

# Learning Multi-View Spatial Reasoning from Cross-View Relations

## Supplementary Material

### 7. Outline

This supplementary material provides additional technical details, experimental analysis, and formal definitions omitted from the main text due to space constraints. The appendices are organized as follows:

- Appendix 8: Formal mathematical definitions of the three task categories and the question-answer generation framework.
- Appendix 9: Detailed task generation pipeline covering geometry-based and metadata-based generation methods.
- Appendix 10: Filtering methodology for Temporal Verification.
- Appendix 11: Dataset statistics including source data composition and XVR distribution analysis.
- Appendix 12: Additional analysis of XVR-Eval benchmark.
- Appendix 13: Task-by-task performance analysis on external benchmarks (MindCube-Tiny and RoboSpatial-Home).
- Appendix 14: Additional experimental results on model and data scaling.
- Appendix 15: Training hyperparameters and procedures for VLM fine-tuning and VLA policy training.
- Appendix 16: Qualitative examples of XVR tasks.

### 8. Formal Task Definitions

We provide formal mathematical definitions for XVR’s three task categories. While Section 3 introduces these categories conceptually, this appendix formalizes the reasoning objectives and generation framework underlying each task. Table 3 shows how the general formulation instantiates into eight specific tasks.

#### 8.1. Category

We formalize cross-view relations as a general relation between multi-view observations. Given a set of images  $\mathcal{I} = \{I_i\}$  captured from different viewpoints, each reasoning objective is represented by a relation function:

$$g_{\text{task}}(\mathcal{I}_r, \mathcal{I}_j, e_j) = \begin{cases} c(\cdot) & \text{for Correspondence} \\ v(\cdot) & \text{for Verification} \\ l(\cdot) & \text{for Localization.} \end{cases} \quad (1)$$

Here,  $e_j$  denotes an optional element associated with view  $I_j$  (e.g., a point or direction), and the functions  $c(\cdot)$ ,  $v(\cdot)$ , and  $l(\cdot)$  capture complementary notions of geometric relationships across views.

We define three representative constraints corresponding to the three task categories:

#### (1) Correspondence:

$$e^* = \arg \max_{e \in E_j} c(\mathcal{I}_r, I_j, e) \quad (2)$$

This objective finds the element  $e^*$  (e.g., a point or direction) in  $I_j$  that best corresponds to the same physical entity observed in  $\mathcal{I}_r$ .

#### (2) Verification:

$$j^* = \arg \min_j v(\mathcal{I}; \Theta) \quad (3)$$

where  $\Theta$  denotes the evaluation criterion (e.g., spatial consistency or temporal alignment). It identifies the view that shows the lowest consistency with the cross-view relations established by  $\mathcal{I}$ .

#### (3) Localization:

$$j^* = \arg \max_j l(\mathcal{I}; \phi) \quad (4)$$

where  $\phi$  represents a spatial or semantic condition (e.g., “left of the reference camera” or a textual description). This constraint selects the view  $I_{j^*}$  that best satisfies the given condition.

Table 3. Taxonomy of cross-view reasoning tasks, summarized with their formal definitions and representative QA prototypes.

Task	Formulation	Question Prototype
<b>Correspondence</b>		
Point Correspondence	$p^* = \arg \max_{p \in P_j} c(\mathcal{I}_r, I_j, p)$	$Q$ : Which point in $I_j$ corresponds to this point in $\mathcal{I}_r$ ?
Directional Correspondence	$a^* = \arg \max_{a \in A_j} c(\mathcal{I}_r, I_j, a)$	$Q$ : Which arrow in $I_j$ points in the same direction as in $\mathcal{I}_r$ ?
<b>Verification</b>		
Spatial Verification	$j^* = \arg \min_j v(\mathcal{I}; \mathcal{P})$	$Q$ : Which point $p$ in each image breaks spatial alignment?
Temporal Verification	$j^* = \arg \min_j v(\mathcal{I}; \mathcal{T})$	$Q$ : Which view was captured at a different timestamp?
<b>Localization</b>		
Viewpoint Localization	$j^* = \arg \max_j l(\mathcal{I}; \mathcal{P})$	$Q$ : Which view corresponds to the camera at point $\mathcal{P}$ ?
Directional View Localization	$j^* = \arg \max_j l(\mathcal{I}; \text{dir})$	$Q$ : Which view lies to the right of the reference camera?
Cross-Scenario Localization	$j^* = \arg \max_j l(\mathcal{I}^{(1)}; \mathcal{I}^{(2)})$	$Q$ : Which camera in Scene 2 matches the viewpoint of Scene 1?
Language-conditioned Localization	$j^* = \arg \max_j l(\mathcal{I}; \phi_{\text{text}})$	$Q$ : Which view matches the description “wrist-mounted camera”?

## 8.2. Question-Answer Generation

The QA generation pipeline instantiates these formal definitions into concrete question-answer pairs from multi-view data. Formally, each generation instance is defined as:

$$\mathcal{G} : (\mathcal{I}, \mathcal{P}, \mathcal{X}, \mathcal{T}, \mathcal{M}) \rightarrow (\mathcal{Q}, \mathcal{A}) \quad (5)$$

where  $\mathcal{P} = \{(R_i, t_i)\}$  represents camera parameters (intrinsic/extrinsic),  $\mathcal{X}$  denotes geometric structure such as 3D point clouds,  $\mathcal{T} = \{t_i\}$  provides temporal indices for synchronization across frames, and  $\mathcal{M}$  contains auxiliary metadata (e.g., robot poses or camera identifiers).

Each QA pair  $(\mathcal{Q}, \mathcal{A})$  is derived from a rule-based template that maps these multimodal inputs into reasoning objectives across the three categories: correspondence, verification, and localization. Specifically,  $\mathcal{Q}$  specifies a relational query following the prototypes in Table 3, and  $\mathcal{A}$  provides the correct answer identifying either a view or an element within a view.

## 9. Task Generation Pipeline

Figure 6 illustrates the complete pipeline for generating XVR tasks from multi-view data. The pipeline branches based on data source characteristics: geometry-based generation for tasks leveraging explicit 3D information (general domain), and metadata-based generation for tasks utilizing trajectory annotations (robotic domain).

### 9.1. Geometry-Based Generation

Geometry-based generation applies to tasks requiring precise 3D geometric information: Point Correspondence, Directional Correspondence, Spatial Verification, and Viewpoint Localization. The pipeline operates on general domain data with calibrated cameras and dense point clouds.

