SEE ONCE, ASK MANY: OBJECT-ANCHORED NEURAL MEMORY FOR MULTI-VIEW SPATIAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-view spatial reasoning requires consistent object identity and reference-frame reasoning across partial observations, which remain difficult for vision–language models (VLMs). Existing methods often use external 3D models to provide spatial features, which VLMs then combine with visual semantics to directly predict spatial answers. We introduce **See Once, Ask Many (SOAM)**, which learns object-addressable neural scene memory within the VLM and uses it to compose question-specific programs for explicit geometric execution. Given sparse views, a single VLM performs neural scene compilation: it organizes cross-view observations into anonymous objects and binds their visual representations into a question-independent neural memory. Object identities provide shared addresses for semantic grounding, view-local localization, and geometric computation. The same VLM then interprets each question through this memory and composes a typed program, while a frozen geometric backend supplies measurements for execution. The resulting scene representation supports subsequent questions. Joint state–program supervision and reuse-aware policy optimization train the shared memory for observation fidelity and utility across questions, with execution feedback assigned to each program individually. SOAM-8B outperforms the strongest baseline on VIEW2SPACE-v1 by 6.96 points in grounding mIoU and 5.28 percentage points in MCQ accuracy, while achieving a macro score of 72.85 on SPAR-Bench. Ablations show higher task scores with object-anchored memory and full training than with their respective controls.

1 INTRODUCTION

Spatial reasoning from sparse images requires integrating partial observations into a coherent understanding of one physical scene (Yeh et al., 2026; Ke et al., 2026). A question may identify its target in one view, disambiguate a reference in another, and request their relation from a third viewpoint. Such questions couple two challenges: forming a consistent scene understanding that preserves object identity and visual semantics across views, and composing question-specific spatial computations in the requested reference frame. Existing evaluations reveal persistent failures in correspondence and spatial alignment, even among strong VLMs (Yeh et al., 2026; Oi et al., 2026).

Spatial supervision and geometric features offer complementary routes to spatial reasoning. SpatialVLM teaches spatial relations, while Spatial-MLLM and SpatialStack introduce geometric representations through dedicated encoders and hierarchical fusion (Chen et al., 2024; Wu et al., 2025; Zhang et al., 2026). These approaches provide VLMs with richer spatial representations. Completing a spatial task also requires connecting this information to the appropriate computation. Appearance and context determine which objects a question refers to, while geometric measurements and reference-frame transformations determine their spatial relations.

Explicit scene representations and tools provide a basis for organizing these computations. Cog3DMap constructs semantic–geometric memory, while pySpatial and Reasmory support spatial reasoning through executable tools and structured memory access (Gwak et al., 2026; Luo et al., 2026; He et al., 2026). Our starting point is a division of responsibilities: the VLM establishes the objects over which reasoning operates and determines what to compute; the geometric pathway supplies measurements and evaluates the corresponding spatial operations. A shared object representation is needed to connect these decisions to the same physical entities across views.

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

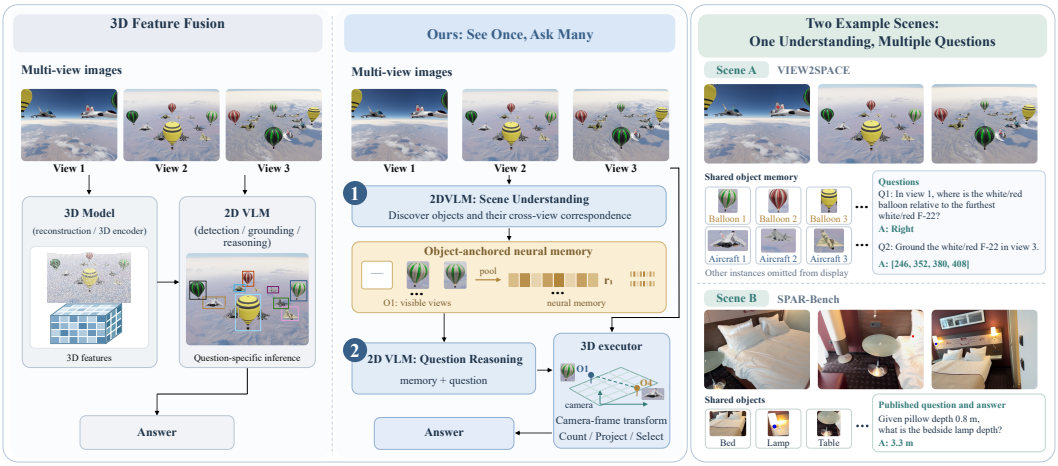

Figure 1: **See once, ask many.** SOAM separates cross-view scene understanding from question-conditioned reasoning. Object-anchored neural memories connect visual semantics with geometric execution and are reused across questions. The example scenes illustrate spatial relations, grounding, and reference-scaled measurement.

