PURSUING MINIMAL SUFFICIENCY IN SPATIAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Spatial reasoning, the ability to ground language in 3D understanding, remains a persistent challenge for Vision-Language Models (VLMs). We identify two fundamental bottlenecks: *inadequate* 3D understanding capabilities stemming from 2D-centric pre-training, and reasoning failures induced by *redundant* 3D information. To address these, we first construct a Minimal Sufficient Set (MSS) of information before answering a given question: a *compact* selection of 3D perception results from *expert models*. We introduce **MSSR** (Minimal Sufficient Spatial Reasoner), a dual-agent framework that implements this principle. A *Perception Agent* programmatically queries 3D scenes using a versatile perception toolbox to extract sufficient information, including a novel **SOG** (Situated Orientation Grounding) module that robustly extracts language-grounded directions. A *Reasoning Agent* then iteratively refines this information to pursue minimality, pruning redundant details and requesting missing ones in a closed loop until the MSS is curated. Extensive experiments demonstrate that our method, by explicitly pursuing both sufficiency and minimality, significantly improves accuracy and achieves state-of-the-art performance across two challenging benchmarks. Furthermore, our framework produces interpretable reasoning paths, offering a promising source of high-quality training data for future models. Source code will be made publicly available.

1 INTRODUCTION

Spatial reasoning—the ability to perceive and reason about object relationships in 3D space—is a cornerstone of general intelligence and a critical prerequisite for deploying AI in the physical world, from robotics to AR/VR (Cheng et al., 2024a; Kim et al., 2024). While modern VLMs (OpenAI, 2024; DeepMind, 2025) have achieved remarkable success, they still consistently fail on spatial reasoning tasks (Yang et al., 2025b; Li et al., 2025a). In this work, we diagnose this critical gap by identifying two fundamental bottlenecks:

Inadequate 3D perception. Trained predominantly on 2D data, VLMs lack geometric priors and thus struggle to perceive 3D information like layout, orientation, and depth (Ma et al., 2023; 2024b).

Redundancy degrades reasoning. 3D environments are information-dense. Naively aggregating all percepts floods the context with weakly relevant details, potentially diluting attention (Liu et al., 2023) and encouraging shortcut heuristics (Xiao et al., 2024)—ultimately degrading performance (Fig. 1).

In the face of these two challenges, cognitive science offers a compelling insight: humans navigate complex scenes not by exhaustively processing all sensory data, but by constructing task-specific, minimal mental models (Tversky, 1991). Based on the mental models, they then selectively attend to the details needed to make a decision, and incrementally update the mental models as required (Byrne & Johnson-Laird, 1989; Johnson-Laird, 2010). This principle is formalized in statistics by the Minimal Sufficient Statistic (Lehmann & Casella, 1998), which captures all relevant information from a sample in the most compressed form possible.

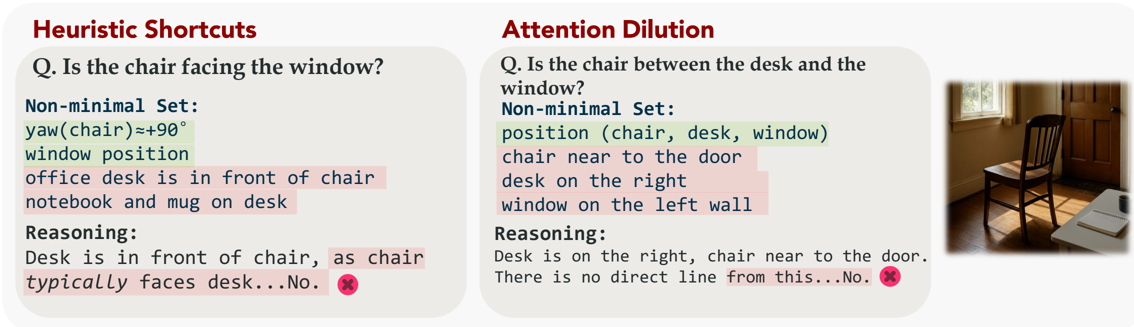


Figure 1: Irrelevant (highlighted in red) information can overwhelm the VLM and hurt its performance. **Left**: VLM ignores crucial information and hallucinates. **Right**: VLM cannot attend to the important part.

We show that the key to robust spatial reasoning lies in a similar pursuit: actively discovering a **Minimal Sufficient Set (MSS)**—the most compact representation of spatial information required to answer a specific query. Motivated by this principle, we introduce **MSSR** (**M**inimal **S**ufficient **S**patial **R**easoner), a zero-shot framework that operationalizes this pursuit through a dual-agent architecture. MSSR disentangles the challenges of perception and reasoning into two specialized, collaborative agents:

To bridge the 3D perception gap, we equip a *Perception Agent* (PA) with a suite of vision modules to programmatically query the scene for spatial primitives (*e.g.*, locations, directions, relations) and return a structured, VLM-friendly state. A key limitation of existing tool-augmented VLMs is their inability to ground complex, situational directions specified by language descriptions. We address this with a **SOG** (**S**ituated **O**rientation **G**rounding) module, which reformulates the orientation estimation task as a multi-choice question, and to overlay candidate 3D directions on 2D images as visual prompting. Through a procedural coarse-to-fine approach and alternative 3D view rendering to eliminate ambiguities, it robustly extracts object orientations ("Is the chair facing the door?") and behavior-centric directions ("Which way is the person facing while ascending the stairs?"). Thus, the PA provides rich and accurate 3D data without costly end-to-end training.

While the PA gathers extensive perceptual data, it risks worsening the second bottleneck: information redundancy that could degrade reasoning accuracy. To solve this, a *Reasoning Agent* (RA) strategically prunes this stream of information and curates the MSS by explicitly filtering redundant or task-irrelevant 3D information. It first formulates a high-level reasoning plan, assesses the collected information, and subtracts non-contributing information. If this set is deemed insufficient to confidently answer the question, the RA issues targeted, specific requests back to the PA to acquire only the missing information. This iterative refinement continues until a MSS is formed. The final answer is derived exclusively from this curated set, sharpening the model's focus and mitigating errors from distracting data.

We evaluate MSSR on two challenging benchmarks: MMSI-Bench (Yang et al., 2025b), which tests situated multi-view reasoning in complex scenes, and ViewSpatial-Bench (Li et al., 2025a), which focuses on perspective relational understanding. MSSR achieves state-of-the-art performance against strong monolithic and agentic baselines, while producing compact, interpretable reasoning paths. Extensive ablation studies demonstrate the harmful effect of redundancy and the efficacy of the RA's pruning in restoring performance. Our contributions are threefold:

- We formulate 3D spatial reasoning as **Minimal Sufficient Set construction** and introduce a **dual-agent** framework that interleaves perception with high-level planning to acquire *just enough* information.
- We design a **perception agent** that has access to a versatile toolbox including a SOG module for robust directional grounding; and a **reasoning agent** that directs the entire process and provides final answers.
- Our MSSR effectively improves the 3D spatial reasoning performance on two challenging benchmarks and produces interpretable reasoning traces that can provide supervision for future 3D-aware models.

2 Related Work

Monolithic VLMs in Spatial Reasoning typically inject 3D knowledge by fine-tuning on synthetic data (Chen et al., 2024a; Cheng et al., 2024b; Ma et al., 2025a) or integrate specialized modules to process 3D modalities like point clouds (Hong et al., 2023; Huang et al., 2024; Ma et al., 2025b). While demonstrating progress, these methods are fundamentally limited: they require prohibitively expensive 3D instruction datasets (Yang et al., 2025a) and risk forgetting pre-trained knowledge, which degrades the VLM's crucial general-purpose reasoning abilities (Kirkpatrick et al., 2017). In contrast, MSSR is a zero-shot, training-free framework that bypasses these issues. By preserving the VLM's full capabilities and instead explicitly structuring the perception-reasoning process, our approach maximizes spatial reasoning performance without compromising the model's versatility or requiring costly data and retraining.

Agentic Frameworks are a prominent line of work where complex problems are decomposed into steps solved via tool-use. Pioneering methods like ReAct (Yao et al., 2023) showed that LLMs can interleave reasoning and action, a paradigm extended to 3D where agents gather spatial information for tasks like embodied exploration (Yang et al., 2025c) and 3D VQA (Ma et al., 2024a). These approaches focus primarily on information gathering, often adopting a purely accumulative strategy. However, for the tasks we address, the dense nature of 3D scenes introduces significant redundancy, where an excess of irrelevant spatial details degrades performance. MSSR is therefore designed to not only gather information, but also prune irrelevance, which is a key departure from prior agentic designs.

Visual Programming is a paradigm often used for 3D attribute querying (Marsili et al., 2025; Yuan et al., 2024). It enhances VLMs by decomposing complex visual tasks into executable programs that leverage specialized modules (Gupta & Kembhavi, 2023; Surís et al., 2023). We adopt the visual programming as the execution backbone for our Perception Agent, leveraging its modularity to integrate specialized tools. Instead of the typical one-shot execution, our framework advances this paradigm by integrating visual programming into a closed loop. By preserving the full execution state across iterations, subsequent perception steps can build upon prior computations, enabling dynamic information refinement while avoiding redundant work.

3 Method

We address *language-conditioned spatial reasoning*: given M views $\mathcal{I} = \{I_1, \ldots, I_M\}$ from the same scene and a natural-language query q, the goal is to produce the answer a. Before answering the 3D reasoning question, our approach first builds a Minimal Sufficient Set (MSS)—a compact representation that is both sufficient to answer the query and minimal to prevent failures from redundant and distracting information.

