IN3D-VLM: FAITHFUL INTEGRATION OF EXPLICIT 3D SPACE FOR METRIC SPATIAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

As spatial reasoning becomes increasingly important for vision-language models (VLMs), incorporating explicit 3D geometry has attracted growing attention. However, existing approaches either rely on computationally expensive geometry encoders or mix compressed geometry with RGB features without ensuring that the injected 3D information remains accessible after fusion. Consequently, even when metric 3D observations are available, VLMs may struggle to accurately infer quantitative properties such as depth, distance, and relative displacement. We argue that effective 3D space injection depends on two fundamental design choices: what geometric representation to inject and how to fuse it with visual features while keeping the injected geometry explicitly accessible. To this end, we introduce In3D-VLM, which directly incorporates metric 3D geometry obtained from depth sensors or metric depth estimation models. The Point-Map Tokenizer (PMT), a lightweight tokenizer pretrained through masked XYZ point-map reconstruction, converts dense observations into compact, geometrically recoverable tokens without relying on a large geometry backbone. Geometry-Preserving Fusion (GPF) then maps these geometry tokens through a fixed orthonormal basis into a dedicated subspace of the LLM embedding space and adds them to the spatially aligned RGB visual tokens. The orthonormal mapping expands the geometry tokens into the LLM space without deforming their learned geometric structure. Geometry-leak regularization further reduces semantic overlap with the reserved subspace, encouraging an explicit linear geometry readout from the fused tokens. We further introduce MD2Bench to evaluate metric depth and distance reasoning from provided 3D geometry under visual-point and text-name grounding protocols. In3D-VLM substantially outperforms existing VLMs on MD^2Bench and achieves state-of-the-art performance on established public spatial-reasoning benchmarks.

1 INTRODUCTION

Spatial reasoning is increasingly important for vision-language models (VLMs), from general scene understanding to physical-world applications such as robotics (Yuan et al., 2024; Qu et al., 2025) and autonomous driving (Wang et al., 2025d; Tian et al., 2024). Many existing spatial tasks focus on recognizing objects and reasoning about qualitative relations, such as whether one object is behind or to the left of another (Chen et al., 2024; Song et al., 2025; Du et al., 2024). Although useful, such relations provide only a coarse understanding of scene geometry. More comprehensive spatial understanding requires quantitative reasoning about metric properties, including object depth, distance, and relative displacement. However, these quantities are fundamentally ambiguous from monocular RGB observations because absolute scale is not directly observable. Recent studies therefore augment VLMs with explicit geometric inputs, such as depth and point maps, or with 3D-aware representations produced by pretrained geometry models (Chen et al., 2025; Zheng et al., 2025b; Wang et al., 2025f).

Effective geometry integration poses two central design questions: *what geometric representation to inject* and *how to fuse it with pretrained visual features*. The representation should compress dense metric observations into compact tokens while retaining recoverable geometric information. The fusion mechanism should preserve access to this information and limit interference with pretrained

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.

visual semantics. Crucially, these goals should be achieved with minimal computational overhead and without increasing the visual-token sequence length.

The first design question concerns geometry representation. Some approaches downsample or pool dense depth or point maps to match the spatial resolution of visual tokens (Chen et al., 2025; Wang et al., 2025f; Zheng et al., 2025b; Li et al., 2026c). Although efficient, naive pooling performs task-agnostic local aggregation without explicitly encouraging geometric recoverability. For example, reducing a 512×512 depth map to a 32×32 feature grid results in a 256-fold reduction in spatial locations, with each output feature effectively summarizing a 16×16 input region. This compression can obscure object boundaries, depth discontinuities, and other fine geometric structures. At the other extreme, some approaches (Zheng et al., 2025a; Hu et al., 2026; Zhao et al., 2026; Zhang et al., 2026b) extract geometry-aware features from RGB images using large pretrained geometry or vision backbones (Wang et al., 2025a;b; Tschannen et al., 2025). Although these backbones can provide rich geometric or visual priors, they introduce substantial parameter, memory, and inference overhead. Moreover, because their geometry is inferred from RGB, they do not naturally incorporate metric observations from depth sensors. Even with a compact geometry representation, a second question remains: how should it be fused with visual features? Existing methods often add or project geometry features directly into RGB features (Liang et al., 2026; Cheng et al., 2026). Although straightforward, such fusion provides no explicit mechanism for keeping the geometry representation explicitly readable from the fused tokens. Geometry and semantic features may therefore become entangled, requiring the language model to implicitly recover the geometry signal while performing metric spatial reasoning.

To address both the representation and fusion challenges, we introduce **In3D-VLM**, a framework for efficiently integrating explicit metric 3D geometry into VLMs. For geometry representation, In3D-VLM uses an image-aligned XYZ point map constructed from either sensor depth or metric depth estimates. Our **Point-Map Tokenizer (PMT)** compresses this dense point map into compact geometry tokens spatially aligned with the RGB token grid. PMT is pretrained through masked XYZ reconstruction to retain recoverable metric information without relying on a large geometry backbone. For geometry fusion, we introduce **Geometry-Preserving Fusion (GPF)**, which maps the geometry tokens through a fixed orthonormal basis into a dedicated subspace of the language-model embedding space and adds them to the spatially aligned RGB visual tokens. The orthonormal mapping expands the geometry tokens into the language-model space without deforming their learned geometric structure. Geometry-leak regularization further reduces the projection of semantic visual features onto the reserved subspace, keeping the geometry linearly readable from the fused tokens. GPF therefore provides an explicit geometry readout without increasing the visual-token sequence length. We further introduce **MD^{2}Bench**, a benchmark designed to determine whether VLMs can faithfully interpret supplied metric 3D geometry. By deriving reference answers from the same geometry provided to the model, MD^{2}Bench reduces input–target mismatch and isolates the model's ability to read and reason over explicit geometric observations.

Our contributions are threefold:

1. We introduce PMT, a lightweight tokenizer that compresses image-aligned XYZ point maps into compact, recoverable geometry tokens through masked reconstruction.
2. We propose GPF, which fuses geometry and RGB tokens through a fixed orthonormal subspace. Geometry-leak regularization keeps geometry linearly readable without increasing the visual-token sequence length.
3. We introduce MD^{2}Bench to evaluate whether VLMs can faithfully reason over supplied metric 3D geometry, and demonstrate strong performance across spatial-reasoning benchmarks.

2 Related Work

2.1 Spatial Reasoning in VLMs

Spatial reasoning in VLMs has primarily advanced through spatially grounded supervision and specialized training. Prior work learns qualitative relations, metric estimation, and embodied spatial understanding from annotated 3D scenes (Tong et al., 2024; Song et al., 2025; Liao et al., 2024; Zhang

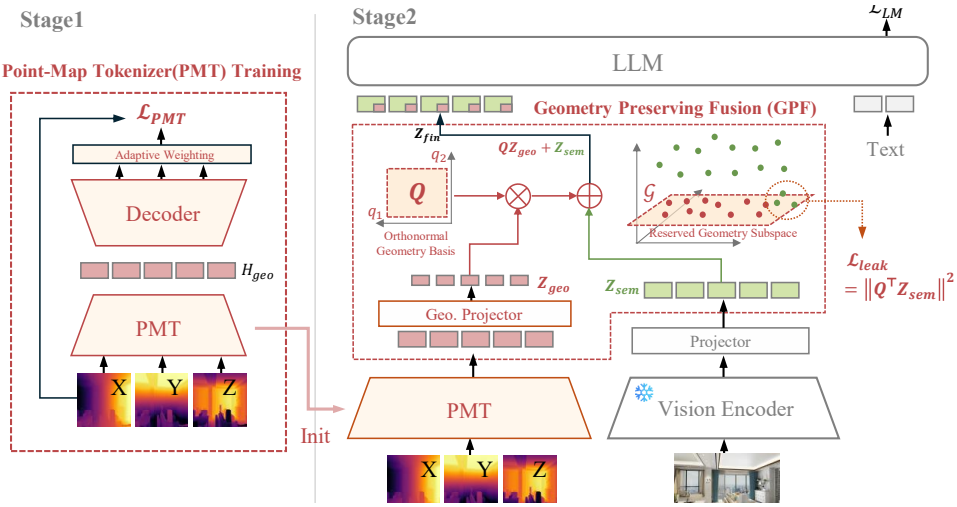


Figure 1: **Overview of In3D-VLM.** PMT encodes an XYZ point map into RGB-aligned geometry tokens. GPF maps and adds them to the RGB tokens through a fixed orthonormal basis $\mathbf{Q}$, while geometry-leak regularization encourages the geometry to remain linearly recoverable via $\mathbf{Q}^\top$.

et al., 2025; Daxberger et al., 2025), or from 3D-aware supervision obtained by lifting 2D observations (Chen et al., 2024; Cheng et al., 2024; Deng et al., 2026; Yang et al., 2025b; Cheng et al., 2026). Other studies address scene-level reasoning over reconstructed 3D environments (Azuma et al., 2022; Ma et al., 2023; Jia et al., 2024; Majumdar et al., 2024; Lyu et al., 2024), as well as multi-step reasoning through progressive training and reinforcement learning (Li et al., 2026b; Wu et al., 2025b; Ouyang et al., 2025). Despite this progress, efficiently integrating observed metric geometry as an explicitly accessible VLM input remains relatively underexplored.

