Token Warping Helps MLLMs Look from Nearby Viewpoints — Supplementary Material

Phillip Y. Lee* Chanho Park* Mingue Park Seungwoo Yoo Juil Koo Minhyuk Sung
KAIST

In this supplementary material, we report additional experimental results with more baseline MLLMs and showcase qualitative examples of warped visualizations with corresponding MLLM responses (Sec. A). We then present implementation and algorithmic details of *backward token warping* with *nearest* and *adaptive* fetching (Sec. B). Finally, we describe the step-by-step data construction pipeline of `ViewBench` in Sec. C.

A. Additional Results

This section presents additional experiments: extended comparisons with specialized MLLMs (Sec. A.1), robustness analysis under estimated geometry (Sec. A.2), evaluation under extreme viewpoint shifts and occlusion (Sec. A.3), a geometry-based oracle analysis (Sec. A.4), and qualitative examples (Sec. A.5).

A.1. Comparison with Additional Baselines

Extending Tab. 1 **of the main paper**, we report a more extensive quantitative comparison against a wider range of specialist and general-purpose MLLMs.

Baselines. We include recent open-source MLLMs: *Qwen3-VL* [28], *InternVL3* [33], *Cambrian-1* [22], *LLaVA-OneVision-1.5* [4], and *Kimi-VL-Thinking* [21]. We further include models explicitly fine-tuned for spatial reasoning via SFT and/or GRPO [13]. *RoboBrain-2.0* [20] and *Ve-Brain* [16] extend Qwen2.5-VL [5] with rich spatial task suites, while *SpaceQwen* [1] and *SpaceThinker* [2] are Qwen2.5-VL variants fine-tuned on spatial VQA data [3] following data synthesis protocol of SpatialVLM [8]. For *MindCube* [31], we used the `Plain-CGMap-FFR-Out` SFT variant, reported as the best-performing configuration by the authors.

We include models from *VST* [30], a concurrent work that fine-tunes Qwen2.5-VL on a curated dataset spanning over 19 spatial tasks, comparing both their SFT *(VST-SFT)* and RL-tuned *(VST-RL)* variants. We further compare with a SFT variant of *SpaceR* [18] and *SpatialLadder* [14], a concurrent work employing a progressive SFT+GRPO training schedule for spatial reasoning. Finally, we evaluate *VG-LLM* [32], which integrates a 3D geometry encoder initialized from VGGT [23] into an MLLM, similar to VLM-3R [11] **in the main paper**, which integrates CUT3R [24] features to provide strong 3D priors.

Results. Full results are shown in Tab. A1. Consistent with Tab. 1 **of the main paper**, our backward token warping methods (*i.e.*, *Backward-Nearest* and *Backward-Adaptive*) achieve the best performance on both `ViewBench-Text` and `ViewBench-Shape`, outperforming all baselines including the newly added models. Notably, recent state-of-the-art general MLLMs (*e.g.*, Qwen3-VL [28], InternVL3 [33]) still struggle to internally shift viewpoint to solve our tasks. Likewise, MindCube [31], despite being designed for multi-view spatial reasoning, shows clear limitations when required to reason about a single view from a nearby target viewpoint. SpatialLadder [14], despite its carefully designed training curriculum, still underperforms our backward token warping, which explicitly and reliably transfers source-view information to the target viewpoint.

Lastly, VG-LLM [32], which integrates rich 3D features from VGGT [23], exhibits highly degraded behavior: the model frequently outputs multiple-choice labels (*e.g.*, *"A"*, *"B"*) even when prompted to answer with *"left"* or *"right"*. We hypothesize that the VGGT-based fine-tuning phase may have compromised the base MLLM's general capabilities, whereas our token warping approach leaves the underlying MLLM unchanged, better preserving its original abilities.

A.2. Robustness Analysis on Estimated Geometry

Our token warping framework relies on the depth map $\mathbf{D}$ and relative camera pose $\mathbf{\Pi}_{T\to S}$ to compute the backward warping function $f_{T\to S}$ (Eq. B.4). A natural concern is whether the method remains effective when geometric inputs are estimated rather than ground-truth. We evaluate this on `ViewBench-Shape` by replacing the ground-truth geometry with predictions from off-the-shelf models.

*Equal contribution.

