# SpatialScore: Towards Comprehensive Evaluation for Spatial Intelligence

Supplementary Material

## Contents

# A. Limitations & Future Works

## A.1. Limitations

While SpatialScore offers a comprehensive and diverse evaluation framework for spatial intelligence, SpatialCorpus provides large-scale, high-quality training samples for spatial understanding tasks, and SpatialAgent demonstrates promising improvements with a multi-agent system, our work is not without its limitations. Although SpatialScore covers evaluation across single-image, multi-frame sequences, and videos, it primarily relies on RGB frames, still lacking samples that take point clouds, depth maps, or surface normals as input. Likewise, despite the effectiveness of SpatialCorpus, its diversity is still limited and cannot fully cover the breadth of spatial understanding tasks, leading to biased performance gains in fine-tuned models. Moreover, while SpatialAgent effectively boosts spatial understanding capabilities of MLLMs in a training-free manner by leveraging specialized tools, its current toolbox remains relatively rudimentary, and fundamental advances in spatial perception abilities of MLLMs are still required. These gaps are left for future work.

## A.2. Future Works

To tackle the potential limitations, we outline several promising directions for advancing spatial intelligence: (i) beyond RGB images/videos from existing datasets, incorporating diverse in-the-wild data and direct 3D inputs (*e.g.*, point clouds and depth maps) will further enhance the evaluation of spatial understanding capabilities and drive progress in related research areas; (ii) expanding training samples to cover a broader range of task categories (*e.g.*, counting, orientation) may enable more stable and comprehensive improvements in spatial reasoning tasks through supervised fine-tuning; (iii) enriching the toolbox with more robust expert models and facilitating better multi-agent collaboration will yield more reliable spatial reasoning systems; and (iv) substantial and holistic improvements in the spatial intelligence of multimodal large language models (MLLMs) still require deeper, foundational advances, such as equipping models with essential 3D representational understanding.

# B. Additional Data Details

In this section, we provide additional details about our established **SpatialScore** benchmark and the **SpatialCorpus** training resources. Specifically, in Sec. B.1, we present several examples of inaccurate annotations found in existing datasets, highlighting the motivation and necessity for our thorough manual verification; Next, we elaborate on how we repurpose existing 3D annotations into spatial-intelligence QA pairs in Sec B.2; Then, Sec B.3 presents more detailed statistics and analyses for both SpatialScore and SpatialCorpus; Finally, we include additional representative visualization examples in Sec B.4.

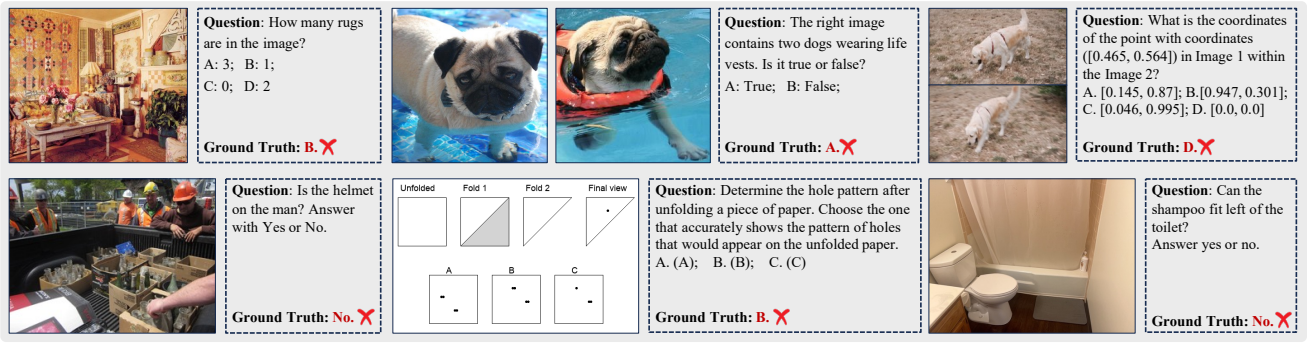



Figure 1. **Annotation Issues in Existing Benchmarks.** We observe that existing datasets contain various annotation errors or ambiguities, including those found in CV-Bench [21], SpatialSense [28], MMIU [15], SRBench [14], SITE-Bench [23], and RoboSpatial-Home [18].

## B.1. Annotation Quality Issues in Existing Datasets

As depicted in Fig. 1, we first observe that existing spatial intelligence benchmarks contain numerous inaccurately annotated samples, which can undermine fair evaluation of model performance. This motivates two key steps in constructing SpatialScore: (i) creating new evaluation samples using high-quality 3D annotations, and (ii) integrating existing benchmarks while manually verifying and filtering their samples to ensure high quality and sufficient challenge. Consequently, we develop **SpatialScore**, the most comprehensive and diverse spatial understanding benchmark to date, comprising **5,025** manually validated samples spanning **30** tasks across **10** categories, providing a comprehensive foundation for evaluating the spatial intelligence of current MLLMs.

## B.2. Data Construction Details

**Repurpose 3D Data for SpatialScore.** Considering that ScanNet has been widely used in prior work [6, 11, 15, 32] and may already be included in the training data of existing models, we intentionally avoid using such heavily reused datasets as metadata for constructing our data. Instead, we first sample 500 scenes from ScanNet++ [30], Omni3D [4], PointOdyssey [33], WildRGB-D [25], and CA-1M [10], and convert their original annotated metadata into question–answer pairs. After extracting essential key information from each scene, we generate corresponding questions and contexts using predefined templates for each task, as presented below.