2.2 3D Geometry Integration in VLMs

Existing geometry-integrated VLMs broadly follow two strategies. The first uses large pretrained geometry or perception backbones (Wang et al., 2025a; Oquab et al., 2023; Wang et al., 2025b) and fuses their geometry-aware representations with visual features (Zheng et al., 2025a; Hu et al., 2026; Zhao et al., 2026; Wu et al., 2025a; Fan et al., 2026; Zhang et al., 2026b; Li et al., 2026a). While providing rich 3D priors, these additional streams introduce substantial parameter, memory, and inference overhead, and geometry inferred from RGB does not directly use metric observations available from depth sensors. The second strategy directly injects depth maps, point maps, or 3D positional information into visual tokens (Chen et al., 2025; Wang et al., 2025f; Zheng et al., 2025b; Cheng et al., 2026; Liang et al., 2026; Li et al., 2026c). Although computationally lighter, these methods often rely on pooling, downsampling, or unconstrained fusion without ensuring that the compressed geometry remains accessible after fusion. In contrast, our method learns compact geometry tokens from observed XYZ point maps through reconstruction pretraining and injects them through an orthonormal subspace that supports an explicit linear geometry readout.

3 In3D-VLM

3.1 Overview

As illustrated in Fig. 1, **In3D-VLM** augments a pretrained VLM with explicit metric geometry obtained from sensor depth or estimated metric depth. **Point-Map Tokenizer (PMT)** compresses an image-aligned XYZ point map into compact geometry tokens, and **Geometry-Preserving Fusion (GPF)** maps them into a reserved orthonormal subspace and adds them to the spatially aligned RGB tokens. **Geometry-leak regularization** discourages RGB-derived semantic features from occupying the reserved subspace, keeping the injected geometry linearly recoverable from the fused tokens. The resulting tokens retain access to both RGB semantics and metric geometry without increasing the visual-token sequence length.

3.2 POINT-MAP TOKENIZER

To incorporate geometric information into a VLM, dense 3D observations must first be converted into compact geometry tokens compatible with the VLM's visual representation. We use an image-aligned XYZ point map as the input representation. Unlike a sparse point cloud, it preserves dense pixel-wise correspondence with the RGB image and can be constructed from either sensor depth or estimated metric depth.

Given a metric depth map $d(u, v)$ and camera intrinsics (f_x, f_y, c_x, c_y), each valid pixel (u, v) is lifted to the camera coordinate system as

$$\mathbf{x}(u, v) = d(u, v) \begin{bmatrix} (u - c_x)/f_x \\ -(v - c_y)/f_y \\ 1 \end{bmatrix}. \tag{1}$$

The resulting point map $\mathbf{X}$ preserves the metric scale and image-space layout of the input depth.

Based on this representation, we develop PMT, a lightweight autoencoder-based tokenizer for point maps. During pretraining, we apply block-wise masking to an intermediate feature map and train the decoder to reconstruct the corresponding XYZ regions. Because the three coordinate channels can differ in reconstruction difficulty, uniformly averaging their losses may under-emphasize a channel that remains poorly reconstructed. We therefore compute an independent reconstruction loss for each channel $c \in \{x, y, z\}$:

$$\mathcal{L}_c = \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} |\widehat{x}_{i,c} - x_{i,c}|, \tag{2}$$

where $\mathcal{M}$ is the set of masked valid locations. We then assign each channel an adaptive weight according to its loss relative to the channel-wise mean:

$$\overline{\mathcal{L}} = \frac{1}{3} \sum_{c \in \{x,y,z\}} \mathcal{L}_c, \qquad w_c = \mathrm{sg}\left[\max\left(\left(\frac{\mathcal{L}_c}{\overline{\mathcal{L}} + \epsilon} \right)^2, w_{\min} \right) \right], \tag{3}$$

Here, $\mathrm{sg}[\cdot]$ stops gradients through the adaptive weights, ϵ ensures numerical stability, and $w_{\min} = 0.1$ prevents any channel from being ignored. The weights are recomputed for each mini-batch but treated as constants during backpropagation. The final PMT objective is the normalized weighted loss

$$\mathcal{L}_{\mathrm{PMT}} = \frac{\sum_{c \in \{x,y,z\}} w_c \mathcal{L}_c}{\sum_{c \in \{x,y,z\}} w_c}. \tag{4}$$

This objective emphasizes channels with larger residual errors, encouraging balanced XYZ reconstruction. After pretraining, the decoder is discarded and the PMT encoder produces a compact geometry grid aligned with the VLM's RGB token grid.

3.3 GEOMETRY-PRESERVING FUSION

RGB-derived semantic tokens and PMT geometry tokens encode distinct signals, yet unconstrained fusion does not ensure that the geometry remains separately recoverable. GPF addresses this issue by injecting geometry tokens into a dedicated subspace of the language-model embedding space. Here, geometry preservation refers to avoiding additional distortion of the learned geometry-token structure during expansion and maintaining a linear geometry readout after fusion. Intuitively, the orthonormal basis preserves the geometry-token structure during injection, while leak regularization reduces semantic interference within the reserved subspace.

3.3.1 GEOMETRY SUBSPACE INJECTION

Let $\mathcal{E}_{\mathrm{vis}}$ and $\mathcal{P}_{\mathrm{vis}}$ denote the pretrained vision encoder and visual projector, respectively. The semantic tokens are

$$\mathbf{Z}_{\mathrm{sem}} = \mathcal{P}_{\mathrm{vis}}(\mathcal{E}_{\mathrm{vis}}(I)) \in \mathbb{R}^{D \times T}, \tag{5}$$

where D is the language-model embedding dimension and T is the number of visual tokens. We apply the same pixel-shuffle operation to both the RGB and geometry streams to align their spatial token grids before projection. In parallel, a geometry projector compresses the PMT features $\mathbf{H}_{\text{geo}} = \text{PMT}(\mathbf{X})$ into k-dimensional geometry tokens:

$$\mathbf{Z}_{\text{geo}} = \mathcal{P}_{\text{geo}}\left(\mathbf{H}_{\text{geo}}\right) \in \mathbb{R}^{k \times T}. \tag{6}$$

To embed these tokens into the language-model space, we define a fixed *Orthonormal Geometry Basis*

$$\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_k] \in \mathbb{R}^{D \times k}, \qquad \mathbf{Q}^\top \mathbf{Q} = \mathbf{I}_k. \tag{7}$$

We construct $\mathbf{Q}$ once by applying QR decomposition to a randomly initialized matrix and keep it fixed throughout training. We found that learning the underlying matrix before orthonormalization provides no meaningful performance gain, suggesting that the trainable visual and geometry projectors can effectively adapt to a fixed orthonormal subspace.

Its columns span the reserved geometry subspace $\mathcal{G} = \text{range}(\mathbf{Q})$. GPF expands the geometry tokens through this basis and injects them into the semantic tokens:

$$\boxed{\mathbf{Z}_{\text{fin}} = \mathbf{Q}\mathbf{Z}_{\text{geo}} + \mathbf{Z}_{\text{sem}}}. \tag{8}$$

We apply GPF after the visual projector to avoid additional nonlinear transformation of the injected geometry and limit interference with the pretrained visual representation. Because $\mathbf{Q}$ is orthonormal, this expansion preserves pairwise distances between geometry tokens:

$$\begin{aligned} \left\|\mathbf{Q}\left(\mathbf{z}_{\text{geo},i} - \mathbf{z}_{\text{geo},j}\right)\right\|_2^2 &= \left(\mathbf{z}_{\text{geo},i} - \mathbf{z}_{\text{geo},j}\right)^\top \mathbf{Q}^\top \mathbf{Q}\left(\mathbf{z}_{\text{geo},i} - \mathbf{z}_{\text{geo},j}\right) \\ &= \left\|\mathbf{z}_{\text{geo},i} - \mathbf{z}_{\text{geo},j}\right\|_2^2. \end{aligned} \tag{9}$$

Thus, the orthonormal expansion introduces no additional distortion to the learned geometry-token space. This property alone, however, does not prevent semantic features from occupying the same subspace. Since both pathways contain the same T spatial locations, the fusion also preserves the original visual-token sequence length.

