WoRld As a Program: Programmable Spatial Reasoning for Multimodal LLMs

Anonymous authors
Paper under double-blind review

Abstract

Although MLLMs have advanced rapidly, spatial reasoning remains limited by implicit, geometrically unconstrained representations, while explicit tool-based approaches often rely on external perception modules for object grounding and geometric acquisition, underusing native visual-semantic capabilities. Recent advances in 3D reconstruction can recover explicit 3D representations of real-world environments, where scene entities are directly accessible and manipulable, suggesting 3D assets as a natural programmable substrate for spatial reasoning. This motivates a natural question: ***Can explicit 3D world states be made directly programmable for MLLMs?*** To this end, we propose ***WoRld As a Program (WRAP)***, a framework that converts standard video observations into an addressable and executable 3D scene state for MLLMs, where visual objects are exposed as *Grounded Variables* for *3D Program Reasoning*. ***WRAP*** explicitly binds visual object instances to their 3D entities as grounded variables, then enables the MLLM to generate executable reasoning programs over these variables while delegating only deterministic computation to an external executor. Extensive experiments on both video spatial reasoning and 3D scene editing benchmarks demonstrate the effectiveness and generality of ***WRAP***. Our method achieves state-of-the-art (SOTA) performance on ReVSI, improving video spatial reasoning accuracy by up to 5.2% over prior SOTA methods, while also outperforming existing approaches on EditRoom-DB test set, achieving an absolute IoU improvement of 0.19 over prior methods. These results demonstrate the programmable 3D representations benefit both video spatial reasoning and scene manipulation.

1 Introduction

Multimodal Large Language Models (MLLMs) have recently demonstrated impressive capabilities in perception and reasoning from images and videos (Liu et al., 2024; Bai et al., 2023; Feng et al., 2026). However, their spatial reasoning is still largely performed over implicit visual representations and autoregressive token generation, without explicitly enforcing the geometric structure underlying the observed scene. As a result, spatial evidence acquired from different views is not necessarily grounded in a single, globally consistent 3D representation, which can lead to failures in tasks that require metric estimation, viewpoint-invariant relations, or cross-view consistency. In this sense, current MLLMs may combine fragmented visual cues to produce plausible answers without necessarily constructing a coherent `spatial mental model` (Tversky, 1993) of the environment.

To address this limitation, recent studies have explored two major paradigms for enhancing spatial reasoning in MLLMs: *Implicit Spatial Modeling* and *Explicit Geometry Modeling*. Implicit spatial modeling methods (Chen et al., 2026b; Fan et al., 2026; Zheng et al., 2025a; Chen et al., 2026c; Yang et al., 2026a; Cui et al., 2026; Gao et al., 2026; Qu et al., 2026; Cai et al., 2026) incorporate spatial knowledge into MLLMs through spatially enriched training data or dedicated spatial encoders, encouraging the models to learn implicit representations of spatial structure. However, such representations are not necessarily stable or globally consistent. For example, as shown in Figure 1, even on absolute-distance tasks that are independent of temporal order, simply reversing the temporal order of the input video frames can lead to substantial changes in

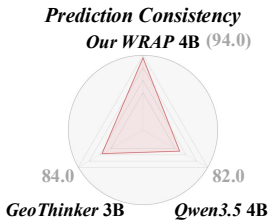


Figure 1: Measured by the concordance correlation coefficient (CCC).

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

model predictions. In contrast, explicit geometry modeling methods (Dai et al., 2026; Cho et al., 2026; Zhang et al., 2026b; Luo et al., 2026; Marsili et al., 2025) rely on external spatial experts to construct explicit geometric representations of the spatial structure, upon which MLLMs perform spatial reasoning. However, query-relevant visual objects are often identified through external perception modules, intertwining semantic object grounding with geometric acquisition inside the tool pipeline. This design limits the direct use of the native visual-semantic capabilities of MLLMs for resolving query-specific object references.

Recent advances in 3D reconstruction models (World Labs, 2026; Leroy et al., 2024; Wang et al., 2026; 2025; 2024) make it possible to recover globally consistent 3D geometry from multi-view observations, transforming transient visual inputs into explicit and structured 3D scene representations. Such representations make scene entities and geometric relations directly accessible to downstream systems, opening up the possibility of operating on the world through explicit, executable computations rather than relying solely on implicit visual inference. This naturally raises a key question: ***Can explicit 3D world states be made directly programmable by MLLMs?*** To explore this question, we introduce ***WoRld As a Program (WRAP)***, which connects video observation to an externalized 3D scene state and exposes grounded scene entities as programmable variables for MLLMs. Our study is organized around three questions:

Q1: How can MLLMs bind objects in videos to addressable 3D entities?

We address this through **Visual-Geometric Entity Binding**, which establishes explicit correspondences between visual object instances and their associated 3D point-cloud entities. Specifically, we reconstruct the scene geometry from the input video, aggregate object-level point clouds across frames, and assign each physical object a persistent symbolic identifier. These identifiers serve as visually grounded and addressable 3D variables, allowing subsequent reasoning to directly reference the corresponding external geometric entities.

Q2: How can MLLMs formulate executable spatial programs over these grounded variables?

We introduce **MLLM-Native Spatial Reasoning**, which enables the model to formulate executable spatial programs over grounded 3D entities while keeping the reasoning procedure within the MLLM. Specifically, given a query and video, the MLLM first identifies the relevant 3D object variables and then explicitly constructs the required reasoning procedure, such as geometric computation or scene manipulation, in the form of code. Importantly, the MLLM determines what should be computed and how, while only final deterministic execution is delegated to an external executor.

Q3: Can WRAP generalize across spatial reasoning and scene editing?

To answer Q3, we evaluate ***WRAP*** across both video spatial reasoning and 3D scene editing tasks. ***WRAP*** achieves state-of-the-art (SOTA) performance on spatial reasoning benchmarks, including ReVSI (Zhang et al., 2026a), while also outperforming existing on the EditRoom-DB (Zheng et al., 2025b) 3D scene editing benchmark. Mechanistic analysis further verifies the importance of accurate object grounding and reliable program generation for spatial reasoning. Together, these results demonstrate the effectiveness of ***WRAP*** across spatial understanding and scene manipulation.

Our contributions can be summarized as:

- We propose ***WRAP***, a spatial reasoning framework that connects standard video observations to an external, executable 3D scene state. Rather than directly feeding 3D representations into the MLLM, ***WRAP*** exposes grounded scene entities as *Grounded Variables*, enabling the model to retain its native video interface while accessing explicit geometry when needed.
- We introduce ***Visual-Geometric Entity Binding*** to grounded visual objects as addressable 3D variables (*Grounded Variables*), and ***MLLM-Native Spatial Reasoning*** to enable *3D Program Reasoning* by formulating executable geometric programs over grounded variables, delegating only deterministic computation to an external executor.
- We construct a 27**K training dataset** for 3D program reasoning, covering grounded variable understanding, adaptive 3D activation, and executable spatial program generation, and introduce a two-stage learning strategy to progressively acquire these capabilities.
- We empirically demonstrate the effectiveness of ***WRAP*** across video spatial reasoning and 3D scene editing, achieving state-of-the-art performance with a relative improvement of **8.5%** on ReVSI and a **25.6%** relative improvement in IoU on EditRoom-DB test sets.

2 Related Work

Recent studies have advanced spatial reasoning beyond visual semantics by incorporating geometric priors and explicit 3D structure, generally falling into two paradigms: ***Implicit Spatial Modeling*** and ***Explicit Geometry Modeling***.

Implicit Spatial Modeling Methods. These methods enhance spatial reasoning either by introducing explicit geometry encoders or by scaling spatial reasoning supervision. Existing methods (Fan et al., 2026; Zheng et al., 2025a; Li et al., 2026) incorporate dedicated geometric representations or encoders into MLLMs, while Cambrian-S (Yang et al., 2026b) and SpaceR (Ouyang et al., 2025) primarily improve spatial reasoning through large-scale spatially enriched training data and reasoning supervision. However, such implicit spatial representations may lack stability and global consistency: as shown in Figure 1, reversing the video frame order reduces the prediction consistency of GeoThinker (Li et al., 2026) to a CCC of 84, while ***WRAP*** remains substantially more stable.