	ViewBench-Text (%)						ViewBench-Shape (%)						ViewBench-Object (1-10)					
View Overlap (%)	5–15		15–25		25–35		5–15		15–25		25–35		5–15		15–25		25–35	
Depth	GT	Pred.	GT	Pred.	GT	Pred.	GT	Pred.	GT	Pred.	GT	Pred.	GT	Pred.	GT	Pred.	GT	Pred.
Target View (Oracle)	100.00	–	100.00	–	100.00	-	100.00	–	100.00	–	100.00	-	6.64	–	7.31	–	7.43	–
Specialist MLLMs																		
SpatialReasoner [17]	46.73	–	53.30	–	53.71	–	33.72	–	38.27	–	48.15	–	–	–	–	–	–	–
VLM-3R [11]	63.82	–	70.56	–	60.57	–	49.22	–	49.79	–	50.21	–	–	–	–	–	–	–
ViLaSR [26]	44.22	–	52.28	–	48.00	–	22.87	–	23.05	–	34.57	–	–	–	–	–	–	–
Qwen2.5-VL [5]	46.23	–	59.39	–	52.00	–	24.42	–	25.10	–	37.86	–	–	–	–	–	–	–
Qwen3-VL [28]	41.71	–	47.21	–	45.14	–	18.60	–	22.22	–	35.80	–	–	–	–	–	–	–
InternVL3 [33]	56.28	–	64.47	–	61.71	–	32.17	–	38.68	–	51.85	–	–	–	–	–	–	–
Cambrian-1 [22]	9.05	–	11.68	–	9.71	–	34.88	–	34.57	–	44.03	–	–	–	–	–	–	–
LLaVA-OneVision-1.5 [4]	48.24	–	51.27	–	61.71	–	27.52	–	30.04	–	38.27	–	–	–	–	–	–	–
Kimi-VL-Thinking [21]	49.25	–	54.31	–	52.00	–	31.78	–	37.86	–	43.21	–	–	–	–	–	–	–
RoboBrain-2.0 [20]	37.69	–	43.65	–	49.71	–	22.48	–	29.63	–	39.92	–	–	–	–	–	–	–
VeBrain [16]	49.25	–	54.31	–	54.29	–	29.84	–	32.51	–	47.33	–	–	–	–	–	–	–
SpaceQwen [8]	68.34	–	72.69	–	62.86	–	48.06	–	46.50	–	49.38	–	–	–	–	–	–	–
SpaceThinker [8]	48.74	–	51.27	–	48.57	–	46.51	–	47.74	–	48.15	–	–	–	–	–	–	–
MindCube [31]	59.30	–	59.39	–	57.14	–	46.90	–	47.74	–	47.33	–	–	–	–	–	–	–
VST-RL [30]	28.14	–	34.01	–	38.29	–	28.29	–	26.34	–	43.62	–	–	–	–	–	–	–
VST-SFT [30]	42.71	–	47.72	–	46.29	–	28.29	–	26.34	–	43.62	–	–	–	–	–	–	–
SpaceR-SFT-7B [18]	67.84	–	73.10	–	64.00	–	44.96	–	48.15	–	53.09	–	–	–	–	–	–	–
SpatialLadder [14]	70.35	–	74.11	–	67.43	–	50.00	–	49.38	–	50.21	–	–	–	–	–	–	–
VG-LLM [32]	5.93	–	13.20	–	9.71	–	13.18	–	14.40	–	24.69	–	–	–	–	–	–	–
Novel View Synthesis																		
GenWarp [19]	69.35	–	71.07	–	66.29	–	53.10	–	47.33	–	55.14	–	4.32	–	4.81	–	4.34	–
Pixel-Wise Warping																		
Forward	70.85	69.35	73.60	73.10	62.86	67.43	56.20	56.20	56.79	56.79	60.49	60.08	3.22	3.22	4.04	3.87	4.78	4.54
Backward	71.86	67.84	75.63	74.62	68.57	68.57	62.40	58.14	58.02	56.79	66.67	64.20	4.53	4.45	5.52	5.48	5.94	5.89
Token Warping																		
Forward	60.30	66.83	64.47	65.48	54.86	60.57	55.04	56.98	55.14	60.91	53.09	56.38	4.09	4.20	4.27	4.37	4.07	3.78
Backward-Nearest	74.87	**75.38**	**80.71**	**81.73**	74.86	**76.00**	**67.44**	63.95	62.96	**62.55**	73.25	**75.31**	4.80	4.86	5.39	5.57	**6.19**	5.97
Backward-Adaptive	**77.89**	73.37	79.70	80.71	**78.86**	74.29	**67.44**	**66.28**	**66.26**	61.32	**75.72**	70.37	**4.97**	**5.18**	**5.76**	**6.29**	6.11	**6.14**

Table A1. **Additional Quantitative Comparisons on ViewBench.** Extended table of Tab. 1 **in the main paper**, with additional baseline MLLMs included in orange ( ). Columns 2-13 report accuracy (%) on spatial reasoning tasks (ViewBench-Text and ViewBench-Shape), and columns 14-19 report target-view object description scores (ViewBench-Object), evaluated by Qwen2.5-VL-14B [5] on a 1–10 scale. Across all tasks and setups, backward token-wise warping achieves the best performance.

Depth Estimation. We compare ground-truth depth (GT) against predictions from two monocular depth estimators: Depth Anything v2 (DA-V2) [29] and Depth Pro (DP) [7]. We additionally include a no-warping reference baseline (Ref.) using the base Qwen2.5-VL [5] on the source image. As shown in Tab. A2, backward token warping with adaptive fetching achieves 65.84% with DA-V2 and 67.74% with DP, compared to 70.99% with GT depth. Pixel-wise backward warping follows the same trend, dropping from 62.35% (GT) to 60.49% (DA-V2) and 62.76% (DP). In both cases, warping with estimated geometry substantially outperforms the no-warping baseline, confirming that the gains from warping persist even without ground-truth geometry. Importantly, the performance gap between token warping and pixel-wise warping is preserved regardless of the depth source, indicating that the advantage of operating in token space is orthogonal to improvements in depth estimation quality.

Joint Depth and Pose Estimation. We further evaluate a more challenging setting where *both* depth and relative pose are predicted from an image pair, using VGGT [23] and DUSt3R [25]. As reported in Tab. A2, token warping with VGGT-estimated geometry achieves 68.95%, compared to 63.58% for pixel-wise warping under the same conditions. With DUSt3R, both methods decline further, yet token warping still outperforms pixel-wise warping. These results confirm that the conclusions of Tab. 1 **of the main paper** hold under realistic conditions where ground-truth geometry is unavailable.

	GT	Depth		Depth & Pose		Ref.
		DA-V2	DP	VGGT	DUSt3R	
Pixel-Wise Warp.	62.35	60.49	62.76	63.58	61.29	31.48
Token Warp.	70.99	65.84	67.74	68.95	65.05	

Table A2. **Robustness to Estimated Geometry.** Accuracy (%) on ViewBench-Shape (averaged across all overlap levels). *Ref.* is a no-warping baseline with base Qwen2.5-VL [5].

A.3. Larger Viewpoint Shifts and Occlusion

To stress-test our method beyond the overlap ranges in Sec. 5 **of the main paper** (5–35%), we construct two additional evaluation splits targeting extreme viewpoint shifts and occlusion.

Larger Viewpoint Shifts. We sample source–target pairs from ScanNet [9] with very low overlap (2–5%), representing nearly disjoint views where only a small portion of the scene is shared. As shown in Tab. A3, backward token warping with adaptive fetching achieves 65.08% with GT depth and 66.14% with estimated depth, substantially outperforming pixel-wise backward warping (61.90% / 61.38%) and the no-warping baseline (34.39%). The consistent trend across all overlap levels suggests that the advantages of token-level warping are not confined to moderate viewpoint changes.