**Task 1: Object Existence.**

Using the data from Omni3D [4], we construct samples for the *object existence* task, *i.e.*, determining whether a specific object category appears in an image, based on the following question templates:

```
IS_THERE: ["Is there any {category} present in the image?"]
DOES_CONTAIN: ["Does the image contain any {category}?"]
IS_VISIBLE: ["Is any {category} visible in this image?"]
CAN_YOU_SEE: ["Can you see any {category} in this image?"]
```

**Task 2: 3D Object Detection.**

We adopt the 3D bounding-box annotations from Omni3D [4] and CA-1M [10] (represented by the coordinates of eight 3D corner points) to construct samples for this task, and generate distractors by adding small random noise to each coordinate of the ground-truth box.

```
DETECT_3D_BBOX: [
    "Detect the 3D bounding box of the {object_name} in the image.",
    "Provide the 3D bounding box for the {object_name}.",
    "What is the 3D bounding box of the {object_name}?",
    "Locate the {object_name} and output its 3D bounding box.",
    "Output the 3D bounding box coordinates for the {object_name}."
]
PROVIDE_3D_BBOX: [
    "Can you provide the 3D bounding box of the {object_name}?",
    "Please detect and output the 3D bounding box for the {object_name}.",
    "Identify the {object_name} and provide its 3D bounding box coordinates.",
    "What are the 3D coordinates of the {object_name}'s bounding box?"
]
WHAT_IS_3D_BBOX: [
    "What is the 3D bounding box of the {object_name}?",
    "Where is the {object_name} located in 3D space?  Provide its bounding box.",
    "Determine the 3D bounding box coordinates for the {object_name}."
]
```

**Task 3-5: Absolute Depth & Absolute Distance & Absolute Size.**

We construct these metric-based task samples (typically using units such as meters, feet, or centimeters) from the 3D bounding-box data in the Omni3D [4] and CA-1M [10] datasets. For distractors, we first generate one value very close to the correct answer (usually within 85%–95% or 105%–115% of the ground truth). We then create additional distractors within the broader ranges of 50%–90% and 110%–180% of the correct value. If the above strategy does not yield enough distractors, we generate additional ones by simply adding or subtracting fixed values (e.g., 0.5 or 1.0) from the correct answer.

```
### Absolute Depth:
HOW_FAR: [
    "How far is the {object_name} from the camera?",
    "What is the distance of the {object_name} from the camera?",
    "How far away is the {object_name}?"
],
DISTANCE_FROM_CAMERA: [
    "What is the approximate distance from the camera to the {object_name}?",
    "How many {unit} away is the {object_name} from the camera?",
```

```
    "At what distance is the {object_name} located from the camera?"
],
APPROXIMATE_DISTANCE: [
    "Approximately how far is the {object_name}?",
    "What is the rough distance to the {object_name}?",
    "How far would you estimate the {object_name} to be?"
]

### Absolute Distance:
[
    "What is the distance between the {object1} and the {object2}?",
    "How far apart are the {object1} and the {object2}?",
    "What is the approximate distance from the {object1} to the {object2}?",
    "How much distance separates the {object1} and the {object2}?",
    "What is the spatial separation between the {object1} and the {object2}?"
]

### Absolute Size:
WHAT_IS_DIMENSION: [
    "What is the {dimension} of the {object_name}?",
    "What is the {dimension} {dimension_type} of the {object_name}?",
    "How much is the {dimension} of the {object_name}?"
],
HOW_DIMENSION: [
    "How {dimension} is the {object_name}?",
    "How {dimension_adj} is the {object_name}?",
],
DIMENSION_OF_OBJECT: [
    "What is the {object_name}'s {dimension}?",
    "How would you measure the {dimension} of the {object_name}?",
    "What dimension represents the {dimension} of the {object_name}?"
]
```

**Task 6-8: Relative Depth & Relative Distance & Relative Size.**

These tasks involving relative comparisons can likewise be constructed from the metadata of Omni3D [4] and CA-1M [10], and the distractors can be easily generated by simply selecting metadata corresponding to other objects or points.

```
### Relative Depth:
[
    "Which object is closest to the camera?",
    "Among the following objects, which one is nearest to the camera?",
    "Which of these objects has the shortest distance from the camera?",
    "Select the object that is closest to the camera:",
    "Which object appears closest in the image?"
]

### Relative Distance:
[
    "Among the following objects, which one is closest to the {reference}?",
    "Which object has the shortest distance to the {reference}?",
    "Select the object that is nearest to the {reference}:",
    "Which of these objects is closest to the {reference}?",
    "What object is positioned closest to the {reference}?"
]

### Relative Size:
(ComparisonDimension.HEIGHT, ComparisonType.LARGER): [
    "Which object is taller, the {object1} or the {object2}?",
    "Between the {object1} and the {object2}, which one is higher?",
    "Which is taller:  the {object1} or the {object2}?",
    "Compare the height of the {object1} and the {object2}.  Which one is taller?"
],
(ComparisonDimension.WIDTH, ComparisonType.LARGER): [
```

```
    "Which object is wider, the {object1} or the {object2}?",
    "Between the {object1} and the {object2}, which one is wider?",
    "Which is wider:  the {object1} or the {object2}?",
    "Compare the width of the {object1} and the {object2}.  Which one is wider?"
],
(ComparisonDimension.LENGTH, ComparisonType.LARGER): [
    "Which object is longer, the {object1} or the {object2}?",
    "Between the {object1} and the {object2}, which one is longer?",
    "Which is longer:  the {object1} or the {object2}?",
    "Compare the length of the {object1} and the {object2}.  Which one is longer?"
],
(ComparisonDimension.HEIGHT, ComparisonType.SMALLER): [
    "Which object is shorter, the {object1} or the {object2}?",
    "Between the {object1} and the {object2}, which one is lower?",
    "Which is shorter:  the {object1} or the {object2}?",
    "Compare the height of the {object1} and the {object2}.  Which one is shorter?"
],
(ComparisonDimension.WIDTH, ComparisonType.SMALLER): [
    "Which object is narrower, the {object1} or the {object2}?",
    "Between the {object1} and the {object2}, which one is narrower?",
    "Which is narrower:  the {object1} or the {object2}?",
    "Compare the width of the {object1} and the {object2}.  Which one is narrower?"
],
(ComparisonDimension.LENGTH, ComparisonType.SMALLER): [
    "Which object is shorter in length, the {object1} or the {object2}?",
    "Between the {object1} and the {object2}, which one is shorter?",
    "Which is shorter:  the {object1} or the {object2}?",
    "Compare the length of the {object1} and the {object2}.  Which one is shorter?"
]
```

