VLM-3R: Vision-Language Models Augmented with Instruction-Aligned 3D Reconstruction

Supplementary Material

A. Additional Implementation Details

A.1. VLM-3R Architecture Specifics

Spatial Encoder Configuration For 3D scene reconstruction/understanding from monocular video, our VLM-3R utilizes a feed-forward (end-to-end) dense 3D reconstruction model as its spatial encoder. This type of encoder directly maps sequences of input RGB images $\{I_i\}_{i=1}^{N}$, where each $I_i \in \mathbb{R}^{3\times H\times W}$, to 3D representations, including estimated camera intrinsics, extrinsics, and point maps that associate pixel information with 3D coordinates. To circumvent the extensive computational costs of using global point cloud and scaling difficulties across different scenes, we propose aligning the implicit encoder tokens instead.

We employ the CUT3R model [6] to process sequences of input images. Given multiple images, such as frames $\{I_t\}$ from a monocular video stream, CUT3R directly outputs corresponding dense 3D point maps (e.g., $\hat{X}_t^{\text{world}}$) and relative camera poses ($\hat{P}_t$) for each view.

Spatial-Visual-View Fusion Design VLM-3R employs a Spatial-Visual-View Fusion stage to integrate diverse multimodal inputs, as depicted in our overall architecture (e.g., Figure 3). This stage utilizes 3D reconstructive tokens from our spatial encoder [6]—specifically, geometry tokens F_t' (encoding scene structure, dimension 729×768) and camera view tokens z_t' (capturing global camera motion, dimension 1×768). These 3D tokens are fused with 2D appearance-based visual tokens H_v (dimension 729×1152) derived from a pre-trained visual encoder, such as a CLIP ViT. Initially, the F_t' and z_t' tokens are concatenated to form a unified 3D representation, $Z_{3D} = \text{Concat}(F_t', z_t')$. This Z_{3D} representation is then processed by a one-layer cross-attention block where it interacts with the visual tokens H_v (e.g., H_v as queries attending to Z_{3D} as keys and values). The output of this attention stage, let's call it H_{attn}, is then residually connected with the original visual tokens H_v to form an enriched representation $H_v' = H_v + H_{attn}$. This enriched H_v' subsequently passes through a two-layer MLP projector, which transforms the feature dimensions (e.g., from 1152 to 3584, and then to 3584) for effective alignment with the Large Multimodal Model (LMM). Finally, these fused and projected 3D-aware visual features, which constitute the final visual input to the LMM, are concatenated with language instruction tokens for processing by the LMM backbone.

A.2. Training Details

The VLM-3R model was trained using the following configuration and hyperparameters.

Base Model and Pretraining The training initialized from the `LLaVA-Video-7B-Qwen2` checkpoint. The vision tower used was `google/siglip-so400m-patch14-384`.

Hardware and Distributed Training Setup Training was conducted using a distributed setup. The specific training run utilized 16 H200 GPUs and lasted for approximately 5 hours. For distributed training, `NUM_GPUS_PER_NODE` was set to 1, and the master address and port were dynamically determined using SLURM environment variables. The `accelerate` library with `torchrun` was used for launching distributed training, employing the DeepSpeed ZeRO stage 2 optimization (`scripts/zero2.json`). CPU affinity was set to 1 for the accelerate launcher.