Depth	GT	Pred.
Qwen2.5-VL [5]	34.39	–
Pixel-Wise Warp.	61.90	61.38
Token Warp.	65.08	66.14

Table A3. **Larger Viewpoint Shift (2–5% Overlap).** Accuracy (%) on a stress-test split with extremely low view overlap, where the source and target views share only 2–5% of visible scene content.

Occlusion. We also collect synthetic image pairs using ProcTHOR [10] where an object visible from the source view becomes *fully occluded* at the target viewpoint. This tests whether warping helps the model reason about visibility changes under viewpoint shifts. As shown in Fig. A1, token warping achieves 46% accuracy with GT depth, compared to 38% for pixel-wise warping and 32% for the base Qwen2.5-VL [5], evaluated on 50 pairs with GT depth. While absolute accuracies are lower due to the difficulty of reasoning under full occlusion, the relative ordering is consistent with our main findings: token warping provides a more reliable basis for viewpoint reasoning even under significant visibility changes.

Depth	GT
Qwen2.5-VL [5]	32.00
Pixel-Wise Warp.	38.00
Token Warp.	46.00

Table A4. **Occlusion Evaluation.** Accuracy (%) on a ProcTHOR-based [10] split where the queried object is fully occluded in the target view. Token warping consistently outperforms pixel-wise warping and the base Qwen2.5-VL [5].

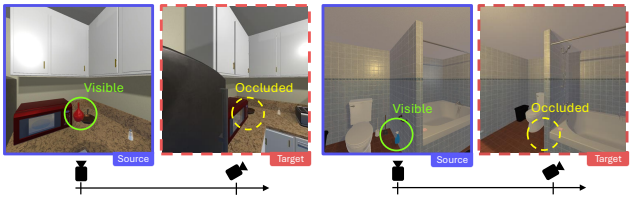

Figure A1. **Occlusion Evaluation.** Example source–target pairs from the ProcTHOR-based [10] occlusion split, where a visible object in the source view becomes fully occluded in the target view.

A.4. Geometry-Based Oracle

To verify the reliability of the geometric pipeline underlying our token warping, we implement a *geometry-based oracle* that bypasses the MLLM entirely. Given a source–target pair, the oracle applies the backward warping function $f_{T\to S}$ (Eq. B.4) to the two annotated keypoints in the source image and determines their left–right ordering by directly comparing the x-coordinates of the warped points, without querying the MLLM.

As shown in Tab. A5, the geometry-based oracle achieves above 93% across all overlap levels for both `ViewBench-Text` and `ViewBench-Shape`. The small gap from 100% is attributable to occasional depth noise near object boundaries and edge cases where the two keypoints project to nearly identical x-coordinates in the target view. These results confirm that the warping geometry is highly accurate, and that the remaining gap between our token warping methods (Tab. 1 **of the main paper**) and the oracle is primarily due to limitations in the MLLM's perception and reasoning capabilities rather than geometric errors.

	Text (%)			**Shape (%)**		
View Overlap (%)	5–15	15–25	25–35	5–15	15–25	25–35
Oracle (Geometry)	93.78	93.81	93.64	95.26	94.54	93.75

Table A5. **Geometry-Based Oracle.** Accuracy (%) of a geometry-only baseline that determines left–right ordering by comparing x-coordinates of the warped source keypoints.

A.5. Additional Qualitative Results

We provide additional qualitative comparisons of our backward token warping with multiple baselines—including pixel-wise warping and forward token warping—on single-view VQA examples that require reasoning under viewpoint changes. The visualizations are shown in Figs. A2– A5, with brief descriptions provided below. For each case, we are given the source image, its depth map, the relative camera pose from source to target, and the camera intrinsics. To obtain the depth and poses, we run VGGT [23] on the source and target view images.

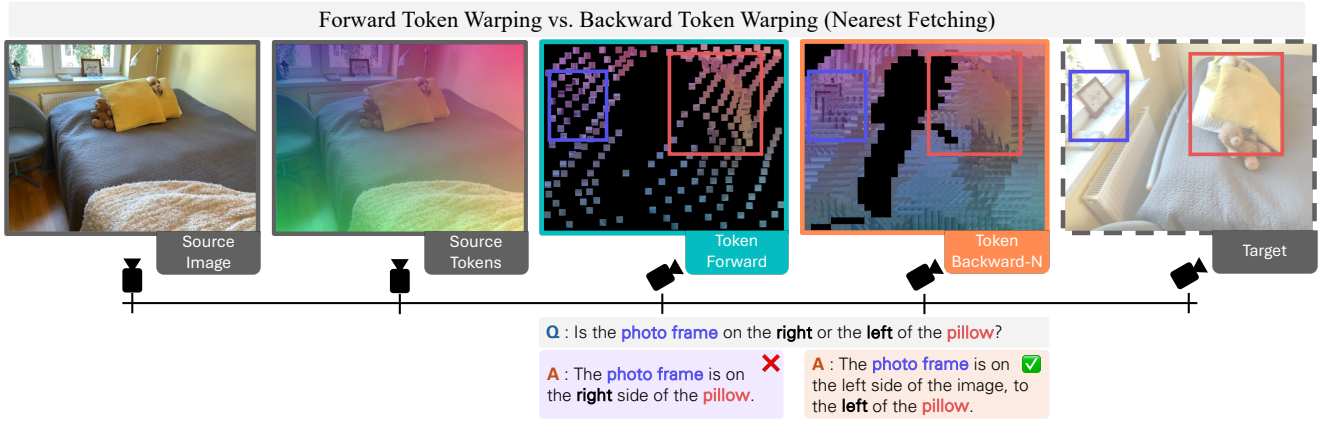

Figure A2. **Qualitative Sample 1.** Given the source image (leftmost), the question asks for the spatial relationship between the photo frame (blue box) and the pillow (red box) as viewed *from the target viewpoint* (rightmost). **To visualize tokens, we color-code each source token by its** (x, y) **position in the source image, and preserve this color after warping, so the color of each token in the warped views indicates its source location.** With **forward token warping**, the projected tokens become sparse and irregular, leading the MLLM to answer incorrectly. In contrast, **backward token warping** with **nearest fetching** produces a dense, regular target token grid, allowing the model to correctly infer the spatial relationship from the target view. (Source and target images are from ARKitScenes [6].)