3.3.2 Geometry-Leak Regularization

Projecting the fused tokens back onto the geometry basis gives

$$\begin{aligned} \mathbf{Q}^\top \mathbf{Z}_{\text{fin}} &= \mathbf{Q}^\top \mathbf{Q}\mathbf{Z}_{\text{geo}} + \mathbf{Q}^\top \mathbf{Z}_{\text{sem}} \\ &= \mathbf{Z}_{\text{geo}} + \underbrace{\mathbf{Q}^\top \mathbf{Z}_{\text{sem}}}_{\text{semantic leakage}}. \end{aligned} \tag{10}$$

The second term quantifies the contribution of semantic tokens to the geometry readout and acts as an additive readout error. We therefore define geometry-leak regularization as

$$\boxed{\mathcal{L}_{\text{leak}} = \frac{1}{Tk}\left\|\mathbf{Q}^\top \mathbf{Z}_{\text{sem}}\right\|_F^2 = \frac{1}{Tk}\left\|\mathbf{Q}\mathbf{Q}^\top \mathbf{Z}_{\text{sem}}\right\|_F^2}. \tag{11}$$

Together with the orthonormal expansion, leak regularization encourages the geometry representation to remain linearly recoverable from the fused input tokens through $\mathbf{Q}^\top \mathbf{Z}_{\text{fin}} \approx \mathbf{Z}_{\text{geo}}$.

3.4 Training Objective

Training proceeds in two stages. We first pretrain PMT with $\mathcal{L}_{\mathrm{PMT}}$ in equation 4. We then initialize the geometry pathway with the pretrained PMT and optimize the VLM for spatial instruction following. The downstream objective combines the standard autoregressive language-model loss with geometry-leak regularization:

$$\mathcal{L} = \mathcal{L}_{\mathrm{LM}} + \lambda_{\mathrm{leak}} \mathcal{L}_{\mathrm{leak}}. \tag{12}$$

The basis $\mathbf{Q}$, pretrained vision encoder, and pretrained PMT remain fixed, while the visual projector, geometry projector, and language model are jointly optimized during downstream training.

4 MD^{2}Bench

4.1 Motivation and Scope

Recent spatial benchmarks have expanded beyond qualitative 2D and 3D spatial relations (Tong et al., 2024) to evaluate fine-grained metric properties such as depth and distance (Li et al., 2026b; Yang et al., 2025a; Zhang et al., 2025). Some protocols additionally provide explicit geometry, such as depth or point maps, but generating the geometric inputs and reference targets from different sources can make it unclear whether an error reflects a failure to interpret the supplied geometry or a reference–input mismatch. To avoid this confound, we introduce MD^{2}Bench (*Metric Depth and Distance Benchmark*), which evaluates metric readout and distance reasoning over supplied 3D geometry rather than monocular depth estimation or agreement with sensor ground truth. Its geometric inputs and reference answers are derived from the same estimated metric geometry, isolating the model's ability to use the supplied 3D information. This design isolates geometry reasoning from geometry estimation, enabling direct evaluation of whether a VLM can faithfully interpret and reason over the provided metric geometry.

4.2 Benchmark Construction and Tasks

MD^{2}Bench is constructed from 1,000 sampled SUN RGB-D images (Song et al., 2015). For each annotated object, we compute the centroid of its 2D polygon annotation and obtain the corresponding 3D coordinate from an image-aligned XYZ point map constructed using estimated metric depth and camera intrinsics. This 3D point serves as the object's reference location, and the resulting targets are rounded to the nearest 0.1 m.

For an object reference coordinate $\mathbf{p} = (X, Y, Z)$ and a pair of coordinates $\mathbf{p}_1, \mathbf{p}_2$, the three tasks are defined as

$$\begin{aligned} \mathit{Depth}(\mathbf{p}) &= Z, \\ \mathit{Dist}_{\mathrm{OC}}(\mathbf{p}) &= \|\mathbf{p}\|_2, \\ \mathit{Dist}_{\mathrm{OO}}(\mathbf{p}_1, \mathbf{p}_2) &= \|\mathbf{p}_1 - \mathbf{p}_2\|_2. \end{aligned} \tag{13}$$

These quantities represent object depth, object–camera distance, and object–object distance in meters; each task contains 1,000 samples per protocol, yielding six evaluation cells.

4.3 Referent Protocols and Evaluation

MD^{2}Bench specifies queried objects through two protocols. In the **visual-point protocol**, the prompt names the queried object or objects and additionally marks their locations in the image. In the **text-name protocol**, the prompt specifies the queried object or objects by name without visual points.

For a prediction $\widehat{y}_n$ and reference target y_n, let $e_n = |\widehat{y}_n - y_n|/y_n$ denote the relative error. Following SPAR-Bench (Zhang et al., 2025), we use the same set $\mathcal{T}$ of relative-error tolerances and define mean relative accuracy (MRA) as

$$\mathrm{MRA} = \frac{100}{N|\mathcal{T}|} \sum_{n=1}^{N} \sum_{\tau \in \mathcal{T}} \mathbb{1}[e_n \le \tau], \tag{14}$$

where N is the number of samples and $\mathbb{1}[\cdot]$ is the indicator function. The overall MD^{2}Bench score is the unweighted macro average of the six cell-level MRA scores.

Model	Size	Additional Geometry	CV-Bench	SP-Bench-SI	SPAR-Bench
Proprietary Models					
GPT-4o	-	-	75.4	42.4	36.4
Gemini-2.5 Flash	-	-	74.0	-	-
Gemini-2.5 Pro	-	-	85.4	-	-
Cosmos 3	-	-	88.0	-	54.9
Open-source General VLMs					
Qwen3-VL Bai et al. (2025)	2B	-	78.0	52.0	33.7
Qwen3-VL	4B	-	85.5	50.0	35.9
Qwen3-VL	8B	-	86.2	52.3	40.3
InternVL3.5 Wang et al. (2025e)	1B	-	67.2	48.1	32.1
InternVL3.5	4B	-	79.0	52.2	33.2
InternVL3.5	8B	-	82.0	53.3	37.6
Spatial VLMs					
SDVLM Chen et al. (2025)	7B	Rel. Depth	47.7	48.7	27.6
SpaceR Ouyang et al. (2025)	7B	-	75.3	48.6	37.6
VILASR Wu et al. (2025b)	7B	-	77.1	50.2	37.6
SpatialMLLM Wu et al. (2025a)	4B	VGGT(1B)	73.8	43.7	35.1
VG-LLM Zheng et al. (2025a)	8B	VGGT(1B)	81.7	-	-
SpaceMind Zhao et al. (2026)	8B	VGGT(1B)	-	59.7	-
SpatialLadder Li et al. (2026b)	3B	-	73.7	70.2	34.4
3DThinker Chen et al. (2026)	3B	VGGT(1B)	78.4	60.2	58.2
3DThinker	7B	VGGT(1B)	81.1	68.3	63.3
SpatialStack Zhang et al. (2026a)	5B	VGGT(1B)	85.5	-	71.4
GeoThinker Li et al. (2026a)	8B	VGGT(1B)	85.1	-	68.2
In3D-VLM	1B	XYZ+PMT(52M)	**89.7**	71.6	69.0
In3D-VLM	4B	XYZ+PMT(52M)	89.3	**76.8**	**71.8**

Table 1: **Comparison on spatial-reasoning benchmarks**. The *Additional Geometry* column lists any geometric input or backbone used beyond RGB, with its parameter count.

5 Experiments

5.1 Experimental Details

Implementation Details. We build In3D-VLM on InternVL3.5-1B and InternVL3.5-4B (Wang et al., 2025e), sharing the same pretrained PMT across model sizes. XYZ point maps are constructed by back-projecting estimated metric depth using the corresponding camera intrinsics. We apply pixel shuffle to both the RGB and geometry streams and implement the geometry projector $\mathcal{P}_{\text{geo}}$ as a single bias-free linear layer. By default, GPF is applied after the RGB visual projector (post-projector fusion), in the LLM embedding space, with geometry dimension $k = 512$ and leak coefficient $\lambda_{\text{leak}} = 0.1$. Training proceeds in two stages. Stage 1 pretrains PMT on approximately 0.96M RGB–XYZ pairs, and Stage 2 performs spatial instruction tuning using 1.4M collected QA pairs. Stage 2 performs full-parameter instruction tuning without LoRA, while the vision encoder and PMT remain frozen. All models are trained on eight NVIDIA H100 GPUs. Additional architecture, data, and optimization details are provided in appendix.

Baselines and Evaluation. Our RGB-only SFT baseline uses the same pretrained VLM, instruction-tuning data, and Stage 2 training settings as In3D-VLM. To compare geometry-injection strategies, we additionally adapt the injection mechanisms of SD-VLM (Chen et al., 2025) and N3D-VLM (Wang et al., 2025f) to our base VLM under the same instruction-tuning setup. These variants evaluate the respective injection mechanisms within a shared framework. We evaluate In3D-VLM on CV-Bench, SP-Bench-SI and SPAR-Bench, using their official evaluation code. Comparisons on public benchmarks include both locally evaluated results and scores reported in the original papers. We additionally evaluate metric geometry readout on MD^{2}Bench under its visual-point and text-name protocols, reporting task-level MRA and the unweighted average over the six task–protocol combinations.