**Input Processing.** The pipeline receives multi-view images  $\mathcal{I}$ , camera parameters  $\mathcal{P}$  (intrinsic and extrinsic), and 3D point clouds  $\mathcal{X}$ . Point clouds provide explicit geometric structure for visibility analysis and 3D-to-2D projection.

**Target Selection.** For correspondence tasks, the pipeline samples 3D points or directions from  $\mathcal{X}$  that are visible across multiple views. For localization tasks, it samples camera positions from  $\mathcal{P}$ . Visibility checking ensures selected targets can be reliably projected onto reference and target views without occlusion.

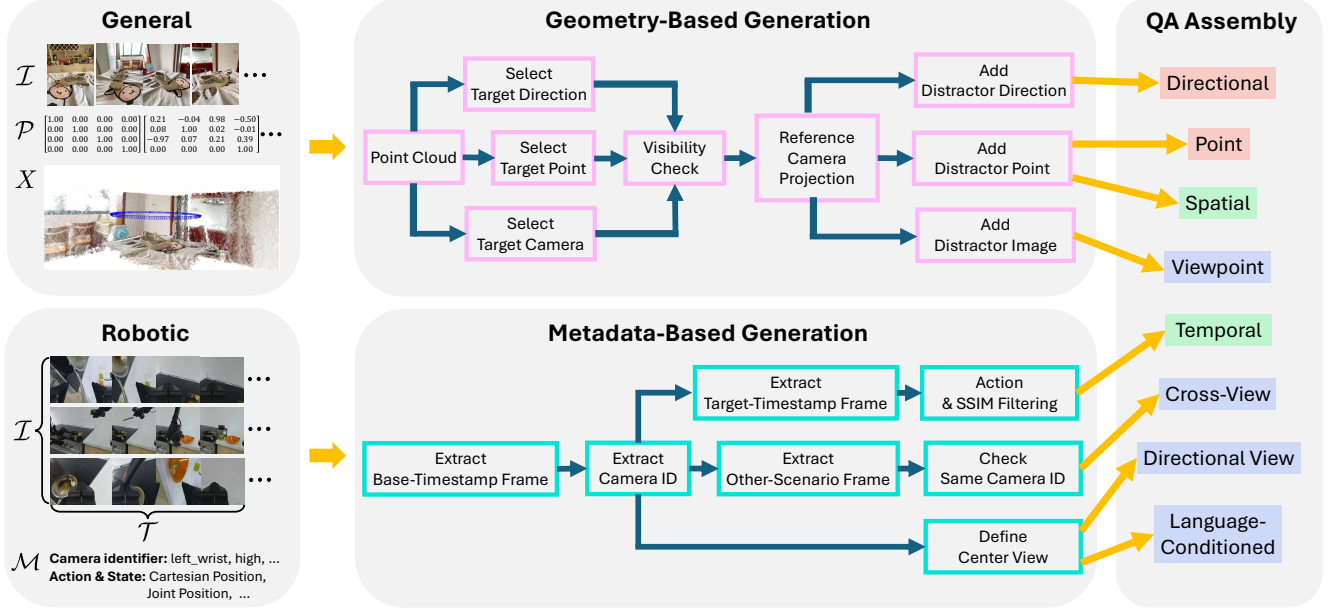


Figure 6. Task generation pipeline for XVR. The pipeline branches into geometry-based generation (top) for tasks using 3D geometric information and metadata-based generation (bottom) for tasks using trajectory annotations. Geometry-based generation processes general domain data ( $\mathcal{I}, \mathcal{P}, \mathcal{X}$ ) through 3D-to-2D projection and visibility checking to create Point, Directional, Spatial, and Viewpoint tasks. Metadata-based generation processes robotic domain data ( $\mathcal{I}, \mathcal{T}, \mathcal{M}$ ) through temporal and camera metadata extraction to create Temporal, Cross-View, Directional View, and Language-Conditioned tasks. Both pipelines converge at QA assembly to produce final question-answer pairs.

**Projection and Distractor Generation.** Selected 3D targets are projected onto 2D image planes using camera parameters. The pipeline then generates spatially separated distractors to create challenging multiple-choice questions. For Point and Directional Correspondence, distractors are alternative points or directions within the target view. For Spatial Verification, one view receives an intentionally inconsistent point. For Viewpoint Localization, distractor images come from other camera positions.

## 9.2. Metadata-Based Generation

Metadata-based generation applies to tasks utilizing trajectory information: Temporal Verification, Directional View Localization, Cross-Scenario Localization, and Language-Conditioned Localization. The pipeline operates on robotic domain data with temporal sequences and camera metadata.

**Base Frame and Camera Identification.** The pipeline first extracts a base-timestamp frame and its associated camera identifier from metadata  $\mathcal{M}$  (e.g., "left\_wrist", "high"). This base frame serves as the reference point for all metadata-based tasks. Camera identifiers enable matching corresponding viewpoints across different trajectories and time steps.

**Target Frame Extraction.** For Temporal Verification, the pipeline searches for candidate frames at different timestamps that exhibit perceptually distinguishable changes, validated through action magnitude and SSIM filtering (Appendix 10). For Cross-Scenario Localization, it extracts frames from different trajectory scenarios and identifies those captured from the same camera identifier as the base frame.

**Spatial and Semantic Matching.** For Directional View Localization, the pipeline defines the base frame as the center reference and identifies which camera positions lie in specified directions (left, right, front, back) based on robot state information in  $\mathcal{M}$ . For Language-Conditioned Localization, it matches camera metadata against natural language descriptions to identify views satisfying textual spatial conditions (e.g., "wrist-mounted camera").

### 9.3. QA Assembly

Both pipelines converge at the QA assembly stage, which constructs question-answer pairs following the templates in Table 3. Each task receives task-specific inputs from its generation pipeline and produces structured QA pairs with reference images, target options, and correct answers.

## 10. Filtering for Temporal Verification

Temporal Verification task requires pairs of frames with sufficient perceptual and physical differences to ensure meaningful reasoning. We employ a two-stage filtering pipeline: (1) trajectory-level pre-filtering to remove statically inactive episodes, and (2) frame-pair validation using action-based and perceptual criteria.