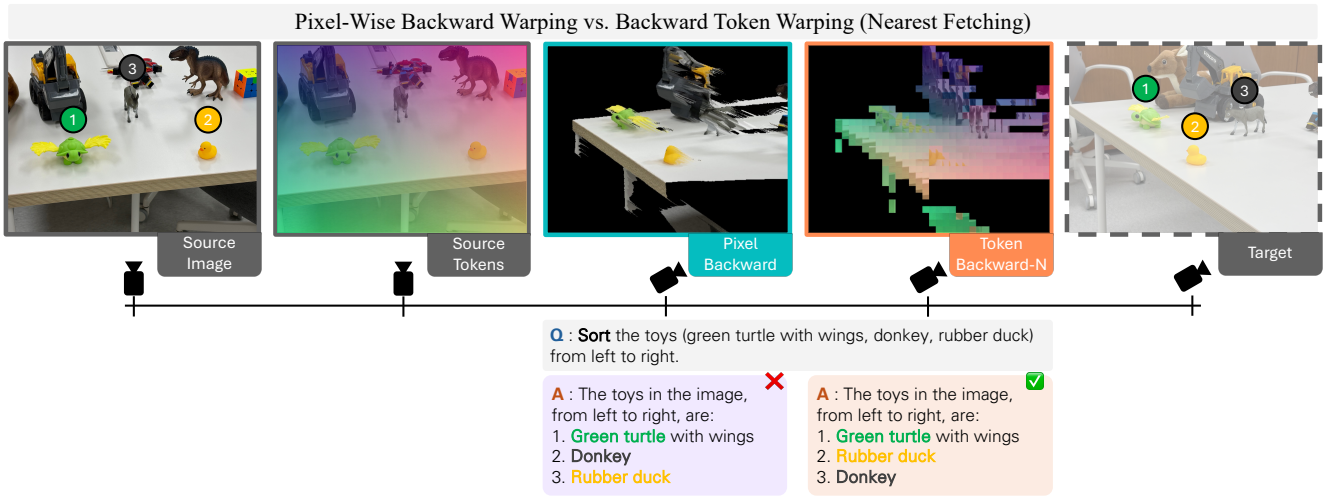

Figure A3. **Qualitative Sample 2.** Given the source image (leftmost), the question asks for the order of the toys from left to right *as seen from the target viewpoint* (rightmost). **To visualize tokens, we color-code each source token by its** (x, y) **position in the source image, and preserve this color after warping, so the color of each token in the warped views indicates its source location.** With **pixel-wise backward warping**, the target-view image suffers from local pixel distortions caused by depth noise, leading the MLLM to answer incorrectly. In contrast, **backward token warping** with **nearest fetching** preserves the semantic content while shifting viewpoint, allowing the MLLM to produce the correct ordering of the toys. (Source and target images were captured manually.)

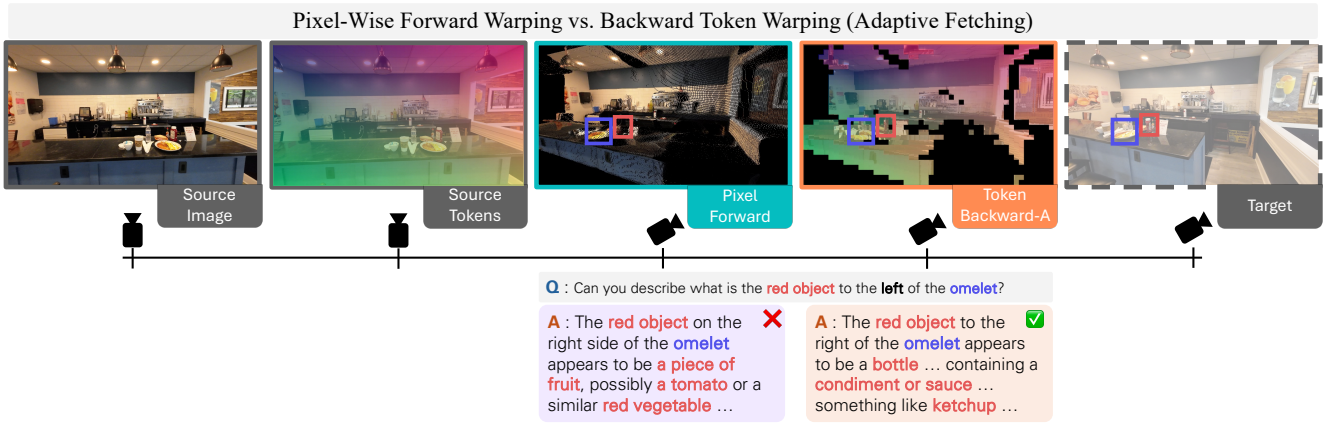

Figure A4. **Qualitative Sample 3.** Given the source image (leftmost), the question asks to describe the *red object* (red box) placed on the left side of the omelet (blue box) *when viewed from the target viewpoint* (rightmost). **To visualize tokens, we color-code each source token by its** (x, y) **position in the source image, and preserve this color after warping, so the color of each token in the warped views indicates its source location.** When using **pixel-wise forward warping**, the warped image exhibits local pixel distortions due to depth prediction noise and holes caused by magnification. Consequently, given this warped RGB image, the MLLM incorrectly answers that the object is *"a piece of fruit"*. In contrast, with **backward token warping** and **adaptive fetching**, the MLLM correctly identifies the object as a *"bottle"*, more specifically *"containing a condiment or sauce"* and *"ketchup"*. This further highlights the advantage of warping in token space rather than pixel space when transferring source content to a target view. (Source and target images are from DL3DV-10K [15].)