**Task 9: Camera Intrinsics.**

Our considered available 3D annotations all provide *camera intrinsics* information. For distractors, we generate random values within predefined proportional ranges: for focal lengths (fx and fy), principal-point coordinates (cx and cy), and the skew parameter (s), we randomly sample distractors using variance ratios of 0.25, 0.20, and 0.10, respectively.

```
FOCAL_LENGTH: [
    "What is the camera's focal length in pixels?",
    "What is the focal length of the camera?",
    "Can you determine the camera's focal length?"
]
PRINCIPAL_POINT: [
    "What are the image center coordinates in the camera intrinsics?",
    "What is the principal point of the camera?",
    "What are the coordinates of the image center?"
]
FOCAL_LENGTH_X: [
    "What is the horizontal focal length (fx) in pixels?",
    "What is the camera's focal length in the x-direction?",
    "Can you determine the horizontal focal length?"
]
FOCAL_LENGTH_Y: [
    "What is the vertical focal length (fy) in pixels?",
    "What is the camera's focal length in the y-direction?",
    "Can you determine the vertical focal length?" ]
ASPECT_RATIO: [
    "What is the aspect ratio of the camera's focal lengths (fx/fy)?",
    "What is the ratio between horizontal and vertical focal lengths?",
    "Can you calculate the aspect ratio from the camera intrinsics?"
]
```

**Task 10: Camera Extrinsics.**

Constructing samples for *camera extrinsic* estimation requires the annotations from CA-1M [10], ScanNet++ [30], and

WildRGB-D [25]. For distractors, we randomly choose from three strategies: (i) Axis swapping or sign flipping: randomly swap axes of the rotation matrix or flip the sign of an axis while keeping the matrix orthogonal. (ii) Translation vector perturbation: keep the rotation unchanged and add random noise to the translation vector. and (iii) Small rotational perturbation: apply a slight rotational disturbance to the original rotation matrix to produce a new, approximately orthogonal matrix.

```
[
    "What is the transformation matrix from the first camera coordinate system to the second
camera coordinate system in OpenCV convention?",
    "Can you provide the relative transformation matrix between the two camera poses in OpenCV
convention?",
    "What is the 4x4 transformation matrix that transforms coordinates from the first camera
frame to the second camera frame in OpenCV convention?",
    "Please calculate the extrinsic transformation matrix from camera 1 to camera 2 in OpenCV
convention.",
]
```

**Task 11: Camera Motion.**

The *camera motion* task is derived from camera extrinsics. We first identify the axes with the most significant rotational or translational changes. If the dominant motion exceeds a high threshold (*e.g.*, rotation greater than 10°), it is considered a salient motion to be described. If the motion is below a low threshold (*e.g.*, rotation less than 5°), the camera is treated as stationary along that axis. Motions falling in between are ignored in the correct answer but may be used when generating distractors. Each degree of freedom (roll, pitch, yaw, x, y, z) is labeled with a state (changed, ignored, stationary) and a direction (e.g., left, up, forward). A standardized, human-readable description is then produced as the correct answer. For example, if roll changes significantly to the left and translation moves significantly backward, the output becomes: "The camera rolled left and moved backward."

Distractors are generated using the following strategies: (i) Opposite motion: For true motions (state = changed), there is a 70% chance of describing them with the opposite direction. (ii) Omission: For true motions, there is a 30% chance of omitting them entirely and treating the camera as stationary. and (iii) Fabrication: For ignored minor motions (state = ignored), there is a 30% chance of fabricating a new motion. These incorrect motion fragments (*e.g.*, "moved backward," "pitched up") are combined into complete sentences. If all true motions are omitted, a fallback option "The camera remained stationary." is generated. Finally, distractors that duplicate each other or the correct answer are removed. If the remaining number is insufficient, additional distractors are sampled from a preset list of generic motions (*e.g.*, "The camera moved forward.") to ensure adequate coverage.

```
multi_choice:  [
    "Which best describes the camera motion between these two images?",
    "How did the camera primarily move?",
    "What type of camera movement occurred?",
    "Which motion pattern best matches the camera transformation?",
]
open_ended:  [
    "What kind of camera motion occurred between the two images?",
    "Describe the relative motion of the camera from the first image to the second image.",
    "How did the camera move between these two frames?",
    "Can you describe the camera movement between the two views?",
    "What is the camera's motion from the first view to the second view?",
]
```

**Task 12: Point Tracking.**

We construct *point tracking* samples by leveraging the metadata from CA-1M [10], PointOdyssey [33], and WildRGB-D [25] to establish correspondences across multiple images. For distractors, we randomly select other non-corresponding points from the images.

```
[
    "In the first image, there is a point at coordinates ({x1}, {y1}).  Which point in the second
image corresponds to this tracked point?",
```

```
    "Given a point at position ({x1}, {y1}) in image 1, which of the following coordinates in
image 2 represents the same tracked point?",
    "A point is tracked from image 1 at ({x1}, {y1}).  Where does this point appear in image 2?",
    "Tracking point from image 1:  ({x1}, {y1}).  Select its corresponding location in image 2:"
]
```

**Task 13: Homography Matrix.**

Data for the *homography matrix* task can be easily constructed from all available metadata, and multiple additional homography matrices can be randomly generated as distractors. The corresponding question templates are shown below.

```
[
    "What is the homography matrix that transforms the original image to the given transformed
image?",
    "Please provide the 3x3 homography transformation matrix between the original and transformed
images.",
    "Calculate the homography matrix that maps the original image to the transformed version.",
    "What is the perspective transformation matrix from the original image to the transformed
image?"
]
```