3.1 Overview

As illustrated in Figure 2, our method actively curates the MSS. The process is iterative, interleaving information acquisition and strategic pruning to converge on a representation that is both sufficient and minimal. Formally, we define the target of this process, the MSS, as follows: Let $\mathcal{W}$ represent the comprehensive set of all spatial and semantic information derivable from the full 3D scene. We consider a set of spatial information $\mathcal{S} \subseteq \mathcal{W}$, which is a subset of $\mathcal{W}$. Our goal is to find the MSS $\mathcal{S}^\star \subseteq \mathcal{W}$, which satisfies two properties:

1. **Sufficiency**: Ideal MSS $\mathcal{S}^\star$ must contain enough information for an oracle reasoning agent, $\mathcal{R}^\star$, to correctly answer the query q. Formally, this means:

$$\mathcal{R}^\star(\mathcal{S}^\star, q) = a^\star \tag{1}$$

where $a^\star$ is the ground-truth answer. This ensures that no essential information for reasoning is omitted.

2. **Minimality**: MSS $\mathcal{S}^\star$ should be free of redundant or irrelevant information that would burden the reasoning model. Ideally, it is the smallest such set that maintains sufficiency. This can be expressed as:

$$\forall \mathcal{S}' \subset \mathcal{S}^\star, \quad \mathcal{R}^\star(\mathcal{S}', q) \neq a^\star \tag{2}$$

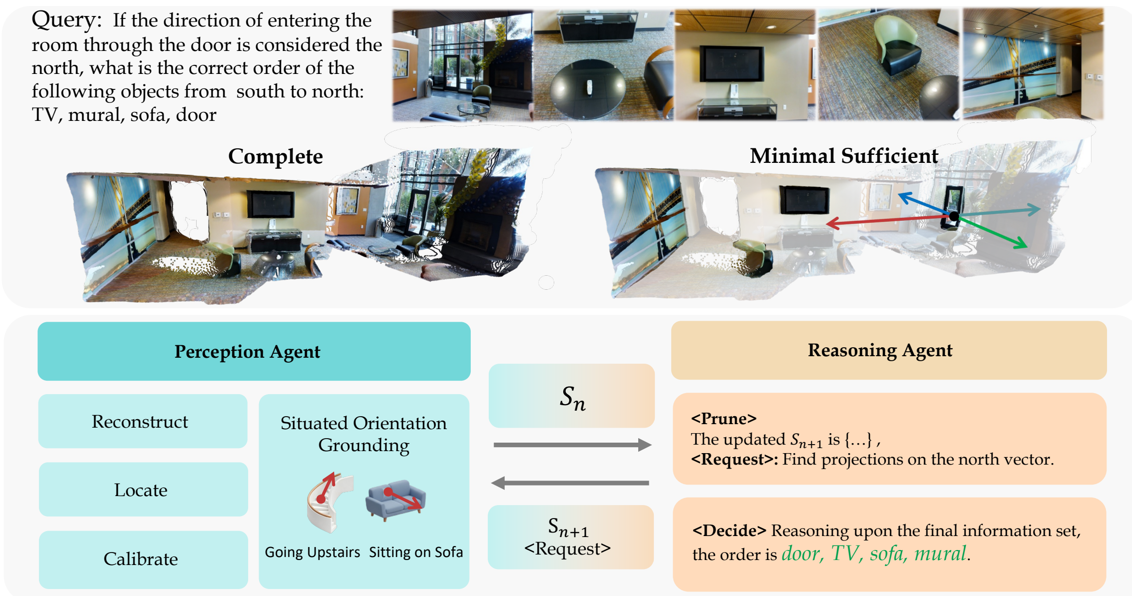


Figure 2: The MSSR dual-agent framework. **Top**: To avoid overwhelming the VLM, MSSR extracts a Minimal Sufficient Set that only contains the necessary information. **Bottom**: Dual-agent loop: the Perception Agent provides a spatial information set (S_n), and the Reasoning Agent prunes irrelevant information, requests missing information, or makes a final decision based on the curated set.

Through updating the spatial information set S, we aim to approximate the MSS $S^{\star}$. In this attempt, MSSR conducts a closed-loop collaboration between the Perception Agent (PA) and Reasoning Agent (RA), as shown in Figure 3. The process begins with an empty $\mathcal{S}$ and proceeds iteratively. Initially, the PA executes a broad perception directive to populate $\mathcal{S}$ with a comprehensive, potentially non-minimal set of spatial primitives. This set is then passed to the RA for curation, where it formulates a reasoning plan and prunes any information not causally linked to the plan, thereby enforcing minimality. If the pruned set is deemed insufficient, the RA performs an assessment and issues a targeted request back to the PA for precisely the missing information, which then augments $\mathcal{S}$. This cycle of curation and targeted augmentation repeats until the RA judges $\mathcal{S}$ to be sufficient for answering the query. At this final stage, the RA discards all prior context and reasons exclusively over the curated MSS to produce the answer, ensuring both focus and interpretability.

3.2 Perception Agent

The Perception Agent (PA) serves as the perception engine in our framework, responsible for bridging the gap between high-level reasoning directives and raw elements from the 3D scene. To equip the PA with robust and precise 3D perception capabilities, we adopt the **Visual Programming** (Gupta & Kembhavi, 2023) paradigm. The PA is provided with a suite of pre-designed modules, which act as specialized tools. These modules leverage vision expert models for tasks like geometric reconstruction and object localization, significantly augmenting the LLM's capabilities for complex spatial computations. Furthermore, the structured nature of code generation within the visual programming framework ensures that the information extraction process is logical, transparent, and reproducible.

At each turn, the PA receives the current $\mathcal{S}$, the original query, scene images $\mathcal{I}$ and a natural language request r from the Reasoning Agent representing the current information-gathering goal. Its objective is to generate a

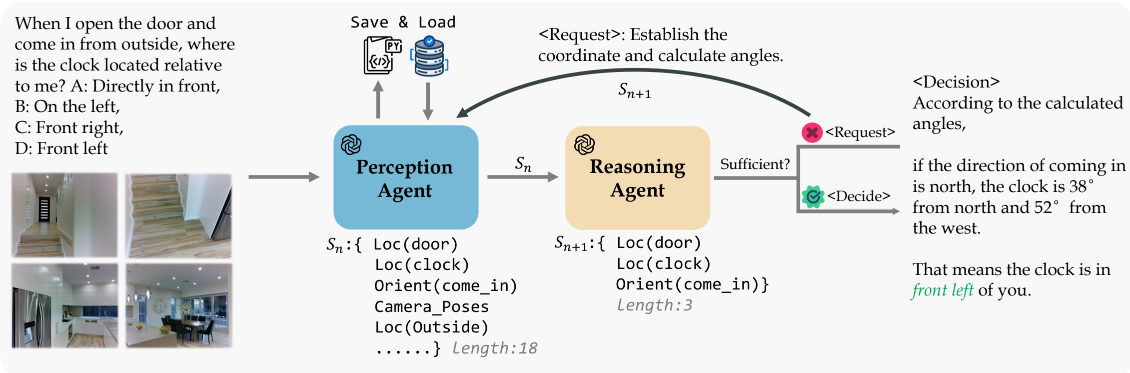


Figure 3: **A detailed example.** The Perception Agent provides an 18-item set S_n, and the Reasoning Agent prunes to 3 essential items in S_{n+1}. Deeming this insufficient, the RA issues a `<Request>` for missing calculations. With sufficient information, it makes a final `<Decision>`.

Python script that invokes the appropriate foundation modules to fulfill the directive. The process begins with an empty $\mathcal{S}$ and an initial, broad instruction, such as:

"Extract all potentially relevant information to solve the problem. Your goal is to find as much information as possible."

The generated script populates a designated dictionary with newly extracted information—such as object coordinates and spatial relationships. This dictionary is then merged into $\mathcal{S}$. Crucially, after each execution, the entire state of the Python environment, including all intermediate variables and data structures, is preserved as a **snapshot**. When the PA is invoked in a subsequent turn, this snapshot is reloaded. This mechanism allows the PA to contextually build upon its previous computations, avoiding redundant processing and enabling a more complex, stateful exploration of the scene. The newly updated $\mathcal{S}$ is then passed to the Reasoning Agent.

Spatial Reasoning Modules. The Perception Agent implements directives by invoking a curated suite of spatial modules, which bridge high-level goals and raw perceptual data to construct the MSS. Following established designs (Surís et al., 2023), our toolkit includes basic modules: a locate module that utilizes vision expert models (Liu et al., 2024; Ravi et al., 2024) to pinpoint object coordinates in 3D, and a computation module that offloads complex numerical tasks, such as coordinate frame transformations. More critically, to address challenges often overlooked by prior work, we propose novel modules designed for more robust spatial reasoning. These advanced modules cover the full pipeline of spatial understanding, from establishing a coherent 3D representation to grounding language-conditioned attributes. The specific implementation of each module is detailed in Appendix B.

Foundational 3D Scene Reconstruction. To bridge the critical gap between sparse 2D images and a coherent 3D scene representation, this module leverages recent breakthroughs in rapid neural-based reconstruction models (Wang et al., 2024b;a; 2025) to estimate the camera parameters, depth maps, and a unified 3D point cloud of the scene. In our experiment, VGGT (Wang et al., 2025) shows robust performance and high speed. This output serves as the foundational canvas upon which subsequent spatial information is extracted. Similar to Chen et al. (2024a), we additionally segment the ground plane during reconstruction.

Global Coordinate System Calibration. To resolve the inherent ambiguity of view-dependent spatial terms (*e.g.*, "left," "behind"), this module establishes a unified global coordinate system. It aligns the scene's axes based on a reference vector, which is derived either from explicit instructions in the query ("assume the window faces east") or from prominent landmarks. This calibration ensures all directional and relational information stored in the MSS is consistent and unambiguous, a prerequisite for reliable multi-step reasoning.