### 10.1. Trajectory-Level Pre-filtering

We first eliminate trajectories with minimal physical activity by computing per-episode action variance. For each trajectory  $\mathcal{T} = \{a_1, a_2, \dots, a_T\}$  where  $a_t$  denotes the action at timestep  $t$ , we calculate:

$$V(\mathcal{T}) = \sum_{t=1}^T \|a_t\|_2 \quad (6)$$

Trajectories in the bottom 20% of action variance are filtered out, as they typically represent statically held positions with insufficient motion dynamics for temporal verification task.

### 10.2. Dataset-Specific Motion Statistics

For datasets with action metadata, we compute motion statistics to establish dynamic thresholds. For each trajectory, we measure the maximum state displacement over 1-second intervals:

$$M_{1\text{sec}} = \max_{t=1}^{T-f} \|\mathbf{s}_{t+f} - \mathbf{s}_t\|_2 \quad (7)$$

where  $\mathbf{s}_t$  denotes the robot state (end-effector position or joint angles) at timestep  $t$ , and  $f$  is the control frequency corresponding to 1 second of motion. We compute percentile statistics of  $M_{1\text{sec}}$  at 10% intervals (10th, 20th, ..., 90th percentiles) across all trajectories in each dataset. The 80th percentile of this distribution, denoted as  $\tau_{\text{act}}^{(d)}$  for dataset  $d$ , serves as the action-based threshold for frame-pair validation.

### 10.3. Frame-Pair Validation

Given a candidate frame pair  $(I_r, I_t)$  at timestamps  $(t_r, t_t)$ , we apply two complementary filters:

**Action Filtering.** For datasets with action metadata, we compute the state displacement between the two frames:

$$\Delta s = \|\mathbf{s}_{t_t} - \mathbf{s}_{t_r}\|_2 \quad (8)$$

The pair is retained only if:

$$\Delta s > \tau_{\text{act}}^{(d)} \quad (9)$$

This ensures sufficient physical motion occurred between frames.

**SSIM Filtering.** To prevent near-identical frames from being included, we compute the Structural Similarity Index (SSIM) [60] between grayscale versions of the two frames. Converting to grayscale emphasizes structural changes over color variations, making the filter more sensitive to meaningful geometric transformations:

$$\text{SSIM}(I_r, I_t) = \frac{(2\mu_r\mu_t + C_1)(2\sigma_{rt} + C_2)}{(\mu_r^2 + \mu_t^2 + C_1)(\sigma_r^2 + \sigma_t^2 + C_2)} \quad (10)$$

where  $\mu_r, \mu_t$  are mean luminance values,  $\sigma_r^2, \sigma_t^2$  are variances,  $\sigma_{rt}$  is the covariance, and  $C_1, C_2$  are stabilization constants. Frames are discarded if:

$$\text{SSIM}(I_r, I_t) > \tau_{\text{ssim}} \quad (11)$$



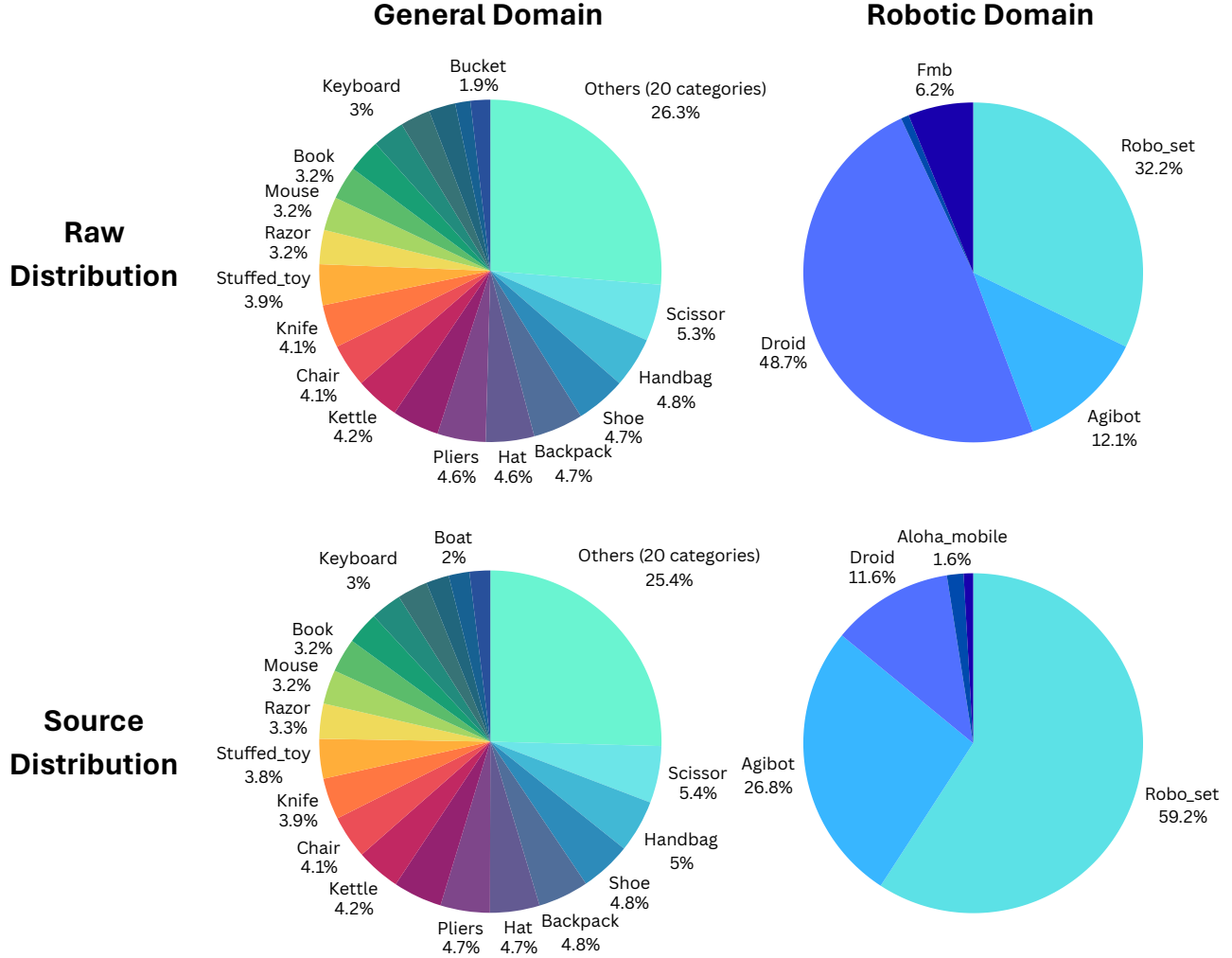


Figure 7. Source data distribution for XVR. **Top row (Raw Distribution):** Original distribution of data sources before task generation. General domain: 18,409 scenes from WildRGB-D. Robotic domain: 35,717 trajectories from DROID, RoboSet, Agibot, MobileAloha, and FMB. **Bottom row (Source Distribution):** Distribution after filtering and task generation. General domain: 51,788 samples (visibility-based filtering affects category proportions). Robotic domain: 51,788 samples (DROID and FMB limited to Temporal Verification due to inconsistent camera metadata; RoboSet and Agibot support all robotic tasks).

