLR, 2025. 1
- [43] Peter Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Adithya Jairam Vedagiri IYER, Sai Charitha Akula, Shusheng Yang, Jihan Yang, Manoj Middepogu, Ziteng Wang, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *Advances in Neural Information Processing Systems*, 37:87310–87356, 2024. 2, 6
- [44] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025. 5, 6, 8, 13
- [45] Ruicheng Wang, Sicheng Xu, Cassie Dai, Jianfeng Xiang, Yu Deng, Xin Tong, and Jiaolong Yang. Moge: Unlocking accurate monocular geometry estimation for open-domain images with optimal training supervision. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5261–5271, 2025. 6
- [46] Ruicheng Wang, Sicheng Xu, Yue Dong, Yu Deng, Jianfeng Xiang, Zelong Lv, Guangzhong Sun, Xin Tong, and Jiaolong

- Yang. Moge-2: Accurate monocular geometry with metric scale and sharp details. *arXiv preprint arXiv:2507.02546*, 2025. 13
- [47] Zehan Wang, Ziang Zhang, Tianyu Pang, Chao Du, Hengshuang Zhao, and Zhou Zhao. Orient anything: Learning robust object orientation estimation from rendering 3d models. In *Forty-second International Conference on Machine Learning*, 2024. 6, 13
- [48] Diankun Wu, Fangfu Liu, Yi-Hsin Hung, and Yueqi Duan. Spatial-mllm: Boosting mllm capabilities in visual-based spatial intelligence. *Advances in Neural Information Processing Systems*, 2025. 1, 2, 6
- [49] Haoning Wu, Xiao Huang, Yaohui Chen, Ya Zhang, Yanfeng Wang, and Weidi Xie. Spatialscore: Towards unified evaluation for multimodal spatial understanding. *arXiv preprint arXiv:2505.17012*, 2025. 1, 2, 3, 4
- [50] Junfei Wu, Jian Guan, Kaituo Feng, Qiang Liu, Shu Wu, Liang Wang, Wei Wu, and Tieniu Tan. Reinforcing spatial reasoning in vision-language models with interwoven thinking and visual drawing. *Advances in Neural Information Processing Systems*, 2025. 1, 2, 6
- [51] Zhenhua Xu, Yujia Zhang, Enze Xie, Zhen Zhao, Yong Guo, Kwan-Yee K Wong, Zhenguo Li, and Hengshuang Zhao. Drivegpt4: Interpretable end-to-end autonomous driving via large language model. *IEEE Robotics and Automation Letters*, 2024. 1
- [52] Ganlin Yang, Tianyi Zhang, Haoran Hao, Weiyun Wang, Yibin Liu, Dehui Wang, Guanzhou Chen, Zijian Cai, Junting Chen, Weijie Su, et al. Vlaser: Vision-language-action model with synergistic embodied reasoning. *arXiv preprint arXiv:2510.11027*, 2025. 1, 2, 6
- [53] Jihai Yang, Shusheng Yang, Anjali W Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10632–10643, 2025. 1, 11, 12
- [54] Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, et al. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. In *Forty-second International Conference on Machine Learning*, 2025. 1, 12
- [55] Sihan Yang, Runsen Xu, Yiman Xie, Sizhe Yang, Mo Li, Jingli Lin, Chenming Zhu, Xiaochen Chen, Haodong Duan, Xiangyu Yue, et al. Mmsi-bench: A benchmark for multi-image spatial intelligence. *arXiv preprint arXiv:2505.23764*, 2025. 1, 2, 4, 6, 11, 12, 14
- [56] Zhutian Yang, Caelan Garrett, Dieter Fox, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Guiding long-horizon task and motion planning with vision language models. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16847–16853. IEEE, 2025. 3
- [57] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022. 4
- [58] Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. Satlm: Satisfiability-aided language models using declarative prompting. *Advances in Neural Information Processing Systems*, 36:45548–45580, 2023. 3
- [59] Baiqiao Yin, Qineng Wang, Pingyue Zhang, Jianshu Zhang, Kangrui Wang, Zihan Wang, Jieyu Zhang, Keshigeyan Chandrasegaran, Han Liu, Ranjay Krishna, et al. Spatial mental modeling from limited views. In *Structural Priors for Vision Workshop at ICCV’25*, 2025. 1, 2, 6, 11, 12, 14
- [60] Songsong Yu, Yuxin Chen, Hao Ju, Lianjie Jia, Fuxi Zhang, Shaofei Huang, Yuhan Wu, Rundi Cui, Binghao Ran, Zabin Zhang, et al. How far are vlms from visual spatial intelligence? a benchmark-driven perspective. *arXiv preprint arXiv:2509.18905*, 2025. 1, 2
- [61] Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, et al. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11142–11152, 2025. 12
- [62] Filippo Ziliotto, Tommaso Campari, Luciano Serafini, and Lamberto Ballan. Tango: Training-free embodied ai agents for open-world tasks. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 24603–24613, 2025. 1, 2

A. Spatial Task Constraint

As introduced in the main paper, the core of Geometrically-Constrained Agent (GCA) paradigm is the **formal task constraint** $\mathcal{C}_{\text{task}}$. It serves as a deterministic bridge to resolve the fundamental semantic-to-geometric gap, effectively decoupling the VLM’s role into a semantic analyst and a constrained task solver. Recall that $\mathcal{C}_{\text{task}}$ is formally defined as a tuple containing two key sub-constraints: the **Reference Frame Constraint** (\mathcal{C}_R), which defines the coordinate system, and the **Objective Constraint** (\mathcal{C}_O), which specifies what to measure within that frame. We guide the Visual Language Model to automatically generate $\mathcal{C}_{\text{task}}$ (for specific prompts used in GCA, refer to Section E). In this section, we first elaborate on the universality of this constraint across different spatial task domains.

A.1. Universality of $\mathcal{C}_{\text{task}}$ in Spatial Tasks

The $\mathcal{C}_{\text{task}}$ is not a rigid definition fixed to a single problem type, but rather a general principle for a wide range of spatial tasks with semantic-to-geometric gap. The core idea is to leverage the VLM’s semantic advantage to formalize the most significant geometric ambiguity of the task. The nature of this ambiguity shifts depending on the task domain.

Spatial Understanding and Reasoning. For spatial reasoning tasks, such as those evaluated in the main paper [18, 22, 53, 55, 59], the objective is typically simple

and explicitly stated in the query (*e.g.*, “what is the relative position?” or “which object is wider?”). Therefore, the objective constraint \mathcal{C}_O is often trivial to formalize. The primary geometric ambiguity lies in the reference frame constraint \mathcal{C}_R . A query like “where is the table relative to you?” is unsolvable until the reference frame (“you”) is geometrically grounded. GCA’s formalization of \mathcal{C}_R (*e.g.*, object-based, camera-based, direction-based frames) is specifically designed to resolve this ambiguity.

Robotic Manipulation and Interaction. Conversely, for robotic manipulation and interaction tasks [5, 23, 28, 54, 61], the reference frame constraint is often simple. The frame is typically the robot’s egocentric perspective or a fixed world frame aligned with the primary camera. The geometric ambiguity shifts entirely to the objective constraint \mathcal{C}_O . A command like “pour tea into the cup” has a trivial \mathcal{C}_R but a highly complex \mathcal{C}_O . The objective is not a simple measurement but a complex, multi-stage procedure involving affordances, contact points, and trajectories. For example, ReKep [16] tackles the manipulation by instructing the VLM to formalize the objective \mathcal{C}_O as a set of *Relational Keypoint Constraints*. These constraints \mathcal{C}_O are literally generated as cost functions written by VLM in Python that map 3D keypoints to a numerical cost. The cost functions are then passed to an inverse kinematics solver ($\mathcal{F}_{\text{compute}}$) to find the optimal robot action. Similarly, EmbodiedCoder [25] formalizes the \mathcal{C}_O as an executable program. The VLM is prompted to first generate code for *geometric parameterization*, fitting point clouds to functional primitives like a rectangle and a hinge axis for a door. Through generating a precise, parametric motion that conforms to the geometric shape just defined, the Python interpreter ($\mathcal{F}_{\text{compute}}$) simply executes the code to produce the final waypoints.