5.2 Results

Public Benchmarks. Table 1 demonstrates that In3D-VLM achieves strong spatial-reasoning performance with compact VLM backbones. The 1B variant attains the highest CV-Bench score among the listed models at 89.7, exceeding SpatialStack by 4.2 points. The 4B variant further establishes the

Model	Overall	Text Prompt			Visual Point Prompt		
		Depth	Distance OC	Distance OO	Depth	Distance OC	Distance OO
SD-VLM-7B	22.9	6.5	25.5	34.8	6.2	28.9	35.2
SpaceR-7B	22.5	19.4	19.4	28.8	19.4	18.8	29.2
SpatialStack	64.4	66.8	66.2	40.6	76.4	74.9	62.0
SDVLM*	64.0	56.4	64.5	38.0	77.3	75.6	72.3
N3DVLM*	66.4	49.0	65.9	**42.3**	83.0	82.3	76.1
Our Models							
In3D-VLM-1B	68.9	58.3	67.3	40.7	**85.1**	81.1	81.3
In3D-VLM-4B	**70.6**	**67.5**	**68.2**	38.9	82.7	**84.3**	**81.8**

Table 2: **Comparison on MD^{2}Bench.** OC and OO denote object–camera and object–object distances, respectively. Models marked with * are our reproductions, obtained by applying each method's depth-injection strategy to the same base VLM used in In3D-VLM.

Representation	X MAE	Y MAE	Z MAE	Mean MAE	Pair-distance MAE
Avg. pooling	0.18439	0.14768	0.26958	0.20055	0.41244
Bilinear resize	0.18950	0.15203	0.37657	0.23937	0.44503
PMT (Ours)	**0.15948**	**0.12955**	**0.20933**	**0.16612**	**0.34454**

Table 3: **Representation quality.** Reconstruction errors (meters, lower is better) on 1,000 held-out SUN RGB-D images. A fixed orthonormal projection is used to match the channel dimensions across representations. Identical convolutional decoders are trained separately on the frozen representations under matched conditions. Pair-distance MAE measures inter-point distance reconstruction error.

competitiveness of the framework, achieving the highest listed scores on SP-Bench-SI and SPAR-Bench at 76.8 and 71.8, respectively. Its 6.6-point advantage over SpatialLadder on SP-Bench-SI highlights the effectiveness of the approach across the spatial tasks covered by this benchmark. Together, these results show that explicit geometry integration enables compact In3D-VLM variants to compete favorably with larger spatial VLMs. The framework also benefits from a larger language backbone while reusing the same pretrained PMT. Moving from 1B to 4B improves SP-Bench-SI and SPAR-Bench by 5.2 and 2.8 points, respectively. These results demonstrate that the shared geometric representation can support effective spatial reasoning across different backbone sizes.

MD^{2}Bench. Table 2 directly evaluates the ability to read metric coordinates and compute distances from supplied geometry. In3D-VLM achieves the highest overall scores among the listed methods, reaching 68.9 with the 1B backbone and 70.6 with the 4B backbone. Under the shared-backbone evaluation, the 1B variant exceeds the adapted SD-VLM and N3D-VLM injection baselines by 4.9 and 2.5 points, respectively. Because these comparisons apply different geometry-injection mechanisms within the same base VLM and instruction-tuning setup, the gains support the effectiveness of the proposed integration design. The results further demonstrate strong performance on relational metric computation. Under visual-point prompting, the 1B model achieves 81.3 on object–object distance, exceeding the adapted N3D-VLM baseline by 5.2 points. This task requires combining the coordinates of two queried locations, complementing the evaluation of individual depth and camera-distance readout. The 4B variant achieves the highest scores in four of the six task–protocol combinations, covering both text-name and visual-point prompting. Together, these findings demonstrate the effectiveness of In3D-VLM in using supplied geometry for both pointwise metric readout and inter-object distance computation.

5.3 Ablation Study

Representation Quality of PMT We first evaluate whether PMT preserves metric geometry more effectively than direct spatial downsampling. Existing geometry-augmented VLMs often use pooling or resizing to align dense geometric inputs with visual-token grids (Zheng et al., 2025b; Wang et al., 2025f). We compare PMT against average pooling and bilinear resizing by freezing each representation and training identical convolutional decoders under matched conditions. Table 3 reports

Geo. Input	Fusion	MD^2 VP	MD^2 TN	CV	SPAR
Baseline (InternVL3.5-1B)		15.2	19.4	67.2	32.1
SFT (RGB-only)		69.3	50.8	86.6	63.8
PE	Add	76.6	52.8	56.3	64.4
PMT	Add	81.8	51.3	85.8	**69.4**
PMT	**GPF**	**82.5**	**55.4**	**89.7**	69.0

(a) Component ablation.

GPF Variant	MD^2	CV	SPAR	Occ. /R-Err. ↓
RGB only	60.1	86.6	63.8	-
+ PMT, Geo. projector	68.4	83.6	66.0	-
+ Orthonormal	67.9	84.4	66.6	50.3/0.55
+ Leak penalty	**68.9**	**89.7**	**69.0**	**15.6 / 0.16**
Pre-projector fusion	68.8	87.0	66.7	-

(b) GPF ablation.

Table 4: **Ablation studies of In3D-VLM.** VP and TN denote visual-point and text-name prompts, respectively. PE denotes positional encoding, PMT denotes the Point-Map Tokenizer, and GPF denotes Geometry-Preserving Fusion. Pre-projector fusion applies geometry fusion before the RGB visual projector.

results on 1,000 held-out SUN RGB-D images. PMT achieves the lowest MAE across all XYZ coordinates, reducing the mean MAE to 0.16612 from 0.20055 and 0.23937 for average pooling and bilinear resizing, respectively. It also achieves the lowest pair-distance MAE (0.34454), indicating better preservation of both absolute coordinates and relative metric structure after compression. Qualitative results are provided in the appendix.

Component Ablation Table 4a studies the effects of geometry representation and fusion on downstream spatial reasoning. While PE with additive fusion substantially improves MD^{2}Bench, it degrades CV-Bench from 86.6 to 56.3, suggesting interference with visual-semantic representations. Replacing PE with PMT under the same additive fusion improves MD^{2}Bench visual-point MRA from 76.6 to 81.8, CV-Bench from 56.3 to 85.8, and SPAR-Bench from 64.4 to 69.4. Combining PMT with GPF further improves MD^{2}Bench to 82.5/55.4 and achieves 89.7 on CV-Bench while maintaining strong SPAR-Bench performance. CV-Bench mainly evaluates general spatial understanding, such as object recognition and relative spatial relations, where visual-semantic representations play an important role, whereas MD^{2}Bench focuses on metric spatial reasoning that more directly depends on geometric information. The consistent gains across both benchmarks suggest that GPF reduces interference between geometry and semantic pathways, improving geometry-heavy metric reasoning without sacrificing general spatial understanding.

GPF Ablation Table 4b isolates the contribution of each GPF component. Adding the geometry projector improves MD^{2}Bench from 60.1 to 68.4 but reduces CV-Bench from 86.6 to 83.6. Introducing the orthonormal basis partially recovers CV-Bench and SPAR-Bench performance. With geometry-leak regularization, performance reaches 68.9 on MD^{2}Bench, 89.7 on CV-Bench, and 69.0 on SPAR-Bench. Semantic subspace occupancy decreases from 50.3% to 15.6%, while relative geometry-readout error drops from 0.55 to 0.16, supporting reduced semantic interference in the reserved geometry subspace. Post-projector fusion also outperforms pre-projector fusion across all three benchmarks, consistent with our design choice to inject geometry after visual projection and thereby avoid subsequent deformation of the geometry subspace.

6 Conclusion

We presented In3D-VLM, a framework for injecting explicit metric 3D space into VLMs while keeping geometry compact and recoverable after fusion. PMT compresses dense XYZ point maps through masked reconstruction pretraining, while GPF injects them into a reserved orthonormal subspace with leak regularization for linear geometry readout. We further introduced MD^{2}Bench, which derives targets from the supplied geometry to isolate metric reasoning. Across CV-Bench, SP-Bench-SI, SPAR-Bench, and MD^{2}Bench, In3D-VLM competes favorably with substantially larger spatial VLMs using compact 1B and 4B backbones.

Limitations. Our experiments primarily use geometry from a single metric depth estimator, and MD^{2}Bench focuses on single-view depth and distance reasoning. Evaluating robustness across sensor depth, different geometry estimators, and multi-view or temporal settings remains future work.

AI USE STATEMENT

We used generative AI tools to assist with drafting and polishing the manuscript, including improving clarity, grammar, and organization, and to support literature retrieval and the identification of relevant prior work. The authors reviewed and revised the AI-assisted text and verified the suggested references and associated claims against the original sources. We take full responsibility for the final content of this paper, including all text, claims, and citations produced with AI assistance.

REFERENCES

Daichi Azuma, Taiki Miyanishi, Shuhei Kurita, and Motoaki Kawanabe. Scanqa: 3d question answering for spatial scene understanding. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19107–19117. IEEE, 2022.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14455–14465, 2024.