Explicit Geometry Modeling Methods. These approaches externalize spatial information into explicit 3D representations or executable geometric operations. pySpatial (Luo et al., 2026) generates Python programs over reconstructed 3D geometry, VADAR (Marsili et al., 2025) dynamically synthesizes spatial APIs, while S-Agent (Dai et al., 2026) and SpatialClaw (Cho et al., 2026) perform tool- or code-based geometric reasoning. However, query-relevant objects are often identified through external perception modules, coupling semantic grounding with geometric acquisition outside the MLLM. Thinking with Spatial Code (Chen et al., 2026a) and GR3D (Yuan et al., 2026) instead expose temporally coherent or ID-linked object geometry as structured textual representations, but the subsequent geometric computation remains implicit in textual reasoning, which may limit reliability for precise metric tasks.

Overall, prior methods either ***lack a globally consistent 3D state*** or ***rely on external object grounding*** or ***implicit textual computation***. In contrast, ***WRAP*** binds visual instances to persistent 3D entities as Grounded Variables, preserving native visual grounding while enabling executable reasoning over globally consistent object geometry.

3 World As a Program

Our ***WRAP*** aims to turn visual objects in videos into persistent, grounded 3D entities that MLLMs can directly reference and program over for spatial reasoning. To this end, ***WRAP*** binds video objects to addressable 3D entities, termed *Grounded Variables*, and enables the MLLM to formulate executable spatial programs directly over them, while delegating only deterministic computation to an external executor. ***WRAP*** consists of two key components: ***Visual-Geometric Entity Binding***, which instantiates 3D objects as grounded variables, and ***MLLM-Native Spatial Reasoning***, which formulates executable spatial programs over these variables (*3D Program Reasoning*). To equip MLLMs with this reasoning paradigm, we further introduce a two-stage learning framework that progressively learns visual-geometric grounding and programmatic spatial reasoning.

3.1 Visual-Geometric Entity Binding

Since current MLLMs are primarily designed for video and image inputs, with limited capability to directly process raw 3D point clouds, we first project object instances observed in the video into the reconstructed 3D space to obtain their corresponding point-cloud representations (**3D Object Instantiation**). We then establish explicit correspondences between the 2D object instances in the video and their instantiated 3D counterparts, enabling 2D visual-to-3D geometric entity binding (**2D-3D Entity Binding**).

3D Object Instantiation. To construct instance-level 3D representations from the input video, we first use an off-the-shelf reconstruction model (*e.g., DepthAnything 3 (Lin et al., 2025)*) to estimate per-frame depth, camera intrinsics, and extrinsics. We then prompt the MLLM to identify scene object categories and use them to guide SAM3 (Carion et al., 2026) for instance segmentation. Since SAM3 may fail to preserve instance identity over long videos, especially under long-term disappearance or multiple same-category instances, we further introduce geometry-based instance merging. Each instance mask is back-projected into a shared world coordinate system and aggregated across

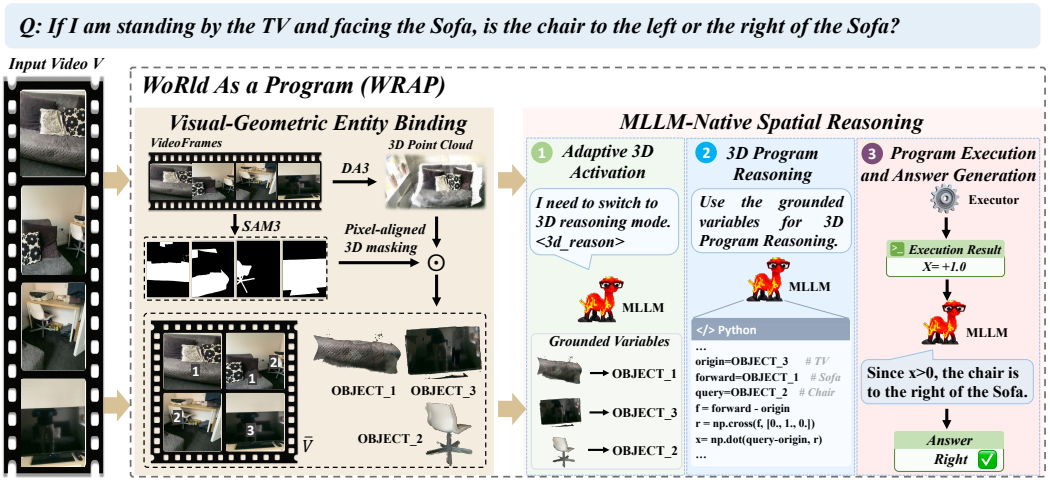


Figure 2: ***Detailed illustration of our proposed WRAP with a running example.*** Given an input video and query, ***WRAP*** performs ***Visual-Geometric Entity Binding*** to associate visual object instances with their corresponding 3D point clouds as grounded variables (*e.g.,* `OBJECT_1`). ***WRAP*** then performs ***MLLM-Native Spatial Reasoning*** in three steps: ❶ ***Adaptive 3D Activation*** determines whether explicit 3D reasoning is required, ❷ ***3D Program Reasoning*** formulates an executable geometric program over the grounded variables, and ❸ The program is executed to obtain the geometric result, which is then fed back to the same MLLM for final answer generation.

frames to form a voxelized point cloud $\mathcal{P}_i$. For same category instance pairs, we first filter candidates by 3D bounding-box distance, then measure bidirectional geometric correspondence with KD-trees. Let $||\mathcal{P}_A||$ and $||\mathcal{P}_B||$ denote the point counts of objects A and B, and $||\mathcal{P}_{A_m}||$ and $||\mathcal{P}_{B_m}||$ denote the numbers of points with a neighbor in the other point cloud within a predefined radius. We define the geometric overlap score as:

$$s(A, B) = \max\left(\frac{||\mathcal{P}_{A_m}||}{||\mathcal{P}_A||}, \frac{||\mathcal{P}_{B_m}||}{||\mathcal{P}_B||}\right). \tag{1}$$

If $s(A, B) > \alpha$, with $\alpha = 0.5$, the two tracks are considered observations of the same physical object and are merged. All pairwise matches are then transitively consolidated using a disjoint-set structure, producing a unified instance identity and an aggregated 3D point cloud for each object.

2D-3D Entity Binding. After constructing an instance-level 3D point cloud $\mathcal{P}_i$ for each physical object, we establish explicit correspondences between visual observations and their geometric entities. Rather than feeding raw point clouds into the MLLM, we preserve the native video interface and expose each 3D object through a symbolic visual reference. Inspired by Set-of-Mark (Yang et al., 2023; Yuan et al., 2026) prompting, we assign each object a unique tag in the video frames $\bar{V}$ and a symbolic identifier $s_i = \texttt{OBJECT_}i$, referred to as a *Grounded Variable*. This defines a deterministic binding $\Phi : s_i \longleftrightarrow \mathcal{P}_i$, with the same identifier rendered alongside the corresponding visual instance. During 3D program reasoning, any reference to $\texttt{OBJECT}_i$ can therefore be directly resolved to its associated 3D point cloud $\mathcal{P}_i$. This visual-symbolic-geometric binding provides a unified interface between video perception and explicit 3D computation without requiring the MLLM to directly process point-cloud representations.

3.2 MLLM-Native Spatial Reasoning

Building on **Visual-Geometric Entity Binding**, we expose video objects and their associated 3D point clouds as grounded variables to the MLLM. The model performs **3D Program Reasoning** based on the tagged video $\bar{V}$ and these grounded variables. Since object tags may introduce unnecessary visual clutter for queries that do not require explicit 3D program reasoning (*e.g.,* appearance ordering), we adopt **Adaptive 3D Activation**: the model operates on clean (untagged) video by default and switches to tagged video only when explicit 3D program reasoning is required. Both modes share the same MLLM and differ only in the system prompt and visual input.

3D Program Reasoning. Conditioned on the tagged video $\bar{V}$ and query Q, the MLLM identifies the required spatial operation and generates an executable program C that explicitly specifies the reasoning procedure, such as relative-direction computation or 3D scene editing. To make grounded variables directly accessible during execution, all grounded variables and their associated 3D representations are registered in the executor `namespace`, allowing identifiers such as `OBJECT_`i to be deterministically resolved to their underlying 3D entities. The generated program can therefore operate directly on scene geometry without additional grounding or task-specific tool selection. Execution is defined as $\text{Exec}(C, E_\Phi, S) \rightarrow (z, S')$, where E_Φ denotes the executor environment, S and S' are the scene states before and after execution, and z is the execution result. The MLLM then produces the final answer conditioned on the tagged video, query, generated program, execution result, and updated scene state.