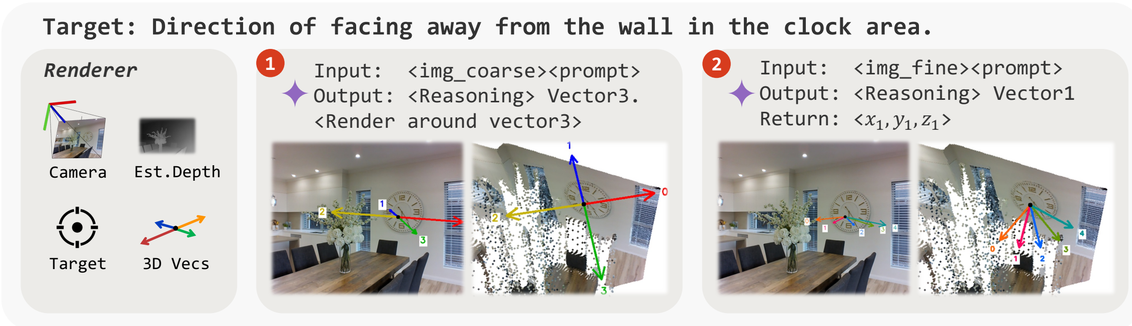


Figure 4: Situated Orientation Grounding as a multi-choice question. We prompt a VLM to select a direction from multiple candidates using the original image and a rendered image, both overlayed with the candidate's 3D orientation vectors. Starting from coarse directions, it then refines this direction to a fine-grained result.

Situated Orientation Grounding (SOG). Beyond simple localization, reasoning often requires understanding orientation. We introduce the SOG module, which utilizes *Visual Prompting* (Qi et al., 2025) to ground complex, language-conditioned directional concepts into 3D vectors. Critically, SOG handles not only intrinsic object orientation ("the front of the chair") but also situation-dependent orientations ("the direction to exit the room")—a capability largely overlooked in prior work. This module dramatically expands the range of addressable queries from static localization to dynamic, perspectival reasoning.

The power of SOG lies in bridging the gap between a VLM's rich semantic scene understanding and its inability to directly regress 3D geometric outputs. Instead of attempting this intractable regression, we reframe orientation grounding as a more reliable *visual selection task*, implemented through a coarse-to-fine strategy as illustrated in Figure 4.

For a query anchored at object position P_o, we first randomly generate a sparse set of four coplanar, orthogonal vectors $\{\vec{d}_i\}_{i=1}^4$ parallel to the ground plane, resembling compass directions. To provide the VLM with sufficient context and resolve perspective ambiguity, we render these vectors onto two distinct views: a **Situated View** using the original image to preserve natural context, and a synthetic **Canonical View** from an elevated perspective to reduce foreshortening. The VLM is prompted to select the candidate that best aligns with the language query. This selection is subsequently refined by generating a denser set of candidates around the chosen vector and repeating the selection process, allowing the system to converge on a precise direction. Empirically, this coarse-to-fine strategy robustly and efficiently identifies the target 3D orientation. While not designed for sub-degree accuracy, our experiments confirm that this level of precision is sufficient for most challenging spatial tasks. The vector is then added to $\mathcal{S}$, empowering the RA to tackle a wide range of orientation-dependent problems that were previously intractable.

3.3 Reasoning Agent

The Reasoning Agent (RA) acts as the cognitive core in MSSR, responsible for ensuring the information set $\mathcal{S}$ is both sufficient and minimal. At each step, the RA receives the current information set $\mathcal{S}_n$ and the original query q. It operates in a two-stage process of information curation and strategic decision-making.

In the first stage, *Plan-Guided Information Curation*, the RA formulates a high-level reasoning plan outlining the necessary steps to answer the query. With this implicit plan, it initializes an empty, updated information set, $\mathcal{S}_{n+1}$. The RA is then prompted to systematically scrutinize each item within the current set $\mathcal{S}_n$, critically evaluating its relevance to the reasoning plan. Only those deemed necessary for the plan are preserved and added to $\mathcal{S}_{n+1}$. This subtractive filtering step is crucial for maintaining the conciseness of $\mathcal{S}$, aggressively pruning any information that is irrelevant to the specific query. After completing the curation, the RA enters the second stage, *Strategic Decision-Making*. Here, it makes one of two decisions:

Table 1: **Comparison with baselines on MMSI-Bench and ViewSpatial-Bench.** Our method performs favorably against previous methods on these challenging spatial reasoning tasks. Blue is to highlight the improvement of our method over GPT-4o backbone.

	MMSI-Bench				ViewSpatial-Bench		
	positional relationship	multi-step reasoning	attribute & motion	overall	camera based	person based	overall
Proprietary LLM							
o3	45.8	34.9	36.4	**41.0**	51.3	51.0	**51.1**
Gemini 2.5 Pro	38.5	34.3	36.1	37.0	44.3	41.6	43.0
Gemini 2.5 Flash	37.4	30.3	33.9	35.0	41.6	36.9	38.4
GPT-4o *(backbone)*	28.0	30.8	34.3	30.3	33.6	36.3	35.0
Open-source LLM							
Llama-3.2-11B-Vision	26.9	19.2	23.9	24.5	23.7	33.6	28.8
LLaVA-OneVision-7B	28.0	11.6	27.1	24.5	28.5	26.6	27.5
Qwen2.5-VL-3B	29.5	23.2	23.2	26.5	39.8	32.1	35.8
Qwen2.5-VL-7B	25.9	25.8	26.1	25.9	40.6	33.4	36.9
Qwen2.5-VL-72B	31.2	27.3	32.1	**30.7**	47.6	38.9	43.1
InternVL2.5-8B	29.9	30.3	25.4	28.7	46.5	40.2	**43.2**
InternVL3-14B	25.5	29.3	27.5	26.8	47.1	33.9	40.3
Specialist							
LEO	42.3	32.3	38.6	39.3	41.5	45.8	43.7
Agentic							
ViLaSR	-	-	-	30.2	42.4	34.4	38.2
VADAR	32.8	22.7	26.1	28.9	34.2	33.2	33.7
MSSR (Ours)	50.6	50.0	47.1	**49.5 (+19.2)**	51.0	54.4	**51.8 (+16.8)**

`<Request>`: If the RA determines that $\mathcal{S}_{n+1}$ is insufficient to complete the reasoning plan, it then formulates a targeted, natural-language directive that precisely articulates the missing information (*e.g.*, '`<Request> The facing direction of someone sitting on the chair.`'). This request, along with the pruned $\mathcal{S}_{n+1}$, is passed back to the Perception Agent. The PA uses this focused guidance and updated set to initiate a new round of programming, generating $\mathcal{S}_{n+2}$ that populates the requested information.

`<Decide>`: Conversely, if the RA concludes that $\mathcal{S}_{n+1}$ contains all necessary information to derive a final answer, it triggers the `<Decide>` action. It then discards all prior context and reasons exclusively over this final, minimal set using Chain-of-Thought (CoT) (Wei et al., 2022) to produce the answer. This disciplined use of only the pruned set ensures that the final reasoning is efficient and shielded from the distracting influence of irrelevant data.

Notably, unlike many current 3D agentic systems (Li et al., 2025b; Marsili et al., 2025), both our agents operate in a zero-shot fashion. They are guided by high-level principles rather than in-context learning (ICL) examples. This design endows our method with strong generalization capabilities and mitigates the risk of overfitting to dataset-specific exemplars.

4 EXPERIMENTS

4.1 BENCHMARKS AND BASELINES

We evaluate our method and various baseline models on two challenging spatial reasoning benchmarks. **MMSI-Bench** (Yang et al., 2025b) is a hand-crafted multi-image dataset focusing on spatial reasoning. The tasks are designed to probe a model's understanding of the positions, attributes, and motions of three elements—cameras, objects, and regions—within real-world environments. Furthermore, it incorporates a multi-step reasoning split, which composes elementary tasks into long-horizon questions to test for deeper spatial reasoning. **ViewSpatial-Bench** (Li et al., 2025a) complements this by addressing a critical limitation of the gap between egocentric and allocentric spatial reasoning. Designed to evaluate multi-viewpoint spatial localization and recognition capability, this benchmark is structured into five distinct task categories to rigorously assess a model's ability to generalize across different spatial viewpoints. These two benchmarks provide a robust, multifaceted platform for testing spatial reasoning capabilities that we aim to advance.

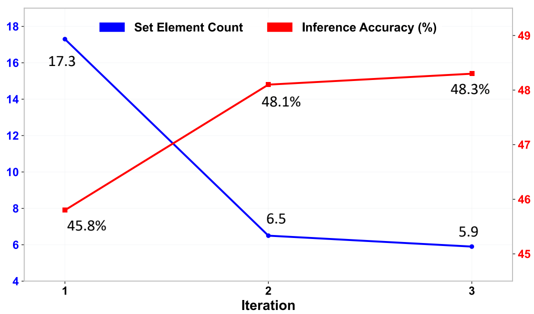


Figure 5: Effects of conciseness on accuracy.

Table 2: **Ablation study of our framework's key components.** We report accuracy (%) on the MSR sub-task and the overall MMSI-Bench.