Pingyi Chen, Yujing Lou, Shen Cao, Jinhui Guo, Lubin Fan, Yue Wu, Lin Yang, Lizhuang Ma, and Jieping Ye. Sd-vlm: Spatial measuring and understanding with depth-encoded vision-language models. In *NeurIPS*, 2025.

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, 2026.

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.

An-Chieh Cheng, Yang Fu, Yukang Chen, Zhijian Liu, Xiaolong Li, Subhashree Radhakrishnan, Song Han, Yao Lu, Jan Kautz, Pavlo Molchanov, et al. 3d aware region prompted vision language model. In *International Conference on Learning Representations*, volume 2026, pp. 123286–123308, 2026.

Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 2432–2443. IEEE, 2017.

Erik Daxberger, Nina Wenzel, David Griffiths, Haiming Gang, Justin Lazarow, Gefen Kohavi, Kai Kang, Marcin Eichner, Yinfei Yang, Afshin Dehghan, et al. Mm-spatial: Exploring 3d spatial understanding in multimodal llms. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7395–7408. IEEE, 2025.

Nianchen Deng, Lixin Gu, Shenglong Ye, Yinan He, Zhe Chen, Songze Li, Haomin Wang, Jinhui Yin, Qi Wei, Tianshuo Yang, et al. Internspatial: A comprehensive dataset for spatial reasoning in vision-language models. In *International Conference on Learning Representations*, volume 2026, pp. 50773–50814, 2026.

Mengfei Du, Binhao Wu, Zejun Li, Xuan-Jing Huang, and Zhongyu Wei. Embspatial-bench: Benchmarking spatial understanding for embodied tasks with large vision-language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 346–355, 2024.

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.

Zhi Helu, Huang Jingjing, Xu Wang, Xu Yangbin, Zhang Wanyue, Jiang Baoyang, Deng Shirui, Zhu Liang, Li Fangfang, Zhao Tiejun, et al. Scaling spatial reasoning in mllms through programmatic data synthesis. *arXiv preprint arXiv:2512.16237*, 2025.

Wenbo Hu, Jingli Lin, Yilin Long, Yunlong Ran, Lihan Jiang, Yifan Wang, Chenming Zhu, Runsen Xu, Tai Wang, and Jiangmiao Pang. G^2 vlm: Geometry grounded vision language model with unified 3d reconstruction and spatial reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9535–9546, 2026.

Baoxiong Jia, Yixin Chen, Huangyue Yu, Yan Wang, Xuesong Niu, Tengyu Liu, Qing Li, and Siyuan Huang. Sceneverse: Scaling 3d vision-language learning for grounded scene understanding. In *European Conference on Computer Vision (ECCV)*, 2024.

Justin Lazarow, David Griffiths, Gefen Kohavi, Francisco Crespo, and Afshin Dehghan. Cubify anything: Scaling indoor 3d object detection. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22225–22233. IEEE, 2025.

Haoyuan Li, Qihang Cao, Tao Tang, Kun Xiang, Zihan Guo, Jianhua Han, JiaWang Bian, Hang Xu, and Xiaodan Liang. Thinking with geometry: Active geometry integration for spatial reasoning. *arXiv preprint arXiv:2602.06037*, 2026a.

Hongxing Li, Dingming Li, Zixuan Wang, Yuchen Yan, Hang Wu, Wenqi Zhang, Yongliang Shen, Weiming Lu, Jun Xiao, and Yueting Zhuang. Spatialladder: Progressive training for spatial reasoning in vision-language models. In *International Conference on Learning Representations*, volume 2026, pp. 76566–76592, 2026b.

Peizheng Li, Zhenghao Zhang, David Holtz, Hang Yu, Yutong Yang, Yuzhi Lai, Rui Song, Andreas Geiger, and Andreas Zell. Spacedrive: Infusing spatial awareness into vlm-based autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 40096–40107, 2026c.

Huizhi Liang, Yichao Shen, Yu Deng, Sicheng Xu, Zhiyuan Feng, Tong Zhang, Yaobo Liang, and Jiaolong Yang. Hispatial: Taming hierarchical 3d spatial understanding in vision-language models. *arXiv preprint arXiv:2603.25411*, 2026.

Yuan-Hong Liao, Rafid Mahmood, Sanja Fidler, and David Acuna. Reasoning paths with reference objects elicit quantitative spatial reasoning in large vision-language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 17028–17047, 2024.

Ruiyuan Lyu, Jingli Lin, Tai Wang, Shuai Yang, Xiaohan Mao, Yilun Chen, Runsen Xu, Haifeng Huang, Chenming Zhu, Dahua Lin, et al. Mmscan: A multi-modal 3d scene dataset with hierarchical grounded language annotations. *Advances in Neural Information Processing Systems*, 37: 50898–50924, 2024.

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

Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, Karmesh Yadav, Qiyang Li, Ben Newman, Mohit Sharma, Vincent Berges, Shiqi Zhang, Pulkit Agrawal, Yonatan Bisk, Dhruv Batra, Mrinal Kalakrishnan, Franziska Meier, Chris Paxton, Sasha Sax, and Aravind Rajeswaran. Openeqa: Embodied question answering in the era of foundation models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

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.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

Chan Hee Song, Valts Blukis, Jonathan Tremblay, Stephen Tyree, Yu Su, and Stan Birchfield. Robospatial: Teaching spatial understanding to 2d and 3d vision-language models for robotics. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15768–15780. IEEE, 2025.

Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 567–576, 2015.

Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Zhiyong Zhao, Yang Wang, Kun Zhan, Peng Jia, Xianpeng Lang, and Hang Zhao. Drivevlm: The convergence of autonomous driving and large vision-language models. *arXiv preprint arXiv:2402.12289*, 2024.

Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai C Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *Advances in Neural Information Processing Systems*, 37:87310–87356, 2024.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

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, 2025a.

Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10510–10522. IEEE, 2025b.

Ruicheng Wang, Sicheng Xu, Yue Dong, Yu Deng, Jianfeng Xiang, Zelong Lv, Guangzhong Sun, Xin Tong, and Jiaolong Yang. MoGe-2: Accurate monocular geometry with metric scale and sharp details, 2025c. URL https://arxiv.org/abs/2507.02546.

Shihao Wang, Zhiding Yu, Xiaohui Jiang, Shiyi Lan, Min Shi, Nadine Chang, Jan Kautz, Ying Li, and Jose M Alvarez. Omnidrive: A holistic vision-language dataset for autonomous driving with counterfactual reasoning. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22442–22452. IEEE, 2025d.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025e.

Yuxin Wang, Lei Ke, Boqiang Zhang, Tianyuan Qu, Hanxun Yu, Zhenpeng Huang, Meng Yu, Dan Xu, and Dong Yu. N3d-vlm: Native 3d grounding enables accurate spatial reasoning in vision-language models. *arXiv preprint arXiv:2512.16561*, 2025f.

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*, 2025a.

Haoning Wu, Xiao Huang, Yaohui Chen, Ya Zhang, Yanfeng Wang, and Weidi Xie. Spatialscore: Towards comprehensive evaluation for spatial intelligence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 31029–31041, 2026.

Junfei Wu, Jian Guan, Kaituo Feng, Qiang Liu, Shu Wu, Liang Wang, Wei Wu, and Tieniu Tan. Reinforcing spatial reasoning in vision-language models with interwoven thinking and visual drawing, 2025b. URL https://arxiv.org/abs/2506.09965.

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 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10632–10643, 2025a.

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. *arXiv preprint arXiv:2511.05491*, 2025b.

Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12–22. IEEE, 2023.

Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. *arXiv preprint arXiv:2406.10721*, 2024.

Jiahui Zhang, Yurui Chen, Yanpeng Zhou, Yueming Xu, Ze Huang, Jilin Mei, Junhui Chen, Yujie 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. *arXiv preprint arXiv:2603.27437*, 2026a.

Shihua Zhang, Qiuhong Shen, Shizun Wang, Tianbo Pan, and Xinchao Wang. Make geometry matter for spatial reasoning. *arXiv preprint arXiv:2603.26639*, 2026b.

Ruosen Zhao, Zhikang Zhang, Jialei Xu, Jiahao Chang, Dong Chen, Lingyun Li, Weijian Sun, and Zizhuang Wei. Spacemind: Camera-guided modality fusion for spatial reasoning in vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16811–16822, June 2026.

Duo Zheng, Shijia Huang, Yanyang Li, and Liwei Wang. Learning from videos for 3d world: Enhancing mllms with 3d vision geometry priors. *arXiv preprint arXiv:2505.24625*, 2025a.

Duo Zheng, Shijia Huang, and Liwei Wang. Video-3d llm: Learning position-aware video representation for 3d scene understanding. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8995–9006. IEEE, 2025b.

Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. Structured3d: A large photo-realistic dataset for structured 3d modeling. In *European Conference on Computer Vision*, pp. 519–535. Springer, 2020.