Moreover, to boost the linguistic diversity of QA pairs while preserving semantic integrity, we employ DeepSeek-V3 [12] to systematically rephrase questions, generate distractors, and convert question types, using the following prompts:

```
### For rephrasing open-ended questions:
  Please rephrase the following question while maintaining its original meaning.  Requirements:
  1.  Keep the core meaning of the question unchanged
  2.  Use natural and fluent language
  3.  Return only the rephrased question, nothing else
  Original question:  {question}
  Rephrased question:

### For rephrasing multi-choice questions:
  Please rephrase the following multiple-choice question while maintaining its original meaning.
Requirements:
  1.  Keep the core meaning of the question unchanged
  2.  If there is an instruction phrase like "Select from the following choices", keep it
  3.  Use natural and fluent language
  4.  Return only the rephrased question, nothing else
  Original question:  {question}
  Rephrased question:

### For generating distractor options:
  Based on the following question and correct answer, generate num_options options (including the
correct answer).  Requirements:
  1.  Options should be reasonable and have distraction value
  2.  The correct answer is:  {correct_answer}
  3.  Other options should be incorrect but plausible answers
  4.  Return in JSON format:  {"options":  ["option1", "option2", ...]}
  5.  Return only JSON, nothing else
  Question:  {question}
  Correct answer:  {correct_answer}
  Required answer:  {required_ans}
  Generated options:

### For converting questions into judgment questions:
  Convert the following question to a yes/no question format.  Requirements:
  1.  Keep the core meaning unchanged.
  2.  The question should be answerable with yes or no.
  3.  The converted question should be as specific as possible, directly incorporating relevant
details and data points (e.g., specific values, coordinates, identifiers) from the original
question or answer.  Avoid asking general questions about detection, presence, or existence if
```

```
more specific information can be queried.
  4. Based on the original answer "{correct_answer}", determine if the yes/no answer should be
"yes" or "no".
  5. Return in JSON format: {"question": "yes/no question", "answer": "yes or no"}.
  6. Return only JSON, nothing else.
  Original question: {question}
  Original answer: {correct_answer}
  Required answer: {required_ans}
  Converted question:
```

**Scaling Data for SpatialCorpus.** Building on the automated pipeline that repurposes 3D annotations into spatial understanding question–answer pairs, we employ the metadata of the training sets of ScanNet++ [30], Omni3D [4], WildRGB-D [25], and PointOdyssey [33] to create additional training samples. Note that CA-1M [10] is excluded at this stage since it only contains class-agnostic object annotations. We enhance data diversity by designing richer question templates, thereby avoiding the heavy cost of large-scale LLM-based rephrasing.

Additionally, to further improve model performance on tasks related to *mental animation*, we utilize simulators to generate scalable data for *spatial map*, *multi-view projection*, and *2D/3D rotations*. For these *mental animation* tasks, distractors are constructed as follows: (i) Spatial Map: We randomly place several non-overlapping locations on a map and compute directional relations based on their coordinates. From this, we generate four types of multi-choice questions: determining directional relations, finding an object in a specified direction, counting objects in a given direction, and identifying the nearest object. Each question contains exactly one correct answer and spans diverse spatial-relation types. (ii) 2D Rotation: We generate a colored grid reference image with no rotational or mirror symmetry, create a single correct option by rotating it (90°/180°/270°), and add three distractors (e.g., flipped or color-shifted versions). All distractors are guaranteed not to match the reference under any rotation, ensuring a strictly single-answer setting. and (iii) 3D Rotation: We construct a colored voxel-based 3D shape without self-symmetry. The correct option is obtained via one of the 24 valid rotation matrices, while distractors include shapes with different voxel counts, mismatched colors, or altered spatial layouts with the same voxel count. Only the correct option remains equivalent to the reference under valid 3D rotations.

The corresponding question templates are provided below:

```
### Spatial Map:
  1. direction_relation: "question": "In which direction is {q1_p1} relative to {q1_p2}?
{DIRECTION_RULE}"
  2. find_object: "question": "Which object is in the {target_dir} of {q2_p1}?
{DIRECTION_RULE}"
  3. count_objects: "question": "How many objects are in the {q3_target_dir} of {q3_p1}?
{DIRECTION_RULE}"
  4. closest_object: "question": "Which object is closest to {q4_p1}?"
### Multi-view Projection:
  1. view_identification: "The first image shows a 3D view of the scene, while the second shows
one of the three orthographic views of this 3D scene. What type of view is displayed in the
second image? The front view is observed from the positive direction of the Y-axis toward the
negative direction, the left view is observed from the positive direction of the X-axis toward
the negative direction, and the top view is observed from the positive direction of the Z-axis
toward the negative direction."
  2. view_matching: "Which option shows the {target_view} view of the 3D scene?" The front
view is observed from the positive direction of the Y-axis toward the negative direction, the
left view is observed from the positive direction of the X-axis toward the negative direction,
and the top view is observed from the positive direction of the Z-axis toward the negative
direction."
### 2D_Rotation:
  "Which option is the rotated version of the reference shape?"
### 3D_Rotation:
  "Which option is the rotated version of the reference 3D shape?"
```

## B.3. Data Statistics

We provide the details on the distributions of question formats, input modalities, data sources, and the samples across categories and tasks for **SpatialCorpus** and **SpatialScore** in Tab. 1 and Tab. 2, respectively. Then we further visualize the