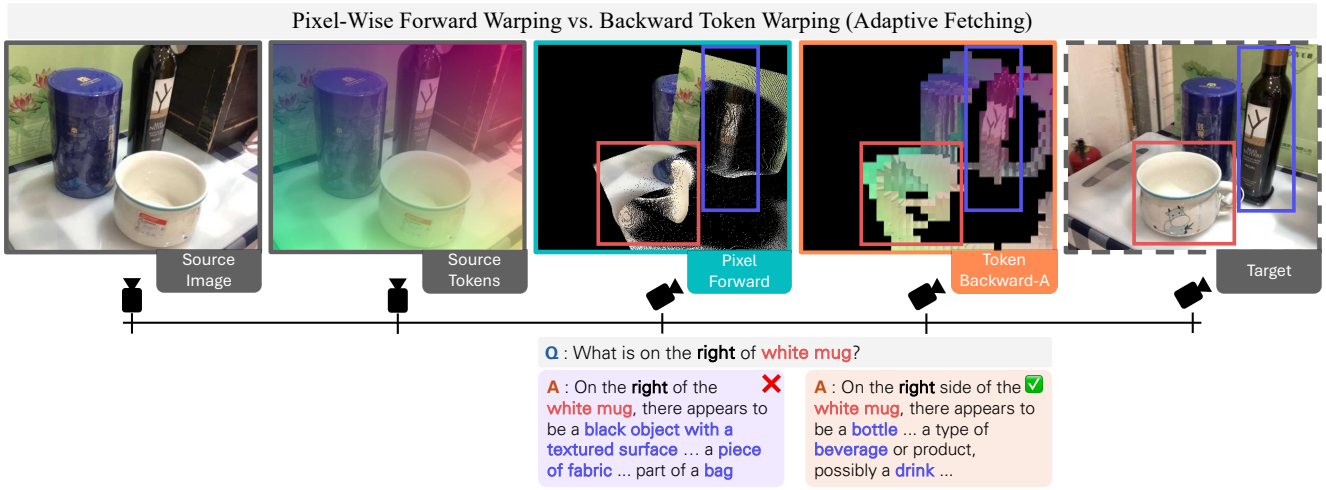

Figure A5. **Qualitative Sample 4.** Given the source image (leftmost), the question asks to describe the object that is located on the right side of the white mug (red box) *when viewed from the target viewpoint* (rightmost). **To visualize tokens, we color-code each source token by its** (x, y) **position in the source image, and preserve this color after warping, so the color of each token in the warped views indicates its source location.** With **pixel-wise forward warping**, the warped image shows distorted local details due as the forward warping distributes the source image pixels to a sparse grid in the target image. Consequently, the MLLM fails to accurately describe the bottle on the right side, and instead replies *"piece of fabric"* and *"part of a bag"*, which are not visible in the target image. On the other hand, when using **backward token warping** with **adaptive fetching**, the MLLM describes the specified object as *"a bottle"* and *"a type of beverage"* which is accurate when seen from the target image. These results again show that our proposes backward token warping can provide a robust way of transferring source image information to the target viewpoint. (Source and target images are from BLINK [12].)

B. Implementation Details

This section extends Sec. 3.3 **of the main paper** and details the implementation of our backward token warping framework, which enables MLLMs to reason under viewpoints changes from a single source image, its depth map, and relative camera pose. For clarity, in this section we use "**c**" to denote coordinates in the *source* view and "**g**" to denote coordinates in the *target* view.

B.1. Details on Backward Token Warping

Recall that in backward warping, we define a dense, regular grid in the target view and fetch the corresponding tokens from the source image $\mathbf{I}$ via the target-to-source mapping $f_{T\to S}$.

Target Grid. For an image $\mathbf{I} \in \mathbb{R}^{H\times W\times 3}$, we impose a regular patch grid of size $l \times l$, yielding $M = (HW)/l^2$ patches*. We denote by $\mathbf{g} \in \mathbb{R}^{M\times 2}$ the set of target-grid centers on the image plane, where each $\mathbf{g}_j$ specifies a location at which we wish to place a token sampled from the source image. In backward token warping, our goal is to assign exactly one token to each grid center. For simplicity, we assume the target image has the same resolution as the source.

Source Proxy from Depth. Because the target-view image is unobserved, we cannot directly compute target-to-source correspondences. Instead, we construct a lightweight 3D triangle mesh $\mathcal{M}_S$ from the source depth map $\mathbf{D} \in \mathbb{R}^{H\times W\times 1}$. Specifically, for each pixel $\mathbf{p}_i = (u_i, v_i)$ in $\mathbf{I}$ with its depth d_i from $\mathbf{D}$, we unproject it using the 3×3 intrinsic matrix $\mathbf{K}_{3\times 3}$ to obtain a 3D point:

$$\mathbf{x}_i = d_i \mathbf{K}_{3\times 3}^{-1} \tilde{\mathbf{p}}_i, \quad \text{where } \tilde{\mathbf{p}}_i = [u_i, v_i, 1]^\top. \tag{B.1}$$

Here, $\mathbf{x}_i = [x_i, y_i, z_i]^\top$. We then triangulate every 2×2 pixel cell into two triangles, forming $\mathcal{M}_S$ in the source camera frame.