Key Hyperparameters

- **LoRA Configuration**:
  - **Enable LoRA**: `True` (`-lora_enable True`).
  - **LoRA rank**: `128` (`-lora_r 128`).
  - **LoRA alpha**: `256` (`-lora_alpha 256`).
- **Training Epochs**: The training was set for 5 epochs (`-num_train_epochs $NUM_TRAIN_EPOCHS`), but concluded after the first epoch.
- **Batch Size**:
  - **Per device training batch size**: `1` (`-per_device_train_batch_size 1`).
- **Gradient accumulation steps**: `8` (`-gradient_accumulation_steps $GRADIENT_ACCUMULATION_STEPS`).
- **Learning rate**: 2×10^{-5} (`-learning_rate 2e-5`).
- **Optimizer and Scheduler**:
  - **Weight decay**: `0.0` (`-weight_decay 0.`).
  - **Warmup ratio**: `0.03` (`-warmup_ratio 0.03`).
  - **LR scheduler type**: `cosine` (`-lr_scheduler_type "cosine"`).
- **Precision**: BF16 was enabled (`-bf16 True`). TF32 was also enabled (`-tf32 True`).
- **Model maximum length**: `32768` tokens (`-model_max_length 32768`).
- **Gradient Checkpointing**: Enabled (`-gradient_checkpointing True`).

Spatial Module Configuration

- **Spatial tower**: `CUT3R` (`-spatial_tower "cut3r"`).
- **Spatial tower feature selection**: `all` (`-spatial_tower_select_feature "all"`).
- **Spatial feature dimension**: `768` (`-spatial_feature_dim "768"`).
- **Fusion block**: `cross_attention` (`-fusion_block "cross_attention"`).
- **Tunable Components**:
  - **Tune spatial tower**: `False` (`-tune_spatial_tower False`).
  - **Tune fusion block**: `True` (`-tune_fusion_block True`).
  - **Tune MM MLP adapter**: `True` (`-tune_mm_mlp_adapter True`).

B. Dataset Curation and Benchmark Design

3D Reconstructive Instructional Tuning relies on large-scale Question-Answer (QA) pairs to fine-tune Large Multimodal Models (LMMs), enabling them to perform tasks related to spatial and temporal reasoning. We explain the construction of our 3D Reconstructive datasets, which are utilized for training models subsequently evaluated on VSI-Bench [7]. Furthermore, we detail the creation of the Visual-Spatial-Temporal Intelligence Benchmark (VSTI-Bench), a new resource containing comprehensive training and evaluation pairs specifically for spatial-temporal tasks.

B.1. Scalable 3D Reconstructive Instructional QA Creation

Data Sources and Preprocessing Methodology Our training dataset for 3D Reconstructive Instructional QA is generated using a methodology inspired by approaches such as those used in the VSI-Bench (Visual-Spatial Intelligence Benchmark). Data is sourced from established 3D scene datasets such as ScanNet [3], ScanNet++ [8], and ARKitScenes [2], which are unified and processed. The core of the data preparation involves a detailed preprocessing pipeline to extract structured scene-level and frame-level metadata, similar to established practices.

The input data requirements for each scene include:

1. **Point Cloud (PCD):** A 3D point cloud (e.g., from `.ply` files) with per-point semantic labels, instance labels, coordinates, and color.
2. **Video:** An RGB video traversal of the scene.
3. **Sampled Frame Data:** A collection of frames sampled from the video, including color images, depth maps, instance segmentation masks, and their corresponding 6DoF camera poses (4x4 transformation matrices) in the world coordinate system.
4. **Camera Intrinsics:** Parameters like focal length (f_x, f_y) and principal point (c_x, c_y).

This raw data undergoes a preprocessing pipeline that generates two primary types of structured metadata files in JSON format:

- **scene_metadata.json:** Contains scene-wide information, including overall scene dimensions, room center, counts of different object categories, and detailed 3D bounding boxes (center, size, orientation, instance ID) for every object instance within the scene.
- **frame_metadata.json:** Contains frame-specific information for each scene, such as camera intrinsics, image dimensions, and for each sampled frame: its ID, paths to color/depth images, the camera pose, and 2D bounding boxes for visible object instances.

These metadata files are crucial for the subsequent automated generation of diverse QA pairs for our training dataset.

Spatio-Temporal Scene Graph Construction from Metadata The generated `scene_metadata.json` and `frame_metadata.json` files effectively serve as a structured spatio-temporal scene graph. This representation includes precise 3D locations, sizes, and orientations of objects, their semantic and instance-level identities, and the dynamic viewpoint of the camera across multiple frames. Such detailed and organized metadata enables the querying of complex spatial relationships (e.g., object-object relations, object-camera distances) and spatiotemporal changes (e.g., camera movement, object appearance order). This structured understanding forms the foundation for the diverse QA tasks in our generated dataset.

