SpatialStack: Layered Geometry-Language Fusion for 3D VLM Spatial Reasoning

Supplementary Material

In this supplementary material, we provide comprehensive implementation details and additional experimental results for *SpatialStack*. The content is organized as follows:

- Sec. A elaborates on the detailed architectural components, including the geometry token extraction pipeline and the masked additive fusion mechanism.
- Sec. B describes the composition and statistics of our training dataset mixture.
- Sec. C describes the training details, including input processing and specific training configurations.
- Sec. D provides the detailed evaluation protocols, including the specific benchmarks and metrics.
- Sec. E presents additional baseline comparisons on zero-shot spatial reasoning in CV-Bench.
- Sec. F offers qualitative visualizations contrasting the feature responses of geometry and vision encoders.

A. Architecture Details

To enable both fine-grained and global spatial reasoning, our architecture integrates multi-level geometric cues extracted from VGGT [9] into the VLM. The overall pipeline consists of three stages: geometry token extraction and spatial alignment (Sec. A.1); geometry merging and projection into the language feature space (Sec. A.2); and masked additive fusion that injects geometry exclusively into the visual-token slice of the decoder state (Sec. A.3). The following subsections describe each component in detail.

A.1. Geometry Token Extraction and Preprocessing

We first outline the end-to-end flow of geometry token extraction and alignment before detailing the reshaping and reordering process. At both training and inference time, images are processed by the vision encoder of the chosen base model (Qwen2.5-VL [1] or Qwen3.5 [8]) to generate visual features. For Qwen2.5-VL, this encoding procedure consists of patch embedding, window based attention, and hierarchical patch merging, and produces a sequence of merged vision tokens; for Qwen3.5, the stock vision encoder produces image embeddings that are inserted into the multimodal sequence. In parallel, VGGT (frozen, evaluation mode) emits geometry tokens or layer-wise geometry features from selected internal aggregator layers. These geometry features are subsequently reshaped and reordered, when needed, to match the layout of the visual tokens before fusion, ensuring spatial consistency.

Token Structuring. VGGT produces a sequence of tokens at multiple internal aggregator layers. Each output contains three types of tokens: a *camera token* encoding global viewpoint information; several *register tokens* acting as global latent slots; and a sequence of *patch tokens* representing per-patch geometric features.

Let $h_{\text{patch}} = H/p$ and $w_{\text{patch}} = W/p$ denote the spatial resolution of the VGGT patch tokens. The patch tokens are originally arranged in a flat row-major sequence of length $h_{\text{patch}} \times w_{\text{patch}}$. To align their traversal order with the vision encoder after the spatial merger step, we partition the spatial grid into windows of size $s \times s$, where $s = \texttt{spatial\_merge\_size}$ (default $s = 2$):

$$(h_{\text{patch}},\, w_{\text{patch}}) \;\to\; \left(\tfrac{h_{\text{patch}}}{s},\, s,\, \tfrac{w_{\text{patch}}}{s},\, s\right), \tag{1}$$

and apply a permutation that moves window indices ahead of within-window positions:

$$\left(\tfrac{h_{\text{patch}}}{s},\, s,\, \tfrac{w_{\text{patch}}}{s},\, s\right) \;\to\; \left(\tfrac{h_{\text{patch}}}{s},\, \tfrac{w_{\text{patch}}}{s},\, s,\, s\right). \tag{2}$$

Finally, the reordered grid is flattened back into a 1D sequence:

$$\left(\tfrac{h_{\text{patch}}}{s},\, \tfrac{w_{\text{patch}}}{s},\, s,\, s\right) \;\to\; h_{\text{patch}} \cdot w_{\text{patch}}. \tag{3}$$

This reordering preserves the total number of tokens while changing their traversal order: tokens are enumerated window-by-window rather than row-by-row. As a result, consecutive groups of s^2 geometry tokens correspond to the same spatial region grouped by one merged visual token, ensuring spatial alignment prior to fusion.