	MSR	Overall
GPT-4o	30.8	30.3
Ours (Full)	50.0	49.5
Only PA	33.8	37.1
Only RA	31.8	31.1
w/o SOG	47.0	46.9
w/o Iteration	44.4	47.2

To provide a comprehensive evaluation, we establish four distinct categories of baseline models for comparison: *Proprietary LLMs:* We include powerful, closed-source models with strong reasoning capabilities, such as GPT-4o (OpenAI, 2024), Gemini-2.5-flash (DeepMind, 2025), Gemini-2.5-pro (DeepMind, 2025), and o3 (OpenAI, 2025). *Open-Source LLMs:* We compare with state-of-the-art models such as LLaMA-3.2-Vision (Meta, 2024), LLaVA-OneVision (Li et al., 2024), Qwen2.5-VL (Bai et al., 2025), InternVL3 (Zhu et al., 2025), InternVL2.5 (Chen et al., 2024b) and DeepSeek-VL2 (Wu et al., 2024). *Specialists:* We compare with LEO (Huang et al., 2024), a multi-modal generalist model recognized for its proficiency in various 3D tasks. *Agents:* We consider VADAR (Marsili et al., 2025), a visual programming framework targeted at spatial inference, and ViLaSR (Wu et al., 2025), a model notable for its ability of visual manipulation.

4.2 Main Results

On the MMSI-Bench, our method achieves an overall accuracy of **49.5%**. We outperform even the strongest LLM evaluated, o3 (41.0%), demonstrating an absolute gain of 8.5 percentage points. Compared to the best-performing open-source LLM, Qwen2.5-VL-72B (30.7%), our approach shows a pronounced relative improvement of over 60%. When contrasted with state-of-the-art specialist models such as LEO (39.3%) and agentic frameworks such as ViLaSR (30.2%), our method maintains a clear lead.

Transitioning to the ViewSpatial-Bench, which focuses on multi-viewpoint spatial localization and recognition, our method again sets a new benchmark with an overall accuracy of **51.8%**. The consistent strength observed in both Camera Based (51.0%) and Person Based (54.4%) tasks highlights our method's robust generalization across varied spatial viewpoints. Excelling in both categories indicates MSSR's proficiency in bridging the gap between egocentric and allocentric spatial understanding, a limitation for current methods.

4.3 Ablation Study

Effects of Minimality. A central idea of our framework is that pursuing minimality is crucial for robust reasoning. To rigorously test this, we conducted a controlled ablation study on a representative subset of MMSI-Bench problems MSSR solved in three iterations. For these problems, we created sufficiency-normalized information sets for each step by retrospectively adding critical information (discovered in iterations) to earlier, larger sets. This isolates the effect of set size on the RA's performance, ensuring each set contains the same level of sufficiency. The RA was then tasked to solve the problem independently using these three sets, corresponding to the state after each iteration.

As illustrated in Figure 5, the results reveal a clear inverse correlation between information set size and accuracy. As our iterative pruning strategy reduces the average set element count from an initial 17.3 to a concise 5.9, the RA's inference accuracy concurrently rises from 45.8% to 48.3%. This finding provides empirical evidence that excess information is a significant distractor for LLM-based agents. This affirms that pursuing minimality is fundamental for high-fidelity spatial reasoning, not merely an efficiency optimization.

Component Analysis. We conduct ablations to quantify the contributions of each component. In Table 2, we report accuracy on the representative Multi-step Reasoning sub-task and the overall score on MMSI-Bench.

Only PA: Removing the RA and tasking the PA to programmatically deduce the answer after information extraction leads to a significant performance drop. In fact, its sequential, top-down execution flow, while effective for information gathering, is less effective in reasoning and question answering, as it often returns sub-optimal results.

Only RA: Conversely, removing the PA and forcing the RA to rely solely on the initial context yields negligible improvement over the baseline. This result affirms that prompting alone cannot substitute for the precise, targeted 3D scene perception provided by the PA, even when using a powerful reasoning model. This highlights the crucial synergy between the two agents.

w/o SOG: We replaced SOG with a baseline directly prompting the VLM to infer directional vectors, causing an overall performance drop to 46.9%. Our observations suggest this degradation stems from the VLM's inability to regress 3D coordinates, rather than a failure in semantic understanding. This result validates our design of SOG, which circumvents this limitation by reframing regression as a visual selection task.

w/o Iteration: Setting the maximum iteration count to 1 forces the RA to make an immediate decision without the ability to request additional information. This leads to a noticeable drop in performance. The degradation is more pronounced on the MSR sub-task, underscoring that for intricate multi-step tasks, the iterative feedback loop is especially vital for achieving information sufficiency.

4.4 Application: Annotating Spatial Reasoning Data

Beyond zero-shot inference, **MSSR** serves as a powerful data annotation engine. In addition to numerical results (Hu et al., 2024), its final output—a Minimal Sufficient Set (MSS) and an explicit reasoning trace—provides a rich foundation for creating CoT (Wei et al., 2022)-style data.

Table 3. Accuracy on MMSI-Bench after fine-tuning.

Model	**Accuracy (%)**
Qwen2.5-VL-7B	25.9
Qwen2.5-VL-7B-SFT	**30.1** (+4.2)

To demonstrate this potential, we curated a dataset by sampling 300 (30%) correctly solved questions from our MMSI-Bench runs. We then employed GPT-4o for automated quality filtering, which scrutinized the RA's reasoning for logical consistency and removed cases of incidental correctness, yielding 258 high-fidelity traces. GPT-4o then synthesized these traces into CoT-style annotations. This process grounds the RA's abstract reasoning in gathered spatial information by systematically interleaving its high-level strategic steps (*e.g.*, "I need to determine the chair's position") with the specific perceptual evidence from the MSS that substantiates each step (*e.g.*, "The Locate module confirms the chair is at [x,y,z]"). We conducted a preliminary fine-tuning experiment using this synthesized dataset on Qwen2.5-VL-7B. As shown in Table 3, despite the modest scale of our experiment, the fine-tuned model's accuracy on MMSI-Bench rose to 30.1%, an absolute improvement of **4.2%**. This result is particularly noteworthy as it elevates the 7B model to a performance level competitive with its much larger 72B counterpart. This validates MSSR as an effective data engine for distilling complex spatial reasoning capabilities into future models (details in Appendix C).

5 Conclusion

We present MSSR, which builds a Minimal Sufficient Set before answering spatial reasoning questions. Specifically, it adopts a dual-agent loop: a programmatic Perception Agent—augmented with **SOG** for directional grounding—extracts spatial information, while a plan-guided Reasoning Agent ensures sufficiency and minimality through targeted pruning and requesting. This design mitigates redundancy-induced errors. Our framework boosts the performance of the backbone VLM significantly, achieving state-of-the-art results on MMSI-Bench and ViewSpatial-Bench. Beyond inference, the MSS reasoning traces also serve as high-quality supervision signals for training future 3D-aware models.

REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025. 8, 19

Ruth MJ Byrne and Philip N Johnson-Laird. Spatial reasoning. *Journal of Memory and Language*, 28(5): 564–575, 1989. 1

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Conference*, pp. 14455–14465, 2024a. 3, 5

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024b. 8

An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Zaitian Gongye, Xueyan Zou, Jan Kautz, Erdem Bıyık, Hongxu Yin, Sifei Liu, and Xiaolong Wang. Navila: Legged robot vision-language-action model for navigation. *arXiv preprint arXiv:2412.04453*, 2024a. 1

An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. Spatialrgpt: Grounded spatial reasoning in vision-language models. *Advances in Neural Information Processing Systems*, 37:135062–135093, 2024b. 3

Google DeepMind. Gemini 2.5: Our most intelligent ai model. `https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/`, 2025. 1, 8

flask. flask. `https://flask.palletsprojects.com/en/stable/`, 2025. 19

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Conference*, pp. 14953–14962, 2023. 3, 4

Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3d-llm: Injecting the 3d world into large language models. *Advances in Neural Information Processing Systems*, 36:20482–20494, 2023. 3

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations*, 1(2):3, 2022. 19

Yushi Hu, Otilia Stretcu, Chun-Ta Lu, Krishnamurthy Viswanathan, Kenji Hata, Enming Luo, Ranjay Krishna, and Ariel Fuxman. Visual program distillation: Distilling tools and programmatic reasoning into vision-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Conference*, pp. 9590–9601, 2024. 9

Jiangyong Huang, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang. An embodied generalist agent in 3d world. In *International Conference on Machine Learning*, 2024. 3, 8

Philip N Johnson-Laird. Mental models and human reasoning. *Proceedings of the National Academy of Sciences*, 107(43):18243–18250, 2010. 1

Yeseung Kim, Dohyun Kim, Jieun Choi, Jisang Park, Nayoung Oh, and Daehyung Park. A survey on integration of large language models with intelligent robots. *Intelligent Service Robotics*, 17(5):1091–1107, 2024. 1

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. 3

E. L. Lehmann and George Casella. *Theory of Point Estimation*. Springer, 2nd edition, 1998. 1

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024. 8

Dingming Li, Hongxing Li, Zixuan Wang, Yuchen Yan, Hang Zhang, Siqi Chen, Guiyang Hou, Shengpei Jiang, Wenqi Zhang, Yongliang Shen, et al. Viewspatial-bench: Evaluating multi-perspective spatial localization in vision-language models. *arXiv preprint arXiv:2505.21500*, 2025a. 1, 2, 7

Rong Li, Shijie Li, Lingdong Kong, Xulei Yang, and Junwei Liang. Seeground: See and ground for zero-shot open-vocabulary 3d visual grounding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Conference*, 2025b. 7

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023. 1

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*, pp. 38–55, 2024. 5, 15, 16

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, 2024a. 3

Wufei Ma, Haoyu Chen, Guofeng Zhang, Yu-Cheng Chou, Celso M de Melo, and Alan Yuille. 3dsrbench: A comprehensive 3d spatial reasoning benchmark. *arXiv preprint arXiv:2412.07825*, 2024b. 1

Wufei Ma, Yu-Cheng Chou, Qihao Liu, Xingrui Wang, Celso de Melo, Jianwen Xie, and Alan Yuille. Spatialreasoner: Towards explicit and generalizable 3d spatial reasoning. *arXiv preprint arXiv:2504.20024*, 2025a. 3