Adaptive 3D Activation. Given a clean video V and a query Q, the MLLM first determines whether explicit 3D program reasoning is necessary. For queries that can be answered directly from the visual input (*e.g.,* appearance ordering), the model remains in the standard reasoning mode. When the model emits the designated trigger `<3d_reason>`, inference switches to 3D program reasoning, and the clean video V is replaced with its tagged counterpart $\bar{V}$, where each visual object is associated with a grounded variable.

3.3 Two Stage Training for WRAP

To equip MLLMs with this reasoning paradigm, we adopt a two-stage learning framework for ***WRAP***. In **Stage 1**, the MLLM is trained to acquire basic spatial perception capabilities and interpret tagged video frames. In **Stage 2**, we construct 27K spatial program learning samples with tagged videos to teach the model when to activate 3D program reasoning and how to reason over grounded variables. Details of the training data are provided in the Appendix A.1.1.

Stage 1: Spatial Grounding. In this stage, we equip the MLLM with basic spatial perception capabilities and the ability to interpret tagged video. Given a video V, query Q, and the corresponding target response $Y = \{y_t\}_{t=1}^{T}$, we optimize the model using standard autoregressive objective:

$$\mathcal{L}_{S1} = -\sum_{t=1}^{T} log p_\theta(y_t | y_{<t}, V, Q). \tag{2}$$

The training data cover diverse spatial perception tasks and include both clean and tagged videos, enabling the model to recognize spatial relationships while learning the semantics of visual tags such as `OBJECT_`i. This stage provides the perceptual and grounding foundation required for subsequent 3D program reasoning.

Stage 2: Spatial Program Learning. In this stage, we further train the MLLM to determine when explicit 3D program reasoning is required and to generate executable programs over grounded variables. The objective consists of three terms: $\mathcal{L}_{adaptive}$ supervises the reasoning-mode decision, encouraging the model to either remain in the standard reasoning mode or emit the `<3d_reason>` trigger. $\mathcal{L}_{3D}$ supervises program generation and the corresponding final answer under tagged video input. $\mathcal{L}_{standard}$ supervises direct answering from the clean video. Formally, we optimize

$$\mathcal{L}_{S2} = \mathcal{L}_{\text{adaptive}} + \mathcal{L}_{\text{3D}} + \mathcal{L}_{\text{standard}}. \tag{3}$$

Here, $\mathcal{L}_{adaptive} = -log p_\theta(r|V, Q)$ supervises the target reasoning mode, where $r \in \{\text{standard}, \text{3D}\}$ denotes the target reasoning mode, with the 3D mode corresponding to emission of the `<3d_reason>` trigger. $\mathcal{L}_{3D}$ and $\mathcal{L}_{standard}$ are standard autoregressive cross-entropy losses over the 3D program reasoning trajectory conditioned on $(\bar{V}, Q)$ and the standard reasoning trajectory conditioned on (V, Q), respectively.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate ***WRAP*** on four benchmarks spanning video spatial reasoning tasks and 3D scene editing tasks. For video spatial reasoning, VSI-Bench (Yang et al., 2025) covers metric

Table 1: ***Comparison of WRAP with existing methods on ReVSI.*** Results are reported across numerical and multiple-choice video spatial reasoning tasks. Best results are highlighted in **bold**.

Method	*Frames*	*Numerical Question*				*Multiple-Choice Question*			*Avg.*
		Obj. Cnt.	*Abs. Dist.*	*Obj. Size*	*Room Size*	*Rel. Dist.*	*Rel. Dir.*	*Route Plan*	
Baseline									
Chance (Random)	ALL	-	-	-	-	23.7	26.8	26.0	-
Chance (Frequency)	ALL	52.2	40.1	17.4	20.9	25.8	31.9	30.2	31.4
Proprietary Models									
GPT-5.2	64	56.2	41.5	73.9	**63.0**	48.4	34.9	38.2	50.9
Gemini 3 Flash	1 FPS	**65.7**	53.1	77.6	52.8	64.6	47.9	41.8	57.6
Gemini 3 Pro	1FPS	60.1	54.7	**79.3**	51.9	68.1	56.0	56.4	60.9
Open-Source General Models									
Qwen3-VL-8B-Instruct	64	40.4	52.3	69.0	45.1	57.1	39.5	40.5	49.1
Qwen3-VL-32B-Instruct	64	46.9	65.0	70.4	55.8	53.8	34.0	47.3	53.3
InternVL3.5-8B	64	43.3	54.6	64.2	47.6	45.0	36.3	44.4	47.9
InternVL3.5-38B	64	43.8	60.6	70.2	58.4	57.4	45.9	42.7	54.1
LLaVA-Video-7B-Qwen2	64	31.3	1.4	52.5	16.7	38.3	33.3	38.4	30.3
LLaVA-Video-72B-Qwen2	64	40.1	29.6	59.3	27.9	39.6	24.8	43.0	37.8
Spatially Specialized Models									
Cambrian-S-7B	128	48.4	60.5	65.5	46.7	37.1	48.5	37.0	49.1
Qwen2.5-VL-7B-Instruct	4 FPS	36.9	15.0	49.7	29.0	31.5	29.5	36.7	32.6
VST-7B-SFT	4 FPS	35.4	52.6	67.9	47.2	49.2	36.9	35.4	46.4
SpaceR-7B	32	30.7	34.5	52.0	18.6	22.8	34.5	20.2	30.5
Qwen2.5-VL-3B-Instruct	16	18.7	15.6	16.8	-	33.2	34.3	-	23.7
Spatial-MLLM-4B-820K	16	41.5	40.0	53.1	-	30.7	39.2	-	40.9
LLaVA-Video-7B-Qwen2	32	29.9	1.5	53.0	19.3	39.1	33.8	38.8	30.8
VLM3R-7B	32	41.6	61.6	64.8	52.5	46.5	49.5	34.1	50.1
GeoThinker 7B	64	38.6	67.0	64.2	54.7	40.5	46.8	41.2	50.4
GeoThinker 8B	64	37.6	73.7	66.8	55.0	62.5	48.5	41.6	55.1
S-Agent	64	54.0	45.6	62.6	53.4	63.6	66.4	66.1	58.8
WRAP 2B	64	35.5	75.9	62.7	45.2	68.7	84.1	60.9	61.9
WRAP 4B	64	40.1	75.8	63.9	53.1	73.4	81.7	58.8	63.8
WRAP 9B	64	40.2	**77.1**	64.3	51.6	**74.7**	**86.6**	**68.5**	**66.1**

and spatiotemporal reasoning from egocentric videos, while VSI-Bench-Debiased (II et al., 2026) reduces non-visual shortcuts to better isolate visually grounded reasoning. ReVSI (Zhang et al., 2026a) further improves evaluation reliability by correcting annotation errors and ensuring answerability under the given frame budget. For 3D scene editing, EditRoom-DB (Zheng et al., 2025b) evaluates language-guided room-layout manipulation, including object translation, rotation, scaling, replacement, addition, and removal.

Models. We adopt Qwen3.5 (Team, 2026) with 2B, 4B, and 9B parameters as the MLLM backbones. SAM3 (Carion et al., 2026) is used to extract object-level segmentation masks from the input videos, while DepthAnything 3 (Lin et al., 2025) reconstructs the corresponding 3D point clouds. Specifically, we employ DA3NESTED-GIANT-LARGE 1.1 for metric 3D reconstruction, enabling geometric measurements in real-world scale. We additionally extract the floor point cloud to support spatial reasoning relative to the scene geometry.

Training Data. For Stage 1, we train on approximately 700K samples from SoM-LLaVA (Yan et al., 2024) and VSI-590K (Yang et al., 2026b) to establish robust spatial grounding and general spatial perception. For Stage 2, we construct 25K program-oriented spatial reasoning samples from ScanNet (Dai et al., 2017) and incorporate an additional 2K EditRoom-DB (Zheng et al., 2025b) examples to extend the learned reasoning paradigm to 3D scene editing. Details of the training data are provided in the Appendix A.1.1.

4.2 Results on Video Spatial Reasoning

We report the results on ReVSI, VSI-Bench in this section while the results on VSI-Bench-Debiased are provided in Appendix A.2.2.