Spatial QA Generation Methodology The QA pairs for our training dataset are systematically generated using dedicated scripts for each task type, leveraging the aforementioned structured metadata. The generation logic for these tasks is the same as that used in VSI-Bench, and the data is intended for training models on various facets of visual-spatial intelligence.

- **Configurational Tasks:** These assess understanding of spatial layout and inter-object relationships.
  - *Object Count:* Counting instances of a specific object category in a room (numerical answer; objects with single instances are excluded).
  - *Relative Distance:* Determining which of four candidate objects is closest in 3D space to a target object (multiple-choice answer).
  - *Relative Direction:* Identifying the direction of a query object relative to an observer at a specific position and orientation (multiple-choice answer, e.g., left-/right/back).
  - *Route Plan:* Completing a sequence of navigation actions (Go forward, Turn) to reach a target object from a starting object (multiple-choice answer).

- **Measurement Estimation Tasks:** These require quantitative estimation of spatial properties.
  - *Object Size:* Estimating the length of the longest dimension of a unique object instance in centimeters (numerical answer).
  - *Absolute Distance:* Estimating the direct Euclidean distance between the closest points of two specified objects in meters (numerical answer).
  - *Room Size:* Estimating the area of the room in square meters (numerical answer).
- **Spatiotemporal Task:** This tests the processing of spatial information over time.
  - *Appearance Order:* Determining the first-appearance order of four object categories in the video sequence (multiple-choice answer).

Route Plan Data Generation and Template Formatting To generate route planning data, we utilize the Habitat simulator [5] in conjunction with 3D scene data from sources including ScanNet [3], ScanNet++ [8], and ARKitScenes [2]. Mesh files from these datasets are converted from the `.ply` format to the `.glb` format using the assimp library [1]. These `.glb` files then serve as environments within Habitat, where its pathfinder function generates diverse navigation trajectories, as illustrated in Figure A.

The pathfinder computes a navigable trajectory between two designated points within a scene, outputting a sequence of waypoints. These trajectories are then categorized: those containing turns (defined as a change in direction greater than 30 degrees) are classified as 'with turns,' while others are designated 'without turns.' Trajectories 'with turns' are further labeled as "Turn Right" or "Turn Left," depending on the specific angle. For such trajectories, anchor points are defined at the start, the turn itself, and the end. The agent is conceptualized as starting at the initial point, facing the turn point (which serves as a midpoint), and then proceeding to the endpoint. If a turning angle exceeds 45 degrees, an alternative navigation mode is supported where the agent begins at this midpoint, faces the original endpoint, and navigates towards what was the original starting point. Trajectories classified as 'without turns' (i.e., without significant directional changes) are assigned a "Turn Back" action. For these "Turn Back" paths, anchor points are defined at the start, midpoint, and end of the overall segment; the agent begins at the midpoint, oriented towards the starting point, and then moves to the ending point.

For each anchor point along these generated trajectories, we identify the nearest object using available 3D bounding box annotations from the source datasets. This closest object then provides a semantic description for that anchor point. Once these descriptive anchor points are established for a given path, we formulate the final question-answer templates for our route planning tasks.

We use *SRC*, *TGT*, and *MID* to represent the object description derived from the annotations. The templates contain two types:

Templates for Route planning

Template 1
"You are a robot beginning at the *SRC* facing the *MID*. You want to navigate to the *TGT*. You will perform the following actions (Note: for each [please fill in], choose either 'turn back,' 'turn left,' or 'turn right.'): 1. Go forward until the *MID*. 2. [please fill in] 3. Go forward until the *TGT*. You have reached the final destination."