We introduce **See Once, Ask Many (SOAM)**, which connects VLM-native neural scene memory with geometric execution (Figure 1). Through neural scene compilation, the VLM organizes observations into anonymous cross-view objects and associates each identity with a continuous memory derived from its own visual hidden states. Object identities provide common addresses that link visual semantics, view-local observations, and geometric measurements. Language can then refer to an object through its appearance and context, while spatial operations access the observations and geometry associated with the same address.

Given a question, the same VLM selects the relevant objects and composes a typed program over the scene representation. A frozen geometry model estimates the required measurements, and a deterministic executor carries out the specified operations. The VLM thus learns scene organization and computational intent, while spatial calculation has explicit numerical semantics (Figure 2). Because the scene representation is formed before individual questions are introduced, it can support different computations without repeating scene understanding.

This division also guides learning. State supervision teaches the model to organize cross-view observations, and program supervision teaches it to use the resulting memory for question-conditioned reasoning. We further optimize their free-running execution through reuse-aware policy optimization. The shared scene receives feedback on observation fidelity and mean utility across sampled questions, while each program receives feedback from its own execution. These signals train the representation to preserve scene content and support the computations performed on it.

Our contributions are threefold:

One neural scene memory, many spatial computations. SOAM organizes cross-view observations once, using common object addresses to connect VLM-native visual memory with question-specific programs. Each question induces a computation over this shared representation, with geometric measurements used during execution.

Joint learning of scene structure and reasoning. We train SOAM-4B and SOAM-8B with joint state–program supervision and reuse-aware execution rewards. Observation fidelity and cross-question utility jointly optimize the shared scene, while question-local feedback trains program composition.

Task performance and representation analysis. SOAM-8B reaches 70.21% MCQ accuracy on VIEW2SPACE-v1 and a 72.85 macro score on SPAR-Bench. Memory and training analyses examine semantic binding and object preservation, including 84.10% recall of objects outside the queried

subset. An analytical cost model further characterizes the benefit of reusing the scene representation.

2 Related Work

2.1 Bringing 3D Capabilities to 2D VLMs

Spatial supervision and spatial tokens strengthen the spatial representations learned by 2D VLMs (Chen et al., 2024; Lou et al., 2025). 3D-LLM connects language models to 3D scene features, while Chat-Scene introduces object identifiers for accessing object-centric representations (Hong et al., 2023; Huang et al., 2024). Feed-forward reconstruction models, including DUSt3R and VGGT, make geometric structure available directly from images (Wang et al., 2024; 2025). These capabilities support feature fusion in Spatial-MLLM and joint reconstruction–reasoning in G^{2}VLM (Wu et al., 2025; Hu et al., 2025).

Recent work further studies where spatial information should enter a VLM. SpatialStack uses hierarchical geometry–language fusion, Qwen-3D constructs world-aligned tokens, and SpatioLM elicits spatial features within the VLM (Zhang et al., 2026; Lin et al., 2026; Wu et al., 2026). These methods primarily enrich the representations used by VLMs for spatial reasoning. SOAM organizes native VLM features into object-addressable memory and uses geometric measurements during execution. Its design separates semantic interpretation from numerical spatial calculation while linking both to the same objects.

2.2 Multi-View Reasoning

Multi-view reasoning requires consistent correspondence and reference-frame interpretation across partial observations. Correspondence prompts and viewpoint-aware training improve cross-view alignment (Liu et al., 2025; Jeong et al., 2026; Oi et al., 2026). VIEW2SPACE further connects intermediate reasoning to view-local boxes (Ke et al., 2026). These approaches strengthen the association between visual observations and the entities involved in a question.

Explicit representations provide another route to structured reasoning. Building on object-based neuro-symbolic learning (Mao et al., 2019), SpatialReasoner and Cog3DMap make spatial representations explicit, while MomaGraph learns instruction-conditioned scene graphs for planning (Ma et al., 2025; Gwak et al., 2026; Ju et al., 2026). pySpatial exposes spatial tools, Reasmory validates programs over reconstructed memory, and RenderMem retrieves query-conditioned renderings (Luo et al., 2026; He et al., 2026; Park & Kang, 2026). Their representations differ in how objects are established and accessed: Reasmory uses queried-category proposals, while RenderMem retrieves rendered observations. SOAM studies the joint learning of anonymous cross-view identities, VLM-derived neural memory, and the programs that operate on this shared representation.

3 Method

SOAM connects learned scene organization with explicit spatial computation (Figure 2). Section 3.1 describes how object states, neural memory, and geometric execution form this architecture. Section 3.2 presents the supervision and execution feedback used to train the shared model.

3.1 Neural Scene Memory and Geometric Execution

Given sparse RGB views $I = (I_1, \ldots, I_V)$ of a static scene, we first construct a question-independent representation. We instantiate SOAM with Qwen3-VL backbones at the 4B and 8B scales (Bai et al., 2025). Within each instantiation, scene encoding and subsequent program generation share the same VLM parameters θ.