A APPENDIX

CONTENTS

A.1 Additional Analysis of Geometry-Preserving Fusion 14
A.2 Implementation Details 15
A.2.1 Training setup 15
A.2.2 Training Data 16
A.2.3 PMT Architecture and Pretraining 16
A.3 MD^{2}Bench Construction 17
A.4 Effect of Adaptive Channel Weighting 19
A.5 Point-Map Reconstruction Analysis 19
A.6 Effect of Geometry Subspace Size 21

A.1 ADDITIONAL ANALYSIS OF GEOMETRY-PRESERVING FUSION

This section provides additional analysis of the explicit geometry readout induced by GPF. We use the notation introduced in Section 3.3.

Leak regularization as readout-error minimization. Define the geometry-code readout error as

$$\mathbf{E}_{\text{geo}} \triangleq \mathbf{Q}^\top \mathbf{Z}_{\text{fin}} - \mathbf{Z}_{\text{geo}}. \tag{15}$$

From equation 10, this error is exactly

$$\boxed{\mathbf{E}_{\text{geo}} = \mathbf{Q}^\top \mathbf{Z}_{\text{sem}}}. \tag{16}$$

Consequently, the leak objective in equation 11 directly minimizes the squared geometry-code readout error:

$$\mathcal{L}_{\text{leak}} = \frac{1}{Tk} \left\| \mathbf{E}_{\text{geo}} \right\|_F^2. \tag{17}$$

For an individual semantic token $\mathbf{z}_{\text{sem}} \in \mathbb{R}^D$, let

$$\mathbf{M}_{\text{sem}} = \mathbb{E}\left[\mathbf{z}_{\text{sem}} \mathbf{z}_{\text{sem}}^\top\right] \tag{18}$$

denote its second-moment matrix. The expected squared readout error is then

$$\begin{aligned} \mathbb{E}\left[\left\|\mathbf{Q}^\top \mathbf{z}_{\text{sem}}\right\|_2^2\right] &= \mathbb{E}\left[\operatorname{tr}\left(\mathbf{Q}^\top \mathbf{z}_{\text{sem}} \mathbf{z}_{\text{sem}}^\top \mathbf{Q}\right)\right] \\ &= \operatorname{tr}\left(\mathbf{Q}^\top \mathbf{M}_{\text{sem}} \mathbf{Q}\right). \end{aligned} \tag{19}$$

Thus, the empirical leak objective estimates the expected semantic interference in the geometry readout.

Orthogonal decomposition. Let

$$\mathbf{P}_{\mathcal{G}} = \mathbf{Q}\mathbf{Q}^\top \tag{20}$$

be the orthogonal projector onto the reserved geometry subspace $\mathcal{G} = \operatorname{range}(\mathbf{Q})$. Projecting the fused tokens onto $\mathcal{G}$ and its orthogonal complement gives

$$\mathbf{P}_{\mathcal{G}} \mathbf{Z}_{\text{fin}} = \mathbf{Q}\mathbf{Z}_{\text{geo}} + \mathbf{P}_{\mathcal{G}} \mathbf{Z}_{\text{sem}}, \tag{21}$$

$$\left(\mathbf{I} - \mathbf{P}_{\mathcal{G}}\right) \mathbf{Z}_{\text{fin}} = \left(\mathbf{I} - \mathbf{P}_{\mathcal{G}}\right) \mathbf{Z}_{\text{sem}}. \tag{22}$$

If $\mathbf{Q}^\top \mathbf{Z}_{\text{sem}} = \mathbf{0}$, then $\mathbf{P}_{\mathcal{G}} \mathbf{Z}_{\text{sem}} = \mathbf{0}$, and the decomposition becomes

$$\mathbf{P}_{\mathcal{G}} \mathbf{Z}_{\text{fin}} = \mathbf{Q}\mathbf{Z}_{\text{geo}}, \qquad \left(\mathbf{I} - \mathbf{P}_{\mathcal{G}}\right) \mathbf{Z}_{\text{fin}} = \mathbf{Z}_{\text{sem}}. \tag{23}$$

Therefore, minimizing semantic leakage encourages the fused tokens toward an orthogonal decomposition of geometry and semantic components.

Setting	Stage 1 : PMT Pretraining	Stage 2 : Instruction tuning
Epochs	10	1
Learning rate	1×10^{-4}	2×10^{-5}
Per-GPU batch size	32	8
Gradient accumulation	1	4
Global batch size	256	256
Input resolution	XYZ: 512^2	RGB: 448^2, XYZ: 512^2

Table 5: **Stage-specific training hyperparameters.**

Role of orthonormality. Consider instead a full-column-rank but non-orthogonal basis $\mathbf{A} \in \mathbb{R}^{D\times k}$. For

$$\widetilde{\mathbf{Z}}_{\text{fin}} = \mathbf{A}\mathbf{Z}_{\text{geo}} + \mathbf{Z}_{\text{sem}}, \tag{24}$$

direct transpose-based readout gives

$$\mathbf{A}^\top \widetilde{\mathbf{Z}}_{\text{fin}} = \underbrace{\mathbf{A}^\top \mathbf{A}}_{\text{Gram matrix}} \mathbf{Z}_{\text{geo}} + \mathbf{A}^\top \mathbf{Z}_{\text{sem}}. \tag{25}$$

Thus, the geometry code is additionally transformed by $\mathbf{A}^\top\mathbf{A}$. Although a general full-rank basis can be inverted using its pseudoinverse, an orthonormal basis satisfies

$$\mathbf{Q}^\dagger = \mathbf{Q}^\top, \qquad \mathbf{Q}^\top\mathbf{Q} = \mathbf{I}_k, \tag{26}$$

making the transpose itself an exact, unit-conditioned readout. Therefore, orthonormal expansion avoids Gram-matrix correction and introduces no additional distortion to the learned geometry-token space.

Subspace diagnostics. The GPF ablation in Table 4b reports two diagnostics: semantic subspace occupancy (Occ.) and relative geometry-readout error (R-Err.). We define semantic occupancy as the percentage of semantic-token energy contained in the reserved geometry subspace:

$$\text{Occ.} = \frac{\left\|\mathbf{Q}^\top \mathbf{Z}_{\text{sem}}\right\|_F^2}{\left\|\mathbf{Z}_{\text{sem}}\right\|_F^2} \times 100. \tag{27}$$

We define the relative geometry-readout error as

$$\begin{aligned} \text{R-Err.} &= \frac{\left\|\mathbf{Q}^\top \mathbf{Z}_{\text{fin}} - \mathbf{Z}_{\text{geo}}\right\|_F}{\left\|\mathbf{Z}_{\text{geo}}\right\|_F} \\ &= \frac{\left\|\mathbf{Q}^\top \mathbf{Z}_{\text{sem}}\right\|_F}{\left\|\mathbf{Z}_{\text{geo}}\right\|_F}, \end{aligned} \tag{28}$$

where the second equality follows from equation 16. Lower values of both metrics indicate less semantic interference and more accurate recovery of the geometry codes from the fused tokens.

A.2 Implementation Details

A.2.1 Training setup

Stage 1 pretrains the PMT encoder and decoder on 0.96M RGB–XYZ pairs using masked XYZ reconstruction with $r_{\text{mask}} = 0.5$. Stage 2 uses 1.4M collected QA pairs for spatial instruction tuning. During Stage 2, the vision encoder, PMT, and fixed orthonormal basis $\mathbf{Q}$ remain frozen, while the visual projector, geometry projector, and language model are jointly optimized without LoRA. We use $k = 512$ and $\lambda_{\text{leak}} = 0.1$.

Both stages use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, cosine learning-rate decay, a 3% warmup ratio, a weight decay of 0.05, and a maximum gradient norm of 1.0. Training uses bfloat16 precision, DeepSpeed ZeRO-1, and a random seed of 42 on eight NVIDIA H100 GPUs.

Dataset	Samples	Ratio
Stage 1: PMT pretraining		
CA-1M Lazarow et al. (2025)	535K	56.0%
ScanNet Dai et al. (2017)	248K	26.0%
Structured3D Zheng et al. (2020)	74K	7.7%
ScanNet++ Yeshwanth et al. (2023)	64K	6.7%
SPRITE Helu et al. (2025)	34K	3.6%
Total	**0.96M**	**100%**
Stage 2: instruction tuning		
SPAR-7M-RGBD-Custom	336K	24.0%
SpatialCorpus Wu et al. (2026)	331K	23.6%
SPAR-7M-RGBD Zhang et al. (2025)	302K	21.5%
VLM-3R Fan et al. (2026)	163K	11.6%
CAVQA Daxberger et al. (2025)	130K	9.3%
InternSpatial Deng et al. (2026)	120K	8.6%
SpatialLadder-26k Li et al. (2026b)	20K	1.4%
Total	**1.40M**	**100%**

Table 6: **Training-data composition.**

A.2.2 Training Data

Table 6 summarizes the data used in the two training stages. Stage 1 uses approximately 0.96M unique RGB–XYZ point-map pairs collected from five data sources to pretrain PMT. For consistent geometry construction, we apply MoGe-2 (Wang et al., 2025c) to every image to estimate metric depth and camera intrinsics. The estimated depth is then back-projected using equation 1 to construct an image-aligned XYZ point map.