These works are not in conflict with our proposed task constraint $\mathcal{C}_{\text{task}}$. Instead, they can serve as complementary examples of its core principle. They demonstrate how the \mathcal{C}_O for complex manipulation and interaction can be formalized as code, cost functions, or geometric constraints. These constraints are then passed to a solver, just as GCA proposes. This confirms the universality of the $\mathcal{C}_{\text{task}}$: whether the ambiguity lies in the reference frame or the objective, the first and most critical step is to use the VLM’s semantic strength to formalize a deterministic, geometrically-sound constraint to bridge the semantic-to-geometric gap.

A.2. Generalizability of \mathcal{R} Definition

We define three types of reference frame \mathcal{R} in GCA: object-based, camera-based and direction-based reference frame, providing a robust and flexible framework. We find that these categories are sufficient to cover the vast majority of static spatial reasoning queries encountered in existing benchmarks [18, 22, 55, 59]. As spatial reasoning advances

Table 3. **Ablation Study on Task Constraint.**

| Tool Integration | Ref. \mathcal{C}_R | Obj. \mathcal{C}_O | MMSI-Bench |
|------------------|----------------------|----------------------|-------------|
| ✓ | ✓ | ✓ | 47.6 |
| ✓ | ✓ | ✗ | 46.4 |
| ✓ | ✗ | ✓ | 41.0 |
| ✓ | ✗ | ✗ | 40.1 |
| ✗ | ✓ | ✓ | 33.5 |

toward more complex, dynamic, and abstract scenarios, we identify key limitations and challenges for the current implementation of \mathcal{C}_R . These represent important avenues for future research.

Dynamic and Time-Varying Reference Frame. A significant challenge arises in video-based spatial reasoning [53], particularly in tasks involving continuous navigation or long-horizon agent actions. For example, an instruction like, “Move forward to the right first, then move backward to the right, and finally turn left,” involves a \mathcal{C}_R that is continuously changing at each step, contingent on the agent’s previous state. Solving this requires extending the \mathcal{C}_R formalization to become a time-dependent function $\mathcal{C}_R(t)$, capable of tracking and updating the agent’s pose and orientation throughout a sequence.

Frames from Abstract Concepts. This challenge emerges when the reference entity is not a rigid, easily-definable object. A query like “the living room is south of the kitchen” poses a significant problem. It is often impossible to compute a meaningful direction vector between the geometric centroids of two abstract areas or regions. Currently, GCA relies on proxies, for example, if such direction (from kitchen to living room) aligns with the camera’s view from the background to the foreground, we might substitute $-Z_{\text{cam}}$ as the direction vector. This workaround, however, can introduce cumulative errors and lacks generalizability. Future work could explore novel methods to address this. One promising direction is to empower the VLM to directly annotate the reference frame, *i.e.*, outputting two points in the image whose corresponding 3D vector defines the abstract direction.

B. More Ablation Studies

To further explore the proposed formal task constraint $\mathcal{C}_{\text{task}}$ and the stability of GCA, we present three additional ablation studies. These experiments are designed to (1) precisely quantify the relative importance of the reference frame constraint (\mathcal{C}_R) versus the objective constraint (\mathcal{C}_O) for spatial reasoning tasks, (2) validate the constraint’s effectiveness in improving the VLM’s internal, tool-free reasoning, and (3) evaluate the stability and robustness of the GCA paradigm.

Importance of \mathcal{C}_R and \mathcal{C}_O in Spatial Reasoning. To validate our claim in Section A.1 that \mathcal{C}_R represents the primary geometric ambiguity in spatial reasoning tasks, we conduct a detailed ablation on the sub-components of $\mathcal{C}_{\text{task}}$. As shown in Table 3, removing the objective constraint (\mathcal{C}_O) results in a minor 1.2 point performance drop. This demonstrates that for spatial reasoning queries, the objective is often simple and clearly stated, allowing the task solver to infer it from the query during the $\mathcal{F}_{\text{compute}}$ stage. In contrast, removing the reference frame constraint (\mathcal{C}_R) causes a 6.6 point performance drop. This suggests that \mathcal{C}_R is the most critical component in spatial reasoning, as it resolves the core geometric ambiguity of “from where” that VLMs cannot solve in their lossy semantic space.

$\mathcal{C}_{\text{task}}$ without Tool Integration. The $\mathcal{C}_{\text{task}}$ is not a prompt that can fix the VLM’s internal spatial reasoning. Instead, it unlocks the VLM’s agentic reasoning capability and coding skills by constraining the subsequent computation stage ($\mathcal{F}_{\text{compute}}$). This is confirmed by the results in Table 3, where we compare GCA with a VLM that receives $\mathcal{C}_{\text{task}}$ as the prompt but relies solely on chain-of-thought (CoT) reasoning. With the constraint as a textual hint, it only yields a negligible 0.9 point improvement. Even when told the reference frame and objective, the VLM still cannot bypass the internal flawed spatial imagination and high-precision computation in its lossy semantic space.

Stability and Robustness Analysis. A key consideration for any agentic framework, especially one involving multiple VLM calls and tool interactions, is the stability of its results. The probabilistic nature of VLMs could potentially lead to high variance in final performance. To assess the robustness of GCA, we conduct $N = 10$ independent evaluation runs on the complete MMSI-Bench dataset, using the same Qwen3-VL-Thinking [36] for each run. All settings are kept identical as in the Table 1 (main paper). The mean accuracy and the standard deviation across all 10 runs is 47.6 ± 0.3 . The results demonstrate a very low standard deviation, indicating that the GCA framework is highly stable. This stability is a direct benefit of our core design: by forcing the VLM to first generate a deterministic formal task constraint $\mathcal{C}_{\text{task}}$, we significantly reduce the stochasticity and ambiguity in the subsequent $\mathcal{F}_{\text{compute}}$ stage. The task solver operates within the non-negotiable geometric bounds defined by $\mathcal{C}_{\text{task}}$, leading to a consistent and verifiable reasoning pathway.

C. More Implementation Details

C.1. Visual Foundation Models

We deploy several Visual Foundation Models (VFM) for agent to parameterize the visual world, facilitating the deterministic $\mathcal{F}_{\text{compute}}$ stage constrained by $\mathcal{C}_{\text{task}}$.

- **VGGT** [44]. Visual Geometry Grounded Transformer

(VGGT) is a large feed-forward model that infers key 3D attributes from one or multiple images. It predicts camera parameters, point maps, and depth maps for all input views. Within GCA, VGGT serves as the primary geometry parameterization engine for 3D reconstruction.

- **MoGe-2** [46]. The Monocular Geometry (MoGe) estimation model is designed to recover 3D point maps with metric scale from a single image. It achieves this by decoupling the problem, predicting both an affine-invariant point map and a separate global scale factor. GCA leverages this unique capability to derive the correct real-world scale for the scene.
- **GroundingDINO** [27]. GroundingDINO is an open-set object detector that combines a transformer-based detector with grounded language pre-training. This architecture enables it to detect arbitrary objects specified by natural language, such as category names or referring expressions. It serves as a specialized detection tool in GCA.
- **SAM-2** [37]. SAM-2 is a foundation model for promptable visual segmentation in both images and videos. It generalizes the original SAM by incorporating a streaming memory architecture to handle temporal data. GCA uses SAM-2 as a bridge connecting pixels and boxes, allowing us to extract object point clouds from the VG-GT output based on the boxes.
- **Orient Anything** [47]. Orient Anything is trained on rendered 3D models to estimate the 3D orientation of an object from a single, free-view image. It predicts the object’s azimuth, polar, and rotation angles relative to the camera. This capability is crucial for GCA to construct a object-based reference frame.

Note that these foundation models are not provided directly to the agent. Instead, we wrap them and offer some abstract tool interfaces as APIs for invocation (see Section C.2).