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition Conference*, pp. 17249–17260, 2025b. 3

Xiaojian Ma, Silong Yong, Zilong Zheng, Qing Li, Yitao Liang, Song-Chun Zhu, and Siyuan Huang. Sqa3d: Situated question answering in 3d scenes. In *International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=IDJx97BC38`. 1

Damiano Marsili, Rohun Agrawal, Yisong Yue, and Georgia Gkioxari. Visual agentic ai for spatial reasoning with a dynamic api. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Conference*, pp. 19446–19455, 2025. 3, 7, 8

Meta. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models. https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/, 2024. 8

OpenAI. Gpt-4o. https://openai.com/index/hello-gpt-4o/, 2024. 1, 8

OpenAI. Introducing o3 and o4 mini. https://openai.com/index/introducing-o3-and-o4-mini/, 2025. 8

Zhangyang Qi, Zhixiong Zhang, Ye Fang, Jiaqi Wang, and Hengshuang Zhao. Gpt4scene: Understand 3d scenes from videos with vision-language models. *arXiv preprint arXiv:2501.01428*, 2025. 6, 17

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. *arXiv preprint arXiv:2408.00714*, 2024. 5, 15, 16

Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *IEEE/CVF International Conference on Computer Vision*, pp. 11888–11898, 2023. 3, 5

Barbara Tversky. Spatial mental models. *Psychology of Learning and Motivation*, 27:109–145, 1991. 1

Jianyuan Wang, Nikita Karaev, Christian Rupprecht, and David Novotny. Vggsfm: Visual geometry grounded deep structure from motion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Conference*, pp. 21686–21697, 2024a. 5

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Conference*, 2025. 5, 15

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Conference*, 2024b. 5

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022. 7, 9

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, 2025. URL https://arxiv.org/abs/2506.09965. 8

Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, Zhenda Xie, Yu Wu, Kai Hu, Jiawei Wang, Yaofeng Sun, Yukun Li, Yishi Piao, Kang Guan, Aixin Liu, Xin Xie, Yuxiang You, Kai Dong, Xingkai Yu, Haowei Zhang, Liang Zhao, Yisong Wang, and Chong Ruan. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding, 2024. URL https://arxiv.org/abs/2412.10302. 8

Junbin Xiao, Angela Yao, Yicong Li, and Tat-Seng Chua. Can i trust your answer? visually grounded video question answering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Conference*, pp. 13204–13214, 2024. 1

Jianing Yang, Xuweiyi Chen, Nikhil Madaan, Madhavan Iyengar, Shengyi Qian, David F Fouhey, and Joyce Chai. 3d-grand: A million-scale dataset for 3d-llms with better grounding and less hallucination. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Conference*, pp. 29501–29512, 2025a. 3

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*, 2025b. 1, 2, 7

Yuncong Yang, Han Yang, Jiachen Zhou, Peihao Chen, Hongxin Zhang, Yilun Du, and Chuang Gan. 3d-mem: 3d scene memory for embodied exploration and reasoning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Conference*, pp. 17294–17303, 2025c. 3

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. 3

Zhihao Yuan, Jinke Ren, Chun-Mei Feng, Hengshuang Zhao, Shuguang Cui, and Zhen Li. Visual programming for zero-shot open-vocabulary 3d visual grounding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Conference*, pp. 20623–20633, 2024. 3

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational Linguistics. URL http://arxiv.org/abs/2403.13372. 19

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Yuchen Duan, Hao Tian, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv:2504.10479*, 2025. 8

This appendix provides additional details and results that complement the main paper:

- **Implementation Details**: hardware setup, inference parallelization, and timing breakdowns.
- **Module Implementations**: detailed descriptions of each perception and reasoning module.
- **Fine-tuning Details**: training configuration for supervised fine-tuning of Qwen2.5-VL-7B.
- **Qualitative Results**: step-by-step execution traces and reasoning case studies.
- **Limitations**: discussion of error propagation and evaluation challenges.
- **A Visualization Tool**: web-based interface for inspecting code, execution, and reasoning.
- **The Use of LLMs**: clarification that LLMs were only used for polishing the text.

A IMPLEMENTATION DETAILS

All experiments were conducted on a server equipped with 8 NVIDIA 3090 GPUs. For evaluations, we ran eight independent inference processes in parallel, with each process exclusively allocated to a single GPU. Consequently, all performance metrics and timing statistics reported hereafter correspond to the execution of a single such process.

A.1 INFERENCE TIME

The per-question inference time is shown in Table 4. The primary computational bottleneck is the latency of API calls to the large language models that serve as the backbone for both the PA's code generation and the RA's deliberation (GPT-4o in our case). These calls account for approximately 81.7% of the total iteration time. The execution of our local perception toolbox constitutes the remaining 18.3%. This breakdown highlights a clear avenue for future optimization through the use of smaller, locally-hosted models or more efficient API endpoints.

Table 4: **Breakdown of average inference time per iteration on MMSI-Bench.** The dominant cost is from API calls.

Component	**Average Time (s)**
Local Vision Modules Execution	8.3
LLM API Calls (PA Code Generation & RA Deliberation)	37.1
Total per Iteration	**45.4**

A.2 ITERATION ANALYSIS

Table 5 presents the average number of iterations required for MSSR to resolve problems across different benchmarks and task types. The average number of iterations required for task completion demonstrates a potential correlation with problem complexity. The more demanding Multi-hop Spatial Reasoning (MSR) sub-task within MMSI-Bench, which often needs intricate multi-step information retrieval and reasoning, demands an average 2.41 iterations. Across all tasks on MMSI-Bench, the average iteration is around 2.15. Problems on the ViewSpatial-Bench were generally resolved with greater efficiency, averaging 1.88 iterations overall.

B MODULE IMPLEMENTATIONS

This section provides a detailed explanation of the implementation specifics for each specialized module within the Perception Agent's toolkit. These modules collectively enable the robust extraction of 3D perceptual

Table 5: **Average number of iterations per task.** Our framework converges to a solution in a small number of steps, demonstrating the efficiency of its iterative information curation process.

Benchmark	Task	Average Iterations
MMSI-Bench	MSR	2.41
MMSI-Bench	Overall	2.15
ViewSpatial-Bench	Camera-based	1.84
ViewSpatial-Bench	Person-based	1.92
ViewSpatial-Bench	Overall	1.88

information, contributing to the construction of the Minimal Sufficient Set (MSS) required for complex spatial reasoning.

B.1 Reconstruction Module

Functionality: This module transforms sparse 2D images $\mathcal{I}$ into a coherent 3D scene representation. It estimates intrinsic and extrinsic camera parameters, predicts depth maps, and produces a unified 3D point cloud. Additionally, it segments and reconstructs the ground plane, providing a spatial reference for downstream reasoning.

3D Point Cloud Generation: We use a rapid, state-of-the-art 3D reconstruction model VGGT (Wang et al., 2025) to estimate intrinsics K, extrinsics (R_{CW}, T_{CW}), and depth maps $d(u, v)$. For each pixel (u, v), back-projection into camera coordinates is defined by:

$$Z_C = d(u, v),$$
$$X_C = Z_C \frac{u - c_x}{f_x}, \quad Y_C = Z_C \frac{v - c_y}{f_y},$$
$$\begin{pmatrix} X_W \\ Y_W \\ Z_W \end{pmatrix} = R_{CW} \begin{pmatrix} X_C \\ Y_C \\ Z_C \end{pmatrix} + T_{CW},$$

where (f_x, f_y) are focal lengths and (c_x, c_y) the principal point. Here (X_C, Y_C, Z_C) are camera-frame coordinates, mapped to world coordinates via R_{CW}, T_{CW}. Aggregating these across images yields the scene point cloud.

Ground Plane Estimation: We detect floor regions using GroundingDINO (Liu et al., 2024) and segment with SAM2 (Ravi et al., 2024). Masked pixels are back-projected to 3D, and PCA is applied to fit the ground plane:

$$\mathbf{n} \cdot \mathbf{p} + D = 0, \quad \mathbf{p} = (X_W, Y_W, Z_W).$$

If the resulting $\mathbf{n}$ points downward (in our setting, the positive Y component points downward), we flip its sign to ensure it points to the 'up' of the real world.

Fallback: If floor detection confidence is low, PCA is applied to the full point cloud; the eigenvector with smallest eigenvalue is used as the approximate ground normal.

Output: The module outputs calibrated camera parameters, depth maps, a dense 3D point cloud, and the estimated ground plane equation. These are used in later modules.

B.2 Object Localization Module

Functionality: This module localizes objects in the 3D scene based on natural language queries. Given a textual description of an object, it outputs the estimated 3D coordinates of the object within the reconstructed scene.

Object-Centric View Selection: For a query description "obj", we first tried to identify bounding boxes across all input images using GroundingDINO (Liu et al., 2024). However, we observe that the box with the highest confidence score often does not correspond to the most complete or unoccluded view of the object. To address this, we employ a low-cost vision-language model (VLM), Gemini-2.5-Flash-No-Thinking, to evaluate all scene images jointly. The VLM is prompted to return the image ID that best captures a clear and complete appearance of the queried object; if no such image exists, the module returns `None`. This step ensures robust selection of the most informative viewpoint before localization.

2D Segmentation and 3D Projection: Given the selected image ID, we run GroundingDINO again with the object description to obtain the bounding box with the highest confidence. The bounding box is then refined into a segmentation mask using SAM2 (Ravi et al., 2024). Pixels within this mask are projected into 3D world coordinates using the estimated camera parameters and depth maps. The object's 3D position is computed as the centroid of all projected points:

$$\mathbf{p}_{obj} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{p}_i,$$

where $\mathbf{p}_i \in \mathbb{R}^3$ denotes the 3D world coordinates of the i-th pixel inside the segmentation mask, and N is the total number of such pixels.