Backward Mapping via Ray Casting. For each target grid center $\mathbf{g}_j$, we cast a ray from the target camera using its pose $\Pi_T \in \mathbb{R}^{4\times 4}$ and intrinsics $\mathbf{K} \in \mathbb{R}^{4\times 4}$, and intersect it with the proxy mesh $\mathcal{M}_S$, obtaining a 3D hit point in the target frame, $\mathbf{x}_j^* \in \mathbb{R}^3$. We then express this point in homogeneous coordinates and project it back into the source image using the relative pose $\Pi_{T\to S} = \Pi_S \Pi_T^{-1}$ and intrinsics $\mathbf{K}$:

$$\tilde{\mathbf{p}}_j^* = \mathbf{K}\, \Pi_{T\to S}\, \tilde{\mathbf{x}}_j^*, \quad \text{where } \tilde{\mathbf{x}}_j^* = [\mathbf{x}_j^*, 1]^\top, \tag{B.2}$$

$$\mathbf{g}_j^* = \pi\left(\tilde{\mathbf{p}}_j^*\right), \tag{B.3}$$

where $\pi([u, v, w, 1]^\top) = (u/w, v/w)^\top$ denotes perspective projection. The resulting $\mathbf{g}_j^* \in \mathbb{R}^2$ is a coordinate on $\mathbf{I}$ and serves as the backward mapping from target to source. If no valid intersection is found (e.g., due to occlusion or field-of-view mismatch), we mark $\mathbf{g}_j^*$ as invalid and omit the corresponding patch.

By applying Eq. B.3 for every target grid center $\mathbf{g}_j \in \mathbf{g}$, we obtain the set of backward-warped coordinates on the source image, $\mathbf{g}^* \in \mathbb{R}^{M\times 2}$. Consistent with Eq. 3.1 **in the main paper**, we denote this backward warping process as

$$\mathbf{g}^* = f_{T\to S}(\mathbf{g}, \Pi_{T\to S}, \mathbf{K}, \mathbf{D})\,. \tag{B.4}$$

Given $f_{T\to S}$, which provides a coordinate for every target grid center, the final step is to *fetch* the corresponding tokens from the source image at these locations. We provide details on the fetching strategies in the next section.

B.2. Nearest vs. Adaptive Fetching

We now detail the *nearest* and *adaptive* token fetching strategies used in the final step of backward token warping.

Nearest Fetching. Recall from Sec. 3.1 **in the main paper** that source image $\boldsymbol{I}$ is partitioned into a fixed, non-overlapping grid of patches $\{\mathbf{u}_i\}_{i=1}^M$. Let the source image $\mathbf{I}$ be patchified on a fixed grid, and let $\mathbf{c} \in \mathbb{R}^{M\times 2}$ denote the set of source grid centers, where M is the number of patches. Given a target grid center $\mathbf{g}_j$ and its backward-warped source coordinate $\mathbf{g}_j^*$ from $f_{T\to S}$, *nearest fetching* selects the existing source patch whose center is closest to $\mathbf{g}_j^*$ in Euclidean distance:

$$i' = \arg\min_i \left\| \mathbf{g}_j^* - \mathbf{c}_i \right\|_2. \tag{B.5}$$

We then assign to $\mathbf{g}_j$ the token that was derived from the patch $\mathbf{u}_{i'}$ centered at $\mathbf{c}_{i'}$. While this introduces a small mismatch as $\mathbf{g}_j^*$ may not not coincide with any $\mathbf{c}_i$ in most cases, it allows us to reuse the original, efficient fixed-grid patchification for the source image.

Adaptive Fetching. Alternatively, we further implement *adaptive fetching*, which re-patchifies the source image $\mathbf{I}$ according to the backward-warped coordinates $\mathbf{g}^*$ so that each patch is centered exactly at $\mathbf{g}_j^*$ with size $l \times l$. For each $\mathbf{g}_j^*$, we obtain a patch $\bar{\mathbf{u}}_j$ via

$$\bar{\mathbf{u}}_j = \mathrm{Crop}(\mathbf{I}, \mathbf{g}_j^*), \quad \bar{\mathbf{u}}_j \in \mathbb{R}^{l\times l\times 3}, \tag{B.6}$$

where $\mathrm{Crop}(\mathbf{I}, \mathbf{g}_j^*)$ extracts an $l \times l$ patch from $\mathbf{I}$ centered at $\mathbf{g}_j^*$. Applying this to all $\mathbf{g}_j^* \in \mathbf{g}^*$ yields a new set of *adaptive* patches $\{\bar{\mathbf{u}}_j\}_{j=1}^M$ that replaces the original fixed-grid patches $\{\mathbf{u}_i\}_{i=1}^M$. Finally, we assign to each target grid center $\mathbf{g}_j$ the token derived from its corresponding adaptive patch $\bar{\mathbf{u}}_j$, which is explicitly centered at $\mathbf{g}_j^*$. Intuitively, this approach more faithfully respects the precise backward mappings in $f_{T\to S}$, at the cost of re-patchifying the image rather than relying on the original, efficient fixed-grid partitioning.

*We assume H and W are divisible by l.

C. Details on ViewBench

In this section, we provide additional details on the data synthesis protocol and evaluation metrics for `ViewBench`, introduced in Sec. 4 **in the main paper**.

C.1. Benchmark Construction

We construct `ViewBench` from real indoor scenes in ScanNet [9], which provides dense RGB-D frames along with ground-truth depth, camera poses, and intrinsics. For evaluations with estimated depth maps, we use Depth Anything v2 [29]. To sample two-view pairs with controlled overlap, we use the MultiSPA data engine from Xu et al. [27], originally introduced for generating multi-view VQA data. We adopt the same notions of *visible points* and *overlap ratio* as in MultiSPA and use them to construct `ViewBench` questions. Below, we detail the benchmark construction procedure, following the notation of MultiSpa [27].

Overlap Computation. For each ScanNet scene [9], we are given a 3D point cloud

$$\mathbf{P}_{\text{scene}} = \{\mathbf{p}^w\}, \quad \text{where } \mathbf{p}^w = [x^w, y^w, z^w]^\top, \tag{C.1}$$

with each point $\mathbf{p}^w$ expressed in the world coordinate system. Each RGB frame $\mathbf{I}_i \in \mathbb{R}^{H\times W\times 3}$ is associated with a depth map $\mathbf{D}_i \in \mathbb{R}^{H\times W\times 1}$, an extrinsic matrix $\mathbf{E}_i \in \mathbb{R}^{4\times 4}$, and an intrinsic matrix $\mathbf{K}_i \in \mathbb{R}^{4\times 4}$. The extrinsic matrix is defined as

$$\mathbf{E}_i := \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad \mathbf{R}_i \in \mathbb{R}^{3\times 3},\ \mathbf{t}_i \in \mathbb{R}^{3\times 1}, \tag{C.2}$$

where $\mathbf{R}_i$ and $\mathbf{t}_i$ denote the camera rotation and translation, respectively.