A.2. Geometry-to-Language Projection

After reordering the geometry patch tokens to match the traversal order of the merged vision tokens (Sec. A.1), we obtain a 1D sequence

$$\mathbf{Z} \in \mathbb{R}^{(h_{\text{patch}} \cdot w_{\text{patch}}) \times D_{\text{geo}}}, \tag{4}$$

where $h_{\text{patch}} \cdot w_{\text{patch}}$ denotes the total number of spatial tokens and D_{g} is the geometry feature dimension.

Normalization. Following the design of Qwen2.5-VL [1], token-wise RMS normalization [13] is first applied:

$$\mathbf{Z}_{\text{norm}} = \mathrm{RMSNorm}(\mathbf{Z}). \tag{5}$$

Window-wise merging. The normalized tokens are grouped into non-overlapping spatial windows of size $s \times s$.

Each window is flattened and concatenated along the channel dimension, producing a 1D sequence of merged geometry tokens:

$$\tilde{\mathbf{Z}} \in \mathbb{R}^{\left(\frac{h_{\text{patch}}}{s} \cdot \frac{w_{\text{patch}}}{s}\right) \times (s^2 D_{\text{geo}})}. \tag{6}$$

Projection to language space. Each flattened window token is projected to the language decoder dimension by a two-layer MLP:

$$\mathbf{G} = W_2\, \sigma(W_1 \tilde{\mathbf{Z}} + b_1) + b_2, \tag{7}$$

where $W_1 \in \mathbb{R}^{D_{\text{mlp}} \times (s^2 D_{\text{geo}})}$, $b_1 \in \mathbb{R}^{D_{\text{mlp}}}$, $W_2 \in \mathbb{R}^{D_{\text{lang}} \times D_{\text{mlp}}}$, and $b_2 \in \mathbb{R}^{D_{\text{lang}}}$. The projected geometry representation has shape

$$\mathbf{G} \in \mathbb{R}^{\left(\frac{h_{\text{patch}}}{s} \cdot \frac{w_{\text{patch}}}{s}\right) \times D_{\text{lang}}}. \tag{8}$$

A.3. Additive Fusion via Vision-Token Mask

Let $H_l \in \mathbb{R}^{N_{\text{tot}} \times D_{\text{lang}}}$ denote the decoder hidden states at layer l, where N_{tot} is the token sequence length (including system prompt, instruction text, vision tokens, and autoregressive text), and D_{lang} is the decoder hidden dimension.

The projected geometry features from Sec. A.2 are $\mathbf{G}_l \in \mathbb{R}^{N_p \times D_{\text{lang}}}$, where N_p denotes the number of vision tokens participating in fusion (in the default setting without camera tokens and assuming h_{patch} and w_{patch} are divisible by s, $N_p = \frac{h_{\text{patch}}}{s} \cdot \frac{w_{\text{patch}}}{s}$). To locate the visual portion of the sequence, we define a binary mask $M_{\text{vis}} \in \{0,1\}^{N_{\text{tot}}}$, where $M_{\text{vis}}[i] = 1$ if and only if position i corresponds to a visual token.

Additive fusion updates only the masked positions:

$$\mathbf{H}_l \leftarrow \mathbf{H}_l + \text{scatter}\big(\mathbf{G}_l,\ M_{\text{vis}}\big), \tag{9}$$

where $\text{scatter}(\mathbf{G}_l, M_{\text{vis}})$ distributes rows of $\mathbf{G}_l$ sequentially to locations where $M_{\text{vis}} = 1$ and inserts zeros elsewhere.

Equivalently, for each token index i,

$$\mathbf{H}_l[i] \leftarrow \begin{cases} \mathbf{H}_l[i] + \mathbf{G}_l[k], & \text{if } M_{\text{vis}}[i] = 1, \\ \mathbf{H}_l[i], & \text{if } M_{\text{vis}}[i] = 0, \end{cases} \tag{10}$$

where k enumerates the N_p masked positions.