Output: The module outputs the estimated 3D location of the queried object, represented as a single point in the world coordinate system.

B.3 Global Coordinate Calibration Module

Functionality: This module establishes a scene-level directional coordinate system (e.g., North, South, East, West or equivalently Front, Back, Left, Right) based on natural language calibration such as "the table is north of the chair." With this calibrated system, relative directional relations of other objects (e.g., "the cabinet is to the west of the chair") can be consistently computed. An example is shown in Figure 6.

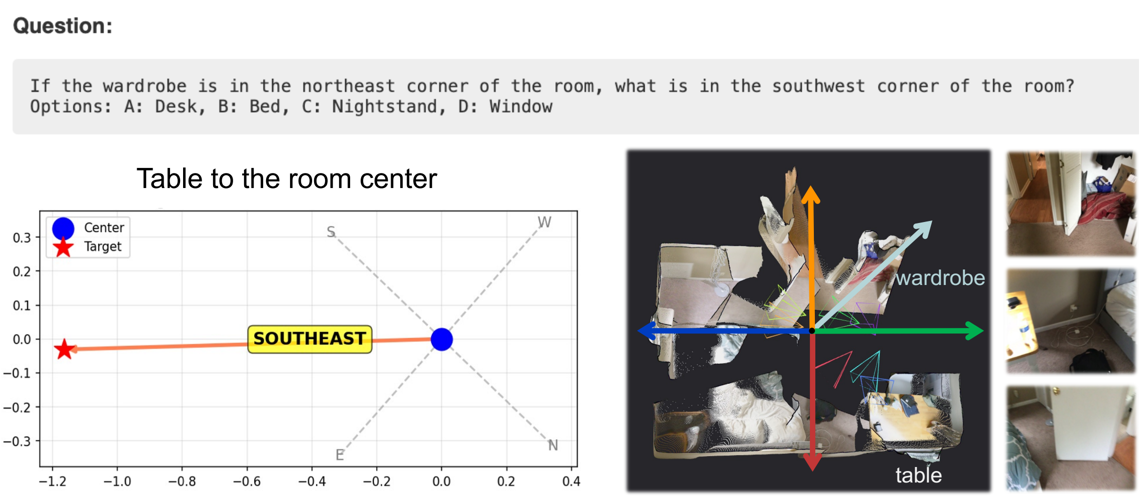


Figure 6: **Coordinate Calibration Example.** Here, the coordinate is calibrated by "wardrobe is in the northeast of the room". Then the relation of the table to the room center can be calculated as "Southeast".

Calibration via Reference Relation: Given a calibration statement "A is north of B," along with the 3D world coordinates of objects A (target) and B (anchor), both objects are first projected onto the ground plane estimated previously:

$$\mathbf{p}_A^G = \Pi(\mathbf{p}_A), \quad \mathbf{p}_B^G = \Pi(\mathbf{p}_B),$$

where $\Pi(\cdot)$ denotes orthogonal projection onto the ground plane. The direction from anchor to target is then defined as the *north* vector:

$$\mathbf{n}_N = \frac{\mathbf{p}_A^G - \mathbf{p}_B^G}{\|\mathbf{p}_A^G - \mathbf{p}_B^G\|}.$$

Constructing the Local Coordinate Frame: Let $\mathbf{n}_G$ denote the ground plane normal. The west vector is obtained as

$$\mathbf{n}_W = \frac{\mathbf{n}_G \times \mathbf{n}_N}{\|\mathbf{n}_G \times \mathbf{n}_N\|}.$$

By symmetry, the east and south vectors are defined as

$$\mathbf{n}_E = -\mathbf{n}_W, \quad \mathbf{n}_S = -\mathbf{n}_N.$$

Thus, the four orthogonal directions $\{\mathbf{n}_N, \mathbf{n}_S, \mathbf{n}_E, \mathbf{n}_W\}$ form the calibrated ground-plane coordinate system.

Directional Querying: To answer queries such as "the cabinet is in which direction relative to the chair," we set the chair as anchor and the cabinet as target. After projecting both to the ground plane, the relative vector

$$\mathbf{v} = \mathbf{p}_{\text{cabinet}}^G - \mathbf{p}_{\text{chair}}^G.$$

The directional relation is determined by measuring the angle between $\mathbf{v}$ and each of the basis vectors $\{\mathbf{n}_N, \mathbf{n}_S, \mathbf{n}_E, \mathbf{n}_W\}$, assigning the label corresponding to the vector with the smallest angular deviation.

Output: The module outputs a calibrated ground-plane coordinate system anchored to a reference relation, together with directional labels for object pairs under this system.

B.4 Situated Orientation Grounding (SOG) Module

Functionality: The Situated Orientation Grounding (SOG) module leverages *Visual Prompting* (Qi et al., 2025) to align complex, language-conditioned directional concepts with explicit 3D vectors.

Grounded Vector Generation: Given an input image, the camera pose, and the 3D location of a target object P_o (obtained from the Object Localization Module), we first select a representative point near the ground plane within the object's point cloud. This ensures that rendered arrows appear closer to the image bottom, mitigating perspective distortion (e.g., avoiding the degenerate case where arrows collapse into a single line at eye level). A sparse set of four orthogonal candidate vectors

$$\mathcal{D}_{\text{coarse}} = \{\vec{d}_i\}_{i=1}^4,$$

parallel to the ground plane and centered at P_o, is then generated to approximate compass-like directions.

Multi-View Visual Prompting: Each candidate arrow is projected into the current camera frame using the estimated camera parameters, preserving true 3D spatial relations. To further mitigate foreshortening, we synthesize a **Canonical View** by rotating the camera 45° upwards around the axis defined by the cross product of the camera-to-object vector and the ground plane normal. Formally, the virtual camera pose is

$$R_{\text{virt}} = \text{Rot}\big(R_{\text{orig}},\ 45^\circ,\ (\mathbf{v}_{co} \times \mathbf{n}_G)\big),$$

where $\mathbf{v}_{co}$ is the camera-to-object vector and $\mathbf{n}_G$ the ground normal. Both the Situated View (original camera) and Canonical View (synthetic top-down-like camera) are rendered and jointly input to the VLM (Gemini-2.5-Pro in our experiments). The VLM is prompted to select the arrow that best matches the query description.

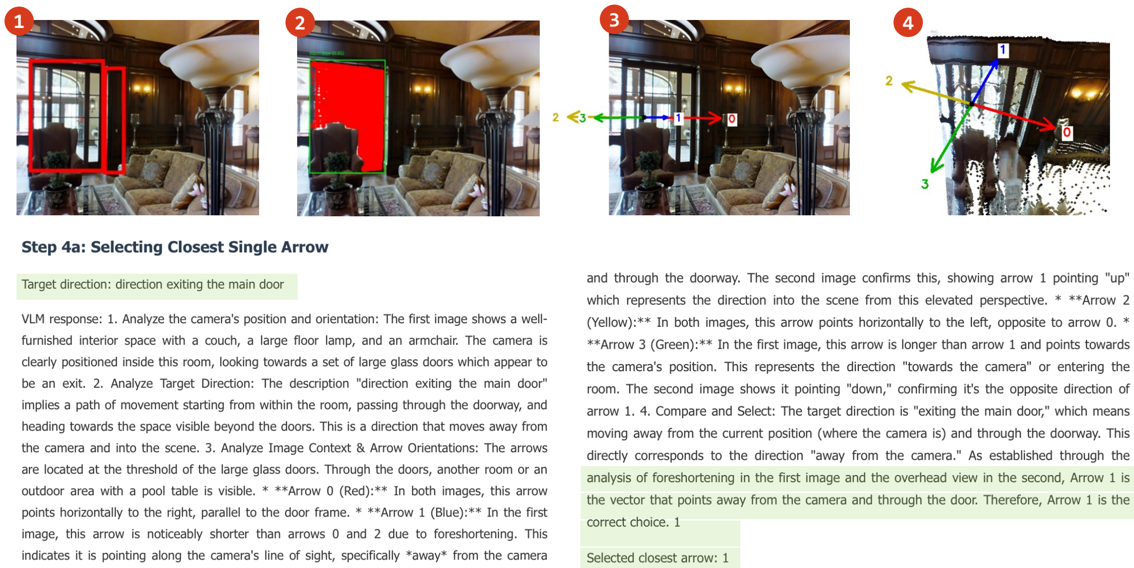


Figure 7: **Situated Orientation Grounding with multi-view overlay.** The target object is first localized and grounded in the scene. Candidate orientation arrows are overlaid on the original camera view (**Situated View**), which suffers from perspective distortion (e.g., arrows collapsing into near-collinear directions). To mitigate this effect, we additionally render a **Canonical View** from a virtual elevated camera pose. Combining both views, the VLM robustly selects the correct orientation vector despite visual foreshortening in the original view.

Iterative Refinement: To improve precision, a denser set of five candidate vectors is generated around the chosen coarse direction:

$$\mathcal{D}_{\text{fine}} = \{\mathrm{Rot}(\vec{d}_j, \theta) \mid \theta \in \{-45^\circ, -22.5^\circ, 0^\circ, 22.5^\circ, 45^\circ\}\},$$

where $\vec{d}_j$ is the vector selected in the coarse stage. The same multi-view prompting and VLM selection process is repeated, yielding the final grounded orientation vector. As illustrated in Fig. 7, overlaying candidate orientation arrows on the original camera view alone can suffer from severe perspective distortion, making disambiguation difficult. By incorporating the synthetic Canonical View, SOG enables the VLM to reason correctly about the intended direction, demonstrating robustness to challenging viewpoint effects.

Output: The resulting 3D vector $\vec{d}^*$ is added to the information set. This equips the reasoning agent with explicit, language-conditioned orientation vectors that support downstream tasks.