Following MultiSPA [27], we map each world point $\mathbf{p}^w$ into the i-th camera coordinate system via

$$\tilde{\mathbf{p}}_i^c = (\mathbf{E}_i)^{-1}\tilde{\mathbf{p}}^w, \quad \text{where } \tilde{\mathbf{p}}^w = [\mathbf{p}^w, 1]^\top, \tag{C.3}$$

and denote $\tilde{\mathbf{p}}_i^c = [x_i^c, y_i^c, z_i^c, 1]^\top$. We then project this point to the image plane :

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{\mathbf{K}_i}{z_i^c} \begin{bmatrix} x_i^c \\ y_i^c \\ z_i^c \end{bmatrix}, \tag{C.4}$$

We define the set of *visible points* in frame i as:

$$\mathcal{V}_i = \left\{ \mathbf{p}^w \in \mathbf{P}_{\text{scene}} \,\middle|\, 0 < z_i^c < d_i(u,v) \right\}, \tag{C.5}$$

where $d_i(u,v)$ is the depth value at pixel (u,v) from $\mathbf{D}_i$. This captures points whose projections fall inside $\mathbf{I}_i$ and are not occluded according to $\mathbf{D}_i$, which is identical to the visibility criterion of MultiSPA [27].

Finally, given two frames $\mathbf{I}_i$ and $\mathbf{I}_j$, we measure how much of the scene they see in common using the IoU of their visible point sets, defining the *overlap ratio* [27]:

$$\text{Overlap}(i,j) = \frac{\left|\mathcal{V}_i \cap \mathcal{V}_j\right|}{\left|\mathcal{V}_i \cup \mathcal{V}_j\right|}. \tag{C.6}$$

We use this overlap ratio to create controlled splits in `ViewBench`.

Two-View Pair Selection. For each ScanNet scene, we enumerate candidate frame pairs and compute the overlap ratio defined above. We retain two-view a pair $(\mathbf{I}_S, \mathbf{I}_T)$ as a candidate if $\text{Overlap}(S,T)$ lies in a moderate range (approximately 5–35%), so that the two views are neither nearly identical nor almost disjoint. Following the overlap-aware sampling strategy of MultiSPA [27], we bin all non-zero-overlap pairs by their overlap ratio and sample an approximately equal number of pairs from each bin to mitigate the natural long-tailed bias toward small overlaps. We then group the selected pairs into three overlap levels: **5–15%**, **15–25%**, and **25–35%**. This categorization allows us to systematically study how viewpoint-conditioned reasoning changes as the amount of shared scene content varies.

Point Annotation. For each selected source–target pair $(\mathbf{I}_S, \mathbf{I}_T)$, we focus on the points that are visible in *both* views, that is, the co-visible set $\mathcal{V}_S \cap \mathcal{V}_T$. For any $\mathbf{p}^w$ in this intersection, we obtain its camera-frame coordinates in each view via

$$\tilde{\mathbf{p}}_S^c = (\mathbf{E}_S)^{-1}\tilde{\mathbf{p}}^w, \quad \tilde{\mathbf{p}}_T^c = (\mathbf{E}_T)^{-1}\tilde{\mathbf{p}}^w, \tag{C.7}$$

and then project them to the image planes using the same camera model as in Eq. C.4. These co-visible projections form the pool of candidate keypoints used to construct task-specific questions, analogous to the *visual correspondence* subset construction in MultiSPA [27].

For `ViewBench-Text`, we randomly sample two co-visible points and annotate them with alphabet labels (*i.e.*, A/B). For `ViewBench-Shape`, we instead mark them with simple geometric symbols (e.g., triangle, star). In all cases, annotations in the two views are guaranteed to correspond to the same underlying 3D locations.

Selecting View-Dependent Point Pairs. Given a source–target pair $(\mathbf{I}_S, \mathbf{I}_T)$ and its co-visible point set $\mathcal{V}_S \cap \mathcal{V}_T$, we construct left–right queries by sampling two co-visible 3D points and projecting them into both images (using the same intrinsics, extrinsics, and visibility checks as above). Let u_A^S, u_B^S and u_A^T, u_B^T denote the u-coordinates of the two keypoints (A and B) in the source and target views, respectively. We retain a pair only if

$$(u_A^S - u_B^S)\,(u_A^T - u_B^T) < 0 \quad \text{and} \quad |u_A^T - u_B^T| \geq \tau, \tag{C.8}$$

with $\tau = 50$ pixels to avoid near-vertical alignments. Thus, we keep only examples where the left–right relation flips between views and is sufficiently separated in the target, ensuring that the correct answer genuinely depends on adopting the target viewpoint.

Instruction Generation. For each source-target pair, we convert the annotations into instruction–answer examples to be input to MLLMs. We render task-specific visual markers: alphabet labels for `ViewBench-Text`, geometric symbols for `ViewBench-Shape`, and a single red circular marker for `ViewBench-Object`.

For `ViewBench-Text` and `ViewBench-Shape`, we pose a binary left–right question about the two markers in the *target* view, randomly ordering the options (*e.g.*, *"left, right"* vs. *"right, left"*). The ground-truth label is computed deterministically from the x-coordinates of the two keypoints in the target image. For `ViewBench-Object`, we instead use a fixed open-ended template (*e.g.*, *"Can you describe the object or feature at the red point?"*) and treat the MLLM's response on the oracle target image as the reference description.

After applying the full data-processing pipeline, we obtain:

- **571** text questions (`ViewBench-Text`),
- **744** shape questions (`ViewBench-Shape`),
- **300** object-description samples (`ViewBench-Object`),

all validated using the target-view oracle and co-visibility constraints.