Cross-view scene encoding. Conditioned on the images and a fixed scene prompt, the VLM autoregressively generates

$$S = \{o_k\}_{k=1}^{N}, \qquad o_k = \{(v, b_{kv}) : v \in \mathcal{V}_k\}, \tag{1}$$

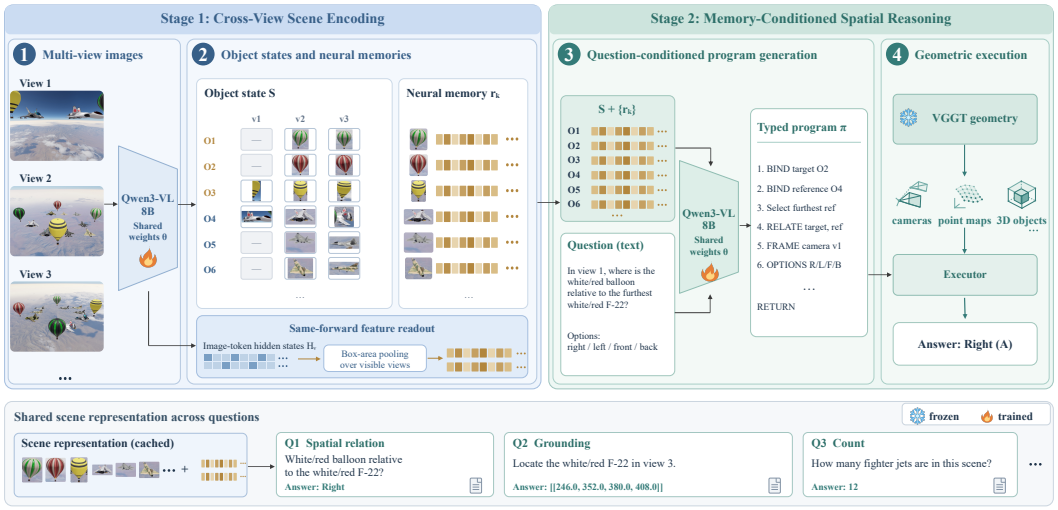

Figure 2: **SOAM architecture.** Stage 1, Cross-View Scene Encoding, constructs anonymous object states and pools object memories from the same forward pass. Stage 2, Memory-Conditioned Spatial Reasoning, uses the same VLM to compose a typed program from the cached representation and a question. Frozen VGGT supplies cameras, point maps, and object geometry for execution. Subsequent questions reuse the completed scene representation.

where $\mathcal{V}_k$ is the set of observed views for object k, and b_{kv} is its predicted box in view $v \in \mathcal{V}_k$. Each object segment, delimited by `OBJECT` and `END_OBJECT`, defines one anonymous object: its grouped view–box records express cross-view identity, and its segment index supplies the object address. Coordinates use categorical actions in $\{0, \ldots, 1000\}$ (Chen et al., 2022); a fixed grammar constrains state structure and coordinate order.

Object-anchored neural memory. Let $H_v(u) \in \mathbb{R}^d$ denote the scene-encoding pass's final-layer hidden state at visual-grid cell u, with spatial support C_{vu}. For a normalized predicted box $\bar{b}_{kv}$, we compute

$$w_{kvu} = \frac{\mathrm{Area}(\bar{b}_{kv} \cap C_{vu})}{\sum_{u'} \mathrm{Area}(\bar{b}_{kv} \cap C_{vu'})}, \qquad r_k = \frac{1}{|\mathcal{V}_k|} \sum_{v \in \mathcal{V}_k} \sum_{u} w_{kvu} H_v(u). \tag{2}$$

Within each observed view, the predicted box determines the spatial support of the readout. Cross-view identity specifies which observations are averaged into the same vector. The resulting r_k complements the explicit view–box records in S with continuous visual information. Because the pooled hidden states are contextualized, this memory can also retain cues about other objects and the surrounding background. The parameter-free readout uses hidden states from the scene-encoding pass, and joint training shapes both their content and their use in reasoning.

Memory-conditioned reasoning. Let $s_{1:T}$ be the serialized action sequence of S, $E_A : \mathcal{A} \to \mathbb{R}^d$ the action embedding map, and t_k the position of object k's `END_OBJECT` action. We attach its memory r_k at this position:

$$m_t = E_A(s_t) + \sum_{k=1}^{N} \mathbf{1}[t = t_k] r_k, \qquad M_\theta(I, S) = (m_1, \ldots, m_T). \tag{3}$$

The complete state sequence provides object structure, while the added memories provide continuous visual context.

For question Q_j, let $\widetilde{Q}_j(S)$ collect its text, options, and query-derived references expressed in the scene interface. The same VLM reads $M_\theta(I, S)$ and $\widetilde{Q}_j(S)$ to generate a typed program π_j. The `BIND`/`MEMBERS` actions associate question roles and phrases with object addresses; subsequent

operators specify the spatial computation. For m questions, the joint distribution factorizes as

$$p_\theta(S, \pi_{1:m} \mid I, Q_{1:m}) = p_\theta(S \mid I) \prod_{j=1}^{m} p_\theta\Big(\pi_j \mid M_\theta(I, S), \widetilde{Q}_j(S)\Big). \tag{4}$$

Scene encoding and program generation share the backbone parameters θ and the action table. Each question reads the same completed scene representation and produces its own program.