**Dataset-Specific Thresholds.** For datasets with action metadata, we apply both filters with  $\tau_{\text{ssim}} = 0.9$ . Both conditions must be satisfied for a frame pair to be retained. For datasets lacking action metadata, we apply only SSIM filtering with a more stringent threshold of  $\tau_{\text{ssim}} = 0.8$  to compensate for the absence of motion-based validation.

This combined filtering strategy ensures that Temporal Verification samples contain pairs of frames with both sufficient physical motion and perceptual distinctiveness, enabling models to learn meaningful temporal reasoning rather than relying on trivial visual cues.

## 11. Dataset Statistics

### 11.1. Source Data Distribution

Figure 7 illustrates the distribution of source data used to construct XVR. We distinguish between raw distribution (the original data sources) and source distribution (how frequently each source contributes to XVR samples). A single raw trajectory or scene may generate multiple QA samples across different tasks, causing the source distribution to differ from the raw

Table 4. XVR Dataset Statistics (103,576 samples, 447,811 images)

Category	Metric	Value
Questions	Avg. length (chars)	257.7
	Median length (chars)	236.5
	Min length (chars)	25
	Max length (chars)	508
Choices	Avg. per question	3.7
	Median	3.0
	Min	3
	Max	6
Views per QA	Avg. views per QA	4.32
	Median views	4.0
	Min views	3
	Max views	6
Image Resolution	Avg. resolution	475×481 px
	Most common	480×640 (53.56%)
Unique Resolutions	480×640	239,837 (53.56%)
	424×240	133,501 (29.81%)
	640×480	54,297 (12.12%)
	320×180	18,276 (4.08%)
	256×256	1,900 (0.42%)
Answer Distribution	1	6,948 (6.71%)
	2	19,951 (19.26%)
	3	21,281 (20.55%)
	4	14,957 (14.44%)
	5	8,337 (8.05%)
	6	6,208 (5.99%)
	red	5,844 (5.64%)
	blue	5,957 (5.75%)
	green	5,756 (5.56%)
	purple	2,538 (2.45%)
	yellow	5,799 (5.60%)

distribution.

**General Domain.** The general domain comprises diverse object categories from WildRGB-D [61]. The raw distribution shows balanced coverage across object categories, with the top categories being Scissor (5.3%), Handbag (4.8%), Shoe (4.7%), and Backpack (4.7%), alongside 20 additional categories. The source distribution shifts due to visibility constraints. Tasks requiring 3D-to-2D projection can only be generated when geometric entities are sufficiently visible across multiple views. Scenes with limited viewpoint coverage contribute fewer samples, altering category proportions.

**Robotic Domain.** The robotic domain draws from DROID [26], MobileAloha [19], RoboSet [29], FMB [38], and AgiBot-World [7]. The source distribution differs substantially from raw due to filtering criteria detailed in Appendix 10. DROID and FMB contain inconsistent camera metadata, limiting their use to Temporal Verification only. RoboSet with consistent metadata supports all robotic tasks and increases proportionally in the source distribution. Additionally, trajectories satisfying multiple task requirements generate more samples, further amplifying representation differences.

## 11.2. Final Dataset Statistics

Table 4 presents comprehensive statistics for the final XVR dataset. The dataset contains 103,576 QA samples derived from 447,811 images, with an average of 4.32 views per question. Questions average 257.7 characters with 3.7 answer choices. Image resolutions vary across five distinct sizes, with 480×640 being most common (53.56%). Answer distribution is approximately balanced across numeric choices (1-6) and color markers (red, blue, green, yellow, purple), ensuring no systematic bias toward specific answer positions.

## 12. Additional XVR-Eval Analysis

### 12.1. Benchmark Composition

Table 5 presents the distribution of samples across the eight tasks in XVR-Eval. The benchmark contains 1,866 samples spanning all three task categories: Correspondence, Verification, and Localization. Random baseline accuracies vary by task structure, ranging from 20% for tasks with average five candidate views to 50% for binary verification tasks. The overall random baseline across all tasks is 32.64%.

Table 5. XVR-Eval task distribution and random baseline accuracy.

Task	Samples	Random Baseline
Point Correspondence	170 (9.11%)	20.00%
Directional Correspondence	221 (11.84%)	25.00%
Spatial Verification	221 (11.84%)	22.22%
Temporal Verification	221 (11.84%)	33.33%
Viewpoint Localization	264 (14.15%)	33.33%
Directional View Localization	264 (14.15%)	50.00%
Cross-Scenario Localization	264 (14.15%)	33.33%
Language-Conditioned Localization	241 (12.92%)	50.00%
Total	1,866 (100%)	32.64%

### 12.2. Out-of-Distribution Design

XVR-Eval is constructed from data sources explicitly excluded from the XVR training set to evaluate generalization capabilities. As mentioned in Section 4.1 of the main paper, we include two distinct held-out sources:

**General Domain.** We use the boat category from WildRGB-D [61], which was completely excluded during XVR training. As shown in Figure 7, the training set comprises 39 other object categories, while boat represents only 2% of the general domain and were reserved for evaluation. This tests whether models can transfer cross-view spatial reasoning learned from objects with structured shapes (chairs, keyboards, scissors, etc.) to a distinct object category with different geometric characteristics.

**Robotic Domain.** We include trajectories from MobileAloha [19], a dataset not present in the XVR training distribution. As shown in Figure 7, the robotic training data consists of RoboSet (59.2%), AgiBot (26.8%), DROID (11.6%), and FMB, while MobileAloha was reserved exclusively for evaluation. This evaluates whether cross-view relation reasoning transfers to an unseen embodiment with different hardware configurations and camera setups.

This OOD design ensures XVR-Eval measures genuine generalization rather than memorization of training distributions. The consistent improvements observed on XVR-Eval (Table 2) despite these distribution shifts validate that cross-view relation supervision enables models to learn transferable multi-view spatial reasoning principles.