C.2. Details on ViewBench-Object Evaluation

As noted in Sec. 4 **of the main paper**, to evaluate MLLM responses on the target-view object description task (`ViewBench-Object`), we use an LLM (Qwen2.5-14B-Instruct [5]) as an evaluator, asking it to rate each response on a 1–10 scale. For this, we query the evaluator LLM with the following prompt template:

```
You are an AI assistant who will help
me to evaluate the response given the
question and the correct answer.
To mark a response, you should output
a single integer between 1 and 10
(including 1, 10).

- 10 means that the response is
describing the same or similar scene
as the answer.

- 1 means that the response is describing
a completely different scene from the
answer.

Question: {question}
Answer: {answer}
Response: {response}

Please output in format
<score>...</score>.
```

References

[1] Remyx AI. Spaceqwen2.5-vl-3b-instruct. https://huggingface.co/remyxai/SpaceQwen2.5-VL-3B-Instruct, 2025. 1

[2] Remyx AI. Spacethinker-qwen2.5vl-3b. https://huggingface.co/remyxai/SpaceThinker-Qwen2.5VL-3B, 2025. 1

[3] Remyx AI. Vqasynth. https://github.com/remyxai/VQASynth, 2025. 1

[4] Xiang An, Yin Xie, Kaicheng Yang, Wenkang Zhang, Xiuwei Zhao, Zheng Cheng, Yirui Wang, Songcen Xu, Changrui Chen, Chunsheng Wu, et al. Llava-onevision-1.5: Fully open framework for democratized multimodal training. *arXiv preprint arXiv:2509.23661*, 2025. 1, 2

[5] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025. 1, 2, 3, 8

[6] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, et al. Arkitscenes: A diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. In *NeurIPS*, 2021. 4

[7] Aleksei Bochkovskii, AmaÃĢl Delaunoy, Hugo Germain, Marcel Santos, Yichao Zhou, Stephan R Richter, and Vladlen Koltun. Depth pro: Sharp monocular metric depth in less than a second. In *ICLR*, 2025. 2

[8] Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *CVPR*, 2024. 1, 2

[9] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 3, 7

[10] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-Scale Embodied AI Using Procedural Generation. In *NeurIPS*, 2022. Outstanding Paper Award. 3

[11] Zhiwen Fan, Jian Zhang, Renjie Li, Junge Zhang, Runjin Chen, Hezhen Hu, Kevin Wang, Huaizhi Qu, Dilin Wang, Zhicheng Yan, et al. Vlm-3r: Vision-language models augmented with instruction-aligned 3d reconstruction. *arXiv preprint arXiv:2505.20279*, 2025. 1, 2

[12] Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. In *ECCV*, 2024. 5

[13] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. 1

[14] 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 *ICLR*, 2025. 1, 2

[15] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *CVPR*, 2024. 5

[16] Gen Luo, Ganlin Yang, Ziyang Gong, Guanzhou Chen, Haonan Duan, Erfei Cui, Ronglei Tong, Zhi Hou, Tianyi Zhang, Zhe Chen, et al. Visual embodied brain: Let multimodal large language models see, think, and control in spaces. *arXiv preprint arXiv:2506.00123*, 2025. 1, 2

[17] Wufei Ma, Yu-Cheng Chou, Qihao Liu, Xingrui Wang, Celso de Melo, Jianwen Xie, and Alan Yuille. Spatialreasoner: Towards explicit and generalizable 3d spatial reasoning. In *NeurIPS*, 2025. 2

[18] 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. 1, 2

[19] Junyoung Seo, Kazumi Fukuda, Takashi Shibuya, Takuya Narihira, Naoki Murata, Shoukang Hu, Chieh-Hsin Lai, Seungryong Kim, and Yuki Mitsufuji. Genwarp: Single image to novel views with semantic-preserving generative warping. In *NeurIPS*, 2024. 2

[20] BAAI RoboBrain Team, Mingyu Cao, Huajie Tan, Yuheng Ji, Xiansheng Chen, Minglan Lin, Zhiyu Li, Zhou Cao, Pengwei Wang, Enshen Zhou, et al. Robobrain 2.0 technical report. *arXiv preprint arXiv:2507.02029*, 2025. 1, 2

[21] Kimi Team, Angang Du, Bohong Yin, Bowei Xing, Bowen Qu, Bowen Wang, Cheng Chen, Chenlin Zhang, Chenzhuang Du, Chu Wei, et al. Kimi-vl technical report. *arXiv preprint arXiv:2504.07491*, 2025. 1, 2

[22] Peter Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Adithya Jairam Vedagiri IYER, Sai Charitha Akula, Shusheng Yang, Jihan Yang, Manoj Middepogu, Ziteng Wang, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. In *NeurIPS*, 2024. 1, 2

[23] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *CVPR*, 2025. 1, 2, 3

[24] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. In *CVPR*, 2025. 1

[25] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *CVPR*, 2024. 2

[26] 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. In *NeurIPS*, 2025. 2

[27] Runsen Xu, Weiyao Wang, Hao Tang, Xingyu Chen, Xiaodong Wang, Fu-Jen Chu, Dahua Lin, Matt Feiszli, and Kevin J Liang. Multi-spatialmllm: Multi-frame spatial understanding with multi-modal large language models. *arXiv preprint arXiv:2505.17015*, 2025. 7

[28] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang,

Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. 1, 2

[29] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. In *NeurIPS*, 2024. 2, 7

[30] 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*, 2025. 1, 2

[31] Baiqiao Yin, Qineng Wang, Pingyue Zhang, Jianshu Zhang, Kangrui Wang, Zihan Wang, Jieyu Zhang, Keshigeyan Chandrasegaran, Han Liu, Ranjay Krishna, et al. Spatial mental modeling from limited views. In *ICLR*, 2026. 1, 2

[32] Duo Zheng, Shijia Huang, Yanyang Li, and Liwei Wang. Learning from videos for 3d world: Enhancing mllms with 3d vision geometry priors. In *NeurIPS*, 2025. 1, 2

[33] Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025. 1, 2