Geometric execution. Frozen VGGT estimates cameras and point maps (Wang et al., 2025). Box-guided, confidence-weighted pooling associates these measurements with the generated objects, forming a geometric cache $\mathcal{G}(I, S)$. The executor applies the program to the object state and this cache to produce $\hat{y}_j = \mathrm{Exec}(\pi_j, S, \mathcal{G})$. Set operations use object identities, grounding reads stored view-local boxes, and spatial operations evaluate measurements in the requested reference frame. This connects semantic selections made by the VLM to explicit spatial calculations. The state, neural memory, and geometry are retained for subsequent questions.

3.2 Learning for Cross-Question Reuse

We first supervise scene and program generation, then optimize their free-running execution. Upper vision/language LoRA adapters and the shared action table are trainable; the remaining backbone and geometry weights are fixed.

State–program supervision. Training begins with image-conditioned state generation. Joint supervision then teacher-forces the scene target, constructs its neural memory, and predicts the associated question program. For scene i, let d_i be its number of question–program pairs. We denote the summed state and program negative log-likelihoods by ℓ_i^S and ℓ_{ij}^π, and their learned-action counts by T_i^S and T_{ij}^π. The joint objective is

$$\mathcal{L}_{\mathrm{SFT}} = \frac{\sum_{(i,j)\in\mathcal{D}} [d_i^{-1}\ell_i^S + \ell_{ij}^\pi]}{\sum_{(i,j)\in\mathcal{D}} [d_i^{-1}T_i^S + T_{ij}^\pi]}. \tag{5}$$

Weighting the state term by d_i^{-1} gives each scene one direct likelihood contribution across its questions. Program likelihood is accumulated over model-predicted actions. Its gradients also pass through the neural-memory readout, training the visual representation for language grounding. Supervision construction and optimization settings are described in Appendix **??**.

Reuse-aware execution rewards. We optimize the shared scene for observation fidelity and its utility across questions. A training group fixes one view tuple I and two questions Q_1, Q_2. Each rollout samples a scene state S, constructs its neural memory and geometric cache, and generates the two programs according to Equation 4. Executing program j gives $\hat{y}_j$ and task reward $R_j = \mathcal{R}_j(\hat{y}_j, y_j^*) \in [0, 1]$, where y_j^* is the annotated answer. Invalid executions receive zero task reward; Appendix B specifies the task-dependent scoring functions.

Let $S^* = \{o_b^*\}_{b=1}^{N^*}$ be the annotated object state and $N = |S|$. The similarity s_{ab} between predicted object a and target object b is the sum of their box IoUs over shared views, divided by the number of views in the union of their observation sets. Let $\mathcal{A}^*$ be the one-to-one assignment maximizing total similarity. We define observation fidelity and the shared return as

$$C_{\mathrm{obs}}(S, S^*) = \frac{2\sum_{(a,b)\in\mathcal{A}^*} s_{ab}}{N + N^*}, \qquad U = C_{\mathrm{obs}}(S, S^*) + \frac{R_1 + R_2}{2}. \tag{6}$$

Observation fidelity rewards preservation of the annotated scene, including objects outside the sampled questions. Missing and redundant objects lower the score through its denominator. Mean task utility rewards the computations supported by the sampled state. The two terms therefore train the representation for both scene preservation and downstream use.

Factorized policy optimization. We apply REINFORCE with leave-one-out baselines (RLOO; Ahmadian et al. 2024). For each view tuple and question pair, we sample $G = 4$ independent rollouts

$(S_g, \pi_{g1}, \pi_{g2})$. Each rollout generates its own scene state, shared by its two question programs, and receives return U_g and task rewards R_{gj}. The advantages are

$$A_g^S = U_g - \frac{1}{G-1}\sum_{h\neq g} U_h, \qquad A_{gj}^\pi = R_{gj} - \frac{1}{G-1}\sum_{h\neq g} R_{hj}.$$

Writing $p_{\theta,g}^S = p_\theta(S_g \mid I)$ and $p_{\theta,gj}^\pi = p_\theta(\pi_{gj} \mid M_\theta(I, S_g), \widetilde{Q}_j(S_g))$, we minimize

$$\mathcal{L}_{\mathrm{RL}} = -\frac{1}{G}\sum_{g=1}^{G}\left[\mathrm{sg}(A_g^S)\log p_{\theta,g}^S + \frac{1}{2}\sum_{j=1}^{2}\mathrm{sg}(A_{gj}^\pi)\log p_{\theta,gj}^\pi\right]. \tag{7}$$

Here sg stops gradients through the advantages. The state term uses observation fidelity and mean task utility, while each program term uses its own execution reward. The factor $1/2$ follows from the two-question mean in Equation 6. This assignment follows the scene–question dependency structure (Schulman et al., 2015). Both terms update the shared parameters, and program gradients also propagate through the neural-memory pathway.

4 Experiments

Section 4.1 describes the setup, Section 4.2 compares benchmarks, Section 4.3 examines memory, decoding, and grounding, and Section 4.4 studies training objectives and scene reuse.