Table 1. **Data Statistics of SpatialCorpus.**

| Question Type | | Categories | | Tasks | | | |
|---|---|---|---|---|---|---|---|
| Multi-choice | 262,601 | Mental Animation | 57,997 | Spatial Map | 40,000 | Relative Size | 11,841 |
| Judgment | 9,776 | Depth Estimation | 57,843 | Multi-view Projection | 7,998 | Absolute Size | 9,773 |
| Open-ended | 58,425 | Object Distance | 51,596 | 2D/3D Rotation | 9,999 | Camera Intrinsics | 49,984 |
| **Input Modalities** | | Object Motion | 20,000 | Absolute Depth | 32,594 | Camera Extrinsics | 4,997 |
| Single-image | 270,812 | Object Size | 21,614 | Relative Depth | 25,249 | Camera Motion | 4,997 |
| Multi-image | 59,990 | Camera | 81,976 | Absolute Distance | 25,712 | Homography Matrix | 21,998 |
| | | Object Localization | 39,776 | Relative Distance | 25,884 | Object Existence | 9,776 |
| | | | | Point Tracking | 20,000 | 3D Object Detection | 30,000 |
| **Total: 330,802** | | | | | | | |

Table 2. **Data Statistics of SpatialScore.** Here, *SpatialScore-Repurpose* denotes the newly constructed samples based on 3D annotations.

| Question Type | | Data Sources | | | | | |
|---|---|---|---|---|---|---|---|
| Multi-choice | 3,686 | SpatialScore-Repurpose | 1,091 | CV-Bench | 171 | QSpatialBench | 39 |
| Judgment | 463 | VSI-Bench | 876 | Space-10 | 169 | RoboSpatial | 35 |
| Open-ended | 876 | MMIU | 419 | BLINK | 143 | SpatialBench | 27 |
| **Input Modalities** | | SPAR-Bench | 475 | SpatialSense | 124 | VSR | 21 |
| | | SpatialEval | 293 | VLM4D | 116 | MIRAGE | 20 |
| Single-image | 2,493 | 3DSRBench | 266 | SpatialViz | 114 | RealWorldQA | 11 |
| Multi-image | 1,339 | SITE-Bench | 233 | MMSI-Bench | 69 | MMVP | 5 |
| Video | 1,193 | OmniSpatial | 205 | SRBench | 54 | STI-Bench | 49 |

| Categories | | Tasks | | | | | |
|---|---|---|---|---|---|---|---|
| Mental Animation | 447 | Spatial Map | 220 | Absolute Distance | 313 | Camera Intrinsics | 112 |
| Counting | 315 | Maze Navigation | 115 | Relative Distance | 263 | Camera Extrinsics | 246 |
| Depth Estimation | 520 | Multi-view Projection | 46 | Point Tracking | 229 | Camera Motion | 174 |
| Object Distance | 576 | Space Folding | 20 | Object Motion | 113 | Homography Matrix | 246 |
| Object Motion | 415 | 2D/3D Rotation | 46 | Velocity Estimation | 73 | Navigation Route | 105 |
| View Reasoning | 446 | Object Counting | 154 | View Perspective | 235 | Appearance Order | 167 |
| Object Size | 559 | Video Counting | 111 | Orientation | 211 | Object Existence | 220 |
| Camera | 778 | Count with Relation | 50 | Relative Size | 189 | 2D Localization | 50 |
| Temporal Reasoning | 272 | Absolute Depth | 325 | Absolute Size | 281 | 3D Object Detection | 212 |
| Object Localization | 697 | Relative Depth | 195 | Size Compatibility | 89 | Spatial Position | 215 |
| **Total: 5,025** | | | | | | | |

distributions of data sources, as well as the distributions across categories and tasks within SpatialScore, in Fig. 2. Here, we denote the newly constructed samples repurposed from 3D annotations as *SpatialScore-Repurpose*.

By integrating these data and conducting manual verification, SpatialScore encompasses a wide range of spatial intelligence tasks with diverse question–answering formats (judgment, multiple-choice, and open-ended) and input modalities (single image, multi-frame sequence, and video), establishing the most comprehensive and heterogeneous benchmark for spatial