Template 2
"You are a robot beginning at the *MID* facing the *TGT*. You want to navigate to the *SRC*. You will perform the following actions (Note: for each [please fill in], choose either 'turn back,' 'turn left,' or 'turn right.'): 1. [please fill in] 2. Go forward until the *SRC*. You have reached the final destination."

For the trajectories with "Turn Right" or "Turn Left", the corresponding description is Template 1 or Template 2. For "Turn Back", the description is Template 2. The descriptions and corresponding actions are paired as question-and-answer (QA) sets and then converted into a multiple-choice format to serve as training data.

Task Type	Count
Object Relative Direction	86,441
Object Absolute Distance	50,757
Object Relative Distance	42,025
Object Size Estimate	12,917
Object Count	9,357
Route Plan	4,225
Room Size	2,057
Total	**207,779**

Table A. VSI-Bench Training Data Distribution by Task Type (Total QA pairs: 207,779).

B.2. VS*Temporal*I-Bench: New Evaluation Benchmark (138.6K Set)

Motivation and Design Principles Current multimodal models often struggle with spatio-temporal reasoning from monocular video. Even for static scene understanding, accurately interpreting camera motion, along with camera-object and object-object relative movements, remains challenging for LMMs. The VS*Temporal*I-Bench is thus mo-

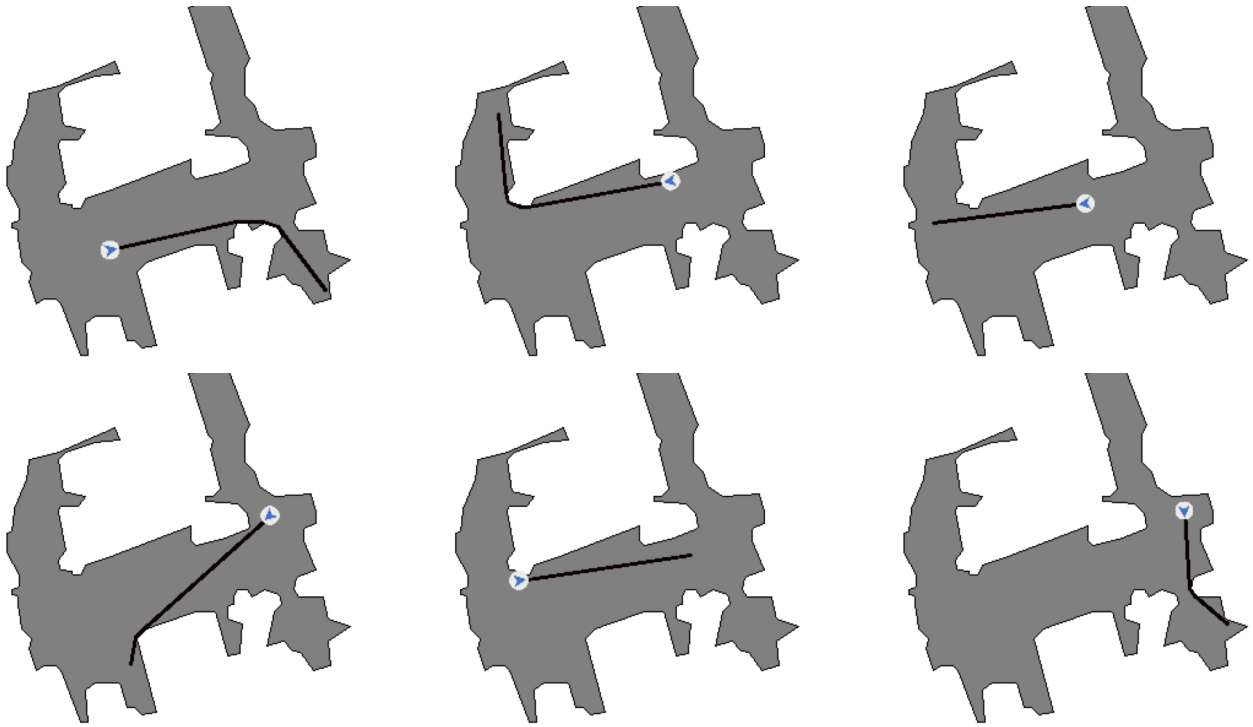

Figure A. Illustrative diverse trajectories generated by the Habitat simulator within a single scene. These demonstrate fundamental navigation actions (e.g., "`Turn Right`", "`Turn Left`", "`Turn Back`"), which are foundational for generating route planning QA data via our specialized templates.