4.1 Experimental Setup

Benchmarks and metrics. We evaluate on VIEW2SPACE-v1 (Ke et al., 2026) and SPAR-Bench (Zhang et al., 2025). VIEW2SPACE-v1 measures sparse multi-view counting, grounding, and spatial decisions through Count accuracy, Detection mIoU, and MCQ accuracy. SPAR-Bench covers 20 tasks across spatial perception, cross-view perception, and compositional reasoning. We report its low-, medium-, and high-level scores and the macro average over all tasks. Auxiliary metrics and task-specific scoring details are provided in the appendix.

Training details. Training uses VIEW2SPACE's released training data and SPAR-7M (Ke et al., 2026; Zhang et al., 2025). The VIEW2SPACE portion provides 96,630 training states and 208,802 question–program pairs, with 7,467 states and 22,352 programs for scene-disjoint validation. For SPAR-7M, we set a sampling budget of 200,000 training question–answer pairs across single- and multi-view spatial tasks.

We train on four NVIDIA H20 GPUs with rank-32 LoRA and an effective SFT batch size of 32. AdamW uses learning rates of 2×10^{-5} for joint SFT and 2×10^{-6} for RL. Each RL group contains four rollouts and two questions per state. The memory, grounding, and training analyses use SOAM-8B. Training subsets, supervision construction, and schedules are provided in Appendix **??**.

Task interfaces. Questions use the released text, options, and public visual prompts. Point and box prompts are associated with object addresses, while reference views, numerical values, and units supply the conditions for program execution. Outputs are serialized in the format required by each task's scorer, including object sets, options, numerical values, and structured camera changes. Invalid outputs receive zero credit in their corresponding task scores.

4.2 Main Results

Multi-view reasoning. Table 1 summarizes performance on both benchmarks. SOAM-8B reaches 60.82% Count accuracy, 76.30% Detection mIoU, and 70.21% MCQ accuracy on VIEW2SPACE-v1. The corresponding improvements over VIEW2SPACE are 5.83, 6.96, and 5.28 percentage points. At the 4B scale, SOAM improves these metrics by 4.45, 1.53, and 4.71 points, respectively.

Spatial perception and reasoning. On SPAR-Bench, SOAM-8B achieves a macro score of 72.85, compared with 63.25 for SPAR, 54.87 for G^{2}VLM-SR, and 71.39 for SpatialStack. SOAM-4B reaches 71.85. The level-wise results show improvements across numerical perception, cross-view reasoning, and higher-level spatial questions at both model scales.

Table 1: Performance on VIEW2SPACE-v1 and SPAR-Bench, covering counting, grounding, spatial decisions, and three levels of spatial perception and reasoning.

		VIEW2SPACE-v1			SPAR-Bench			
Method	VLM backbone	Count Acc.	Det. mIoU	MCQ Acc.	Low	Medium	High	Avg.
InternVL2	8B	20.98	0.57	29.57	26.83	36.49	37.37	33.02
InternVL2.5	8B	22.50	0.79	30.93	29.46	31.88	43.80	36.28
LLaVA-OneVision	7B	20.81	0.94	29.86	21.79	26.13	40.14	30.70
Qwen2.5-VL	7B	13.03	2.68	30.29	28.75	22.97	40.27	33.07
Qwen3-VL	8B	24.92	10.05	37.41	43.27	38.94	48.13	44.81
GPT-5	–	38.10	8.18	59.86	48.76	57.23	66.41	57.97
Spatial-MLLM	3B	9.98	0.00	27.93	52.47	44.83	51.26	50.78
OpenView	–	19.80	1.08	30.72	–	–	–	–
MindCube (Plain)	3B	19.12	0.12	30.21	34.67	42.53	44.91	40.46
VIEW2SPACE	4B	54.99	69.34	64.93	46.83	54.16	55.49	51.83
SPAR	8B	47.38	5.26	46.21	65.53	63.01	61.29	63.25
G^2VLM-SR	2B	–	–	–	59.99	36.27	56.51	54.87
SpatialStack	4B	–	–	–	–	–	–	71.39
SOAM-4B	4B	59.44	70.87	69.64	70.28	71.36	73.41	71.85
SOAM-8B	8B	**60.82**	**76.30**	**70.21**	**71.43**	**72.19**	**74.33**	**72.85**

Table 2: Ablations of neural-memory construction, output representation, and execution on VIEW2SPACE-v1 under the same full-training schedule.

Model configuration			VIEW2SPACE-v1		
Neural memory	Output format	Executor	Count Acc.	Det. mIoU	MCQ Acc.
None	Typed	Explicit	41.29	50.83	45.64
Global pool	Typed	Explicit	57.02	68.47	64.71
Object-anchored	Text	Explicit	52.07	62.86	59.21
Object-anchored	Typed	Neural	49.92	58.73	52.79
Object-anchored	Typed	Explicit	**60.82**	**76.30**	**70.21**

4.3 Neural Memory and Geometric Reasoning

Architecture ablations. Table 2 compares neural-memory source, output format, and answer computation under the full training schedule. Object-anchored memory improves MCQ accuracy by 5.50 points over global pooling and also improves counting and grounding. This comparison examines the value of associating visual information with individual object addresses.