Results on ReVSI. As shown in Table 1, our ***WRAP*** consistently outperforms both general-purpose MLLMs and spatially specialized baselines, demonstrating the benefit of programmable 3D reasoning. Among the evaluated variants, ***WRAP*** 9B achieves the best overall performance with an average score of 66.1%, surpassing the strongest proprietary baseline, Gemini 3 Pro, by 5.2%, and the strongest spatially specialized baseline, S-Agent (Dai et al., 2026), by 7.3%. The gains are particularly pronounced on geometry-intensive tasks, including absolute distance, relative distance, relative direction, and route plan, where ***WRAP*** benefits from explicit 3D object representations

Table 2: ***Comparison of WRAP with existing methods on VSI-Bench.*** Results cover numerical and multiple-choice video spatial reasoning tasks, with the best results in bold.

Method	***Numerical Question***				***Multiple-Choice Question***				***Avg.***
	Obj. Cnt.	*Abs. Dist.*	*Obj. Size*	*Room Size*	*Rel. Dist.*	*Rel. Dir.*	*Route Plan*	*Appr. Order*	
Baseline									
Chance (Random)	-	-	-	-	25.0	36.1	28.3	25.0	-
Chance (Frequency)	62.1	32.0	29.9	33.1	25.1	47.9	28.4	25.2	34.0
General-purpose Models									
GPT-4o	46.2	5.3	43.8	38.2	37.0	41.3	31.5	28.5	34.0
Gemini-2.5 Pro	43.8	34.9	64.3	42.8	61.1	47.8	45.9	71.3	51.5
InternVL-3 8B	68.1	39.0	48.4	33.6	48.3	36.4	27.3	35.4	42.1
Qwen2.5VL 7B	25.2	10.5	36.4	29.6	38.4	38.0	29.8	26.8	29.3
Spatially Specialized Models									
3DThinker 3B	-	-	-	-	-	-	-	-	59.1
3DThinker 7B	-	-	-	-	-	-	-	-	63.7
VST 7B	71.6	43.8	75.5	69.2	60.0	55.6	44.3	69.2	61.2
VLM-3R 7B	70.2	49.4	69.2	67.1	65.4	80.5	45.4	40.1	60.9
VG-LLM 8B	67.9	37.7	58.6	62.0	46.6	40.7	32.4	59.2	50.7
Cambrian-S 3B	70.7	40.6	68.0	46.3	64.8	61.9	27.3	78.8	57.3
Cambrian-S 7B	**73.2**	50.5	**74.9**	**72.2**	**71.1**	76.2	41.8	**80.1**	67.5
GeoThinker 7B	-	-	-	-	-	-	-	-	68.5
GeoThinker 8B	-	-	-	-	-	-	-	-	**72.6**
WRAP 2B	69.1	68.1	71.6	65.2	65.2	85.8	47.9	76.9	68.7
WRAP 4B	68.8	**69.2**	74.2	67.1	67.5	**87.5**	**50.0**	73.3	69.7
WRAP 9B	68.5	68.9	74.5	66.9	67.6	87.2	45.4	76.7	69.5

and executable geometric reasoning. Notably, even the 2B variant reaches 61.9% average accuracy, outperforming substantially larger general-purpose and specialized models, indicating that ***WRAP*** improves spatial reasoning capability rather than relying solely on model scale.

Results on VSI-Bench. As shown in Table 2, our ***WRAP*** achieves strong performance across model scales, with ***WRAP*** 4B reaching an average score of 69.7% and showing particularly strong results on geometry-intensive tasks such as absolute distance and relative direction. Notably, even ***WRAP*** 2B attains an average score of 68.7%, demonstrating that programmable 3D representations can substantially enhance spatial reasoning with a compact backbone. Although ***WRAP*** does not surpass GeoThinker 8B (Li et al., 2026), GeoThinker is trained on approximately 1.8M samples, compared with about 727K for ***WRAP***, while ***WRAP*** remains competitive on several geometry-intensive tasks.

4.3 Results on 3D Scene Editing

To demonstrate that our ***WRAP*** generalizes beyond video spatial reasoning, we further evaluate ***WRAP*** on 3D scene editing using the EditRoom-DB. We first render the original 3D scene into images and extract the objects together with their corresponding point clouds. Each sample is paired with an editing instruction specifying the desired operation, such as rotating the TV by 90°. ***WRAP*** then identifies the target object from the tagged images, generates the corresponding program, and passes it to the executor for execution. The modified scene is subsequently saved as the final output. In addition, the original EditRoom-DB metrics are computed over the entire scene, which can yield relatively high scores even when the original scene is left unchanged. To provide a more object-focused evaluation, we introduce Edit-IoU and Edit-S-IoU, which compute IoU and S-IoU only for the objects that are required to be edited. These metrics more directly measure whether correctly modifies the target object states and complement the original scene-level evaluation metrics.

As shown in Table 3, ***WRAP*** outperforms prior methods (Tang et al., 2024; Zheng et al., 2025b) across all three scene categories, with particularly strong gains in IoU and S-IoU, while achieving lower LPIPS and higher CLIP scores. On the object-focused evaluation, in Table 4, ***WRAP*** consistently surpasses EditRoom on both Edit-IoU and Edit-S-IoU across all available categories. These results show that the benefits of ***WRAP*** extend beyond video spatial reasoning to 3D scene editing.

4.4 Ablation Study

To better understand the contribution of each component in ***WRAP***, we conduct ablation studies with the 4B model on ReVSI and VSI-Bench. As shown in Table 5, introducing ***Tagged Video*** improves ReVSI from 45.5% to 50.8%, while slightly reducing VSI-Bench from 65.8% to 65.3%. This suggests that object tags facilitate explicit object referencing but may also introduce visual clutter, as

Table 3: ***Comparison of WRAP with prior methods on EditRoom-DB*** across bedroom, dining room, and living room scenes using IoU, S-IoU, LPIPS, and CLIP metrics. *No-Op* is a baseline that leaves the input scene unchanged, measuring the score achievable without executing the requested edit. † denotes results reproduced by us using the official implementation.

Method	*Bedroom*				*Dining room*				*Living room*			
	IOU ↑	*S-IOU* ↑	*LPIPS* ↓	*CLIP* ↑	*IOU* ↑	*S-IOU* ↑	*LPIPS* ↓	*CLIP* ↑	*IOU* ↑	*S-IOU* ↑	*LPIPS* ↓	*CLIP* ↑
No-Op	0.8539	0.8506	0.1007	0.9651	0.9257	0.9240	0.0492	0.9807	0.9287	0.9269	0.0470	0.9836
DiffuScene-N	0.6220	0.6125	0.1431	0.9415	0.4473	0.4321	0.1921	0.9254	0.4332	0.4161	0.1810	0.9236
SceneEditor-N	0.7055	0.6942	0.1261	0.9510	0.5262	0.5091	0.1557	0.9366	0.4618	0.4498	0.1738	0.9329
EditRoom	0.7342	0.7236	0.1090	0.9597	0.5360	0.5196	0.1460	0.9460	0.4710	0.4629	0.1616	0.9446
EditRoom†	0.7278	0.7063	0.1326	0.9503	0.5402	0.5193	0.1529	0.9416	0.4699	0.4136	0.1726	0.9369
WRAP 2B	**0.9224**	**0.9221**	0.0432	**0.9853**	**0.9333**	**0.9330**	0.0323	**0.9885**	**0.9438**	**0.9434**	**0.0265**	**0.9895**
WRAP 4B	0.9215	0.9212	**0.0417**	0.9852	0.9293	0.9289	0.0338	0.9866	0.9360	0.9356	0.0287	0.9845
WRAP 9B	0.9143	0.9140	0.0445	0.9811	0.9324	0.9320	**0.0311**	0.9861	0.9375	0.9371	0.0283	0.9886

Table 4: ***Object-focused evaluation on EditRoom-DB*** using the proposed Edit-IoU and Edit-S-IoU metrics across bedroom, dining room, and living room scenes. *No-Op* denotes leaving the original scene unchanged without performing the requested edit. † denotes results from our reimplementation using the official codebase.