C.2. Tool Interfaces

The GCA agent’s $\mathcal{F}_{\text{compute}}$ stage is driven by a discrete set of 8 exposed tool APIs. These APIs form the agent’s action space, encapsulating the underlying VFMs.

- **reconstruct**. It ingests one or more images and produces a comprehensive 3D reconstruction. Internally, it leverages VGGT and automatically selects the optimal reconstruction strategy. If multiple, non-static images are provided, it first consults the VLM to identify common static objects for alignment. The output includes the 3D world points, camera extrinsics, and intrinsics.
- **detect**. It detects target objects in a single image based on a text prompt. For capable VLMs like Qwen3-VL-Thinking [36], we directly instruct the VLM itself to locate the target object through prompts. Otherwise, we use GroundingDINO as the detector. It returns the bounding boxes and corresponding labels.

- `project_box_to_3d_points`. It takes a 2D bounding box and projects it into the 3D world coordinate system defined by the VGGT model output. Internally, it first uses SAM-2 to convert the bounding box into a precise pixel mask, then filters the points using this mask.
- `predict_obj_pose`. It computes the 6-DoF semantic pose of an object, which is essential for establishing an object-based reference frame. This tool first calls `project_box_to_3d_points` to find the object’s 3D centroid, and then calls Orient Anything to determine its 3D orientation. It then combines these to return the final object-to-world transformation matrix.
- `estimate_scale`. This tool is called when metric measurements (*e.g.*, “meters”, “feet”) are required. It aligns MoGe-2’s metric depth with the VGGT model’s relative depth prediction to compute a single scale factor that converts the entire reconstruction into meters.
- `ocr`. It performs optical character recognition (OCR) on an image using the EasyOCR library. It returns a list of recognized texts and their bounding boxes.
- `analyze_motion`. It analyzes pixel-level motion between two sequential images using a Farneback optical flow algorithm [7]. It is used to infer subtle camera movements that may be too small for full 3D reconstruction.
- `code`. This is the agent’s primary computation engine. It generates and executes Python code within a sandbox environment. It tasks a set of context variables (*e.g.*, poses, points) and a natural language description (*e.g.*, request and description of the variables) as input. The code is generated using a knowledge-augmented strategy, where relevant geometric formulas are injected into the prompt, ensuring the computation is both deterministic and sound.

C.3. Agentic Framework

The GCA paradigm is implemented as a high-throughput, modular system. The core VLM deployment and the perception tool suite are physically decoupled to ensure scalability and robustness.

System Backend and State Management. The entire system is built using Ray [32] and LangGraph. LangGraph is used to define and manage the agent’s state and orchestrate the two-stage, graph-based reasoning flow (*i.e.*, $\mathcal{F}_{\text{formalize}}$ followed by the $\mathcal{F}_{\text{compute}}$ loop).

Tool Suite and VFM Deployment. The perceptual and computational tools (listed in Section C.2) are encapsulated as independent Ray Serve actors. This microservice architecture allows GCA to make concurrent perception requests (*e.g.*, running reconstruct and detect in parallel), enabling high parallelism and automatic scaling. This entire tool suite is deployed on 2 NVIDIA A100 GPUs.

VLM Roles and Deployment. In GCA, a single VLM fulfills the three distinct roles within the GCA paradigm:

- **Semantic Analyst.** In the $\mathcal{F}_{\text{formalize}}$ stage, it interprets the query and visual context to generate the formal $\mathcal{C}_{\text{task}}$.

- **Tool Orchestrator.** In the $\mathcal{F}_{\text{compute}}$ stage, it manages the ReAct-style tool call loop, resolves ambiguities, and generates natural language descriptions for the coder.

- **Coder.** It generates Python code for the `code` tool.

The VLM deployment is separate from the tool suite. For open-source models (*e.g.*, Qwen3-VL-Thinking [36]), we use vLLM [19] for efficient, high-throughput inference on 8 NVIDIA A100 GPUs. For closed-source models (*e.g.*, Gemini-2.5-Pro [6]), we access them via their standard commercial APIs. We employ different sampling parameters based on the VLM’s role. For the semantic analyst and tool orchestrator roles, which require reasoning and flexibility, we use `TEMPERATURE=0.6` and `TOP_P=0.95`. For the coder role, which demands deterministic and reliable output, we set `TEMPERATURE=0.0`. All roles use `MAX_TOKENS=32768`.

D. Evaluation Benchmark Details

We evaluate GCA on multiple challenging spatial reasoning benchmarks. This section provides a detailed description of each benchmark and its subcategories, corresponding to the results presented in Table 1 of the main paper.

D.1. MMSI-Bench

MMSI-Bench [55] is a comprehensive benchmark designed to evaluate a VLM’s ability to perform spatial reasoning by integrating information from multiple, distinct images. It is organized into four distinct subcategories:

- **PR. (Positional Relationship).** This subcategory evaluates the model’s ability to understand the relative positions between different objects, cameras and semantic regions (*e.g.*, a kitchen) across multiple views.
- **Attr. (Attribute).** This subcategory evaluates the model’s ability to identify object attributes related to spatial properties, such as geometric properties (*e.g.*, size, length) or visual characteristics (*e.g.*, shape).
- **Mot. (Motion).** This subcategory evaluates the model’s ability to understand object or camera’s movement.
- **MSR (Multi-Step Reasoning).** This subcategory evaluates the model’s ability to perform complex reasoning by chaining multiple spatial understandings described above together to arrive at a final answer.

D.2. MindCube-tiny

MindCube-tiny is a subset of the MindCube benchmark [59], which is designed to test Spatial Mental Modeling (SMM). The core task evaluates a VLM’s ability to construct and manipulate a 3D mental model of a scene using only a limited set of 2D images as input. The “tiny”

version is a smaller-scale version of the full benchmark. We evaluate on its three primary sub-tasks:

- **Rot. (Rotation).** This task requires the model to infer the complete environment based on partial visual information, testing its understanding of sequential views and consistent spatial cues between images (*e.g.*, lighting).
- **Ard. (Around).** This task requires the model to infer the scene from a novel viewpoint, testing its ability to interpolate and extrapolate its mental 3D model.
- **Amg. (Among).** This task requires the model to infer the 3D spatial arrangement based on four orthogonal views characterized by significant occlusion, testing its ability to establish consistency relationships across perspectives and reason about the relative positions of unseen objects.

D.3. OmniSpatial

OmniSpatial [18] is a comprehensive benchmark designed to evaluate a broad spectrum of visual-based spatial intelligence capabilities in VLMs. The full benchmark consists of four categories. We primarily focus on the two subcategories most closely related to geometric perception:

- **Pers. (Perspective Taking).** This subcategory assesses the model’s understanding of 3D spatial relationships by adopting varied viewpoints, *e.g.*, egocentric, allocentric, and hypothetical perspective.
- **Dyn. (Dynamic Reasoning).** This subcategory assesses the model’s understanding of object motion and judgments in uncertain or rapidly changing environments.

We exclude the other two subcategories: “Spatial Interaction” (which focuses on diagrams and user-interface, *e.g.*, terrain map) and “Complex Logic” (which involves abstract spatial reasoning, *e.g.*, puzzles), as they are more centered on abstract or symbolic reasoning rather than the high-fidelity geometric perception that GCA is designed to solve.

D.4. SPBench

SPBench [22] is a benchmark designed to evaluate both VLM’s spatial reasoning in single view and multiple views:

- **SI (Single Image).** This subset contains questions that test the model’s understanding and reasoning capabilities within a single image, including absolute distance, object size, relative distance, and relative direction.
- **MV (Multiple Views).** This subset requires the model to integrate information from multiple viewpoints to answer questions about relative position and object counts within a overlapping scene.