### 12.3. Qualitative Examples

We present representative examples from XVR-Eval to illustrate the cross-view reasoning challenges and model performance patterns. Figures 8 and 9 show two samples with predictions from four models: GPT-5, Gemini-Robotics-ER-1.5, Claude-4.5-Sonnet, and our Qwen3-VL-2B-XVR.

**Cross-Scenario Localization (Figure 8).** This task requires matching corresponding viewpoints across structurally similar but distinct scenes. The example demonstrates the challenge of identifying equivalent camera positions when object arrangements differ between scenarios. Among the evaluated models, only Qwen3-VL-2B-XVR and Gemini-Robotics-ER-1.5 correctly identify the matching viewpoint, demonstrating that explicit cross-view relation supervision enables robust viewpoint localization even under scene-level variations. GPT-5 and Claude-4.5-Sonnet fail on this sample, suggesting that scale alone does not guarantee consistent cross-scenario reasoning capabilities.

**Spatial Verification (Figure 9).** This task requires identifying which view contains a marker at an inconsistent 3D location compared to the others. The example presents multiple views where one view shows a marker placed at a spatially inconsistent position that violates cross-view geometric constraints. Only Qwen3-VL-2B-XVR successfully identifies the inconsistent view, while all other models fail. This demonstrates that XVR training enables precise geometric coherence checking across views—a capability that closed-source models struggle with despite their larger scale.

These examples validate the effectiveness of explicit cross-view relation supervision. While closed-source models show inconsistent performance, XVR-trained models demonstrate robust reasoning on both localization across scenarios and spatial consistency verification—the core components of cross-view relation reasoning.

**Cross-Scenario Localization**

**[Question]**

You will be shown several images captured during a robotic manipulation task. Each image comes with a description of the camera position from which it was taken. Please use this information to help answer the following question.

Scenario 1:  
 Image 1 (ego\_left): This image was captured from a wrist-mounted camera on the robot's left arm.  
 <Image 1>

Image 2 (fix\_center): The center fixed camera captures this front-facing view of the manipulation scene.  
 <Image 2>


Image 3 (ego\_right): This image was captured from a wrist-mounted camera on the robot's right arm.  
 <Image 3>


Scenario 2:  
 Image 4:  
 <Image 4>


Image 5:  
 <Image 5>


Image 6:  
 <Image 6>


In the first scenario, Image 1 represents the ego\_left camera perspective. Now examine Scenario 2 to find the image that shows the same camera orientation. Consider the spatial positioning, viewing angle, and visual characteristics of each image. Which image number in Scenario 2 displays the ego\_left view corresponding to Scenario 1's Image 1?

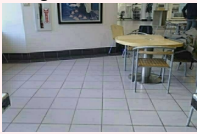
<Image 1>  


<Image 2>  



<Image 3>  


<Image 4>  



<Image 5>  


<Image 6>  



**[Answer]**




④



⑤



④



⑤





 GPT-5
 Gemini-Robotics-ER-1.5
 Claude-4.5-Sonnet
 Qwen3-VL-2B-XVR (Ours)

Figure 8. Cross-Scenario Localization example from XVR-Eval. The task requires identifying the corresponding viewpoint across two different scenarios. Only Qwen3-VL-2B-XVR (Ours) and Gemini-Robotics-ER-1.5 correctly predict the answer.

Spatial Verification

**[Question]**  
 You will be shown multiple synchronized camera views with annotations.  
 Use these images to answer the question below.

Image 1:  
<Image 1>

Image 2:  
<Image 2>

Image 3:  
<Image 3>

Image 4:  
<Image 4>


Image 5:  
<Image 5>

Image 6:  
<Image 6>


Images 1 through 6 mark the same 3D feature with colored dots. One view appears inconsistent.  
 Which image number seems to place the marker most incorrectly?


<Image 1>  


<Image 2>  


<Image 3>  


<Image 4>  


<Image 5>  


<Image 6>  



**[Answer]**



①



②



⑥



⑤


 GPT-5
 Gemini-Robotics-ER-1.5
 Claude-4.5-Sonnet
 Qwen3-VL-2B-XVR (Ours)

Figure 9. Spatial Verification example from XVR-Eval. The task requires identifying which view contains a marker placed at an inconsistent 3D location compared to the others. Only Qwen3-VL-2B-XVR (Ours) correctly identifies the spatially inconsistent view, demonstrating the effectiveness of explicit cross-view relation supervision.

## 13. External Benchmark Analysis

### 13.1. MindCube

**Around (+0.5pp: 34.25% → 34.75%).** This task requires predicting which objects become visible after the camera rotates in a specified direction and then moves forward, combining rotation and translation transformations sequentially. XVR does not contain examples of directional camera motion followed by translation. XVR’s data consists of static multi-view observations where all cameras are fixed simultaneously. The task’s assumption about sequential viewpoint changes differs fundamentally from XVR’s multi-view setup, resulting in minimal transfer.

**Rotation (+0.5pp: 27.0% → 27.5%).** This task provides images captured as a camera rotates 360° around a central point and requires reasoning about the complete circular spatial layout. While this task involves multi-view reasoning, the

camera configuration is outside-looking-inward, which is completely absent from XVR’s training data. XVR focuses on inside-looking-outward setups (objects in front of cameras) from both general domain scenes and robotic manipulation. This distribution mismatch limits transfer despite both tasks requiring discrete multi-view understanding.

**Among (+7.0pp: 32.5% → 39.5%).** This task provides disjoint camera views where objects may be occluded or partially visible, and requires localizing a target object’s position relative to other objects by integrating information across views. The substantial improvement demonstrates that XVR successfully teaches models to understand relationships between spatial information and camera viewpoints. XVR’s three task categories collectively train models to establish geometric relationships between views, verify spatial consistency, and reason about camera-relative positions. Although XVR uses geometric points rather than objects as training targets, this learned capability to reason about how spatial information appears across different viewpoints generalizes to object-level localization.

### 13.2. RoboSpatial-Home

**Compatibility (+7.62pp: 49.52% → 57.14%).** This task provides a single-view image and requires determining whether a target object can physically fit into specified empty spaces, evaluating 3D size, shape, and spatial constraints from 2D observations. This represents the largest improvement across all external benchmarks. We attribute this to XVR creating 3D-aware representations through multi-view training. To establish correspondence across multiple 2D projections, models must learn to reason about underlying 3D structure, including spatial extent, depth relationships, and geometric constraints. This 3D reasoning capability, learned from multi-view supervision, transfers to single-view 3D tasks. The model can evaluate spatial fit and identify empty spaces from single views by applying the same geometric reasoning developed for multi-view consistency.