Thus, geometry information is injected exclusively into the vision-token slice of the decoder state, while non-vision tokens (e.g., system prompt and text tokens) remain unchanged. During autoregressive generation, this fusion is applied at the initial prefill step, after which standard decoding proceeds with the updated hidden states.

B. Dataset Details

We construct a balanced dataset of approximately 200k samples, blending spatial expertise with general instruction-following capabilities to facilitate efficient experimentation. Specifically, we sample subsets from **SPAR-234k** and **LLaVA-Hound-64k** (both from *VG-LLM* [16]), as well as the **ScanNet split** of the *VLM-3R* dataset [7], which provides explicit spatial supervision. To enhance perception of object sequences, we additionally include approximately 2k appearance-order instances from the **VSI-590k** [11] collection. As summarized in Tab. A, this composition ensures broad task coverage suitable for controlled architectural ablations.

B.1. Spatial Instruction-Following Data

SPAR (Spatial Perception and Reasoning). SPAR [14] is a large-scale vision–language dataset designed for *spatial perception and reasoning* in complex indoor scenes, featuring diverse question–answer pairs across 33 spatial task types spanning low-level perception to high-level reasoning, and covering single-view, multi-view, and video formats. We build upon the publicly released **SPAR-234k** subset introduced in [16]; the detailed task-type distribution of our sampled training set is illustrated in Fig. A.

VLM-3R. VLM-3R is a spatial QA construction framework based on open-source 3D datasets with geometry, semantic labels, and instance-level annotations, including ScanNet [6], ScanNet++ [12], and ARKitScenes [2]. We use only the ScanNet split, which provides six spatial QA task types: Object Counting, Relative Distance, Relative Direction, Object Size, Absolute Distance, and Room Size. This split does not include Route Planning or Appearance Order tasks.

VSI-590K. VSI-590k is a large-scale spatial instruction-tuning dataset consisting of 590k QA examples from real and simulated indoor environments across 12 task types. For training, we extract a 2k subset corresponding to the appearance-order task derived specifically from the ScanNet portion of VSI-590k, which supplements the absence of appearance-order supervision in the VLM-3R ScanNet split.

We refer to this combined compilation of spatial tasks as **VSI-Type Data**. As visualized in Fig. B, these seven tasks are categorized into three major groups: Configuration, Measurement, and Spatiotemporal, following the taxonomy in the VSI-Bench setting.

B.2. General Video Instruction-Following Data

LLaVA-Hound. LLaVA-Hound [15] is a dataset for video captioning, instruction tuning, and preference alignment, curated from 900k videos sourced from WebVid, VIDAL, and ActivityNet. High-quality captions are produced using GPT-4V from uniformly sampled frames, followed by 240k instruction–answer pairs generated using ChatGPT and 17k preference pairs for Direct Preference Optimization. We use the 64k LLaVA-Hound subset released in VG-LLM, from which 60 percent is sampled to retain general instruction-following and object-grounded reasoning capability while