D.5. CV-Bench

CV-Bench is a visual-centric benchmark that evaluates the spatial understanding capabilities of VLMs. It is broadly composed of 2D and 3D reasoning tasks:

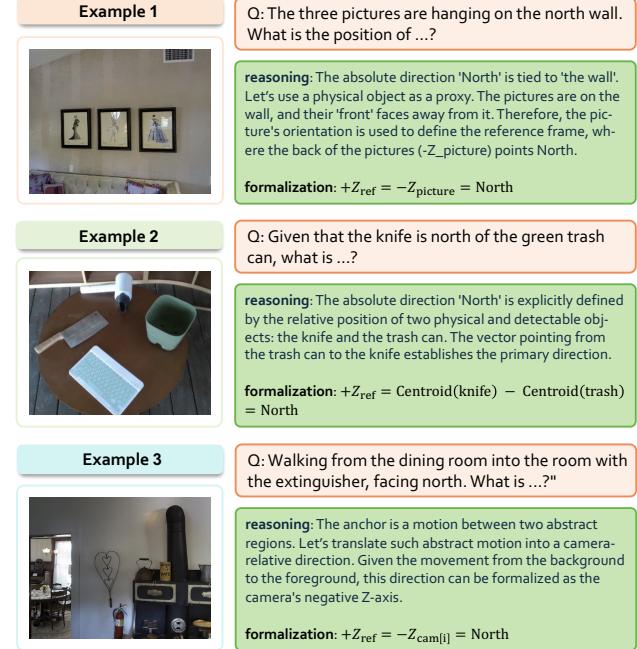


Figure 7. In Context Examples Used in Formalizing Reference Frame. The output format follows the prompt in Table 4.

- **2D (2D Relationship).** This subcategory tests fundamental 2D spatial understanding, including 2D positional relationships and object counting.
- **3D (3D Relationship).** This subcategory assesses the model’s grasp of 3D concepts, such as depth analysis and 3D distance comparisons between objects in the scene.

E. Prompts Used in GCA

We provide detailed prompts used in GCA, including task formalization (Table 4 and 5), tool orchestration (Table 6) and knowledge-augmented code generation (Table 7 and 8). Besides, we also provide the in context examples used in the reference frame formalization (see Figure 7 and Table 4).

F. Qualitative Case Study

We provide several qualitative case studies on how GCA effectively tackles spatial reasoning queries. These challenging cases includes unique object counting across multiple views (Figure 8), direction-based reference frame (Figure 9), object-based reference frame (Figure 10), camera rotation analysis (Figure 11), object movement analysis (Figure 12), and metric-scale estimation (Figure 13).

G. Broader Impacts

Advancing Embodied AI Systems. Applications in robotics and AR/VR depend on an agent’s ability to translate ambiguous human instructions (*e.g.*, “sit on the sofa”)

into precise geometric actions. GCA provides a robust framework for this translation, potentially facilitating the development of robots and AR/VR interfaces that can interact with the physical world with high fidelity.

Trust, Interpretability, and Verification. Unlike opaque end-to-end spatial VLMs, GCA’s reasoning process is highly traceable. $\mathcal{C}_{\text{task}}$ serves as an explicit, human-readable artifact that can be verified before a high-stakes action is executed. This verify-then-execute capability is critical for safety in real-world applications. Furthermore, the structured outputs from both the $\mathcal{F}_{\text{formalize}}$ and $\mathcal{F}_{\text{compute}}$ stages can serve as a valuable source of process-level supervision, acting as a reliable validator to guide the training of more efficient end-to-end spatial VLM.

Inheritance and Amplification of Bias. The reliance on VFM_s for perception and geometry (*e.g.*, 3D reconstruction, object orientation) creates a new dependency chain for bias and failure. If these perceptual tools perform poorly on objects, scenes, or lighting conditions, GCA will not only inherit these biases but may also amplify them, leading to incorrect outcomes in real-world interactions.

Table 4. Prompts Used for Formalizing Reference Frame Constraint. Here, {example} is the in-context examples (see Figure 7), and {question} is the placeholder that will be replaced.

[CORE MISSION]

You are an expert spatial reasoning analyst. Your sole mission is to analyze a user's question and define the final **Reference Frame**. Your goal is to find the single element (an object, a camera, or a vector) that provides the ultimate, non-negotiable definition for absolute directions (e.g., North, South) or relative perspectives (e.g., "front", "left") in the final answer. Ask yourself: "*What element holds the final authority on what 'north', 'front', or 'left' means in the question?*"

[OUTPUT FORMAT]

Your response **MUST** be a single, valid JSON object:

```
```json
{
 "reasoning": "A brief, step-by-step logical deduction, explaining WHY this anchor
 is the arbiter of direction.",
 "formalization": "The precise mathematical mapping of a semantic direction to one
 of the Solvable Geometric Primitive listed below, e.g.,
 $-Z_cam0, +Z_toaster$."
}
```

```

[FORMALIZATION]

1. Identify the Final Arbiter of Direction

- **Priority 1: Absolute Direction.** If an absolute direction (North, South, etc.) is explicitly tied to an element, that element is the arbiter, overriding everything else.

- **Priority 2: Relative Query.** If no absolute direction is given, the arbiter is the object of the relative question.

2. Formalize Reference Frame Using Solvable Geometric Primitives:

o construct a mathematical formalization of reference frame, you **MUST** use following three types of **Solvable Geometric Primitives**:

- **A. Camera Axes:** A vector from a specific camera's coordinate system. Format: $\pm X_{cam[i]}, \pm Y_{cam[i]}, \pm Z_{cam[i]}$. The camera coordinate system follows OpenCV convention: +Z points forward, +Y points down, and +X follow right-hand rules.

- **B. Object Axes:** A vector from a specific object's semantic coordinate system. Format: $\pm X_{[obj]}, \pm Y_{[obj]}, \pm Z_{[obj]}$. The object's local coordinate system is defined by: +Z points its semantic "front", +Y points its semantic "down", +X follows right hand rules, and origin at centroid.

- **C. Inter-Object Vector (Direction):** A vector connecting the centroids of two concrete, detectable objects. Format: Centroid(B) – Centroid(A).

3. Semantic Formalization

- **A. Object-based Reference Frame:** Usually can be defined by corresponding object's axes. Examples:

- "when using the toaster" suggests user's "forward" is opposite the toaster's semantic "front", i.e., $+Z_{ref} = -Z_{toaster}$.
- You must choose a **Physical and Detectable** object as the object anchor. Don't use abstract concepts like room/region/area.

- **B. Camera-based Reference Frame:** Usually can be defined by corresponding camera's axes. Examples:

- "from the perspective of Figure 1" suggests reference frame is identical to camera 0's, i.e., $+Z_{ref} = +Z_{cam0}$.

- **C. Direction-based Reference Frame**

- For spatial relationship between two **Physical and Detectable Objects**, it can be defined by inter-object vector. Examples: "object A is north of object B" suggests the direction from object B to A is north, i.e., $+Z_{ref} = \vec{BA} = \text{Centroid}(A) - \text{Centroid}(B) = \text{North}$

- For spatial relationship between two **Abstract Concepts**, you must use a physic object's axes or a camera's axes as the proxy to tie this direction. Examples: "moves from room A to room B, facing north". Assume this motion aligns with moving from background towards the foreground, formalized as $+Z_{ref} = -Z_{cam[i]} = \text{North}$.

[EXAMPLES]

{examples}

[QUESTION]

{question}

Now, please analyze the above question and provide your response in the specified JSON format.

Table 5. **Prompts Used for Formalizing Objective Constraint.** Here, {question} is the placeholder that will be replaced.

[CORE MISSION]

You are an expert spatial reasoning analyst. Your sole mission is to analyze a user's question and define the final **Objective**. Your goal is to rephrase the user's natural-language question into a single and precise sentence. This sentence describes the specific value or piece of information that definitively answer the question. Ask yourself: "*What is the single, final piece of information (e.g., a scalar value, a 3D vector; a sequence of rotations) that the user is finding?*"

[OUTPUT FORMAT]

Your response **MUST** be a single, valid JSON object:

```
'''json
{
    "reasoning": "A brief, step-by-step logical deduction that breaks down the user's
                  question into its final objective.",
    "formalization": "A single, concise sentence that defines the ultimate goal of the
                     question, stated in technical terms."
}
'''
```

[OBJECTIVE]

- **Identify the target variable:** What type of answer is being sought? Is it a distance, a speed, an orientation, a direction, a count, a relationship, or a sequence of actions?
- **Identify the Key Entities:** What are the specific, concrete objects, cameras, or locations involved in the question?
- **Synthesize the Objective:** Combine the target variable and entities into a single, unambiguous sentence. This sentence must be a statement or a noun phrase, not a question. Example:
 - Bad: Which way the object are going?
 - Good: The 3D direction vector of the object' movement.