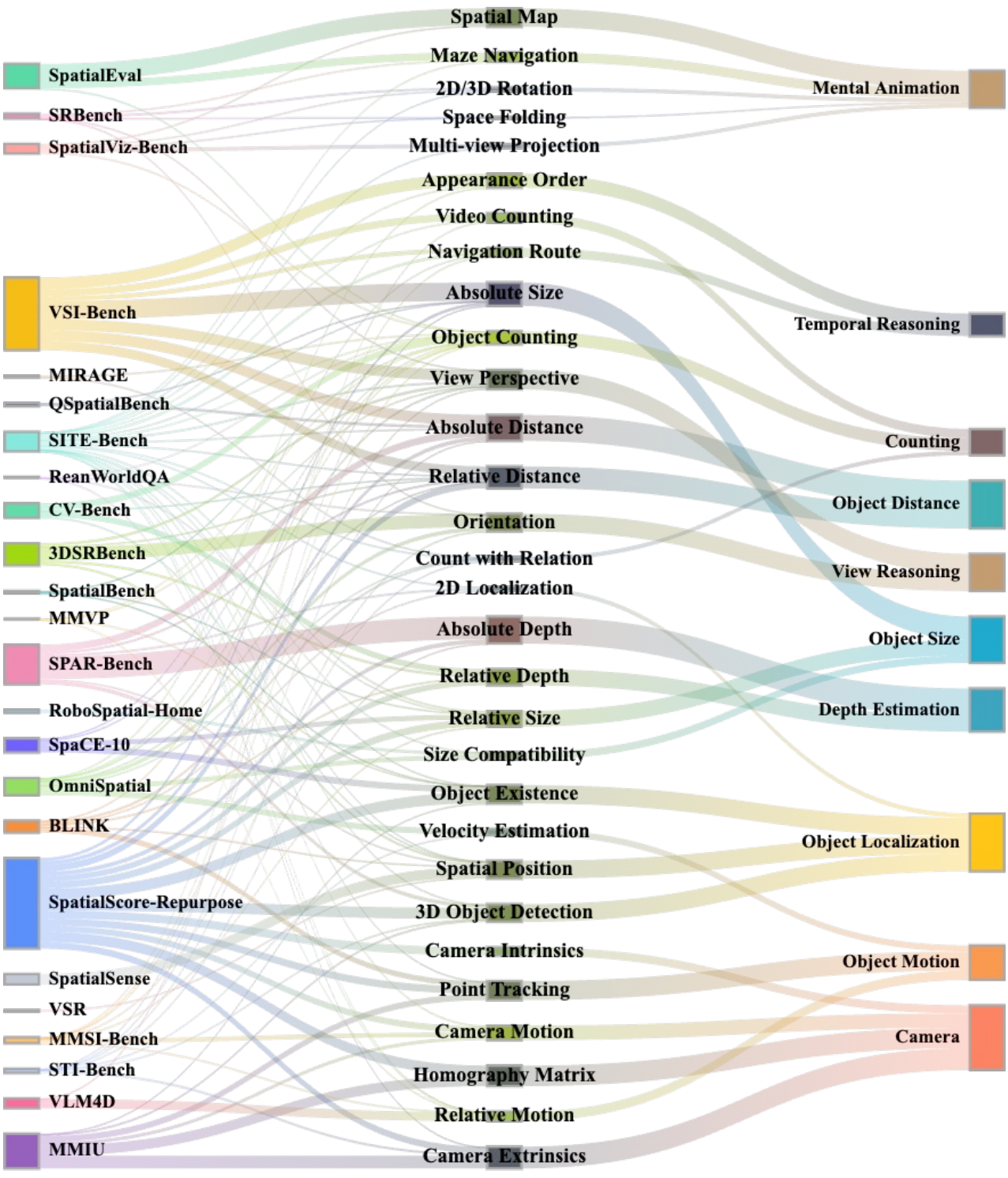



Figure 2. **Data Sources and Task Category Statistics Visualization of SpatialScore.** Here, we denote the newly constructed samples repurposed from 3D annotations as *SpatialScore-Repurpose*.

understanding to date. This makes it an effective testbed for evaluating the spatial reasoning capabilities of current MLLMs. We believe this advancement will further drive research progress in the field of spatial intelligence.


Figure 3. **More Representative Examples in SpatialScore.** Here, some questions have been slightly rewritten for clarity of presentation.

### B.4. Additional Representative Visualizations

We further present representative examples from each task in SpatialScore and SpatialCorpus to demonstrate their comprehensiveness and diversity. As depicted in Fig. 3, SpatialScore contains **5,025** samples covering **30** tasks across **10** categories, while Fig. 4 illustrates that SpatialCorpus includes over **331K** samples spanning **16** tasks across **7** categories. This diversity ensures that SpatialScore serves as the most comprehensive benchmark to date for spatial intelligence, and that SpatialCorpus provides the high-quality data necessary for supervised fine-tuning (SFT) on spatial reasoning tasks.

## C. More Implementation Details

In this section, we provide additional technical details of our work. Concretely, we first describe the evaluation setup used for the SpatialScore benchmark in Sec C.1; Next, we outline the procedures for supervised fine-tuning with SpatialCorpus in Sec C.2; Then, we present the development details of SpatialAgent in Sec C.3; Finally, Sec C.4 introduces the design of the spatial perception expert models included in the toolbox, particularly highlighting instruction prompts.

### C.1. SpatialScore Evaluation

**Hyper-parameters.** To ensure reproducibility, we standardize the following configurations: all models adopt deterministic sampling (TEMPERATURE=0.0, DO_SAMPLE=False) and a maximum output length of 512 tokens, except for reasoning-oriented models such as KimiVL-16B-A3B-Thinking [19] and LLaMA-3.2V-11B-CoT [26], which are allocated 2048 tokens. For our **SpatialAgent**, we set the maximum attempt limit to 3 iterations under the *Plan-Execute* paradigm and permit 10 dialogue turns for *ReAct* interactions. To accommodate the extended reasoning requirements in multi-agent collaboration, the token limit is correspondingly increased to 4096 for these cases. For video inputs, we follow [27] and uniformly sample 32 frames for open-source models. For proprietary models, we provide 16 frames to GPT-5 [16] and Gemini [5, 7], while Claude-4.5-Sonnet [2] supports only 8 input frames.