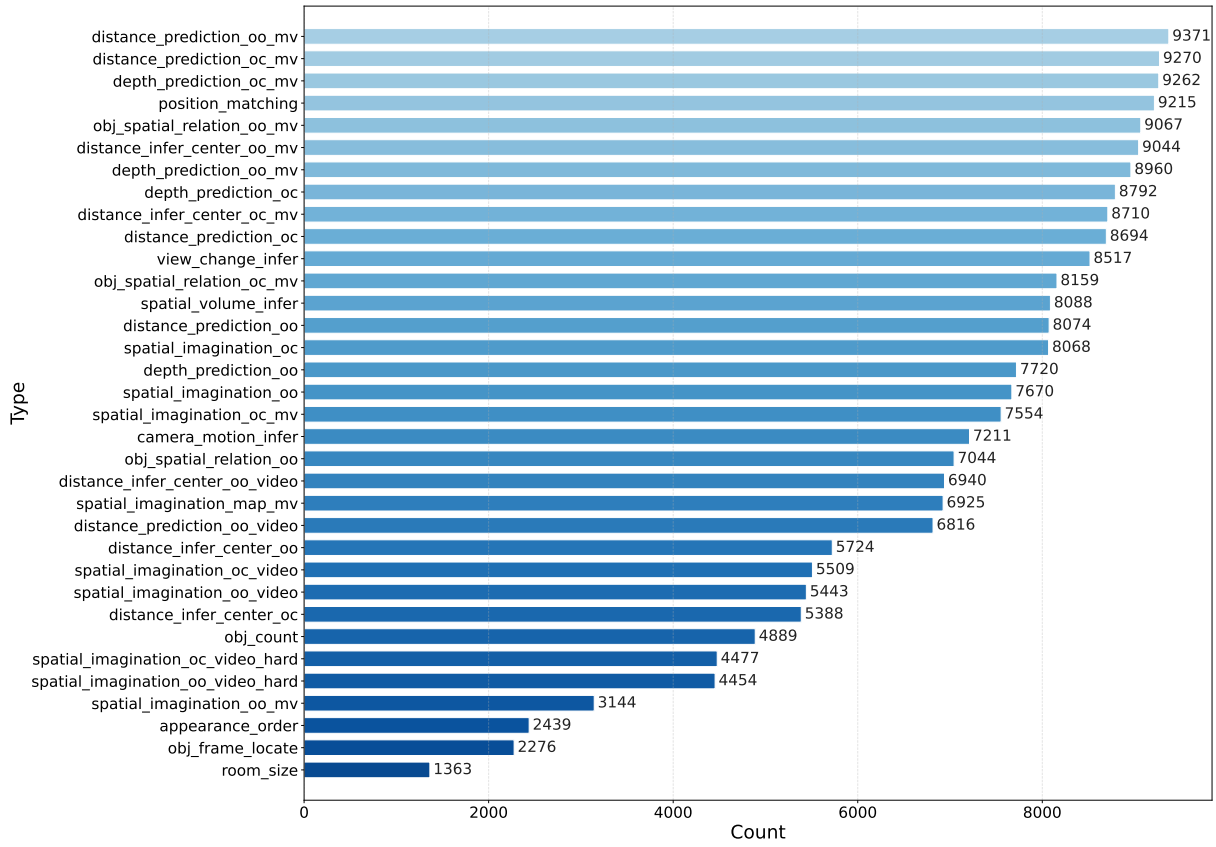

Figure A. **Task-type distribution of the sampled SPAR subset.** The bar chart reports the counts of all 33 spatial task types after randomly sampling 60% of SPAR-234k for training.

Dataset	Raw	Train Subset
SPAR 234k	234k (66.3%)	140k (66.4%)
LLaVA-Hound 64k	63.8k (18.0%)	38.3k (18.1%)
VLM3R-ScanNet	51.8k (14.6%)	31.1k (14.7%)
VSI App-Order	3.8k (1.1%)	1.9k (0.9%)
Total	353k (100%)	212k (100%)

Table A. **Dataset scales and sampled subsets used in our $\sim$ 200k training mixture.** We sample 60% from SPAR-234k, LLaVA-Hound-64k [16], and the ScanNet split of VLM-3R [7], and add $\sim$ 2k appearance-order instances from VSI-590k [11] to compensate for the missing ordering supervision. Percentages indicate each dataset's contribution to the final mixture.

keeping the training scale computationally manageable.

C. Training Details

This section details the implementation of *SpatialStack*, focusing on (1) input processing and (2) training settings. The model is trained via large-scale geometry-aware instruction tuning, where only the language tower and geometry-merger modules are updated, while the vision tower and VGGT remain frozen. All experiments are conducted on 32 NVIDIA A100 GPUs (80GB).