**Configuration (+1.80pp: 73.17% → 74.8%).** This task evaluates understanding of object-object spatial relations through classifying relations (on, in, next to), reasoning about composed relations, and localizing objects from spatial descriptions. XVR does not contain supervision on object-object spatial relationships. The training focuses on view-view correspondence and geometric consistency rather than semantic relations between objects within scenes. The modest improvement likely reflects general spatial reasoning transfer, though the high baseline limits potential gains. The lack of explicit object-relation supervision in XVR explains why this task shows smaller improvements compared to Compatibility.

## 14. Additional Experimental Results

Figure 10 presents additional experiments on XVR-Eval, including baseline comparisons across model scales and the impact of XVR training on model and data scaling.

**Spatial Reasoning Baselines (a).** We compare Qwen3-VL across model scales (2B, 8B, 30B-A3B) against models specifically designed for spatial reasoning: SpatialOM-3B (SO, 30.4%), SpatialMLLM-3B (SM, 31.2%), and RoboBrain2.0-32B (RB, 35.8%). Despite being purpose-built for spatial tasks, these models underperform even the smallest general-purpose Qwen3-VL-2B (36.8%), suggesting that cross-view spatial reasoning is not adequately covered by existing spatial reasoning datasets. Notably, RoboBrain2.0-32B, despite its 32B parameters, scores only 35.8% overall—below Qwen3-VL-2B on most tasks including Point Correspondence (31.4% vs. 46.6%) and Cross-Scenario Localization (41.2% vs. 41.6%).

**Model Scaling (b).** XVR training benefits extend beyond the 2B scale. Qwen3-VL-2B improves from 36.8% to 68.1% (+31.3pp) after XVR fine-tuning, and Qwen3-VL-8B similarly improves from 40.4% to 71.1% (+30.7pp), demonstrating consistent gains across model scales. Task-level improvements are also consistent: Point Correspondence improves from 65.5% to 98.1% (+32.6pp) on 8B, and Spatial Verification improves from 23.9% to 91.3% (+67.4pp), confirming that explicit cross-view supervision is broadly effective regardless of model capacity.

**Data Scaling (c).** We investigate scaling beyond 100K samples by generating additional data from the same source distribution. Overall performance improves consistently from 68.1% (100K) to 72.3% (300K) to 73.8% (500K). Task-level gains are particularly notable on geometry-intensive tasks: Point Correspondence reaches 98.9% at 500K, and Viewpoint Localization improves from 57.7% to 88.0%, confirming that cross-view reasoning benefits from larger training sets without saturating at 100K.



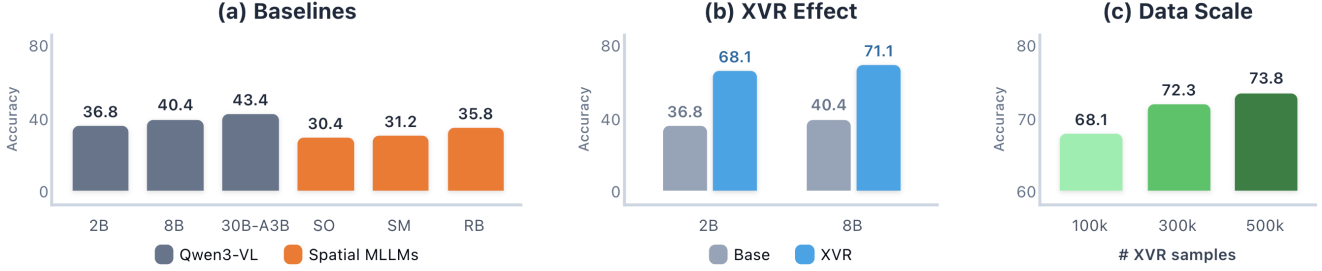


Figure 10. **Additional experiments on XVR-Eval.** (a) Baseline evaluation across Qwen3-VL model scales (2B, 8B, 30B-A3B) and spatial reasoning models: SpatialOM-3B (SO), SpatialMLLM-3B (SM), and RoboBrain2.0-32B (RB). (b) Impact of XVR training on Qwen3-VL-2B and 8B. (c) Scaling behavior of XVR training data (100k, 300k, 500k samples) on Qwen3-VL-2B.

## 15. Training Details

This section provides complete implementation details for reproducing our experimental results. We describe two training stages: (1) fine-tuning the vision-language model on XVR to acquire cross-view spatial reasoning capabilities (§15.1), and (2) extending the XVR-trained VLM into a Vision-Language-Action model for embodied manipulation tasks (§15.2).

The first stage trains Qwen3-VL-2B-Instruct on the 100K XVR dataset through full-parameter supervised fine-tuning, producing the Qwen3-VL-2B-XVR backbone used in all main paper experiments. The second stage integrates this XVR-trained backbone into a VLA architecture by adding a diffusion-based action head following the GR00T-N1.5 design. We evaluate the resulting policies on RoboCasa simulation tasks to assess the transfer of learned spatial representations to robotic control. All experiments are conducted on NVIDIA H100 GPUs using distributed training frameworks. We provide full hyperparameter specifications below to ensure reproducibility.

### 15.1. Fine-Tuning VLM on XVR

To equip the base vision-language model with explicit cross-view spatial reasoning capabilities, we fine-tuned Qwen3-VL-2B-Instruct on the full XVR dataset. All experiments were conducted using HuggingFace Accelerate, which internally adopts Distributed Data Parallel (DDP) across a single node with  $8 \times$  NVIDIA H100 GPUs. We perform full-parameter supervised fine-tuning and enabled both gradient checkpointing and gradient accumulation to efficiently process variable-resolution multi-view inputs.

The XVR dataset contains 103,576 multi-view QA samples. While the image resolutions vary, the average resolution is  $475 \times 481$ , and the most common resolution is  $480 \times 640$  (53.6% of all samples). A unified training script was used across all runs, with only the dataset configuration differing between experiments to ensure comparability. The full set of optimization hyperparameters is summarized in Table 6. The resulting model, denoted as **Qwen3-VL-2B-XVR**, serves as the spatial reasoning backbone for all VLM and VLA experiments presented in the main paper.