[QUESTION]

{question}

Now, please analyze the above question and provide your response in the specified JSON format.

Table 6. **Prompts Used for Tool Orchestration.** Here, {api_documents} is the detailed documentation of provided APIs. {history} includes the initial user question, task formalization, previous planning and corresponding execution results.

[CORE MISSION]

You are an expert spatial intelligence agent. Your mission is to generate a sequence of tool calls that rigorously computes the answer. It follows an iterative Plan → Update cycle, using a workspace as your computational memory.

- **Plan:** Decide the next tool calls based on the goal and current workspace.
- **Update:** Tool results are saved as new variables in the workspace.
- **Repeat:** Continue until the workspace contains enough information to conclude the final answer.

[AVAILABLE APIS]

{api_documents}

[A TYPICAL WORKFLOW]

1. Strictly Follow the Task Formalization

- **Task Formalization:** The input question is pre-formalized and consists of two parts: **Reference Frame Constraint** and **Objective Constraint**. You MUST strictly follow this formalization to solve the question.

- **Reference Frame:** Reference frame is the only one coordinate system that matters for interpreting the final answer (left/right, north/south, etc.). The “formalization” is the equation you must solve with the specified geometric tools. E.g., $+Z_{\text{ref}} = -Z_{\text{toaster}}$ indicates reference frame is defined by object toaster’s local frame, so we MUST perform all calculations in toaster’s frame.

- **Objective:** Objective is the ultimate goal of the question. You must calculate this objective within the reference frame.

2. Acquire Geometric Data: Based on user’s question and pre-defined task formalization, plan the necessary tool calls to gather all data required for the final calculation. This involves two parallel goals:

- **A. Solve for the Reference Frame:** The formalization mathematically defines the **World-to-Reference Transformation**. To solve this formalization, your plan MUST gather all the geometric ingredients in the world frame.

- “reconstruct” tool provides the 3D reconstruction context in a unified world frame, bridging the gap between input 2D images and geometric perception.

- If formalization involves an object’s axes (e.g., $+Z_{\text{ref}} = -Z_{\text{toaster}}$), call “predict_obj_pose” to solve that object’s local frame. The resulting “T_obj2world” is required to establish the reference frame.

- If formalization involves a camera’s axes (e.g., $+Z_{\text{ref}} = -Z_{\text{cam0}}$), you must acquire reconstruction context (include extrinsic matrix) to implement the formalization.

- If formalization involves a vector between two objects (e.g., Centroid(B) – Centroid(A) = North), your plan MUST include calls to “project_box_to_3d_points” for both object A and B.

- **B. Solve for the Objective:** Follow the objective to identify the target data that need to be analyzed within the reference frame.

3. Perform Final Calculation in Reference Frame: Once all required variables are available in the workspace, call “code”.

4. Conclusion: Conclude the final answer using “generate_final_answer”.

[OUTPUT FORMAT]

Your response **MUST** be a single, valid JSON object:

```
'''json
{
    "analysis": "Briefly analyze how you will implement the formalization and what
                target data is need. State the immediate next tool(s) you will call",
    "tool_calls": [
        {
            "api": "API name", "args": {...},
            "output_variable": "A unique name for output, stored in the workspace"
        }, ...
    ]
}
'''
```

[HISTORY]

Here is the history so far:

{history}

Please analyze current situation and history messages, and then generate your response. Your plan **MUST** only includes the immediate next one step.

Table 7. **Prompts Used for Coder.** Here, {question}, {formalization}, {objective} and {var_docs} are the placeholder that will be replaced. {knowledge} includes a set of relevant, fixed formulas based on the type of input variables.

[CORE MISSION]

You are an expert Python programmer. Your goal is to write a single Python function that correctly implements the computational objective based on the provided context and documentation.

[User's Question]

This provides the high-level context for your task.

{question}

[Reference Frame]

All geometric data are defined in the world frame (defined by camera 0) unless specified. All final interpretations MUST be expressed in the reference frame. The reference frame is defined:

{formalization}

[Objective]

The ultimate goal from the high-level question. You MUST write code to calculate this objective to answer the question.

{objective}

[Documentation of Available Variables]

{var_docs}

[Additional Knowledge]

This information is always true for the environment your code runs in.

{knowledge}

[Available Libraries]

You can use “numpy”, “torch”, “scipy”, “math”, and other standard Python libraries.

[Critical Rules and Output Format]

1. Synthesize and Self-Correct: Your primary duty is to write correct code. Use the objective as your goal, but critically verify and implement the logic using the provided documentation.

2. Handle Multiple-Choice Questions: If the user’s question is multiple-choice, your code MUST systematically evaluate the conditions for every option (e.g., A, B, C, D). Besides, the logic of your code should focus ONLY on the given options.

3. DO NOT add any explanation in your final output. Your output MUST follow this format:

```
'''python
def execute(func_signature):
    # Your code here
    ...
    return serializable_value  # return value MUST be a serializable type
'''
```

Table 8. **Knowledge and Formulas Used in Knowledge-Augmented Code Generation.** We will inject the relevant knowledge based on the type of input variable.

[Output of “reconstruct”]

Extrinsic Transformation (World \leftrightarrow Camera):

- **World \rightarrow Camera:** To transform a world point “P_world” into camera s’s frame, use its extrinsic matrix $E_s = \text{extrinsic}[s]$. The formula is: $P_{\text{cam_homo}} = P_{\text{world_homo}} @ E_s.T$.
- **Camera \rightarrow World:** The pose of camera “s” in the world, “Pose_s”, is the inverse of its extrinsic matrix: $\text{Pose}_s = \text{np.linalg.inv}(\text{extrinsic}[s])$. To transform a point from camera s’s local frame to the world frame, use the formula: $P_{\text{world_homo}} = P_{\text{cam_homo}} @ \text{Pose}_s.T$.
- **Relative Rotation Analysis (Camera \rightarrow Camera):** To describe the rotation of camera j’s pose relative to camera i’s pose, use the camera poses in the world frame (“Pose_i”, “Pose_j”). The relative rotation from i to j is: $R_{\text{rel}} = R_{\text{j_pose}} @ R_{\text{i_pose}}.T$, which simplifies to $R_{\text{rel}} = (R_{\text{j}}.T) @ (R_{\text{i}}.T).T = R_{\text{j}}.T @ R_{\text{i}}$.

[Output of “predict_obj_pose”]

Object Pose Transformation (World \leftrightarrow Object):

- **Object \rightarrow World:** The “T_obj2world” matrix (aliased as “Pose_obj”) transforms points from the object’s local frame to the world frame using the formula: $P_{\text{world_homo}} = P_{\text{local_homo}} @ \text{Pose}_{\text{obj}}.T$.
- **World \rightarrow Object:** To transform “P_world” into the object’s local frame, use the inverse matrix: $T_{\text{world_to_obj}} = \text{np.linalg.inv}(\text{Pose}_{\text{obj}})$. The formula is: $P_{\text{local_homo}} = P_{\text{world_homo}} @ T_{\text{world_to_obj}}.T$.