C.1. Input Processing

Videos are first decomposed into individual frame images before entering the multimodal pipeline. A single video token in the prompt is expanded into K consecutive image tokens. For a clip of duration T_{sec} containing F total frames, we uniformly sample $K = \text{clip}(\text{round}(T_{\text{sec}}/\Delta), K_{\min}, K_{\max})$ frame indices from $[0, F-1]$, where Δ denotes the temporal sampling interval.

Each frame (and standalone image) undergoes a unified visual preprocessing pipeline. For SPAR-style training samples, optional task-specific marking is first applied on the original-resolution image: task cues such as points or bounding boxes are drawn according to the provided annotation metadata before any resizing. Transparency is then composited onto a white background and the image is converted to RGB.

Next, we resize the image while preserving aspect ratio to a target size of 518 pixels. In the default crop-based set-

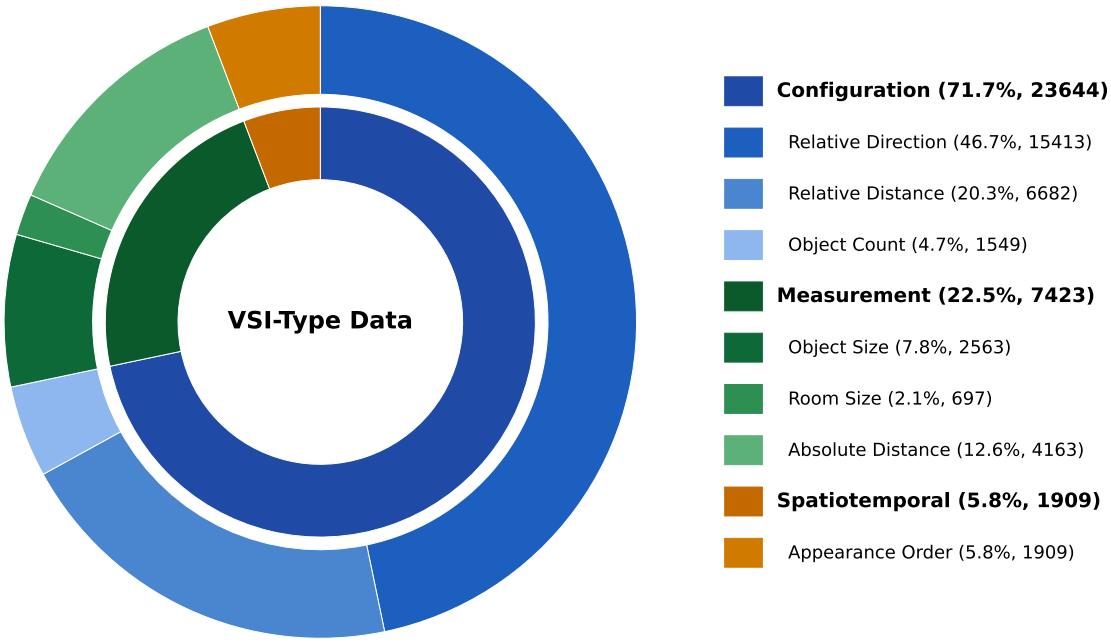

Figure B. **Task-type distribution of the seven tasks in the VSI-Bench setting.** The pie chart summarizes the combined composition from VLM3R-ScanNet and the sampled Appearance-Order subset from VSI-590K, which are merged for unified reporting.

ting, one side is resized to 518 pixels and the other side is scaled proportionally, with center cropping applied when needed. We then apply, when necessary, patch-aligned spatial trimming so that the final height and width satisfy $H \bmod (p \cdot m) = 0$ and $W \bmod (p \cdot m) = 0$, ensuring that the resolution becomes an integer multiple of the effective patch unit $p \cdot m$ (e.g., $14 \cdot 2 = 28$). This alignment is required because the merge stage groups $m \times m$ adjacent patches into a single token.