For output representation, the textual variant expresses the same state and program targets using the native language vocabulary. Typed actions improve Count accuracy, Detection mIoU, and MCQ accuracy by 8.75, 13.44, and 11.00 points, respectively. Both executor variants receive the same predicted geometry. Explicit execution improves MCQ accuracy by 17.42 points over neural answer computation, supporting the separation between program composition and spatial calculation.

Neural-memory binding. We fix the checkpoint, predicted state, geometry, and question, then reassign neural memories between object addresses. Figure 3(b) reports MEMBERS set-F1 of 81.60 with original memory, 31.40 after reassignment, and 47.80 with zero memory. Reassignment is more disruptive than removing memory, indicating that semantic selection depends on how visual representations are associated with object identities.

Multi-target grounding. Table 3 groups questions by the number of target instances in the requested output view. Alongside mIoU and F1, exact-set success measures whether all targets are recovered with no extra prediction. Full training improves all three metrics in every group. For

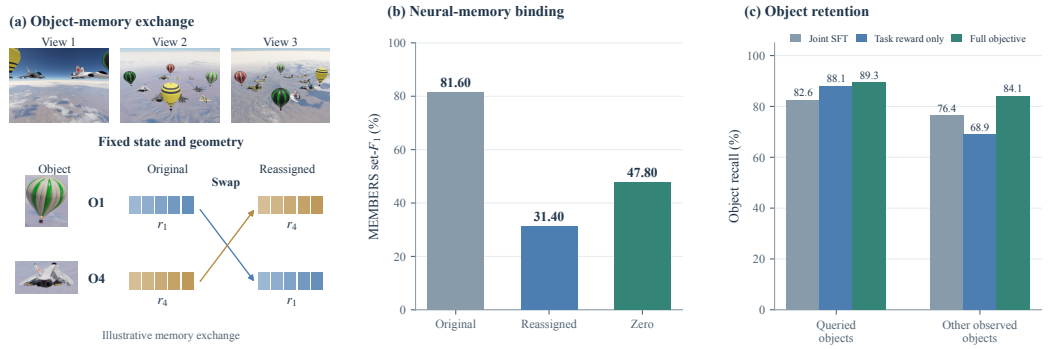

Figure 3: **Neural-memory binding and object retention.** (a) Illustration of exchanging neural memories between two object addresses while retaining the predicted state and geometry. (b) Semantic MEMBERS set-F1 with original, reassigned, and zero memory. (c) Recall of queried and other observed objects under three training objectives.

Table 3: Grounding by target count in the requested view, comparing joint SFT and full training.

	mIoU		F1@0.5		Exact-set success	
Targets	SFT	Full	SFT	Full	SFT	Full
1	73.36	77.31	75.25	79.15	67.81	74.02
2–3	64.87	69.94	66.82	71.37	45.70	54.84
4–8	51.83	58.72	54.67	61.24	30.77	38.46

Table 4: Effect of training objectives on scene fidelity and spatial task performance, starting from the same joint-SFT checkpoint.

Objective	C_{obs}	Count	mIoU	MCQ
Joint SFT	0.744	55.99	72.20	65.42
Continued SFT	0.753	56.58	69.49	65.97
One-question state credit	0.774	58.61	70.27	67.29
Task reward only	0.712	58.94	71.46	68.00
Broadcast joint return	0.796	59.45	71.11	68.36
Full objective	0.807	60.82	76.30	70.21

four to eight targets, exact-set success increases from 30.77% to 38.46%, although performance remains substantially below the single-target setting. The decline in both localization and set recovery identifies multi-instance grounding as a remaining difficulty.

4.4 Learning and Reusing Neural Memory

Training objectives. Table 4 examines the effect of continued supervision and different execution rewards. Continued SFT raises Count and MCQ accuracy slightly, while Detection mIoU decreases. Task reward alone further improves Count and MCQ accuracy, but observation fidelity falls from 0.744 to 0.712.

The full objective reaches an observation fidelity of 0.807 and improves Count accuracy, Detection mIoU, and MCQ accuracy over joint SFT by 4.83, 4.10, and 4.79 points. Its MCQ score is also 1.85 points higher than broadcasting the joint return to all branches. These comparisons connect the combined objective to both scene preservation and successful execution.

Preserving unqueried objects. Figure 3(c) separates objects referenced by a fixed question pair from other observed objects. Task-reward-only training increases queried-object recall from 82.60% to 88.10%, while other-object recall decreases from 76.40% to 68.90%. The full objective reaches 89.30% and 84.10%, respectively. The 15.20-point improvement on other objects over task reward alone shows the benefit of observation fidelity for retaining scene content beyond the sampled questions.