Stage 2 uses approximately 1.40M spatial instruction-tuning samples drawn from seven sources. Each training sample is paired with an XYZ point map generated through the same MoGe-2 preprocessing pipeline used in Stage 1. SPAR-7M-RGBD-Custom denotes our relabeled version of SPAR-7M-RGBD, in which metric targets are represented at $0.1\,\mathrm{cm}$ precision to provide finer-grained metric supervision.

A.2.3 PMT Architecture and Pretraining

Encoder architecture. As illustrated in Fig. 2, PMT receives an XYZ point map $\mathbf{X} \in \mathbb{R}^{3\times512\times512}$. The convolutional stem consists of two convolutional layers and produces a $32\times256\times256$ feature map. The first downsampling block reduces it to $64\times128\times128$, after which feature-space masking is applied.

Two additional downsampling blocks produce feature maps of sizes $128\times64\times64$ and $256\times32\times32$, respectively. Each downsampling block consists of a stride-2 3×3 convolution, layer normalization, GELU, and one or more lightweight convolutional blocks. Each convolutional block uses a 7×7 depthwise convolution followed by layer normalization and two 1×1 pointwise convolutions with an expansion ratio of two. The first two downsampling blocks contain one convolutional block, while the final block contains two.

A 1×1 convolution projects the resulting 256-dimensional features to the encoder dimension $D_{\mathrm{enc}} = 1024$. The 32×32 feature grid is flattened into $T = 1024$ tokens, and a fixed two-dimensional sine–cosine positional embedding is added. The tokens are then processed by four Transformer blocks with eight attention heads and an MLP expansion ratio of four. The final encoder output is

$$\mathbf{H}_{\mathrm{geo}} \in \mathbb{R}^{1024\times1024}. \tag{29}$$

Feature-space masking. We apply block-wise masking to the intermediate feature map $\mathbf{F} \in \mathbb{R}^{64\times128\times128}$, rather than directly to the input XYZ map. The spatial dimensions are partitioned into non-overlapping 8×8 regions, resulting in a 16×16 grid of 256 feature blocks. For each training sample, we independently generate a random spatial mask with a masking ratio of $r_{\mathrm{mask}} = 0.5$.

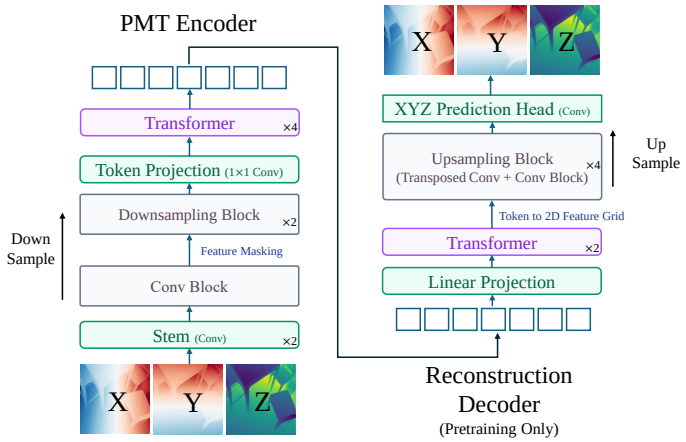


Figure 2: **Architecture used for PMT pretraining.** The encoder combines convolutional downsampling with Transformer blocks to produce geometry tokens. A lightweight reconstruction decoder is used only during pretraining and is discarded when integrating PMT into the VLM.

Consequently, 128 blocks are retained, while the remaining 128 blocks are replaced with a shared learnable mask token $\mathbf{m} \in \mathbb{R}^{64}$.

Let $\mathbf{M} \in \{0,1\}^{128\times128}$ denote the expanded binary mask, where $\mathbf{M}_{u,v} = 1$ indicates a masked location. The masked feature map is constructed as

$$\widetilde{\mathbf{F}}_{:,u,v} = (1 - \mathbf{M}_{u,v})\,\mathbf{F}_{:,u,v} + \mathbf{M}_{u,v}\mathbf{m}. \tag{30}$$

Each 8×8 feature block corresponds to a 32×32 region in the original 512×512 point map. The binary mask is upsampled to the original resolution using nearest-neighbor interpolation, and the reconstruction loss is evaluated only over masked valid locations. A new random mask is sampled independently for every input at each training iteration.

Reconstruction decoder. The reconstruction decoder first projects each 1024-dimensional encoder token to 256 dimensions and applies two Transformer blocks with four attention heads. The token sequence is then reshaped into a $256 \times 32 \times 32$ feature grid. Four upsampling blocks progressively reconstruct the original spatial resolution:

$$\begin{aligned} 256 \times 32 \times 32 &\to 128 \times 64 \times 64 \\ &\to 64 \times 128 \times 128 \\ &\to 32 \times 256 \times 256 \\ &\to 16 \times 512 \times 512. \end{aligned} \tag{31}$$

Each upsampling block consists of a stride-2 transposed convolution followed by layer normalization, GELU, and a lightweight convolutional block. Finally, an XYZ prediction head comprising a 3×3 convolution and a 1×1 output projection reconstructs $\widehat{\mathbf{X}} \in \mathbb{R}^{3\times512\times512}$. The decoder is used only during pretraining and is discarded afterward.

A.3 MD^{2}BENCH CONSTRUCTION

Figure 3 summarizes the construction pipeline of MD^{2}Bench, from image selection and geometry generation to object reference extraction and question generation.

Image selection and geometry generation. We randomly sample 1,000 images from SUN RGB-D (Song et al., 2015). For each image, MoGe-2 (Wang et al., 2025c) predicts metric depth and

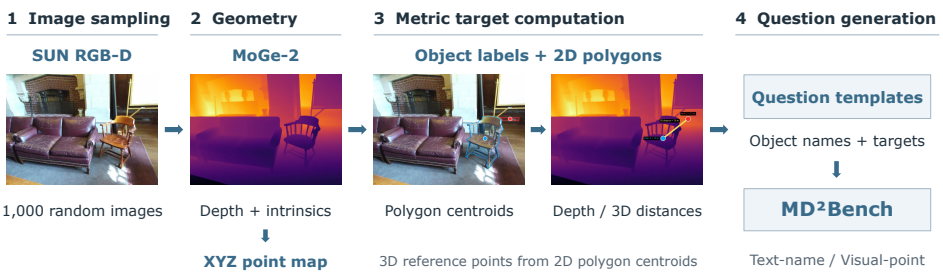


Figure 3: **Construction pipeline of MD²Bench.** We randomly sample 1,000 images from SUN RGB-D and use MoGe-2 to predict metric depth and camera intrinsics, from which we construct image-aligned XYZ point maps. We obtain 3D object reference points by querying these maps at 2D polygon centroids and compute object depth, object–camera distance, and object–object distance. Finally, task-specific templates generate evaluation questions using annotated object names, paired with the computed metric targets, under the text-name and visual-point protocols.

camera intrinsics, which we use to construct an image-aligned XYZ point map following equation 1. The resulting geometry is cached and used both as the source of geometric input and to derive the reference answers, ensuring consistency between the supplied geometry and the evaluation targets.

Object reference points and target computation. We identify objects using the object labels and 2D polygon annotations provided by SUN RGB-D. For each object, we compute the polygon centroid in image coordinates and obtain its corresponding 3D point from the XYZ map. This point serves as the object's geometric reference location for benchmark construction.

We derive the three metric targets according to equation 13. Object depth is the Z coordinate of the reference point, object–camera distance is its Euclidean distance from the camera origin, and object–object distance is the Euclidean distance between two object reference points. All targets are expressed in meters and rounded to the nearest 0.1 m.

Question generation and referent protocols. We adapt the prompt templates from SPAR-Bench (Zhang et al., 2025) to our three metric tasks, populating them with annotated object names and pairing each question with its computed metric target. In the text-name protocol, the queried objects are identified by name in the question, and the RGB image is provided without point markers. The visual-point protocol additionally marks the corresponding object reference locations in the image. Figure 4 presents an example of each task under the visual-point protocol. Representative question templates are provided below



Figure 4: **Examples from MD²Bench.** The examples illustrate object depth, object–camera distance, and object–object distance under the visual-point protocol. Colored points mark the queried object reference locations. In the text-name protocol, the RGB images contain no point markers, and the questions identify the queried objects only by name. All reference answers are expressed in meters.