### 15.2. VLA Policy Training with XVR-Trained VLM

To evaluate whether XVR supervision can improve downstream robotic control, we train a Vision-Language-Action (VLA) policy using the Qwen3-VL-2B-XVR as the perceptual backbone. Our architecture follows the design of Isaac GR00T N1.5: visual observations and language instructions are processed through the VLM, while robot proprioceptive states and noised actions are fed as inputs to a DiT-based action head. The VLM output embedding (taken from the 12th transformer layer) is used as a conditioning signal inside the DiT layers via cross-attention, allowing the action transformer to attend to the XVR-encoded scene representation throughout the denoising process.

We evaluate two backbone variants: (1) replacing GR00T’s original Eagle2.5 VLM with Qwen3-VL-2B-Instruct, and (2) replacing it with our **Qwen3-VL-2B-XVR**. For both variants, we freeze the language model while fine-tuning the vision encoder and training the DiT action head from scratch. All policies are trained for 60,000 steps on three tasks—*CoffeePressButton*, *TurnOffMicrowave*, and *PnP CabToCounter*—sourced from the `nvidia/PhysicalAI-Robotics-GR00T-X-Embodiment-Sim` dataset. All evaluations are conducted in the RoboCasa simulator, and performance is measured using success rate at the 60k-step checkpoint.

The hyperparameters used for VLA policy training are summarized in Table 7.



Table 6. Training hyperparameters for XVR fine-tuning of Qwen3-VL-2B-Instruct.

Parameter	Value
Model	Qwen3-VL-2B-Instruct
Dataset size	103,576 QA pairs
Epochs	3
Learning rate	5e-5
Scheduler	Cosine
Fine-tuning type	Full-parameter
Global batch size	256
Per-device batch size	4
Gradient accumulation steps	8
GPUs used	8× NVIDIA H100 (1 node)
Distributed strategy	Accelerate (DDP)
Gradient checkpointing	True
Max grad norm	1
Precision	BF16
Optimizer	AdamW
Warmup ratio	0.05
Average image resolution	475×481
Most common resolution	480×640 (53.6%)

Table 7. Training hyperparameters for VLA policy learning.

Hyperparameter	Value
Backbone VLM	Qwen3-VL-2B-Instruct / Qwen3-VL-2B-XVR
Frozen components	Language model (LLM)
Fine-tuned components	Vision encoder, Projector, DiT action head
Action head	DiT (trained from scratch)
VLM feature layer used	Transformer layer 12
Tasks	CoffeePressButton, TurnOffMicrowave, PnPCabToCounter
Dataset	nvidia/PhysicalAI-Robotics-GR00T-X-Embodiment-Sim
Training steps	60,000
Batch size (per GPU)	16
Total batch size	128 (8 GPUs)
Learning rate	$1 \times 10^{-4}$
Weight decay	$1 \times 10^{-5}$
Optimizer	AdamW ( $\beta_1=0.95, \beta_2=0.999$ )
Scheduler	Cosine
Warmup ratio	0.05
Mixed precision	BF16
Hardware	8× NVIDIA H100
Evaluation metric	Success rate at 60k checkpoint

## 16. XVR Examples

This section provides visual examples for each of the eight tasks in XVR, illustrating the question format, reference images, and answer choices across the three task categories: Correspondence, Verification, and Localization (Figures 11–18).

## Point Correspondence

### [Question]

You will be shown multiple synchronized camera views with annotations.  
Use these images to answer the question below.

Image 1 (Reference):

<Image 1>

Image 2 (Reference):

<Image 2>

Image 3 (Reference):

<Image 3>

Image 4 (Reference):

<Image 4>

Image 5:

<Image 5>

The reference images (4 images) all show the same 3D point marked with colored dots. In the target image, identify which colored marker corresponds to this same 3D location.

Note: The marked point represents a 3D location in space that may not be on a visible object surface and can be behind objects.

Options: blue, green, purple, red, yellow.

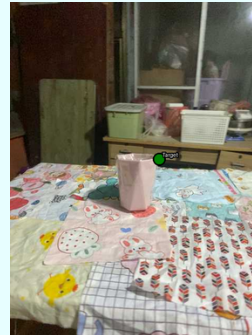
<Image 1>



<Image 2>



<Image 3>



<Image 4>



<Image 5>



### [Answer]

blue

green

purple

red

yellow

Figure 11. Point Correspondence task example. The question asks which point in the target image corresponds to a marked point in the reference images.

## Directional Correspondence

### [Question]

You will be shown multiple synchronized camera views with annotations.  
Use these images to answer the question below.

Image 1 (Reference):

<Image 1>

Image 2 (Reference):

<Image 2>

Image 3 (Reference):

<Image 3>

Image 4:

<Image 4>

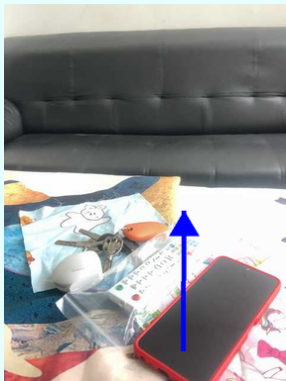
The reference images (3 images) all show the same 3D direction with colored arrows. In the target image, identify which arrow color represents this same direction.

Options: blue, green, red, yellow.

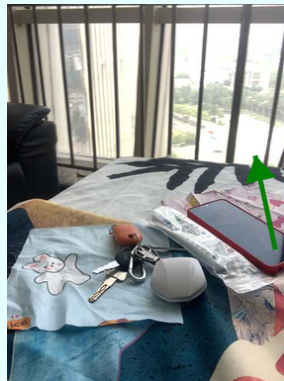
<Image 1>



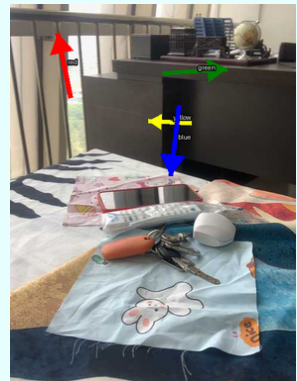
<Image 2>



<Image 3>



<Image 4>



### [Answer]

blue

green

red

yellow

Figure 12. Directional Correspondence task example. The question asks which arrow in the target image points in the same direction as arrows in the reference images.

## Spatial Verification

### [Question]

You will be shown multiple synchronized camera views with annotations.  
Use these images to answer the question below.