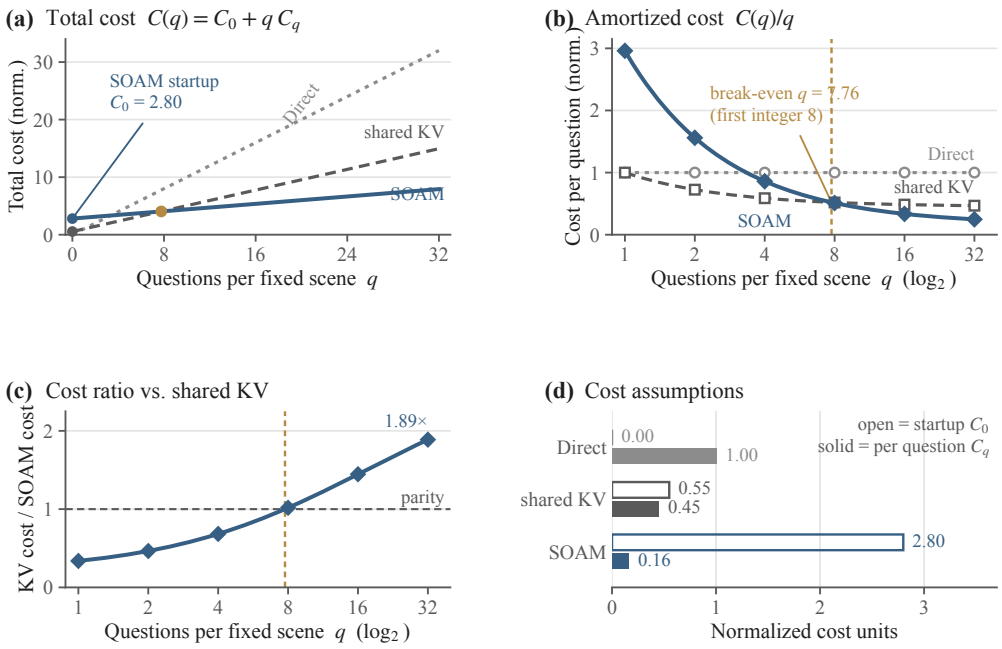

Figure 4: **Amortized inference cost.** Analytical total cost, amortized cost, shared-KV cost ratio, and startup/marginal decomposition, normalized to one uncached direct answer. Setup and marginal constants are $(0, 1)$ for direct answering, $(0.55, 0.45)$ for shared KV, and $(2.80, 0.16)$ for SOAM.

Amortized inference. Figure 4 compares direct answering, shared image-prefix KV, and SOAM using the same analytical cost model. The average cost for m questions is

$$\bar{C}(m) = \frac{C_{\text{setup}}}{m} + C_{\text{question}}. \tag{8}$$

SOAM's setup includes scene decoding, neural-memory readout, and geometry construction. Each subsequent question adds program generation and execution. Under the specified cost constants, its amortized cost becomes lower than shared-KV answering at eight questions. At sixteen questions, the shared-KV cost is $1.45\times$ the SOAM cost. This benefit grows as the initial scene computation is shared across more questions.

5 Conclusion

We presented SOAM, which connects VLM-native neural scene memory with geometric execution. The VLM organizes cross-view objects and reasons over their visual representations, while explicit operations perform the required spatial calculations. Common object addresses connect scene understanding, question-specific programs, and geometry. Joint state–program supervision and reuse-aware execution rewards train the representation for observation fidelity and downstream use. Results on VIEW2SPACE-v1 and SPAR-Bench, together with memory and training ablations, support this organization of spatial reasoning. The shared representation also allows scene computation to be amortized across subsequent questions.

AI USE STATEMENT

We used AI tools to assist with manuscript polishing, literature retrieval and discovery, and the execution of predefined experiments. Execution support included launching experiments on servers, monitoring their progress, and organizing experimental outputs and results. AI tools also assisted with the visual refinement of figures and tables, particularly color selection and presentation consistency. The authors supervised these activities and remain responsible for all methodological decisions, reported results, and scientific conclusions.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *ACL*, 2024.

Shuai Bai, Yuxuan Cai, Runnan Chen, Keqin Chen, Xie Chen, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. SpatialVLM: Endowing vision-language models with spatial reasoning capabilities. In *CVPR*, 2024.

Ting Chen, Saurabh Saxena, Lala Li, David J. Fleet, and Geoffrey Hinton. Pix2Seq: A language modeling framework for object detection. In *ICLR*, 2022.

Chanyoung Gwak, Yoonwoo Jeong, Byungwoo Jeon, Hyunseok Lee, Jinwoo Shin, and Minsu Cho. Cog3DMap: Multi-view vision-language reasoning with 3d cognitive maps. *arXiv preprint arXiv:2603.23023*, 2026.

Jixuan He, Xueting Li, Chieh Hubert Lin, and Ming-Hsuan Yang. Reasmory: 3d reconstruction as explicit memory for vlms spatial reasoning. In *British Machine Vision Conference*, 2026. URL https://kakituken.github.io/Reasmory-project-page/.

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. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2307.12981.