B.5 NUMERICAL COMPUTATION MODULES

Functionality: In addition to high-level reasoning modules, we design lightweight numerical utilities that provide interpretable relative spatial relationships. These modules support both camera motion analysis and object positioning, serving as building blocks for downstream reasoning.

Relative Camera Movement. Given two camera extrinsic matrices $E_0, E_1 \in \mathbb{R}^{4\times4}$, the module computes the relative transformation $T = E_1 E_0^{-1}$. From T, we decompose translation (t_x, t_y, t_z) and rotation angles $(\theta_x, \theta_y, \theta_z)$, which correspond to intuitive notions such as *forward/backward*, *right/left*, *up/down*, and yaw/pitch/roll. The output is a dictionary of movement descriptors, e.g.,

$$\{\text{forward}: 1.0,\ \text{right}: 0.0,\ \text{up}: 0.0,\ \text{rotate\_right}: 30^\circ,\ \text{rotate\_up}: 10^\circ\}.$$

Relative Object Position. Given a camera extrinsic matrix $E \in \mathbb{R}^{4\times4}$ and an object position $\mathbf{p} \in \mathbb{R}^3$ in world coordinates, the module transforms $\mathbf{p}$ into the camera coordinate frame:

$$\mathbf{p}^C = E \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}.$$

The resulting (x, y, z) is then expressed as *right*, *up*, and *forward* distances relative to the camera, yielding an interpretable relation such as $\{\text{forward}: 1.0,\ \text{right}: 0.0,\ \text{up}: 0.0\}$.

Output: These numerical descriptors provide a low-level geometric interface that complements higher-level grounding modules, enabling fine-grained spatial reasoning.

C Fine-tuning Details

We fine-tuned Qwen2.5-VL-7B (Bai et al., 2025) using LLaMAFactory (Zheng et al., 2024) on 4 NVIDIA A6000 GPUs. We adopted LoRA (Hu et al., 2022) with a global batch size of 32 and trained for 5 epochs, resulting in approximately 45 optimization steps in total. A peak learning rate of 2×10^{-5} with cosine decay and 3% warmup was used.

D Qualitative Results

To further illustrate the effectiveness of our system, we provide qualitative case studies. Figures 8 and 9 visualize the step-by-step execution trace of the Perception Agent, including its intermediate variables and the process of incrementally populating the information set (*analysis data*). Figure 10 then demonstrates how the Reasoning Agent consumes this structured information and performs detailed reasoning over it, ultimately producing the correct answer.

E Limitations

While our framework demonstrates strong performance across diverse spatial reasoning tasks, there are several limitations. First, our tools rely on geometric information provided by 3D reconstruction models. Although modern reconstruction models are highly advanced, they may still produce noisy or unstable results under challenging conditions. Such errors inevitably propagate to downstream modules. Second, our approach lacks an explicit mechanism for mitigating error accumulation between perception modules. For instance, a mis-localization in the Object Localization Module might be passed into incorrect directional reasoning in subsequent modules. Moreover, evaluation of situated spatial reasoning itself remains an open challenge, as standardized benchmarks and metrics are still underdeveloped.

We view these limitations as opportunities for future research, particularly toward more robust, error-tolerant reasoning pipelines for spatial reasoning.

F A Visualization Tool

The dual-agent pipeline naturally produces multiple forms of intermediate artifacts, including programmatic code snippets, execution traces, and reasoning trajectories across iterative steps. While these outputs are valuable for analysis, inspecting them separately can be cumbersome. To address this, we developed a lightweight web-based visualization tool built on Flask (flask, 2025) (Figure 11). This interface allows users to conveniently observe the entire problem-solving process of a given query, including code generation, execution results, reasoning iterations, and the evolution of the MSS. Such visualization serves as a useful resource for debugging, qualitative evaluation, and future research. We will release the complete source code of both the core framework and this visualization tool to the community.

G The Use of Large Language Models (LLMs)

In this work, LLMs were employed solely for language polishing and editing purposes. They were not involved in the design of methods, the development of algorithms, or the analysis of experimental results. All technical contributions, experiments, and conclusions are the product of the authors.

Question ID: 27

Question Type: Positional Relationship (Obj.–Reg.)

Question:

```
In the living room, the main door faces north; in which direction is the coffee table located in the dining room?
Options: A: West, B: East, C: South, D: North
```

Answer: A

Thought:

```
The direction in which the main door of the living room faces outward is north, and the coffee table is to the west of the
dining room.
```

Perception Agent

2: `import numpy as np`

5: `extrinsics, intrinsics, depth_maps, world_points = get_geo_info(images)`

Getting geometric info...

Successfully extracted geometric info for 2 images. Extrinsics: (2, 3, 4), Intrinsics: (2, 3, 3), Depth Maps: (2, 420, 518, 1)

6: `ground_info = ground_plane_detection(images)`

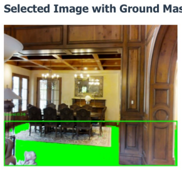
Ground Plane Detection

Step 1: Finding image with 'ground floor' using VQA

Finding objects: ground floor

Step 2: Best ground detection in image 2 with confidence 0.342

Selected Image with Ground Mask



Step 3: Running VGGT 3D reconstruction

Saving Point Cloud and Ground Plane Data

7: `ground_normal = ground_info["ground_normal"]`

8: `ground_centroid = ground_info["ground_centroid"]`

14: `analysis_data = {}`

18: `"context": "Identifying the image IDs and 3D world coordinates for the 'main door' and 'coffee table' to use them for spatial reasoning. The 'main door' is used for calibration, and the 'coffee table' is the target object."`

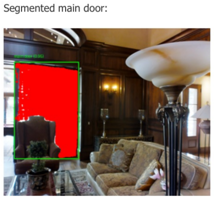
17: `analysis_data["object_detection_and_localization"] = {`

22: `object_prompts = ["main door", "coffee table"]`

23: `object_image_ids = find_obj(images, object_prompts)`

Finding objects: main door, coffee table

Segmented main door:



Success: Object 3D position from segmentation: [-0.340, -0.084, 1.137]

31: `"value": main_door_3d_position.tolist() if main_door_3d_position is not None else None,`

32: `"description": "3D world coordinates of the main door."`

30: `analysis_data["object_detection_and_localization"]["main_door_3d_position"] = {`

41: `coffee_table_image_id = object_image_ids.get("coffee table")`

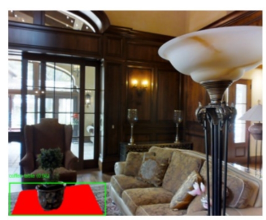
42: `if coffee_table_image_id is not None:`

43: `coffee_table_3d_position = get_object_3d_position(images[coffee_table_image_id], extrinsics, intrinsics, depth_maps, coffee_table_image_id, "coffee table")`

Getting 3D position of 'coffee table' in image 0 using segmentation

Locate: coffee table

Segmented coffee table:



45: `"value": coffee_table_3d_position.tolist() if coffee_table_3d_position is not None else None,`

46: `"description": "3D world coordinates of the coffee table."`

44: `analysis_data["object_detection_and_localization"]["coffee_table_3d_position"] = {`

56: `"context": "Grounding the orientation of the 'main door faces north' to establish the scene's cardinal directions. This involves determining the direction vector representing the 'facing' of the main door."`

55: `analysis_data["situated_orientation_grounding"] = {`

59: `main_door_facing_orientation = "direction facing the main door" # Assuming "faces north" implies the direction *from* the door *into* the room, which is often how such statements are interpreted.`

65: `main_door_orientation_data = situated_orientation_grounding(`

66: `images, extrinsics, intrinsics, depth_maps, world_points, ground_normal, ground_centroid, "direction exiting the main door"`

65: `main_door_orientation_data = situated_orientation_grounding(`

Situated Orientation Grounding: direction exiting the main door

Step 1: Unified Object and Image Selection

Analyzing 2 images for orientation: direction exiting the main door

Origin projected to pixel: [76.0, 210.0]

Successfully created overlay with 4 directional arrows

Original overlay image with directional arrows:

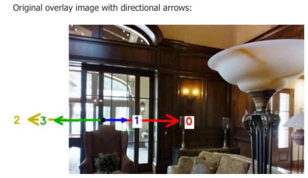
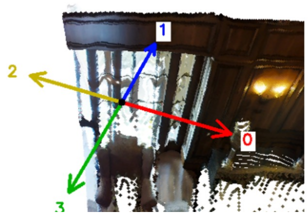


Figure 8: **Execution trace of the Perception Agent (part I).** This figure shows the first part of the Perception Agent's execution trace. The step-by-step code execution and intermediate variables are visualized, highlighting how raw observations are processed before being integrated into the shared analysis data.

Creating Elevated Camera View

Elevated view overlay image with directional arrows:



Selected closest arrow: 1

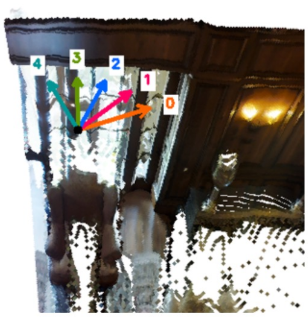
Camera x-axis in world: [1.000, -0.000, -0.000]

Projected camera x (v1): [1.000, 0.007, -0.000]

Perpendicular vector (v2): [0.000, -0.002, 1.000]

Created 5 fine-grained directions around arrow 1 (22.5° spacing):

Elevated fine-grained overlay image:




```
69: if main_door_orientation_data and main_door_orientation_data["direction_vector"] is not None:
71: "value": main_door_orientation_data["direction_vector"].tolist(),
72: "description": "3D unit vector representing the direction exiting the main door, which is calibrated as North."
70: analysis_data["situated_orientation_grounding"]["main_door_exiting_direction_vector"] = {
75: "value": main_door_orientation_data["position"].tolist(),
76: "description": "3D world coordinates of the main door used as reference for its orientation."
74: analysis_data["situated_orientation_grounding"]["main_door_position_for_orientation"] = {
90: "context": "Calibrating the scene's cardinal directions (North, South, East, West) based on the main door facing north. This establishes a consistent directional framework for the entire scene."
89: analysis_data["cardinal_direction_calibration"] = {
93: calibration_info = None
94: if main_door_orientation_data and main_door_orientation_data["direction_vector"] is not None:
95: calibration_info = calibrate_from_vector(
96: world_points,
97: main_door_orientation_data["direction_vector"],
98: main_door_orientation_data["position"],
99: "north", # The main door faces north
100: ground_normal
95: calibration_info = calibrate_from_vector(
```

Calibrating directions from direction vector...