Method	*Bed Room*		*Dining Room*		*Living Room*	
	EDIT-IOU↑	*EDIT-S-IOU*↑	*EDIT-IOU*↑	*EDIT-S-IOU*↑	*EDIT-IOU*↑	*EDIT-S-IOU*↑
No-Op	0.2377	0.2202	0.2120	0.1939	0.2242	0.2047
EditRoom†	0.4594	0.4344	0.2456	0.2233	0.2659	0.2233
WRAP 2B	**0.6784**	**0.6774**	**0.5589**	**0.5568**	**0.5708**	**0.5688**
WRAP 4B	0.6683	0.6673	0.5211	0.5189	0.5555	0.5535
WRAP 9B	0.6654	0.6645	0.5503	0.5482	0.5509	0.5489

reflected by a 6% drop on the Appearance Order task. Incorporating ***3D Program Reasoning*** further improves performance to 61.9% on ReVSI and 66.0% on VSI-Bench, highlighting the benefit of executable reasoning over grounded variables. Finally, ***Adaptive 3D Activation*** achieves the best performance on both benchmarks, reaching 63.8% and 69.7%, respectively, indicating the benefit of selectively invoking tagged videos and explicit 3D reasoning.

Table 5: ***Ablation study of the key components in WRAP 4B on ReVSI and VSI-Bench.*** The baseline predicts directly from original video frames. We progressively add Tagged Video, 3D Program Reasoning over grounded variables, and Adaptive 3D Activation to invoke explicit 3D reasoning only when needed.

Tagged Video	*3D Program Reasoning*	*Adaptive 3D Activation*	*ReVSI*	*VSI-Bench*
			45.5	65.8
✓			50.8	65.3
✓	✓		61.9	66.0
✓	✓	✓	**63.8**	**69.7**

4.5 Why Does Our Method Work?

Our ***WRAP*** involves two critical intermediate processes: (1) grounding queried visual objects to their corresponding 3D entities as grounded variables, and (2) generating executable spatial programs over these variables. To examine how these components shape the final spatial reasoning performance, we follow ReVSI (Zhang et al., 2026a) to construct a 1K-sample test set and further annotate each sample with oracle grounded objects and oracle programs derived from ScanNet (Dai et al., 2017). Based on this test set, we conduct a series of analyses around the following questions:

(1) Can WRAP Accurately Ground Objects to Grounded Variables? To assess the object grounding accuracy of ***WRAP*** during spatial reasoning, we extract the grounded variables (*i.e.,* `OBJECT_X`) generated during reasoning together with their semantic object names (*e.g.,* TV and Bed), and compare the predicted grounded objects against the oracle grounded objects. As shown in Figure 3 (a), ***WRAP*** achieves an average grounding accuracy of 86.7%, with most samples concentrated around 90%. To further isolate the effect of grounding quality, we keep the generated programs fixed and

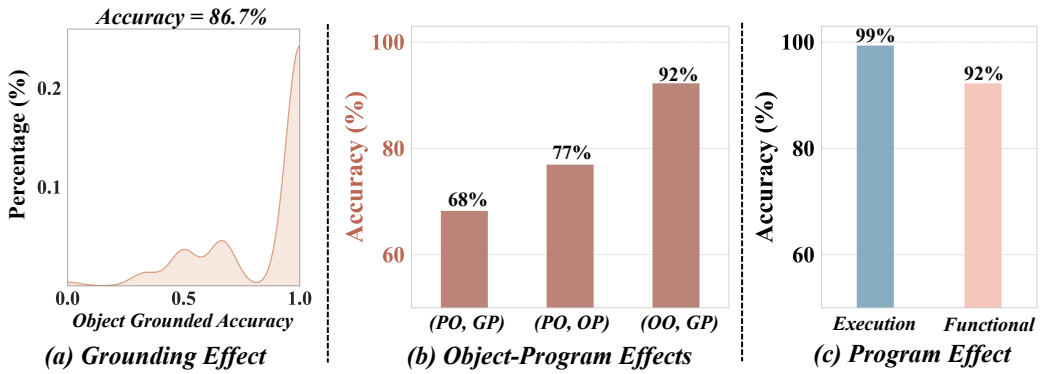


Figure 3: ***Mechanistic analysis of Object Grounding and Program Reasoning in WRAP 4B.*** The "***PO***" denotes the predicted grounded object, the "***GP***" denotes the generated program, the "***OP***" denotes the oracle program, and the "***OO***" denotes the oracle grounded object.

replace the predicted grounded objects with oracle grounded objects, *i.e.,* $(PO, GP) \rightarrow (OO, GP)$. As shown in Figure 3 (b), this substitution improves spatial reasoning accuracy from 68% to 92%. These results demonstrate that ***reliable visual-geometric grounding is a key prerequisite for accurate spatial reasoning and provides a strong foundation for subsequent 3D program reasoning.***

(2) Can WRAP Generate Correct 3D Reasoning Programs? To assess the quality of 3D reasoning programs generated by ***WRAP***, we evaluate them from two complementary perspectives: ***executability*** and ***functional correctness***. Executability measures whether a program is syntactically and operationally complete enough to run successfully, which we quantify as the fraction of programs that terminate without execution errors. The functional correctness further requires it to produce the same output as the oracle program given the same oracle grounded objects. As shown in Figure 3 (c), the generated program achieve a 99% execution success rate and 92% functional correctness. Moreover, replaciing the generated programs with oracle programs while keeping the predicted grounded objects fixed improves spatial reasoning accuracy from 68% to 77%. These results show that ***WRAP*** generates ***highly executable and functionally reliable 3D reasoning programs***, while further improving program generation can still yield ***stronger spatial reasoning performance.***

(3) How Do Grounding and 3D Program Reasoning Jointly Affect Spatial Reasoning Performance? To examine the joint effect of object grounding and program correctness on final spatial reasoning performance, we categorize each sample into four combinations based on whether object grounding is correct or erroneous (**OGC/OGE**) and whether the generated program is correct or erroneous (**PC/PE**). As shown in Figure 4, the bubble size denotes the number of samples, while color intensity indicates answer accuracy. The results show that high spatial reasoning accuracy is achieved primarily when ***both object grounding and program generation are correct***.

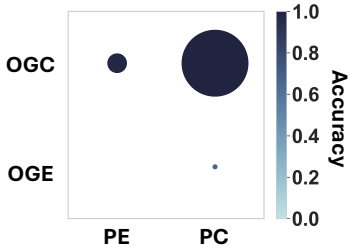


Figure 4: Analysis of grounding and program correctness.

5 Conclusion

In this paper, we propose ***WoRld As a Program (WRAP)***, a novel framework that transforms reconstructed 3D scenes into addressable and programmable reasoning spaces for MLLMs, enabling more reliable spatial reasoning. Extensive experiments on video spatial reasoning and 3D scene editing benchmarks demonstrate the effectiveness and generality of ***WRAP***. Our analyses lead to three main findings: (1) Accurate visual-geometric grounding provides a strong foundation for subsequent 3D program reasoning. (2) Higher-quality 3D reasoning program are consistently associated with stronger downstream spatial reasoning performance. (3) Final spatial reasoning performance is closely linked to the joint quality of visual-geometric grounding and 3D program reasoning. In future work, we aim to extend ***WRAP*** toward richer 4D world representations.

AI USE STATEMENT

Generative AI tools were used in a limited supporting role during the preparation of this work. For mathematical content, their use was restricted to reviewing author-developed expressions and proofs for presentation issues, including notation consistency and potential typographical errors. They were not used to derive mathematical results, generate proof strategies, or construct proofs.

We also used generative AI tools for language polishing, assistance with software development and code editing, and organization of experimental records. All AI-assisted outputs were subsequently examined by the authors. Mathematical suggestions were independently verified, generated or modified code was manually reviewed and tested, and edits to the manuscript and experimental records were cross-checked against the original materials and experiment logs.

No other disclosure-relevant uses of generative AI apply to this work. The authors remain fully responsible for the final manuscript and all associated claims, proofs, code, experimental records, and other AI-assisted artifacts.

REFERENCES

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

Zhongang Cai, Ruisi Wang, Chenyang Gu, Fanyi Pu, Junxiang Xu, Yubo Wang, Wanqi Yin, Zhitao Yang, Chen Wei, Tongxi Zhou, et al. Scaling spatial intelligence with multimodal foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7879–7890, 2026.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. In *International conference on learning representations*, volume 2026, pp. 138846–138923, 2026.

Jieneng Chen, Wenxin Ma, Ruisheng Yuan, Yunzhi Zhang, Jiajun Wu, and Alan Yuille. Thinking with spatial code for physical-world video reasoning. *arXiv preprint arXiv:2603.05591*, 2026a.