[Output of “reconstruct”, “predict_obj_pose”]

Interpreting Rotation:

- **General Rotation Direction:** For most questions about rotation direction, convert the relative rotation matrix to a rotation vector [rx, ry, rz] via `scipy.spatial.transform.Rotation.from_matrix(R).as_rotvec()`.
 - The component with the largest absolute value indicates the primary axis of rotation.
 - Based on the OpenCV coordinate system (+X right, +Y down, +Z forward) and the right-hand rule: ry > 0 corresponds to a pan to the right, rx > 0 corresponds to a tilt upward, rz > 0 corresponds to a clockwise roll.
- **Sequential Rotations:** Use `scipy.spatial.transform.Rotation.from_matrix(R).as_euler(order)` to compute sequential rotation. Verify the signs of the resulting angles. It must match the option’s description.

[If formalization includes cardinal direction]

1. Identify the Cardinal Anchor Axis from the formalization.

- The formalization string (e.g., $+Z_{\text{ref}} = -Z_{\text{obj}}$ = South) links one of your reference axes to a cardinal direction.
- Parse this string to find the anchor. In the example, the anchor is South, and it corresponds to the $Z_{\text{ref_axis}}$. This defines your first cardinal vector: $\text{South_axis} = Z_{\text{ref_axis}}$.

2. Derive the Complete Set of Cardinal Axes: You must find the remaining cardinal axes via applying cross product:

- If South_axis is known, starting with $\text{West_axis} = \text{np.cross}(Y_{\text{ref_axis}}, \text{South_axis})$.
- If West_axis is known, starting with $\text{North_axis} = \text{np.cross}(Y_{\text{ref_axis}}, \text{West_axis})$.
- If North_axis is known, starting with $\text{East_axis} = \text{np.cross}(Y_{\text{ref_axis}}, \text{North_axis})$.
- If East_axis is known, starting with $\text{South_axis} = \text{np.cross}(Y_{\text{ref_axis}}, \text{East_axis})$.

3. Project and Determine the Final Quadrant: Project the target vector onto the primary horizontal cardinal axes (North and East).

- `projection_north = np.dot(disp_vec_world, cardinal_map["N"])`
- `projection_east = np.dot(disp_vec_world, cardinal_map["E"])`
- Use the signs of these projections to determine the final answer.

[Output of “detect”]

Bounding Box Format: All bounding box in input variable is provided in the `[x1, y1, x2, y2]` format.

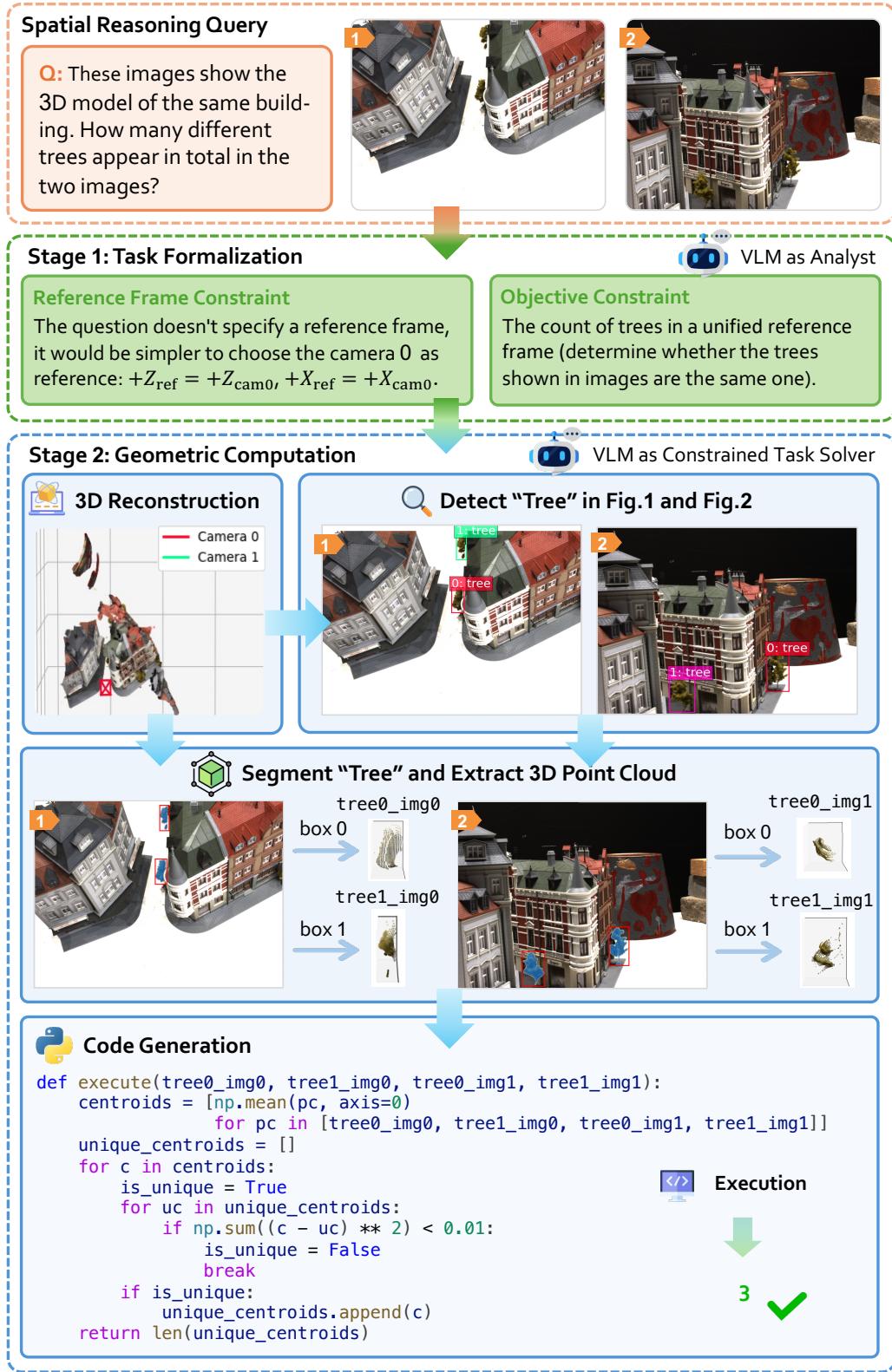


Figure 8. Case Study #1. Unique object counting across multiple views.