Wenbo Hu, Jingli Lin, Yilin Long, Yunlong Ran, Lihan Jiang, Yifan Wang, Chenming Zhu, Runsen Xu, Tai Wang, and Jiangmiao Pang. G^2VLM: Geometry grounded vision language model with unified 3d reconstruction and spatial reasoning. *arXiv preprint arXiv:2511.21688*, 2025. URL https://arxiv.org/abs/2511.21688.

Haifeng Huang, Yilun Chen, Zehan Wang, Rongjie Huang, Runsen Xu, Tai Wang, Luping Liu, Xize Cheng, Yang Zhao, Jiangmiao Pang, and Zhou Zhao. Chat-Scene: Bridging 3d scene and large language models with object identifiers. In *Advances in Neural Information Processing Systems*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/cebbd24f1e50bcb63d015611fe0fe767-Abstract-Conference.html.

Suchae Jeong, Jaehwi Song, Haeone Lee, Hanna Kim, Jian Kim, Dongjun Lee, Dong Kyu Shin, Changyeon Kim, Dongyoon Hahm, Woogyeol Jin, Juheon Choi, and Kimin Lee. Learning multi-view spatial reasoning from cross-view relations. In *CVPR*, pp. 2570–2581, 2026.

Yuanchen Ju, Yongyuan Liang, Yen-Jen Wang, Nandiraju Gireesh, Yuanliang Ju, Seungjae Lee, Qiao Gu, Elvis Hsieh, Furong Huang, and Koushil Sreenath. MomaGraph: State-aware unified scene graphs with vision–language model for embodied task planning. In *ICLR*, 2026.

Fucai Ke, Zhixi Cai, Boying Li, Long Chen, Beibei Lin, Weiqing Wang, Pari Delir Haghighi, Gholamreza Haffari, and Hamid Rezatofighi. VIEW2SPACE: Studying multi-view visual reasoning from sparse observations. *arXiv preprint arXiv:2603.16506*, 2026.

Lucy Lin, Ayush Jain, Yifan Liu, and Katerina Fragkiadaki. Qwen-3D: A generalist 3d vision-language model for spatial understanding. *arXiv preprint arXiv:2608.02980*, 2026. URL https://arxiv.org/abs/2608.02980.

Benlin Liu, Yuhao Dong, Yiqin Wang, Zixian Ma, Yansong Tang, Luming Tang, Yongming Rao, Wei-Chiu Ma, and Ranjay Krishna. Coarse correspondences boost spatial-temporal reasoning in multimodal language model. In *CVPR*, 2025.

Haoran Lou, Chunxiao Fan, Ziyan Liu, Yuexin Wu, and Xinliang Wang. LLaVA-SP: Enhancing visual representation with visual spatial tokens for MLLMs. *arXiv preprint arXiv:2507.00505*, 2025.

Zhanpeng Luo, Ce Zhang, Silong Yong, Cunxi Dai, Qianwei Wang, Haoxi Ran, Guanya Shi, Katia Sycara, and Yaqi Xie. pySpatial: Generating 3d visual programs for zero-shot spatial reasoning. In *ICLR*, 2026.

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*, 2025.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *ICLR*, 2019. URL https://arxiv.org/abs/1904.12584.

Masanari Oi, Koki Maeda, Ryuto Koike, Daisuke Oba, Nakamasa Inoue, and Naoaki Okazaki. From correspondence to actions: Human-like multi-image spatial reasoning in multi-modal large language models. *arXiv preprint arXiv:2602.08735*, 2026.

JooHyun Park and HyeongYeop Kang. RenderMem: Rendering as spatial memory retrieval. *arXiv preprint arXiv:2603.14669*, 2026. URL https://arxiv.org/abs/2603.14669.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In *NeurIPS*, 2015.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *CVPR*, 2025.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jérôme Revaud. DUSt3R: Geometric 3d vision made easy. In *CVPR*, 2024.

Diankun Wu, Fangfu Liu, Yi-Hsin Hung, and Yueqi Duan. Spatial-MLLM: Boosting MLLM capabilities in visual-based spatial intelligence. *arXiv preprint arXiv:2505.23747*, 2025.

Jing Wu, Jianhua Wu, Jiayi Guan, Jiahong Chen, Jinghui Lu, Hangjun Ye, Bingzhao Gao, and Long Chen. SpatioLM: Towards general physical spatial intelligence in vision-language models. *arXiv preprint arXiv:2608.01899*, 2026.

Chun-Hsiao Yeh, Chenyu Wang, Shengbang Tong, Ta-Ying Cheng, Ruoyu Wang, Tianzhe Chu, Yuexiang Zhai, Yubei Chen, Shenghua Gao, and Yi Ma. Seeing from another perspective: Evaluating multi-view understanding in MLLMs. In *AAAI Conference on Artificial Intelligence*, 2026.

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

Jian Zhang, Shijie Zhou, Bangya Liu, Achuta Kadambi, and Zhiwen Fan. SpatialStack: Layered geometry-language fusion for 3d VLM spatial reasoning. In *CVPR*, pp. 38678–38688, 2026.