Yiming Chen, Zekun Qi, Wenyao Zhang, Xin Jin, Li Zhang, and Peidong Liu. Reasoning in space via grounding in the world. In *International Conference on Learning Representations*, volume 2026, pp. 65549–65570, 2026b.

Zhangquan Chen, Manyuan Zhang, Xinlei Yu, Xufang Luo, Mingze Sun, Zihao Pan, Xiang An, Yan Feng, Peng Pei, Xunliang Cai, et al. Think with 3d: Geometric imagination grounded spatial reasoning from limited views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2613–2624, 2026c.

Seokju Cho, Ryo Hachiuma, Abhishek Badki, Hang Su, Byung-Kwan Lee, Chan Hee Song, Sifei Liu, Subhashree Radhakrishnan, Seungryong Kim, Yu-Chiang Frank Wang, et al. Spatialclaw: Rethinking action interface for agentic spatial reasoning. *arXiv preprint arXiv:2606.13673*, 2026.

Kelly Cui, Nikhil Prakash, Ayush Raina, David Bau, Antonio Torralba, and Tamar Rott Shaham. The dual mechanisms of spatial reasoning in vision–language models. In *The First Workshop on Efficient Spatial Reasoning*, 2026.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

Yalun Dai, Hao Li, Shulin Tian, Runmao Yao, Yuhao Dong, Fangzhou Hong, Zhaoxi Chen, Fangfu Liu, Baoliang Tian, Dingwen Zhang, et al. S-agent: Spatial tool-use elicits reasoning for spatial intelligence. *arXiv preprint arXiv:2606.20515*, 2026.

Zhiwen Fan, Jian Zhang, Renjie Li, Junge Zhang, Runjin Chen, Hezhen Hu, Kevin Wang, Peihao Wang, Huaizhi Qu, Shijie Zhou, et al. Vlm-3r: Vision-language models augmented with instruction-aligned 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 31054–31065, 2026.

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*, 38:99114–99137, 2026.

Xiangjun Gao, Zhensong Zhang, Dave Zhenyu Chen, Songcen Xu, Long Quan, Eduardo Pérez-Pellitero, and Youngkyoon Jang. Map2thought: Explicit 3d spatial reasoning via metric cognitive maps. *arXiv preprint arXiv:2601.11442*, 2026.

Ellis L Brown II, Jihan Yang, Shusheng Yang, Rob Fergus, and Saining Xie. Benchmark designers should "train on the test set" to expose exploitable non-visual shortcuts. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=ggtdDv1l9s.

Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European conference on computer vision*, pp. 71–91. Springer, 2024.

Haoyuan Li, Qihang Cao, Tao Tang, Kun Xiang, Zihan Guo, Jianhua Han, Jia-Wang Bian, Hang Xu, and Xiaodan Liang. Thinking with geometry: Active geometry integration for spatial reasoning. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=2mEC3bPBlS.

Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *CVPR*, 2024.

Zhanpeng Luo, Ce Zhang, Silong Yong, Cunxi Dai, Qianwei Wang, Haoxi Ran, Guanya Shi, Katia P. Sycara, and Yaqi Xie. pyspatial: Generating 3d visual programs for zero-shot spatial reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=yv15C8ql24.

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

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.

Kevin Qu, Haozhe Qi, Mihai Dusmanu, Mahdi Rad, Rui Wang, and Marc Pollefeys. Loc3r-vlm: Language-based localization and 3d reasoning with vision-language models. *arXiv preprint arXiv:2603.18002*, 2026.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model, April 2026a. URL https://qwen.ai/blog?id=qwen3.6-27b.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026b. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

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

Barbara Tversky. Cognitive maps, cognitive collages, and spatial mental models. In *European conference on spatial information theory*, pp. 14–24. Springer, 1993.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5294–5306. IEEE, 2025.

Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schönberger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. Vggt-ω. *arXiv preprint arXiv:2605.15195*, 2026.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20697–20709. IEEE, 2024.

World Labs. Marble: A multimodal world model. `https://www.worldlabs.ai/blog/marble-world-model`, 2026. Accessed: 2026-09-17.

An Yan, Zhengyuan Yang, Junda Wu, Wanrong Zhu, Jianwei Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Julian McAuley, Jianfeng Gao, and Lijuan Wang. List items one by one: A new data source and learning paradigm for multimodal LLMs. In *First Conference on Language Modeling*, 2024. URL `https://openreview.net/forum?id=UfWwBaLuXV`.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*, 2023.

Jihan Yang, Shusheng Yang, Anjali W Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10632–10643. IEEE, 2025.

Rui Yang, Ziyu Zhu, Yanwei Li, Jingjia Huang, Shen Yan, Siyuan Zhou, Zhe Liu, Xiangtai Li, Shuangye Li, Wenqian Wang, et al. Visual spatial tuning. In *European Conference on Computer Vision*, pp. 192–211. Springer, 2026a.

Shusheng Yang, Jihan Yang, Pinzhi Huang, Ellis Brown, Zihao Yang, Yue Yu, Shengbang Tong, Zihan Zheng, Yifan Xu, Muhan Wang, et al. Cambrian-s: Towards spatial supersensing in video. In *International Conference on Learning Representations*, volume 2026, pp. 78185–78225, 2026b.

Jiangye Yuan, Gowri Kumar, and Baoyuan Wang. Boosting mllm spatial reasoning with geometrically referenced 3d scene representations. *arXiv preprint arXiv:2603.08592*, 2026.

Yiming Zhang, Jiacheng Chen, Jiaqi Tan, Yongsen Mao, Wenhu Chen, and Angel X Chang. Revsi: Rebuilding visual spatial intelligence evaluation for accurate assessment of vlm 3d reasoning. In *Forty-third International Conference on Machine Learning*, 2026a.

Zaibin Zhang, Yuhan Wu, Lianjie Jia, Yifan Wang, Zhongbo Zhang, Yijiang Li, Binghao Ran, Fuxi Zhang, Zhuohan Sun, Yizhuang Peng, et al. Think3d: Thinking with space for spatial reasoning. *arXiv preprint arXiv:2601.13029*, 2026b.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. Swift:a scalable lightweight infrastructure for fine-tuning, 2024. URL `https://arxiv.org/abs/2408.05517`.

Duo Zheng, Yanyang Li, Liwei Wang, et al. Learning from videos for 3d world: Enhancing mllms with 3d vision geometry priors. *Advances in neural information processing systems*, 38:20560–20586, 2025a.

Kaizhi Zheng, Xiaotong Chen, Xuehai He, Jing Gu, Linjie Li, Zhengyuan Yang, Kevin Lin, Jianfeng Wang, Lijuan Wang, and Xin Wang. Editroom: Llm-parameterized graph diffusion for composable 3d room layout editing. In *International Conference on Learning Representations*, volume 2025, pp. 86791–86808, 2025b.

A APPENDIX

This appendix presents a detailed description of the implementation details, along with additional results from comprehensive experiments and case analyses. The structure of the appendix is summarized as follows.

➤ Appendix A.1 provides the implementation details of ***WRAP***, including the construction details of training data (Appendix A.1.1) and training details (Appendix A.1.2).

➤ Appendix A.2 provides additional experimental results, including the impact of 3D reconstruction quality A.2.1, the results on VSI-Bench-Debiased (Appendix A.2.2), the effect of Stage 2 training data (Appendix A.2.3), and qualitative analysis of the proposed ***WRAP*** (Appendix A.2.4) on video spatial reasoning task and 3D scene editing task.

➤ Appendix A.3 presents and discusses the limitations of our ***WRAP***, providing directions for future research.

A.1 IMPLEMENTATION DETAILS

A.1.1 TRAINING DATA CONSTRUCTION

Our training framework for ***WRAP*** comprises two stages: **(1) Spatial Grounding** and **(2) Spatial Program Learning**.

Stage 1: Spatial Grounding. The first stage equips the MLLM with robust Set-of-Mark (SoM) grounding capabilities and basic spatial perception and reasoning abilities. We adopt the training data introduced by SoM-LLaVA (Yan et al., 2024), including 10K SoM-based image-captioning samples, 20K question-answering samples with SoM-annotated images, and a subset of 290K OCR samples from LLaVA-665K (Liu et al., 2024). Since these data are primarily image-based, we additionally construct 11K video samples from ScanNet (Dai et al., 2017) to extend SoM grounding from static images to temporally continuous visual inputs. Specifically, object instances are annotated using the provided ScanNet metadata, while Qwen3.6-35B-A3B (Qwen Team, 2026b) is employed to generate object-level descriptions. To further establish fundamental video-based spatial perception and reasoning capabilities, we incorporate 374K video samples from VSI-590K (Yang et al., 2026b), retaining only its video subset for training.