Calibrating directions...

Using pre-computed ground plane normal

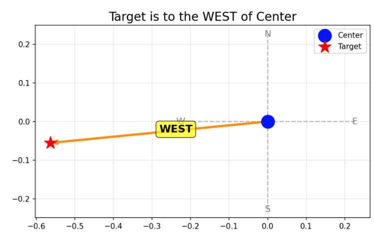
Ground plane normal: [0.007, -1.000, -0.002]

```
103: "north": calibration_info["north"].tolist(),
104: "south": calibration_info["south"].tolist(),
105: "east": calibration_info["east"].tolist(),
106: "west": calibration_info["west"].tolist(),
107: "description": "Calibrated 3D unit vectors for North, South, East, and West based on the main door's orientation."
102: analysis_data["cardinal_direction_calibration"]["calibrated_directions"] = {
117: "context": "Calculating the cardinal direction of the 'coffee table' relative to the overall scene (or a neutral center if no specific reference is given), using the established cardinal direction calibration."
116: analysis_data["coffee_table_direction_calculation"] = {
120: if calibration_info and coffee_table_3d_position is not None:
121: coffee_table_direction_relative_to_scene = calculate_direction(
122: calibration_info,
123: coffee_table_3d_position,
124: None, # Relative to the scene center (default behavior if center_3d_point is None)
125: ground_normal
121: coffee_table_direction_relative_to_scene = calculate_direction(
```

Calculating relative direction...

Using pre-computed ground plane normal

Using scene center for center point



Direction: Target is to the **west** of center

Angles - N: 95.6°, S: 84.4°, E: 174.4°, W: 5.6°

```
128: "value": coffee_table_direction_relative_to_scene["direction"],
129: "angles_deg": coffee_table_direction_relative_to_scene["angles_deg"],
130: "description": "The cardinal direction of the coffee table relative to the scene's calibrated center."
```

Figure 9: **Execution trace of the Perception Agent (part II).** A continuation of Figure 8, showing how the Perception Agent progressively fills the analysis data structure. This illustrates the transition from low-level perception results to a coherent, structured representation of the scene.

Reasoning Agent

```
STAGE 1: PLAN-GUIDED INFORMATION CURATION

  Reasoning Plan:
  1. The task is to determine the coffee table's cardinal direction in the dining room, given
that the main door faces north.
  2. Steps needed:
     - Use the main door to establish north.
     - Calibrate the cardinal directions (north, south, east, west).
     - Identify the coffee table's direction relative to the calibrated system.
  3. Only keep the information directly needed to confirm these steps.

  Information Curation Process:
  - object_detection_and_localization.context → General description, not needed. Discard.
  - object_detection_and_localization.object_image_ids → IDs not needed. Discard.
  - object_detection_and_localization.main_door_3d_position → Raw coordinates, superseded by
orientation grounding. Discard.
  - object_detection_and_localization.coffee_table_3d_position → Raw coordinates, superseded by
final calculation. Discard.
  - situated_orientation_grounding.context → General description, not needed. Discard.
  - situated_orientation_grounding.main_door_exiting_direction_vector → Establishes north. Keep.
  - situated_orientation_grounding.main_door_position_for_orientation → Reference point for
calibration. Keep.
  - cardinal_direction_calibration.context → General description, not needed. Discard.
  - cardinal_direction_calibration.calibrated_directions → Defines cardinal directions. Keep.
  - coffee_table_direction_calculation.context → General description, not needed. Discard.
  - coffee_table_direction_calculation.direction_relative_to_scene → Provides final answer for
coffee table direction. Keep.

  Refined mss Keys:
  - situated_orientation_grounding.main_door_exiting_direction_vector
  - situated_orientation_grounding.main_door_position_for_orientation
  - cardinal_direction_calibration.calibrated_directions
  - coffee_table_direction_calculation.direction_relative_to_scene

  STAGE 2: STRATEGIC DECISION-MAKING

  Decision Analysis:
  The refined set contains:
  - North orientation from the main door.
  - Calibrated cardinal directions.
  - Coffee table's computed relative direction.
  This is sufficient to solve the problem.


  {
      "action": 1,
      "refined_keys": [
          "situated_orientation_grounding.main_door_exiting_direction_vector",
          "situated_orientation_grounding.main_door_position_for_orientation",
          "cardinal_direction_calibration.calibrated_directions",
          "coffee_table_direction_calculation.direction_relative_to_scene"
      ],
      "reasoning": "The main door orientation grounds north. Calibration establishes the scene's
cardinal directions. The coffee table's computed relative direction is 'west', which matches
option A.",
      "final_answer": "A"
  }
```

1 stands for <Decide>

Figure 10: **Reasoning over analysis data.** The Reasoning Agent receives the populated analysis data from the Perception Agent and performs detailed multi-step reasoning. By grounding its inference in the structured information, the agent arrives at the correct final answer, demonstrating the synergy between perception and reasoning.

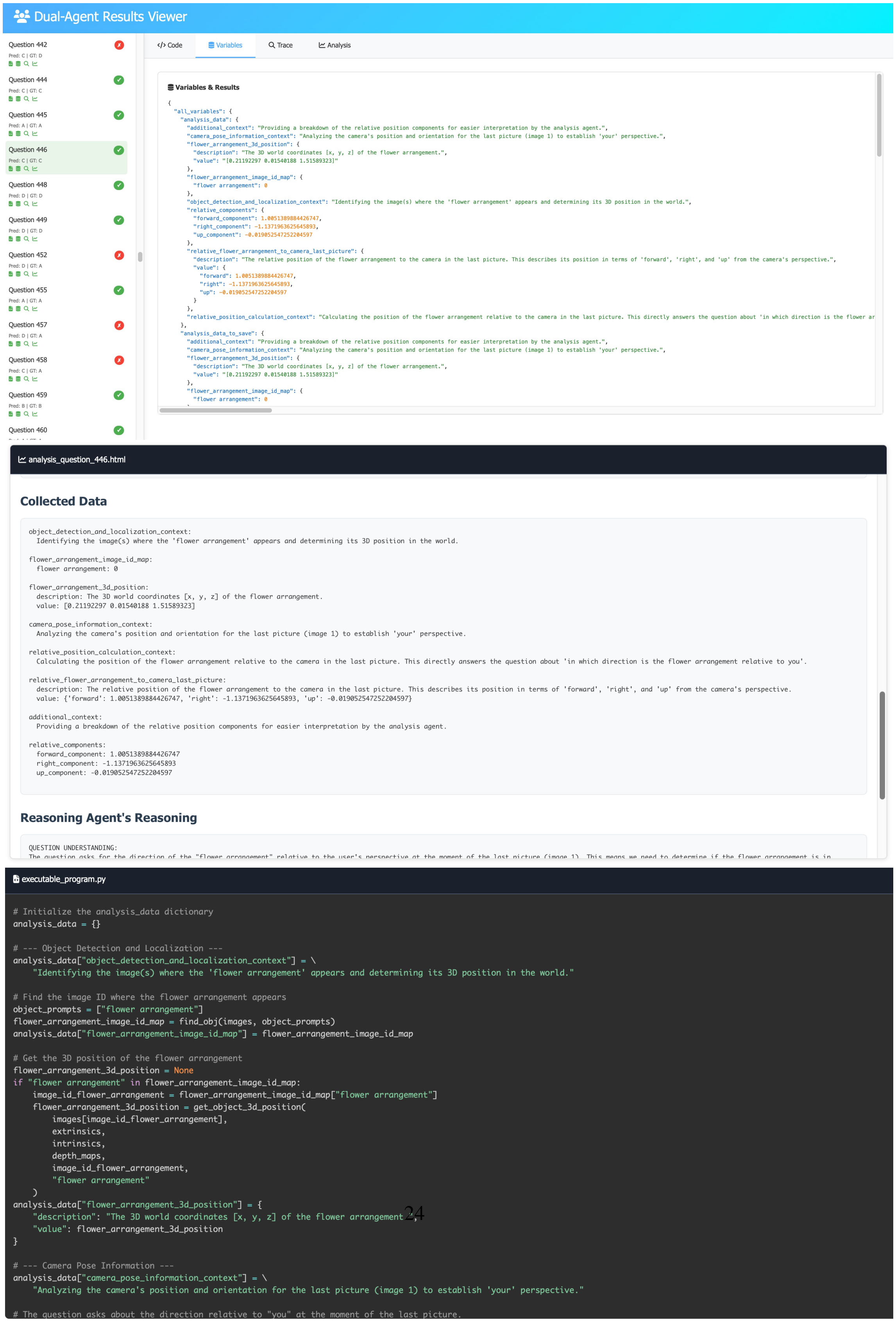


Figure 11: Web-based visualization tool for MSSR. The interface integrates code, execution traces, and reasoning trajectories in a unified view, enabling convenient inspection of the full problem-solving process.