Finally, the resized image is used to construct inputs for both the vision encoder and the geometry encoder (VGGT), with additional patch/merge alignment applied where needed to maintain spatial consistency between the two branches.

C.2. Training Settings

We train SpatialStack using `torchrun` with DeepSpeed ZeRO-2. Optimization uses AdamW with cosine decay scheduling and warmup. bfloat16 precision is employed for training efficiency and numerical robustness. Tab. B summarizes the configuration.

Category	Setting
Base model	Qwen2.5-VL-3B or Qwen3.5-4B
Geometry encoder	VGGT-1B (frozen)
Fusion strategy	SpatialStack (multi-depth)
Trainable modules	Language tower + fusion modules
Precision	bfloat16
Optimizer	AdamW (wd=0.01)
Learning rate	1×10^{-5}
Scheduler	Cosine decay, warmup 3%
Epochs	1
Batch size	effective 64
Sequence length	12,800 tokens
Frames per video	4–8
Pixels/sample	$16\cdot28^2$–$576\cdot28^2$
Distributed	torchrun + DeepSpeed ZeRO-2
Checkpoint save interval	every 1000 steps
Logging	every 10 steps
Hardware	32×A100 GPUs (80GB)

Table B. **Training hyperparameters for SpatialStack.** Geometry-aware instruction tuning is performed on Qwen2.5-VL-3B or Qwen3.5-4B with VGGT-1B using the proposed SpatialStack fusion. The language tower and fusion modules are trainable, while the geometry encoder remains frozen. Training uses AdamW (bfloat16, cosine schedule) with an effective batch size of 64 under ZeRO-2 parallelism.

D. Evaluation Details

Our evaluation pipeline closely follows established protocols to ensure fair comparison. Specifically, we adopt the data preprocessing methodology from *VG-LLM* [16] and