Stage 2: Spatial Program Learning. Stage 2 trains the MLLM to generate executable spatial reasoning programs over grounded 3D entities. We first convert ScanNet into a unified scene representation and sample RGB frames to construct video clips. Using the parsed scene metadata, we automatically generate multiple-choice and numerical spatial questions while retaining object-level geometric annotations. We then prompt an advanced LLM (*e.g.,* GPT-6) to generate task-specific programs, execute them with the corresponding scene metadata, and retain only those that produce the ground-truth answers as *seed programs*. To improve diversity, we generate structurally varied program variants and preserve only functionally consistent ones through execution-based verification. Finally, we construct complete training trajectories following the ***WRAP*** inference pipeline, covering object grounding, grounded-variable binding, spatial program generation and execution, and final answer prediction. Note that we explicitly exclude all physical scenes appearing in VSI-Bench, ReVSI, and VSI-Bench-Debiased from the construction of the Stage-2 training set.

Data Filtering. We apply several quality-control procedures to curate the generated spatial reasoning data. To improve scene diversity, we impose an upper bound on the number of QA pairs generated from each video. For multiple-choice questions, the correct options are randomly permuted across answer positions to reduce positional bias and maintain a balanced answer distribution. For numerical questions, we regulate the target-value distribution to avoid severe imbalance and implausible numerical ranges. After filtering and execution-based verification, we retain 27K high-quality QA pairs for *Spatial Program Learning*.

A.1.2 TRAINING DETAILS

Training details for the two-stage learning framework are summarized in Table 6. We adopt ms-swift (Zhao et al., 2024) as the training framework. Both stages are trained for one epoch with a

Table 6: ***Hyper-parameters for training*** both Stage 1 and Stage 2.

Hyper-Parameters	***Stage 1***	***Stage 2***
Epochs	1	1
Learning Rate	$1e-5$	$1e-5$
Train Batch Size	16	16
Grad Accumulation	2	2

learning rate of 1×10^{-5}. We use a training batch size of 16 and a gradient accumulation step of 2 for both Stage 1 and Stage 2, keeping the optimization settings consistent across the two stages.

A.2 More Experiment Results

A.2.1 Impact of 3D Reconstruction Quality on WRAP

To isolate the effect of 3D reconstruction quality on ***WRAP***, we vary the number of frames provided to DepthAnything 3 for scene reconstruction while keeping the MLLM input fixed at 64 video frames across all settings. This design ensures that performance differences primarily reflect changes in the reconstructed 3D representation rather than variations in visual input to the MLLM. As shown in Figure 5, the accuracy consistently increases with the number of reconstruction frames across ***WRAP*** 2B, 4B, and 9B. The improvement is particularly pronounced from 8 to 32 frames, with further gains at 64 frames. These results demonstrate that higher-quality 3D reconstruction provides a more reliable geometric basis for ***WRAP***, leading to improved spatial reasoning performance.

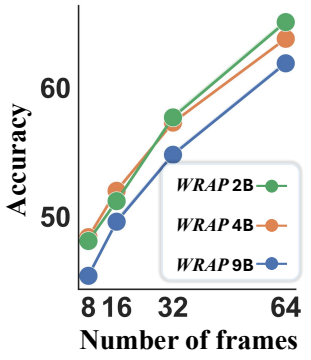


Figure 5: Effect of the number of frames used for 3D reconstruction on ReVSI.

A.2.2 Results on VSI-Bench-Debiased

Table 7 compares ***WRAP*** with general-purpose and spatially specialized models. ***WRAP*** 9B achieves an average score of **63.1**, outperforming Gemini-2.5 Pro, Cambrian-S-7B (Yang et al., 2026b), and VG-LLM 8B by 14.0%, 3.2%, and 7.9% points, respectively. It also achieves the best performance on *Absolute Distance* (**67.1%**) and *Relative Direction* (**84.8%**), demonstrating strong metric and directional reasoning capability.

SpatialClaw (Cho et al., 2026) decomposes object perception into a sequence of interactions, where the MLLM first identifies task-relevant objects, external modules such as SAM3 (Carion et al., 2026) then localize or segment them, and the resulting observations are subsequently fed back to the MLLM for further reasoning. Although each component is individually capable, such repeated transitions between semantic recognition, external perception, and model verification introduce additional opportunities for object mismatches and interface errors. In particular, identifying task-relevant objects is already a strong capability of modern MLLMs, making this multi-step perception loop potentially unnecessary. In contrast, ***WRAP*** exposes persistent visual-geometric entities as *Grounded Variables*, allowing the MLLM to directly identify and reason over the corresponding objects without repeatedly reconstructing their identity through external perception tools. Consequently, ***WRAP*** 9B achieves **63.1%**, outperforming Qwen3.6-27B (Qwen Team, 2026a) with SpatialClaw by **5.2%** despite using a substantially smaller backbone.

A.2.3 Effect of Stage 2 Training Data

To study the effect of Stage 2 data scale, we vary the sampling ratio of the 27K Spatial Learning dataset from 20% to 100% and evaluate ***WRAP*** on VSI-Bench, ReVSI, and VSI-Bench-Debiased. As shown in Figure 6, the performance improves with more training data. Larger models, especially ***WRAP*** 9B, benefit more consistently from increased supervision.

Table 7: ***Comparison of WRAP with existing methods on VSI-Bench-Debiased.*** The best results are highlighted in bold.

Method	*Numerical Question*				*Multiple-Choice Question*				*Avg.*
	Obj. Cnt.	*Abs. Dist.*	*Obj. Size*	*Room Size*	*Rel. Dist.*	*Rel. Dir.*	*Route Plan*	*Appr. Order*	
General-purpose Models									
LLaVA-Video-7B	38.6	15.4	34.7	22.1	42.6	37.9	20.2	34.0	30.7
LLaVA-Video-72B	37.3	22.3	45.8	33.7	43.9	37.8	25.4	48.1	36.8
LLaVA-OneVision-7B	37.1	16.3	31.4	18.0	39.7	34.2	23.7	27.7	28.5
LLaVA-OneVision-72B	32.0	21.8	44.3	40.8	39.0	37.3	21.9	47.8	35.6
Gemini-2.5 Pro	40.8	34.4	56.7	40.9	61.0	46.8	43.0	69.5	49.1
Spatially Specialized Models									
Qwen3.6-27B w/ SpatialClaw	-	-	-	-	-	-	-	-	57.9
Cambrian-S-7B	-	-	-	-	-	-	-	-	59.9
VG-LLM 8B	-	-	-	-	-	-	-	-	55.2
GeoThinker 3B	41.4	26.9	42.2	52.5	50.6	38.5	21.1	73.3	43.3
GeoThinker 7B	**60.4**	53.3	60.7	73.9	68.7	79.2	40.4	62.6	62.4
GeoThinker 8B	59.8	59.7	**64.6**	**75.6**	**71.0**	81.2	**48.2**	**72.3**	**66.6**
WRAP 2B	53.1	64.3	57.2	65.3	63.2	84.8	34.2	76.4	62.3
WRAP 4B	59.5	66.8	60.2	67.9	67.4	82.4	33.3	71.1	63.6
WRAP 9B	55.8	**67.1**	59.9	66.5	65.2	**84.8**	35.1	70.4	63.1

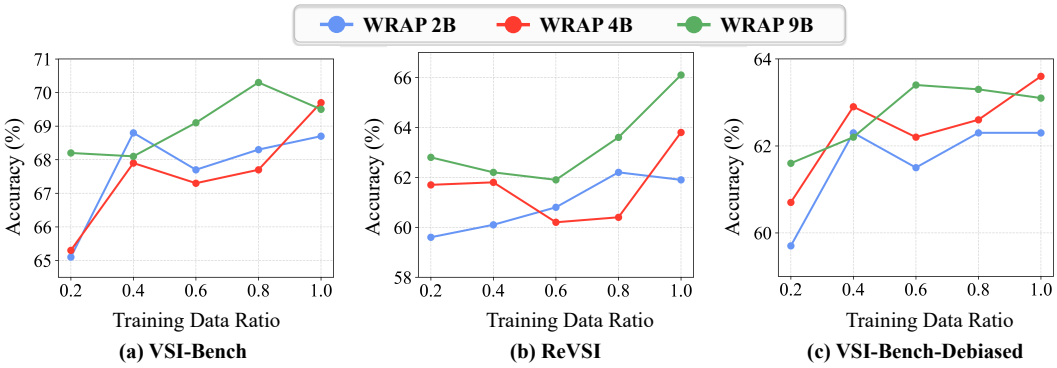


Figure 6: Effect of Stage 2 training data scale on video spatial reasoning performance. We vary the sampling ratio of the 27K Spatial Program Learning training data and evaluate ***WRAP***.