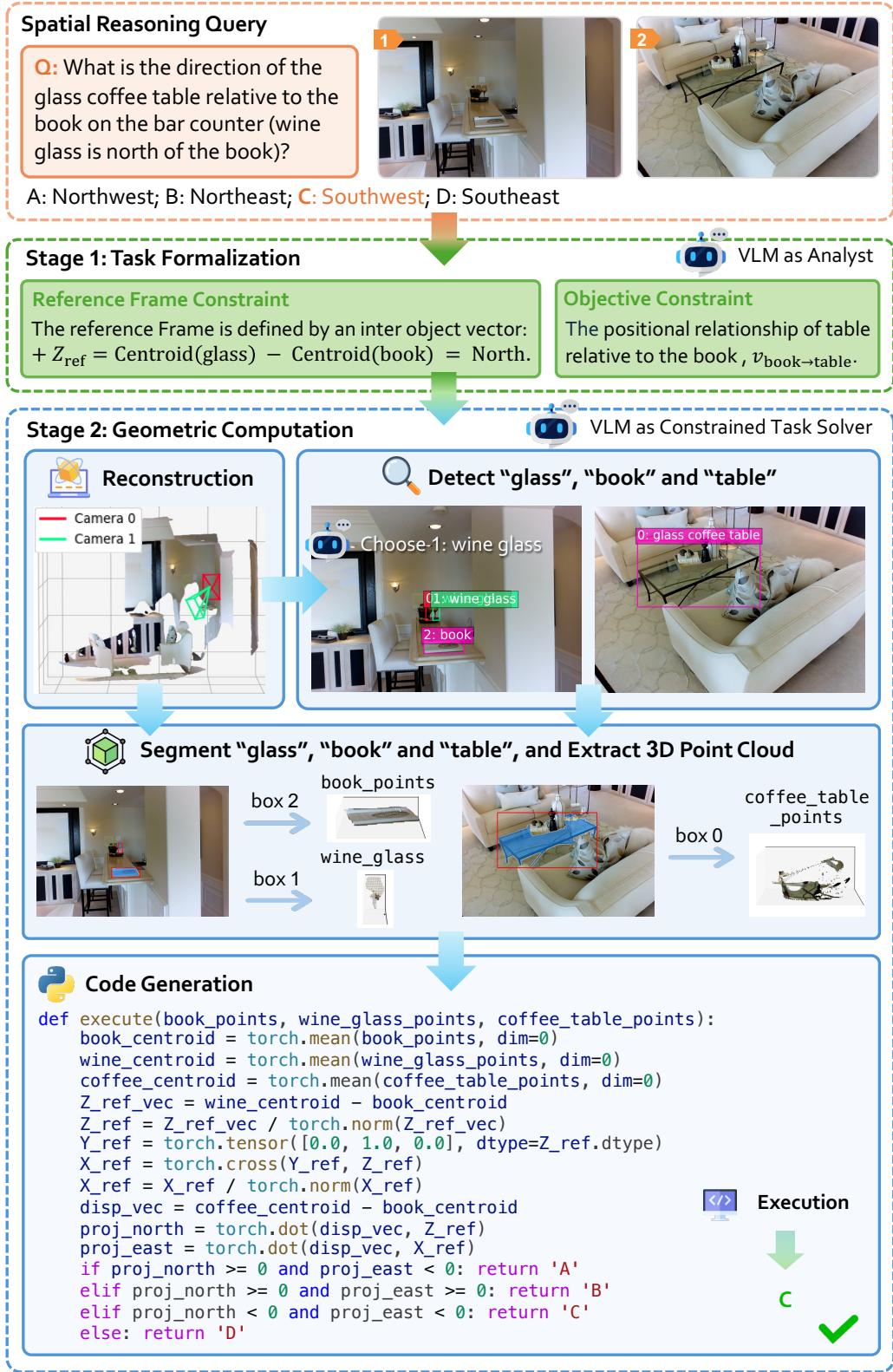


Figure 9. Case Study #2. Direction-based reference frame.

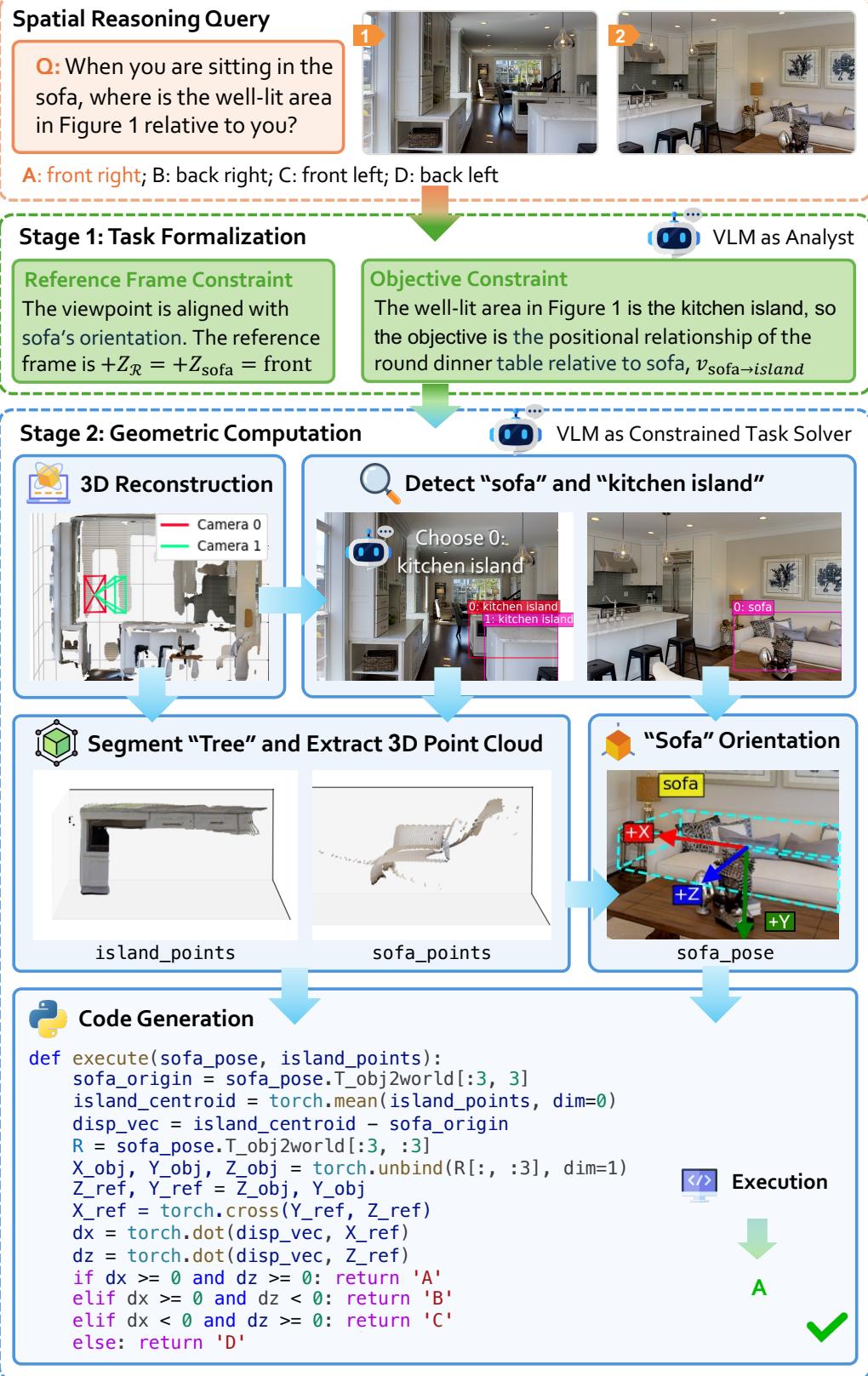


Figure 10. **Case Study #3.** Object-based reference frame

Spatial Reasoning Query

Q: The camera coordinate system is defined as +Y up, -Z forward, and is a right-handed system. How can the viewpoint in the second image be obtained from the viewpoint in the first image in its corresponding camera coordinate system?



- A: Rotate a positive angle around the Z axis, then a positive angle around the Y axis
- B: Rotate a positive angle around the Y axis, then a negative angle around the X axis
- C: Rotate a positive angle around the Y axis, then a positive angle around the X axis**
- D: Rotate a positive angle around the Z axis, then a negative angle around the Y axis

Stage 1: Task Formalization

Reference Frame Constraint

The question asks for the viewpoint transformation, establishing cam_0 as the anchor. The custom coordinate system (+Y up, -Z forward) must be mapped to cam_0: $+Z_{\text{ref}} = -Z_{\text{cam0}}$, $+Y_{\text{ref}} = -Y_{\text{cam0}}$.

VLM as Analyst

Objective Constraint

The sequence of axis rotations required to transform the camera viewpoint from the first image to the second image.

Stage 2: Geometric Computation

VLM as Constrained Task Solver



Code Generation

```

def execute(extrinsic):
    R_0, R_1 = extrinsic[0][:3, :3], extrinsic[1][:3, :3]
    T = np.diag([1, -1, -1]) # y_user = -y_cam0, z_user = -z_cam0
    R_user = T @ R_1.T @ R_0 @ T

    options = {
        'A': {'order': 'zyx', 'signs': [1, 1]}, 'B': {'order': 'yxz', 'signs': [1, -1]},
        'C': {'order': 'yxz', 'signs': [1, 1]}, 'D': {'order': 'zyx', 'signs': [1, -1]}
    }
    for opt, params in options.items():
        angles = Rotation.from_matrix(R_user).as_euler(params['order'], degrees=False)
        sign1_match = (params['signs'][0] == 1 and angles[0] > 1e-6) or \
                      (params['signs'][0] == -1 and angles[0] < -1e-6)
        sign2_match = (params['signs'][1] == 1 and angles[1] > 1e-6) or \
                      (params['signs'][1] == -1 and angles[1] < -1e-6)
        if sign1_match and sign2_match:
            return opt
    return None

```

Execution



Figure 11. Case Study #4. Camera rotation analysis.