Image 1:

<Image 1>

Image 2:

<Image 2>

Image 3:

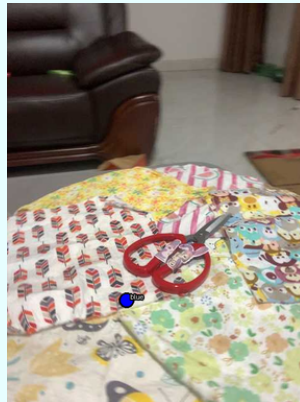
<Image 3>

Images 1 through 3 mark the same 3D feature with colored dots. One view appears inconsistent. Which image number seems to place the marker most incorrectly?

<Image 1>



<Image 2>



<Image 3>



### [Answer]

1

2

3

Figure 13. Spatial Verification task example. The question asks which marked point across multiple images breaks spatial alignment.

### Temporal Verification

#### [Question]

You will be shown several images captured during a robotic manipulation task. Each image comes with a description of the camera position from which it was taken. Please use this information to help answer the following question.

Image 1 (ego\_leftside):

<Image 1>

Image 2 (ego\_rightside):

<Image 2>

Image 3 (fix\_right): A right-side mounted camera provides this perspective of the robotic operation.

<Image 3>

Image 4 (fix\_left): This camera angle provides a left-side observation point of the manipulation task.

<Image 4>

Multiple camera views of a robotic operation are presented here.

While most images are temporally consistent, one frame was taken at a different moment.

Observe the spatial relationships, environmental conditions, and robot state variations.

Identify the image number that shows temporal inconsistency.

<Image 1>



<Image 2>



<Image 3>



<Image 4>



#### [Answer]

1

2

3

4

Figure 14. Temporal Verification task example. The question asks which image was captured at a different timestamp compared to others.

## Viewpoint Localization

### [Question]

You will be shown multiple synchronized camera views with annotations.  
Use these images to answer the question below.

Image 1 (Reference):

<Image 1>

Image 2 (Reference):

<Image 2>

Image 3 (Reference):

<Image 3>

Image 4:

<Image 4>

Image 5:

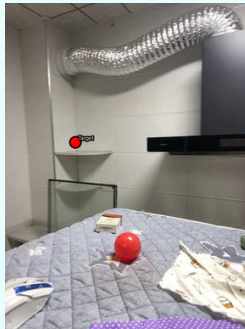
<Image 5>

Image 6:

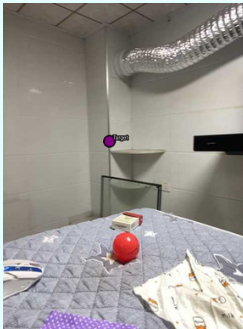
<Image 6>

The reference images (3 images) show where a target camera is located in 3D space (marked with colored dots). Which of the 3 candidate images (numbered) was taken from this camera location? Answer with the image number.

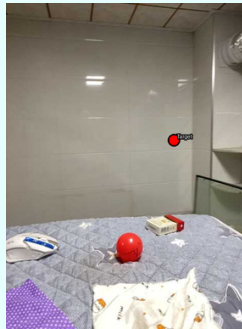
<Image 1>



<Image 2>



<Image 3>



<Image 4>



<Image 5>



<Image 6>



### [Answer]

4

5

6

Figure 15. Viewpoint Localization task example. The question asks which camera view corresponds to a specific spatial position marked in 3D space.



### Directional View Localization

#### [Question]

You will be shown several images captured during a robotic manipulation task. Each image comes with a description of the camera position from which it was taken. Please use this information to help answer the following question.

Image 1 (fix\_center): The central camera provides a front-facing view of the robotic manipulation workspace.

<Image 1>

Image 2:

<Image 2>

Image 3:

<Image 3>

The center image (fix\_center) represents the agent's viewpoint.

The left and right images are from cameras positioned on the left and right sides of the agent.

The center image (fix\_center) establishes the robot's perspective in this manipulation task.

From this reference point, determine which image number shows the left direction.

Evaluate the spatial arrangement and relative positioning of elements in the scene.

Which image number displays the left side relative to the agent's central viewpoint?

<Image 1>



<Image 2>



<Image 3>



#### [Answer]

2

3

Figure 16. Directional View Localization task example. The question asks which camera view lies in a specified direction (e.g., left, right) relative to the reference camera.



## Cross-Scenario Localization

### [Question]

You will be shown several images captured during a robotic manipulation task. Each image comes with a description of the camera position from which it was taken. Please use this information to help answer the following question.

Scenario 1:

Image 1 (fix\_right): This perspective shows the manipulation scene from a fixed right-side camera angle.

<Image 1>

Image 2 (fix\_center): A fixed camera positioned centrally captures this view of the robotic operation.

<Image 2>

Image 3 (fix\_left): This camera angle provides a left-side observation point of the manipulation task.

<Image 3>

Scenario 2:

Image 4:

<Image 4>

Image 5:

<Image 5>

Image 6:

<Image 6>

Two different scenarios show similar robotic manipulation tasks from various camera angles. Scenario 1's Image 1 demonstrates the fix\_right camera view. Your challenge is to identify the equivalent image in Scenario 2 that captures the identical viewpoint. Analyze the camera positioning and visual perspective in each Scenario 2 image. Select the image number in Scenario 2 that matches the fix\_right view from Scenario 1.

<Image 1>



<Image 2>



<Image 3>



<Image 4>



<Image 5>



<Image 6>



### [Answer]

4

5

6

Figure 17. Cross-Scenario Localization task example. The question asks which camera view in one scenario matches the viewpoint of a reference image from another scenario.

### Language-Conditioned Localization

#### [Question]

You will be shown several images captured during a robotic manipulation task. Each image comes with a description of the camera position from which it was taken. Please use this information to help answer the following question.

The center image (Image 1) represents the agent's viewpoint, providing a reference for understanding left and right relative to the robot.

Image 1 (fix\_center): This perspective shows the manipulation task from the center of the camera setup.  
<Image 1>

Image 2:  
<Image 2>

Image 3:  
<Image 3>

The camera perspective is described as:

"This camera angle provides the robot's right arm perspective of the task."

Your task is to examine each image and determine which one was taken from this specific camera position. Consider the spatial orientation, viewing angle, and visual characteristics of each image. Which image number displays the camera view described above?

<Image 1>



<Image 2>



<Image 3>



#### [Answer]

1

2

3

Figure 18. Language-Conditioned Localization task example. The question asks which camera view matches a natural language spatial description (e.g., "wrist-mounted camera").