**Baselines.** Our chance-level (Random) baseline is implemented as follows: For judgment and multi-choice questions, we

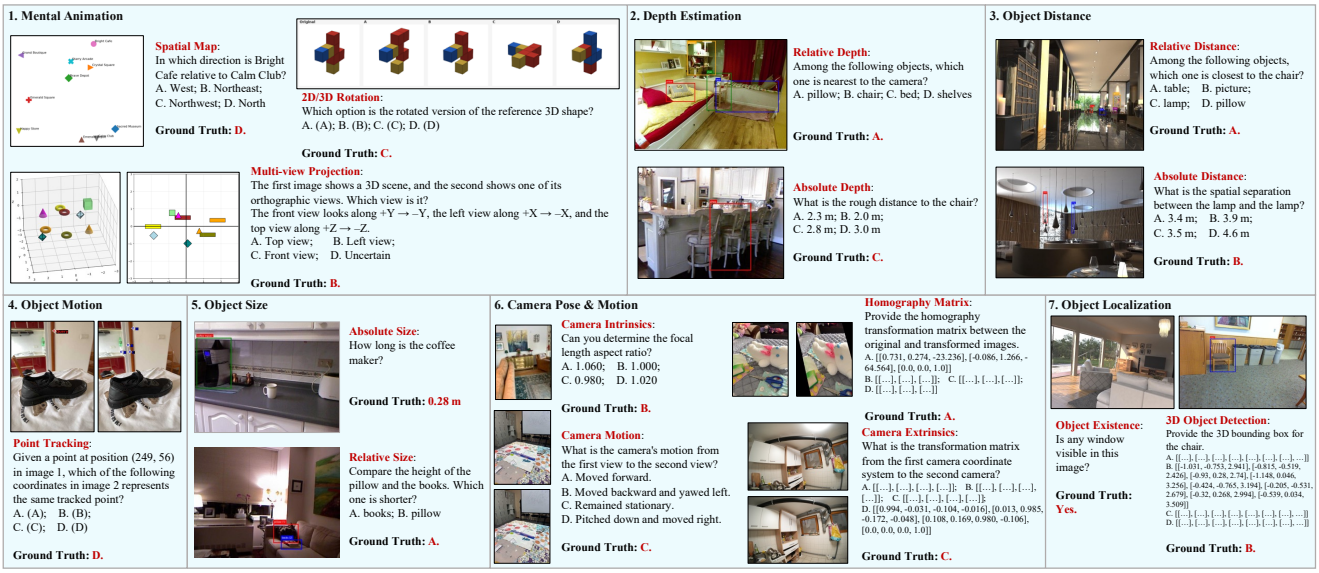



Figure 4. **More Representative Examples in SpatialCorpus.** Here, some questions have been slightly rewritten for clarity of presentation.

randomly sample an answer based on the number of available options. For open-ended questions, to ensure that the baseline yields a reasonably meaningful result, we uniformly sample a value within a range of 0.25 to 4 times the ground-truth value as the final answer. Additionally, we employ a powerful model (*i.e.*, GPT-5 [16]) with text-only input to serve as a blind baseline. For the human-level evaluation, we invite three PhD students with extensive experience in 3D vision research to answer the questions using basic computational tools.

**Prompts.** Since SpatialScore encompasses samples from diverse data sources, covering judgment, multi-choice, and open-ended questions, we carefully design tailored system prompts for each format to ensure models can properly follow instructions and produce correctly formatted answers. The detailed prompts are as follows:

For judgment questions, we employ the following prompt to guide models to provide their determined correct answers:

```
**Please answer with yes or no based on the image.**
**Respond ONLY with 'yes' or 'no'.**
Question: {question}
```

For multi-choice questions, we expect models to concisely output their selected option, with the following prompt:

```
**Please select the most appropriate answer from the given options.**
**Respond ONLY with the capital letter and its parentheses.**
Question: {question}
```

For open-ended questions, we first address those that require estimating metric-based distances or sizes. In these cases, we use the following prompt to guide the model to provide both a numerical value and its corresponding unit of measurement.

```
Please answer the question by measuring the precise distance in 3D space through 2D images or
videos.
Respond ONLY with a numeric answer consisting of a scalar and a distance unit in the format of
**scalar {scalar} distance_unit {distance unit}**.
Question: {question}
```

For other types of open-ended questions, such as counting, we adopt the following prompt:

```
Please answer the question based on the given image or video.
Respond ONLY with a concise and accurate scalar or a scalar with corresponding unit.**
Question: {question}
```

Notably, we have carefully designed answer-parsing functions for all models to ensure accurate extraction of final answers. However, some responses still fail to comply with instruction prompts or result in refusal to answer (primarily in certain fine-tuned models and smaller-scale architectures). Therefore, when computing overall accuracy, we adopt the following strategy: For judgment and multi-choice questions, we apply the parsing functions to extract the models' answers and directly compare them with the ground truth. For open-ended questions, we report the average Mean Relative Accuracy (MRA) [27] obtained by two methods: (i) extracting the models' answers with the parsing function to calculate the score, and (ii) use an off-the-shelf LLM (GPT-OSS-20B [1]) to score the response with the following prompt:

```
You are an evaluator.
Your ONLY job is to compute a score using the following algorithm.
Do NOT answer or solve the question.

TASK TYPE:
- If Type == "counting": treat both GT and PRED as plain scalar numbers (no unit conversion).
- If Type == "distance": parse numeric value + unit; if PRED unit is missing, borrow GT unit;
if both are missing and both look like plain numbers, treat as scalar.
- If a numeric RANGE like "10-15" appears, use the MAX value (e.g., 15).

ALGORITHM (VSI-Bench MRA):
1) Compute abs_dist_norm:
- For scalar/counting: abs_dist_norm = abs(pred - gt) / gt (if gt == 0, set abs_dist_norm =
+Infinity)
- For distance: convert both to centimeters using: meter (m): 100 cm; centimeter (cm): 1 cm;
millimeter (mm): 0.1 cm; inch (in): 2.54 cm; foot (ft): 30.48 cm.
Then abs_dist_norm = abs(pred_cm - gt_cm) / gt_cm (if gt_cm == 0, set +Infinity).

2) For thresholds C = linspace(start, end, steps) with steps = int((end-start)/interval+2):
   accuracy(C) = 1 if abs_dist_norm <= (1 - C) else 0
   mean_relative_accuracy = average of accuracy(C) over all thresholds.

3) The final score is this mean_relative_accuracy, a float in [0,1].

IMPORTANT OUTPUT RULE:
- After you finish the calculation, OUTPUT EXACTLY ONE LINE at the end in the form: output:
<float> For example: output: 0.83

Config:
- start={start}
- end={end}
- interval={interval}

Inputs:
- Type: {open_type} # "counting" or "distance"
- gt_answer: {gt_answer}
- pred_answer: {pred_answer}
```

## C.2. Supervised Fine-tuning with SpatialCorpus

To efficiently and effectively enhance MLLMs' performance on spatial understanding tasks, we construct **SpatialCorpus**, a large-scale training resource consisting of 331K multimodal QA pairs spanning 16 task types. It includes both single-frame and multi-frame inputs and covers judgment, multi-choice, and open-ended QA formats, serving as the supervised fine-tuning (SFT) dataset. To be specific, we fine-tune Qwen3-VL-4B and Qwen3-VL-8B for one epoch on SpatialCorpus using $8\times$ Nvidia A100 GPUs, with bfloat16 precision, a batch size of 512, a peak learning rate of $1 \times 10^{-5}$, and a warm-up ratio of 3%. The visual encoder is kept fixed, while the MLP that projects visual features to language space and the LLM parameters are jointly optimized.