tivated by the critical need to rigorously evaluate and drive progress in LMMs' abilities to comprehend dynamic changes within 3D environments. Our core design principle is to create diverse question-answer pairs that probe the understanding of evolving spatial relationships, such as how camera displacement affects perceived object distances, or how the relative positions of objects change from the camera's perspective over a sequence of frames. The benchmark therefore includes both frame-level tasks requiring precise spatial localization at specific moments and sequence-level tasks demanding an aggregation of information over time to determine motion, direction, or relational changes. This dataset aims to facilitate the examination and advancement of LMMs towards more reliable and human-like visual-spatial-temporal intelligence. Note that this benchmark currently focuses on static 3D scenes, where all depicted motion is from camera movement.

Meta-classes, Task Categories, and Detailed Distribution The VSTemporalI-Bench (VSTIbench) is structured into two primary task categories based on the temporal scope of reasoning required: Frame-Level tasks and Sequence-Level tasks. These categories encompass five distinct types of questions designed to probe various aspects of visual-spatial and temporal understanding. The benchmark contains a total of approximately 138.6K question-answer pairs.

The generation of these QA pairs relies on processed metadata derived from raw 3D scene data (point clouds, videos, and sampled frames with depth, instance masks, and camera poses). Specifically, `scene_metadata.json` (containing 3D object bounding boxes, instance IDs, scene properties) and `frame_metadata.json` (containing per-frame camera poses as 4x4 matrices, camera intrinsics, and 2D object bounding boxes) serve as primary inputs to task-specific generation scripts.

The task categories and their definitions are as follows:

- **Frame-Level Tasks:** These tasks generate questions based on information available within a single frame or the scene's overall point cloud.
  1. **Camera-Object Absolute Distance QA:** Generates questions asking for the approximate Euclidean distance (in meters, numerical answer) between the camera's position in a specific frame and the closest point on the 3D bounding box of a target unique object instance. *Generation Logic:* The 3D Euclidean distance is calculated between the camera's world coordinates (from the frame's pose in `frame_metadata.json`) and the closest point on the 3D bounding box of the object (derived from

scene_metadata.json).

2. **Camera-Object Relative Distance QA:** Generates multiple-choice questions asking which of several candidate object instances is closest to the camera in a specific frame. *Generation Logic:* The 3D Euclidean distance is calculated between the camera's position (from the frame's pose in frame_metadata.json) and the closest point on the 3D bounding box of each candidate object instance (derived from scene_metadata.json). The candidate object with the minimum distance to the camera is the correct answer.
3. **Object-Object Relative Position QA:** Generates multiple-choice questions asking whether a specific target object instance is Near/Far, Left/Right, or Up/-Down relative to another unique target object instance, from the camera's perspective in that frame. *Generation Logic:* The 3D bounding box vertices of both unique objects (from scene_metadata.json) are transformed into the camera's coordinate system for the given frame (using the camera pose from frame_metadata.json). For Near/Far, Z-coordinates are compared. For Left/Right, X-coordinates are compared. For Up/Down, Y-coordinates are compared. Only pairs where one object is entirely nearer/farther, left/right, or up/down than the other (by a defined threshold) are used.