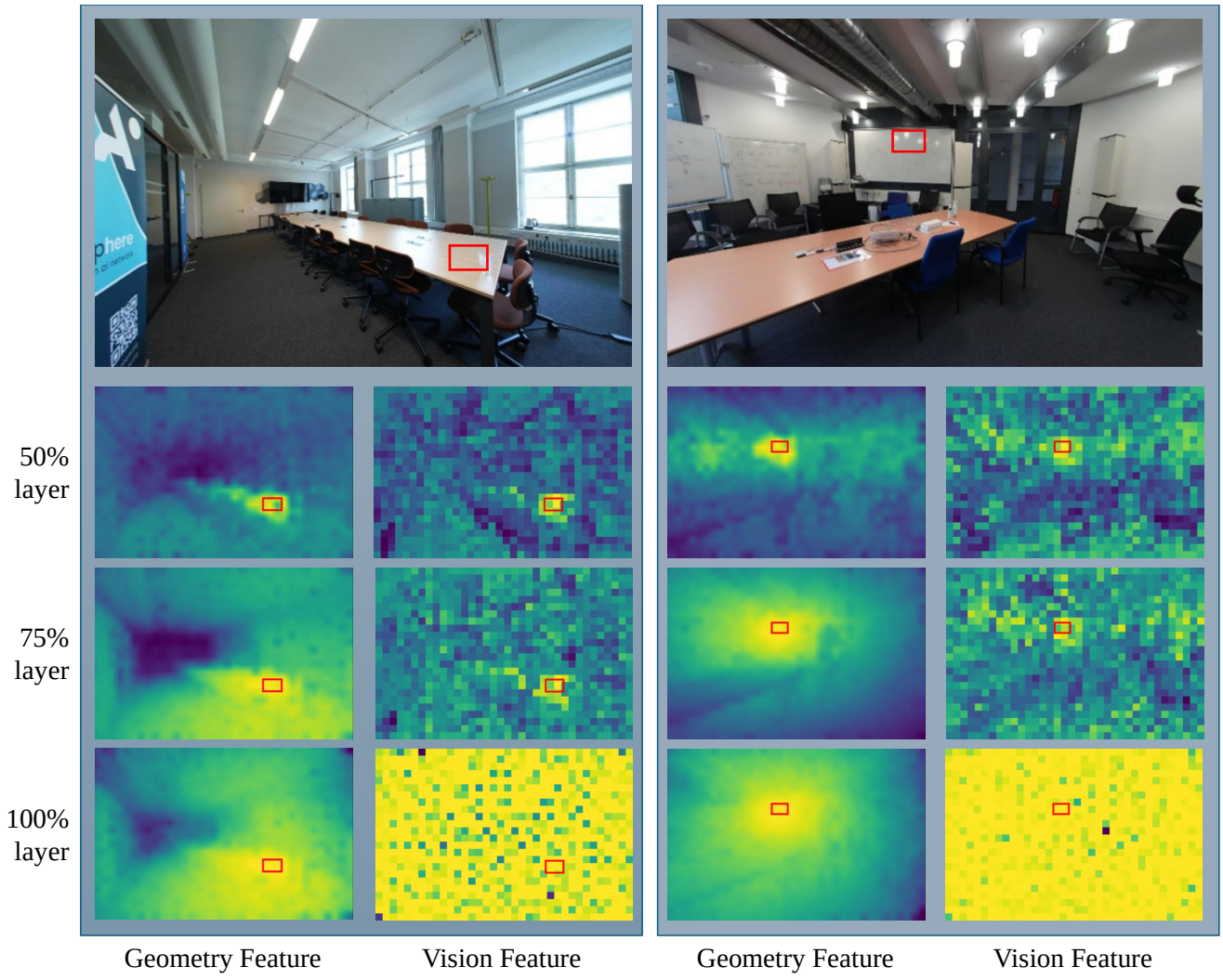

Figure C. **ROI similarity comparison between geometry and vision features across encoder depths.** For two indoor scenes, the top row shows the RGB image with the ROI marked in red. The lower rows display similarity maps (brighter means more similar) at 50%, 75%, and 100% depths of the geometry encoder (left) and the vision encoder (right). Geometry features preserve meaningful spatial structure, while vision features are noisy and become nearly uniform at deeper layers.

adhere to the standard evaluation parameter settings defined in *VSI-Bench* [10].

Implementation Details. Visual inputs (single images, image lists, or videos) are first decomposed into sampled frames with a capped count K, using uniform frame sampling in the evaluation pipeline. Following the preprocessing pipeline of [16], geometry-aware evaluation uses a 518-pixel image preprocessing step. To ensure compatibility with our token merging mechanism, patch/merge alignment is enforced when required so that the patch grid dimensions are divisible by the merge factor m:

$$(W/p) \bmod m = 0 \quad \text{and} \quad (H/p) \bmod m = 0, \tag{11}$$

where p denotes the patch size. When geometry is enabled, the geometry encoder inputs are constructed from the same visual content as the vision branch to maintain spatial correspondence.

Geometry tokens are computed once per sample in evaluation mode. Geometry fusion is injected at predefined decoder layers after self-attention and MLP execution, replacing the vision-aligned slice before decoding continues.

Decoding adopts greedy generation by default (`temperature` $=0$, `num_beams` $=1$) with task-specific generation limits unless specified otherwise. Key/value caching is enabled for efficiency, and outputs are trimmed to remove the prompt prefix before evaluation. All benchmark results in the main paper are produced under this evaluation configuration.

E. More Results

We evaluate zero-shot spatial reasoning on CV-Bench in Tab. C. Our method consistently outperforms both SpatialRGPT [5] and Spatialbot [3] across all metrics. Note that SpatialVLM [4] is excluded as its code is unavailable.