Task	Example prompt templates
Depth	Based on the photo, what is the depth of {obj} **(red point)** at its center? Report in meters. Calculate or judge based on the 3D center points of these points. Enter a single numeric response and no other details.
	What is the depth of {obj} **(red point)**, measured in meters? Calculate or judge based on the 3D center points of these points. Provide your input as exactly one number.
	From the observer's standpoint, what is the depth of {obj} **(red point)**? Ensure the answer is in meters. Calculate or judge based on the 3D center points of these points. Answer by providing just one numeric value.
Distance OC	What is the observer's measured distance in meters to {obj} **(red point)**? Calculate or judge based on the 3D center points of these points. Submit your response as one numeric value only.
	From the perspective of the person who took the image, how far is {obj} **(red point)**? Provide the distance in meters. Calculate or judge based on the 3D center points of these points. Provide your input as exactly one number.
	Please provide the distance in meters between the observer and {obj} **(red point)** at its center. Calculate or judge based on the 3D center points of these points. The answer should be a single number, nothing more.
Distance OO	What is the linear distance between the center of {obj1} **(red point)** and the center of {obj2} **(blue point)**? State the result in meters. Calculate or judge based on the 3D center points of these points. Ensure your answer contains only one number.
	How far, measured in meters, is the center of {obj1} **(red point)** from the center of {obj2} **(blue point)** in the image? Calculate or judge based on the 3D center points of these points. Provide your input as exactly one number.
	Express the Euclidean distance between the center of {obj1} **(red point)** and {obj2} **(blue point)** in meters. Calculate or judge based on the 3D center points of these points. Submit your response as one numeric value only.

Table 7: **Representative prompt templates for MD^{2}Bench.** Three examples are shown for each task. OC and OO denote object–camera and object–object distances, respectively. Placeholders denote annotated object names. Colored, bold parenthetical phrases refer to image markers and appear only in the visual-point protocol. In these templates, "3D center point" denotes the 3D reference point obtained from the XYZ map at the 2D polygon centroid, rather than the volumetric object center. The text-name protocol identifies the queried objects by name without marker references. All prompts request a single numerical answer in meters.

A.4 Effect of Adaptive Channel Weighting

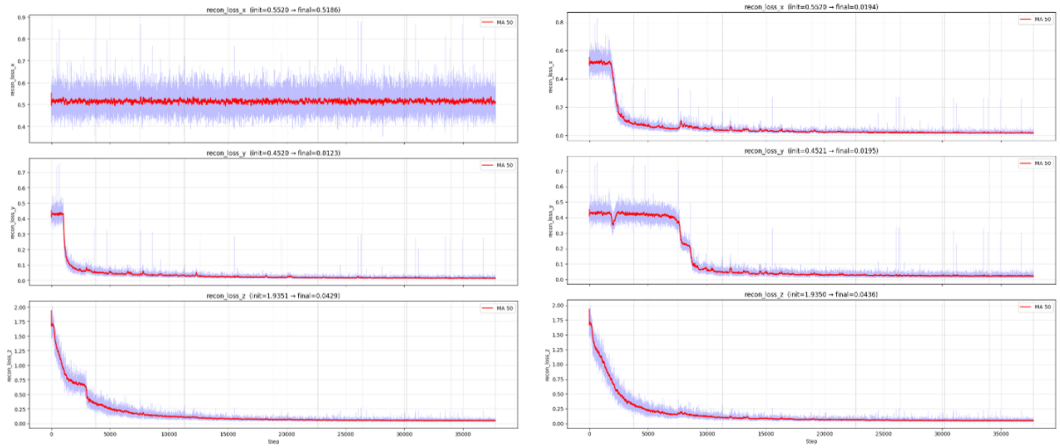
We compare uniform channel averaging with the adaptive weighting strategy in equation 3, while keeping all other training configurations fixed.

As shown in Fig. 5, uniform weighting yields final moving-average losses of 0.5186, 0.0123, and 0.0429 for the x, y, and z coordinates, respectively. The persistently large x-channel error indicates that uniform averaging can produce substantially imbalanced reconstruction across coordinate channels.

Adaptive weighting reduces the final x-channel loss to 0.0194, while obtaining 0.0195 and 0.0436 for y and z. Accordingly, the mean channel loss decreases from 0.1913 to 0.0275. These results demonstrate that adaptive weighting prioritizes the under-optimized coordinate and produces more balanced XYZ reconstruction.

A.5 Point-Map Reconstruction Analysis

Reconstruction probe decoder. For average pooling and bilinear resizing, a fixed orthonormal projection maps the three-channel XYZ features to 1024 channels to match the PMT features. We train identical convolutional probes on each frozen representation. Each probe projects the 32×32



(a) Uniform channel weighting. (b) Adaptive channel weighting.

Figure 5: **Per-channel reconstruction losses during PMT pretraining.** Blue curves show step-wise losses, and red curves show moving averages over 50 steps.

input features to 256 channels, applies a residual convolutional block, and reconstructs a 512×512 XYZ map through four bilinear upsampling blocks with 3×3 convolutions.

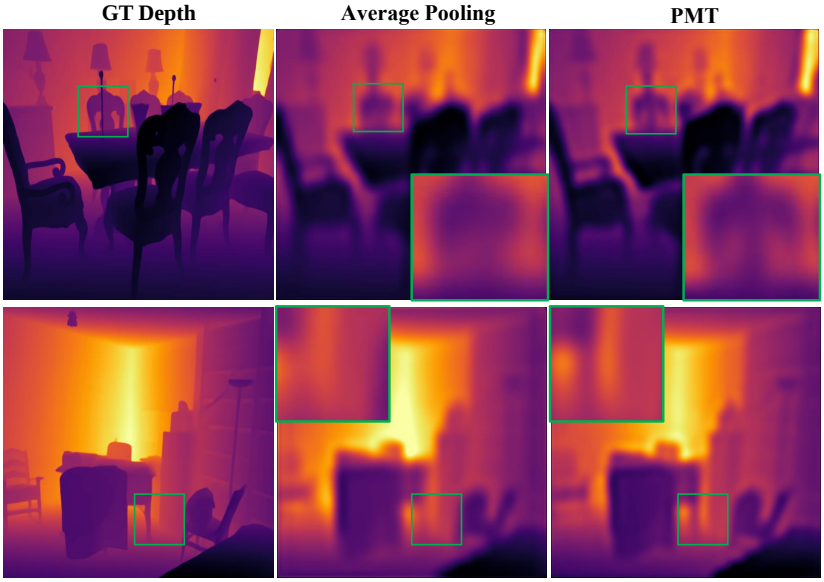


Figure 6: Qualitative comparison of point-map reconstruction on two representative scenes. Each row shows the ground-truth Z-coordinate map (*left*), reconstruction from the average-pooled point-map representation (*middle*), and reconstruction from PMT (*right*). Green boxes and enlarged regions highlight object boundaries and fine geometric structures. Compared with average pooling, PMT better preserves local depth discontinuities and object contours.

Qualitative results. Figure 6 provides qualitative comparisons of point-map reconstruction on two representative scenes. Each row shows the ground-truth depth map, the reconstruction obtained from the average-pooled point-map representation, and the reconstruction produced by PMT, from left to right. Although both representations recover the overall scene layout, average pooling tends to smooth object boundaries and thin structures. In contrast, PMT more clearly recovers local depth discontinuities, narrow gaps, and object contours, as highlighted by the green boxes and enlarged

regions. These observations are consistent with the lower coordinate and pair-distance reconstruction errors reported in Table 3.

A.6 Effect of Geometry Subspace Size

Subspace size k	k/D	MD^2	CV	SPAR
128	1/8	67.3	84.3	66.4
256	1/4	66.4	86.8	67.1
512	1/2	**68.9**	**89.7**	**69.0**
1024	1	68.8	80.5	65.1

Table 8: **Effect of the geometry-subspace dimension.** The default setting is $k = 512$, while $k = 1024$ reserves the full fusion space for geometry.

We study the effect of the geometry-subspace dimension k while keeping all other training and evaluation settings fixed. The full visual-token embedding dimension is $D = 1024$, and our default configuration uses $k = 512$.

As shown in Table 8, $k = 512$ achieves the best performance across all three benchmarks. Smaller subspaces retain a larger complementary space for semantic features but provide less capacity for encoding geometry. Although the results are not strictly monotonic, increasing k from 128 or 256 to 512 provides the strongest overall balance between metric geometry and visual semantics.

In contrast, setting $k = D = 1024$ reserves the entire fusion space for geometry. In this case, $\mathbf{Q}$ is a square orthogonal matrix and

$$\left\|\mathbf{Q}^\top \mathbf{Z}_{\text{sem}}\right\|_F^2 = \|\mathbf{Z}_{\text{sem}}\|_F^2 . \tag{32}$$

Therefore, the leak regularizer no longer penalizes only semantic occupancy of a dedicated geometry subspace; it penalizes the complete semantic representation. Equivalently, the orthogonal complement $\mathcal{G}^\perp$ has zero dimension, leaving no separate subspace for semantic features. This preserves strong MD^2 performance but substantially reduces CV-Bench and SPAR-Bench performance. We consequently adopt $k = 512$, which reserves half of the fusion space for geometry while retaining a 512-dimensional complementary space for semantic representation.