- **Sequence-Level Tasks (Temporal Tasks):** These tasks generate questions based on information aggregated across a sequence of frames. (The generation logic for these tasks is detailed in the subsequent paragraph).
  1. **Camera Displacement QA:** Generates questions asking for the approximate Euclidean distance (numerical answer) the camera traveled between two specified frames, calculated from the camera's world positions.
  2. **Camera Movement Direction QA:** Generates multiple-choice questions about the primary direction of the camera's translation during a sequence, relative to its starting orientation. We first compute the overall camera displacement and express it in the starting camera coordinate system, then assign the dominant direction label (e.g., Forward, Backward, Left, or Right). To increase diversity, we instantiate this task as a mixture of 2-choice, 3-choice, and 4-choice questions by sampling different numbers of distractors from the remaining candidate directions. Accordingly, the random baseline is computed by averaging the per-question chance rate over the mixed choice settings, rather than assuming a fixed 4-way setup.

Data Splits (Train/Test) The VSTemporalI-Bench comprises a total of 138,610 question-answer pairs, which are divided into training and testing splits as detailed below:

VSTI Train (Total: 132,568 QA pairs)

Task Group / Filename	Total Length
camera_displacement	32,292
camera_movement_direction	34,641
camera_obj_abs_dist	38,407
camera_obj_rel_dist	12,155
obj_obj_relative_pos	15,073

Table B. VSTemporalI-Bench Training Set Distribution

VSTI Test (Total: 6,042 QA pairs)

Task Group / Filename	Total Length
camera_displacement	839
camera_movement_direction	913
camera_obj_abs_dist	905
camera_obj_rel_dist	1,740
obj_obj_relative_pos	1,645

Table C. VSTemporalI-Bench Test Set Distribution

C. Additional Results and Analysis

In this section, we provide additional analysis of VLM-3R from three perspectives. First, we examine task-level behaviors on VSI-Bench to understand which capabilities benefit most from reconstructive spatial modeling. Second, we compare different pretrained geometry encoders to justify our choice of CUT3R. Finally, we evaluate zero-shot generalization on OpenEQA, including comparisons with both the base model and broader multimodal baselines.

C.1. Task-wise Analysis on VSI-Bench

Our error analysis, together with the quantitative ablation results reported in the main paper, reveals how VLM-3R behaves across different categories of spatial reasoning tasks. We observe that the gains from our method are most pronounced on tasks that require explicit geometric perception and metric-aware reasoning.

For example, on **Absolute Distance**, the full VLM-3R achieves 49.38, substantially outperforming the LLaVA-NeXT-Video ft (w/o C&G Tok.) baseline, which scores 43.67. Removing either camera tokens or geometry tokens also leads to performance drops (48.66 and 49.27, respectively), indicating that both components contribute to accurate absolute-scale distance estimation. These results suggest that reconstructive spatial representations are particularly beneficial when the model must reason about metric geometry rather than relying only on appearance cues.

Metric	Base	VGGT	CUT3R
Abs. Dist.	43.6	**49.8**	49.4
Obj. Count	**70.6**	68.8	70.2
Obj. Size	**70.8**	**70.8**	69.2
Room Size	63.7	54.0	**67.1**
Rel. Dist.	64.9	61.5	**65.4**
Rel. Dir.	68.9	**80.8**	80.5
Route Plan	40.7	44.8	**45.4**
Appr. Order	38.5	34.5	**40.1**
Overall	57.7	58.1	**60.9**

Table D. **Geometry encoder ablation on VSI-Bench.** CUT3R achieves the best overall performance, showing the benefit of metric scale and temporal reasoning.

By contrast, on **Object Counting**, VLM-3R obtains 70.16, which is comparable to both the baseline (70.64) and the variant without geometry tokens (70.30). This suggests that, under the current VSI-Bench formulation, object counting depends less on fine-grained 3D geometry and more on 2D visual recognition and cross-frame instance tracking. Overall, these observations indicate that VLM-3R mainly improves tasks requiring stronger geometric grounding, while bringing smaller gains on tasks that are already well supported by conventional visual features.

C.2. Ablation on Pretrained Geometry Encoders

To study how the choice of geometric prior affects performance, we compare our default geometry encoder, CUT3R [6], with another strong multi-view geometric foundation model, VGGT. The results on VSI-Bench are summarized in Table D.