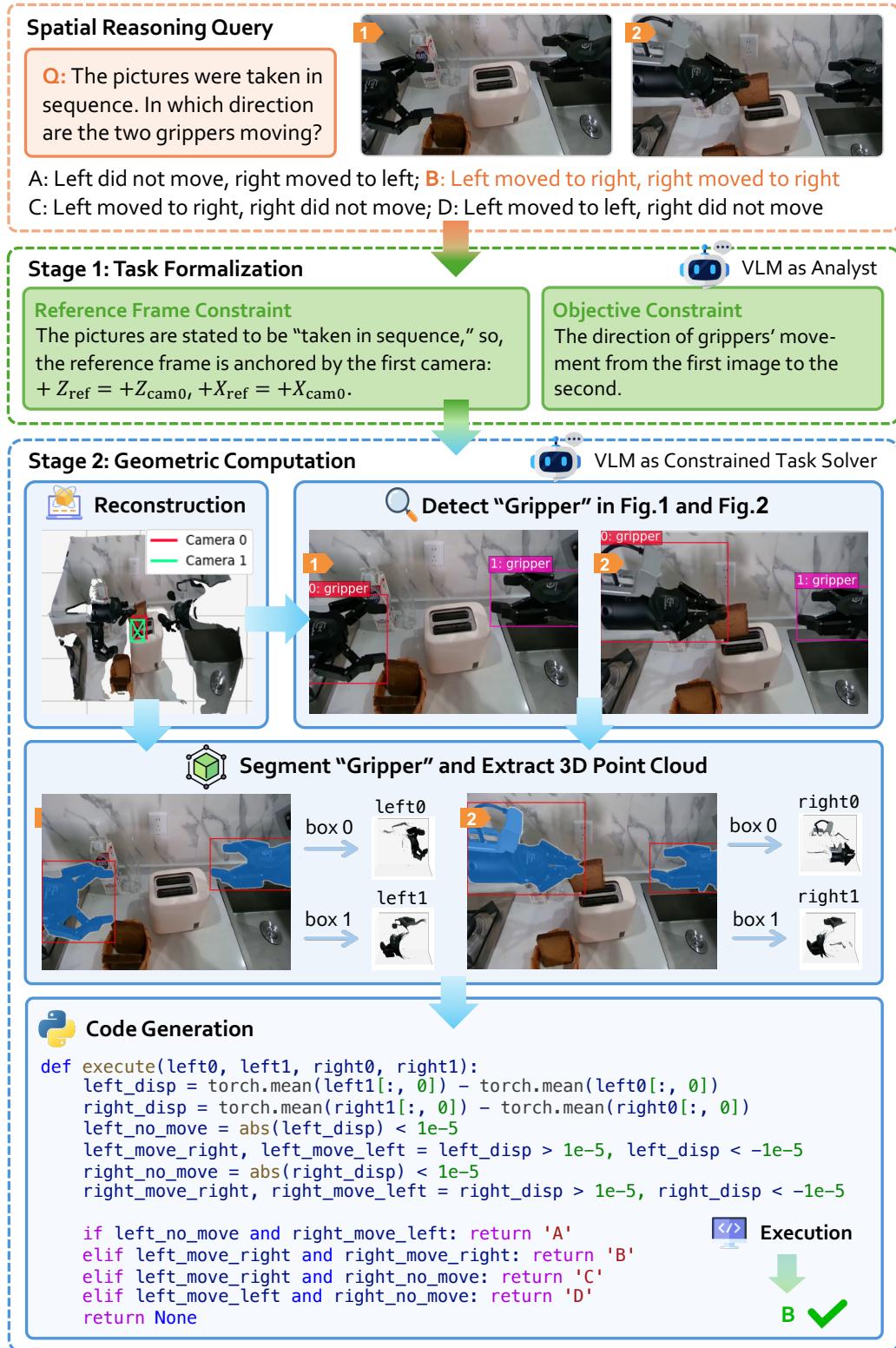


Figure 12. Case Study #5. Object movement analysis.

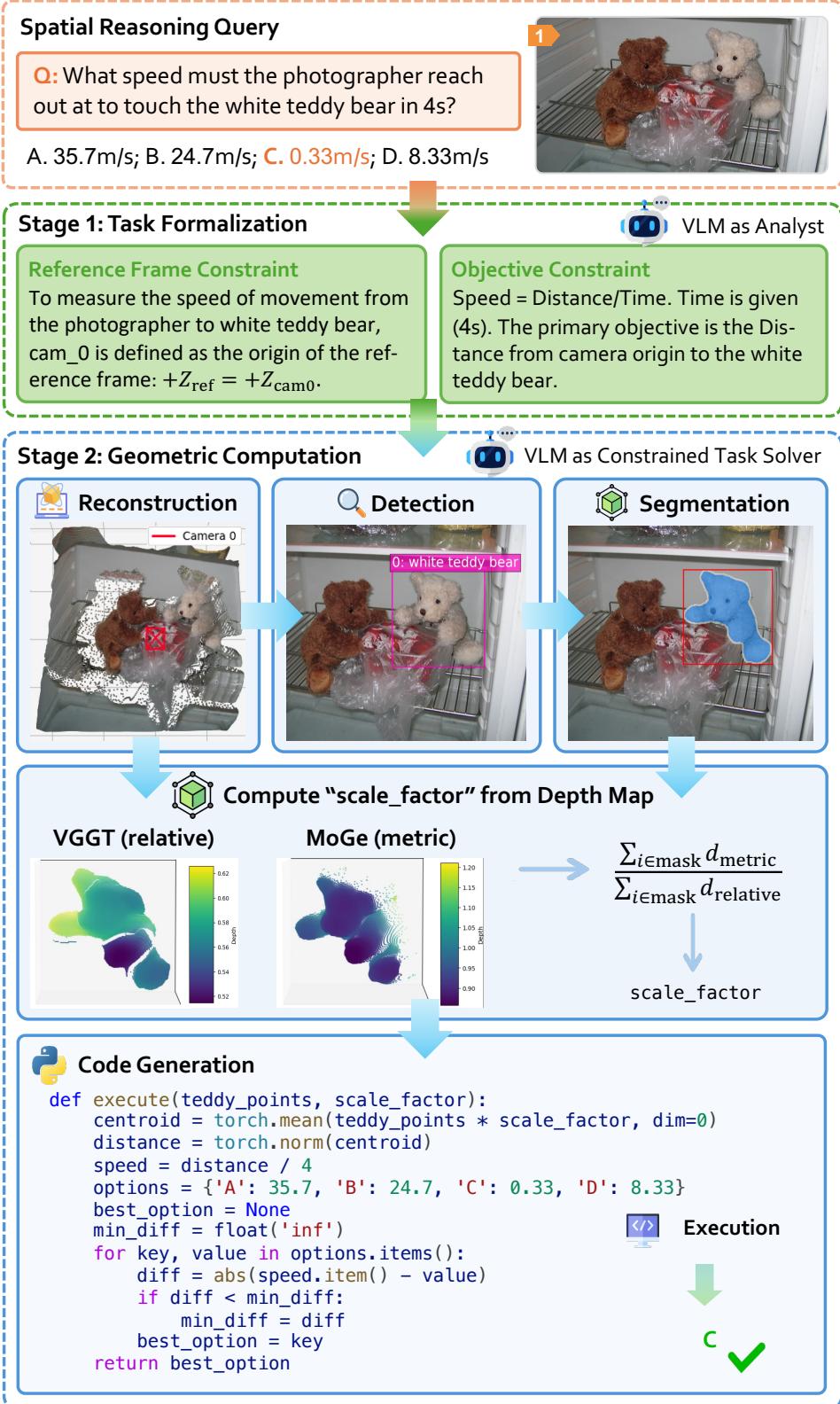


Figure 13. Case Study #6. Metric-scale estimation.