Method	Count	Relation	Depth	Distance	Overall
SpatialRGPT	60.4	78.9	80.0	71.3	72.7
Spatialbot	61.4	73.1	76.5	61.0	68.0
Ours	**69.0**	**92.5**	**93.7**	**90.7**	**86.5**

Table C. **Additional Baseline Comparison on CV-Bench.**

F. More Visualizations

F.1. Geometry vs. Vision Feature Responses

To analyze the difference between geometry and vision representations, we visualize ROI-based similarity maps derived from features at different encoder depths, as shown in Fig. C. For each scene, a red box marks a region of interest (ROI) in the RGB image. We compute patch-wise similarity between this ROI and all other spatial locations using features extracted at 50%, 75%, and 100% depth of the geometry encoder and compare them with features from the native vision encoder at corresponding relative depths.

Here, the percentages refer to proportional positions within the encoder stack rather than absolute layer indices. For example, the geometry encoder contains 24 layers, so 50%, 75%, and 100% depths correspond to layers 12, 18, and 24. The vision encoder contains 32 layers, where the same relative depths map to layers 16, 24, and 32. This proportional alignment allows a fair comparison between encoders with different depths.

The similarity maps reveal a consistent trend: shallow geometry layers preserve fine-grained spatial distinctions and clear geometric boundaries, whereas deeper geometry layers become increasingly homogeneous, causing many regions to appear similar despite different physical geometry. In contrast, similarity maps from the native visual encoder are noisy and spatially fragmented across depths, and at the deepest layers they collapse into nearly uniform responses without meaningful spatial differentiation.

These results demonstrate that internal visual features alone lack explicit spatial structure and are insufficient for reasoning about relative geometry. External geometry encoders provide structured spatial cues at different levels that are missing from the native visual pathway, motivating the use of multi-level geometry fusion in spatial reasoning.

References

[1] Shuai Bai et al. Qwen2.5-VL Technical Report, 2025. 1

[2] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, et al. Arkitscenes: A diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. *arXiv preprint arXiv:2111.08897*, 2021. 2

[3] Wenxiao Cai, Iaroslav Ponomarenko, Jianhao Yuan, Xiaoqi Li, Wankou Yang, Hao Dong, and Bo Zhao. Spatialbot: Precise spatial understanding with vision language models. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9490–9498. IEEE, 2025. 6

[4] Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *CVPR*, pages 14455–14465, 2024. 6

[5] An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. Spatialrgpt: Grounded spatial reasoning in vision-language models. *Advances in Neural Information Processing Systems*, 37:135062–135093, 2024. 6

[6] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 2

[7] Zhiwen Fan, Jian Zhang, Renjie Li, Junge Zhang, Runjin Chen, Hezhen Hu, Kevin Wang, Huaizhi Qu, Shijie Zhou, Dilin Wang, 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*, 2026. 2, 3

[8] Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, 2026. 1

[9] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025. 1

[10] 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 *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10632–10643, 2025. 5

[11] 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. *arXiv preprint arXiv:2511.04670*, 2025. 2, 3

[12] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12–22, 2023. 2

[13] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019. 1

[14] Jiahui Zhang, Yurui Chen, Yanpeng Zhou, Yueming Xu, Ze Huang, Jilin Mei, Junhui Chen, Yu-Jie Yuan, Xinyue Cai, Guowei Huang, et al. From flatland to space: Teaching vision-language models to perceive and reason in 3d. *arXiv preprint arXiv:2503.22976*, 2025. 2

[15] Ruohong Zhang, Liangke Gui, Zhiqing Sun, Yihao Feng, Keyang Xu, Yuanhan Zhang, Di Fu, Chunyuan Li, Alexander G Hauptmann, Yonatan Bisk, et al. Direct preference optimization of video large multimodal models from language model reward. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 694–717, 2025. 2

[16] Duo Zheng, Shijia Huang, Yanyang Li, and Liwei Wang. Learning from videos for 3d world: Enhancing mllms with 3d vision geometry priors. *Advances in neural information processing systems*, 2025. 2, 3, 4, 5