Both geometry encoders improve relative spatial reasoning over the finetuned base model, showing that pretrained geometric priors are broadly beneficial. However, the choice of encoder becomes important for tasks involving temporal ordering and absolute metric estimation. VGGT adopts a permutation-equivariant design and relies more heavily on relative-scale priors, which makes it less suitable for sequence-aware reasoning and metric-scale judgments. As a result, it underperforms on Appearance Order ($38.5 \rightarrow 34.5$) and Room Size ($63.7 \rightarrow 54.0$), despite remaining competitive on some relational tasks.

In contrast, CUT3R naturally preserves temporal structure and provides metric-scale information, which better matches the needs of spatio-temporal reasoning. With CUT3R, VLM-3R achieves the best overall performance (60.9%), with particularly clear advantages on Appearance Order (40.1%), Room Size (67.1%), and Route Plan (45.4%). These results justify our adoption of CUT3R as the default geometry encoder.

Metric	LLaVA-NXT-Video	VLM-3R
Spatial	49.95	**51.60**
Non-Spatial	**67.22**	65.54
Overall	**62.4**	61.7

Table E. **Base-model comparison on OpenEQA.** VLM-3R improves spatial performance over its base model, with a small trade-off on non-spatial questions.

C.3. Generalization to OpenEQA

To further assess whether the spatial capabilities learned by VLM-3R transfer beyond VSI-Bench, we evaluate it zero-shot on OpenEQA [4], an open-vocabulary embodied question answering benchmark containing 1,600 human-written questions from more than 180 scenes. We analyze the results from two perspectives: comparison with the direct base model, and comparison with broader multimodal baselines.

Comparison with the base model. A key concern in spatial instruction tuning is whether stronger geometric reasoning comes at the expense of general video understanding. To examine this trade-off, we compare VLM-3R directly with its base model, LLaVA-NeXT-Video. As shown in Table E, VLM-3R improves performance on spatial questions from 49.95% to 51.60%, while causing only a modest decrease on non-spatial questions from 67.22% to 65.54%. Although this leads to a slightly lower overall score, the results confirm that our reconstructive instruction tuning successfully injects stronger spatial awareness into the base model. To reduce the loss in general-domain capability, we adopt a mixed-domain training strategy combining *200k VSI-Bench* samples with *30k LLaVA-Video* samples.

Comparison with broader multimodal baselines. Beyond the base-model comparison, VLM-3R also remains competitive against strong general-purpose multimodal models. As shown in Table F, VLM-3R achieves 61.7% overall zero-shot accuracy, outperforming GPT-4V, Gemini-Pro, Claude 3, GPT-4, and LLaMA2. These results suggest that the proposed geometry-aware fusion and mixed-domain training strategy improve spatial understanding without sacrificing competitiveness on a broader embodied benchmark.

References

[1] Open-asset-importer-library (assimp). https://github.com/assimp/assimp, 2024. 3

[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

	Method	Perf. (%)		Method	Perf. (%)
1	Human	86.8	5	Gemini-Pro	44.9
2	**VLM-3R**	**61.7**	6	Claude 3	36.3
3	GPT-4V (50 frames)	55.3	7	GPT-4	33.5
4	GPT-4V (15 frames)	54.6	8	LLaMA2	28.3

Table F. **Zero-shot performance against general baselines on OpenEQA.** VLM-3R remains highly competitive among strong multimodal baselines.

dataset for 3d indoor scene understanding using mobile rgb-d data. *arXiv preprint arXiv:2111.08897*, 2021. 2, 3

[3] 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, 3

[4] Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, et al. Openeqa: Embodied question answering in the era of foundation models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16488–16498, 2024. 6

[5] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347, 2019. 3

[6] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. *arXiv preprint arXiv:2501.12387*, 2025. 1, 6

[7] 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. *arXiv preprint arXiv:2412.14171*, 2024. 2

[8] 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, 3