A.2.4 Case Study

Qualitative Example of Video Spatial Reasoning Task. Figure 7 demonstrates the 3D Program Reasoning process of ***WRAP*** for video spatial reasoning. Given a metric-distance query, the MLLM first grounds the queried objects to their corresponding grounded variables and generates an executable program that computes the closest-point distance between their point clouds. The external executor performs the deterministic geometric computation and returns the numerical result, which is subsequently interpreted by the MLLM to produce the final answer.

Qualitative Examples of 3D Scene Editing Task. Figure 8 demonstrates the effectiveness of ***WRAP*** on 3D scene editing tasks. Given natural-language editing commands, ***WRAP*** more accurately identifies the target object and performs the requested spatial modification while preserving the surrounding scene structure. Compared with EditRoom (Zheng et al., 2025b), which exhibits noticeable localization and editing errors, ***WRAP*** produces results that are substantially closer to the ground truth, achieving higher IoU.

A.3 Limitation Discussion

Admittedly, the proposed ***WRAP*** has limitations, despite its promising performance on video spatial reasoning benchmarks and EditRoom-DB test set. In particular, the current version of ***WRAP*** is trained solely with supervised fine-tuning. As a result, it performs reliably on task types observed during training, but exhibits limited generalization to previously unseen, more complex tasks. In future work, we plan to incorporate RLVR to first acquire fundamental 3D program reasoning skills (*e.g.,* distance measurement and directional reasoning), and then learn to flexibly compose these primitive skills to solve novel spatial tasks.



Question:

Measuring from the closest point of each object, what is the direct distance between the coffee table and the tv (in meters)?

Turn 1:

I first ground the natural-language object names and captions to the numbered marks. The coffee table described as the coffee table is positioned in front of a brown sofa, with a white cabinet to its left and a rug beneath it, corresponds to OBJECT_19. The tv described as the tv is mounted on the wall above a white cabinet, with a wooden staircase to its right, corresponds to OBJECT_16. I therefore bind target = OBJECT_19 and anchor = OBJECT_16. I will calculate their closest-point distance from the two point clouds. The executable code is inside the tool_call block.

```python
import json
import numpy as np
from scipy.spatial import cKDTree
target = OBJECT_19
anchor = OBJECT_16
def is_invalid_points(points):
    points = np.asarray(points, dtype=np.float32)
    return points.shape == (1, 3) and np.isinf(points).all()
def validate_points(points):
    points = np.asarray(points, dtype=np.float32)
    if is_invalid_points(points):
        return points
    if points.ndim != 2 or points.shape[1] != 3:
        raise ValueError(f"Expected object point data with shape [N, 3], but got {points.shape}")
    if not len(points):
        raise ValueError("Object point data must not be empty.")
    if not np.isfinite(points).all():
        raise ValueError("Object point data contains NaN or Inf values.")
    return points
def object_distance(points1, points2):
    points1 = validate_points(points1)
    points2 = validate_points(points2)
    nearest_distances, _ = cKDTree(points2).query(points1, k=1)
    return float(nearest_distances.min())
if is_invalid_points(target) or is_invalid_points(anchor):
    result = {
        "result": "Invalid spatial object data. Please solve the problem by direct reasoning."
    }
else:
    result = {"result": round(object_distance(target, anchor), 4)}

print(json.dumps(result, ensure_ascii=False))
```

MLLM

Turn 2:

Execution result:

```json
{"result": 2.9423}
```

Now analyze the execution result and answer the task.

Executor

Turn 3:

The execution result gives the closest-point distance between the coffee table and the tv as 2.9423 meters. So the answer is 2.9423.

✅

MLLM

Figure 7: ***Qualitative example of Video Spatial Reasoning task.*** We conduct experiments on ReVSI using ***WRAP*** 9B with 3D Program Reasoning.

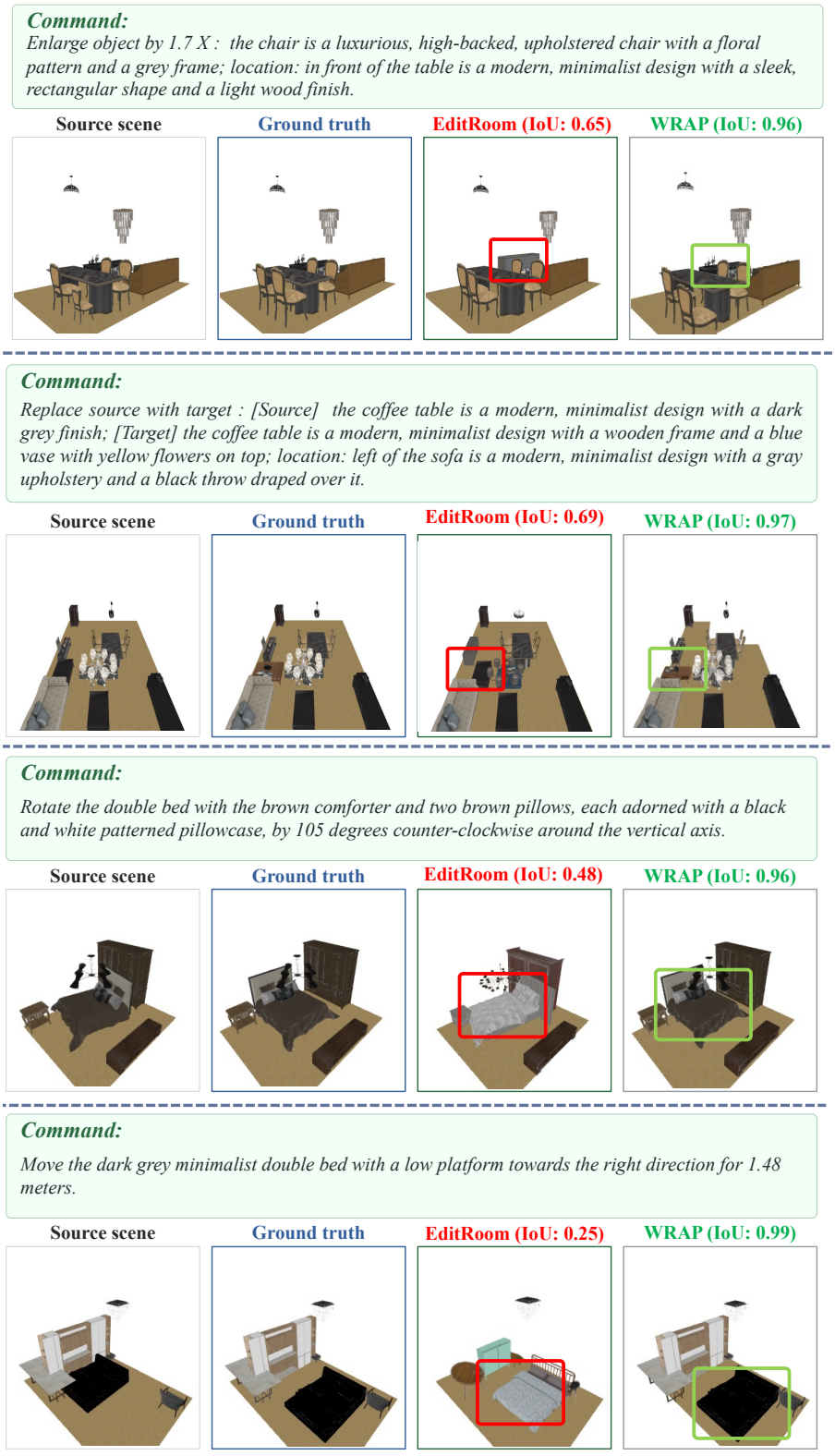


Figure 8: Qualitative example of 3D Scene Editing task. We conduct experiments on EditRoom-DB test set with EditRoom and our ***WRAP***.