## C.3. SpatialAgent Development

To build **SpatialAgent**, a multi-agent system tailored for spatial intelligence and capable of using open-source MLLMs (*e.g.*, Qwen3-VL) as the agent core, we have meticulously designed a series of instruction prompts that guide the agent core ($\Phi$) to serve as different components within the system. These prompts enable SpatialAgent to think step-by-step in two distinct paradigms: *Plan-Execute* and *ReAct*, thereby improving the spatial understanding capabilities of MLLMs in a training-free manner. Details are presented below.

**Plan-Execute Paradigm.** For the *Plan-Execute* paradigm, SpatialAgent primarily consists of three components: *planner* ($\Phi_{\text{plan}}$), *executor* ($\Phi_{\text{exe}}$), and *summarizer* ($\Phi_{\text{sum}}$). First, we use the following prompt to guide the *planner* ($\Phi_{\text{plan}}$) in formulating a detailed tool invocation plan based on the descriptions in the toolbox:

```
[BEGIN OF GOAL]
Generate a JSON-formatted tool-calling plan to solve spatial understanding questions about given
images or videos.
[END OF GOAL]

[BEGIN OF TOOLBOX]
{action_details}
[END OF TOOLBOX]

[BEGIN OF TASK INSTRUCTIONS]
Generate a step-by-step plan to answer the given spatial understanding question about given
images or videos.
***Use ONLY the tools listed in the TOOLBOX section (e.g., GetObjectOrientation,
EstimateObjectGeometryProperties, LocalizeObjects, EstimateObjectDepth)***
***Follow their argument specifications EXACTLY as defined in the toolbox, and try to give
detailed and comprehensive instructions in queries.***
Do NOT invent new tools or modify the existing tool interfaces.
The plan should strictly follow what these tools can and cannot do.
[END OF TASK INSTRUCTIONS]

[BEGIN OF FORMAT INSTRUCTIONS]
You are a helpful assistant tasked with solving spatial reasoning questions.  Think step by
step.
***
Return a JSON list of tool calls inside ```json``` tags, where each call is a dictionary with
'name' and 'arguments'.
The 'name' MUST match exactly one of the tool names provided in the toolbox.
The 'arguments' MUST include ALL required parameters for that specific tool with EXACT parameter
names.
The 'images' or 'image' argument must be specified as 'image-0', 'image-1', and 'image-2', to
refer to the provided images.
Do not answer the question directly, and do not use absolute paths for the 'images' or 'image'
argument.
***
[END OF FORMAT INSTRUCTIONS]

[BEGIN OF EXAMPLES]
Example for 'Which is closer to the camera, the dog or the cat?':
```json  [
   {"name":  "LocalizeObjects", "arguments":  {"image":  "image-0", "objects":  ["dog", "cat"]}},
   {"name":  "EstimateObjectDepth", "arguments":  {"image":  "image-0", "objects":  ["dog",
"cat"], "indoor_or_outdoor":  "outdoor"}},
  ]
```
[END OF EXAMPLES]

***
Do not answer the question directly.  Instead, think step-by-step, and output the tool-calling
plan inside ```json``` tags.
***
```

Next, the *executor* ($\Phi_{\text{exe}}$) follows the prompt below to sequentially execute tool invocations according to the plan.

```
[BEGIN OF GOAL]
Generate a Chain of Thought (CoT) reasoning process using the provided tool execution results.
[END OF GOAL]

[BEGIN OF TASK INSTRUCTIONS]
You are a helpful assistant tasked with solving spatial reasoning questions. Analyze the given
question and tool execution results. Think step by step.
Generate a step-by-step reasoning process that shows how the tools contribute to solving the
question.
Use ONLY the tools and results provided, following their specifications STRICTLY.
The results of tool calls can sometimes be incomplete or incorrect, so please be critical and
decide how to make use of them.
If a tool failed, note the failure and proceed with your prior knowledge and reasoning.
Repeat for each tool result in order.
[END OF TASK INSTRUCTIONS]

[BEGIN OF FORMAT INSTRUCTIONS]
***
Output a CoT with:
  - <thinking> Explain why this tool was used and how its result contributes to the answer.
</thinking>
  - <tool> The tool call in JSON format, e.g., {{"name": "LocalizeObjects", "arguments":
{{"image": "image-0", "objects": ["dog", "cat"]}}}}. </tool>
  - <observation>: The tool result as a string. </observation>
Repeat for each tool result in order.
***
[END OF FORMAT INSTRUCTIONS]

[BEGIN OF EXAMPLES]
Example for 'In image-0, which is closer to the camera, the dog or the cat?':
  <thinking> To determine which object is closer to the camera, I need first localize the dog and
cat in the image. </thinking>
  <tool> {{"name": "LocalizeObjects", "arguments": {{"image": "image-0", "objects": ["dog",
"cat"]}}}} </tool>
  <observation> {{"results": [{{"label": "dog", "region": [0.5, 0.6, 0.6, 0.8], "confidence":
0.95}}, {{"label": "cat", "region": [0.4, 0.5, 0.45, 0.7], "confidence": 0.87}}]}}
</observation>

  <thinking> The bounding box for the dog is [0.5, 0.6, 0.6, 0.8], and for the cat is [0.4, 0.5,
0.45, 0.7]. Then, I need estimate the depth of them to reflect their distances to the camera.
</thinking>
  <tool> {{"name": "EstimateObjectDepth", "arguments": {{"image": "image-0", "objects":
["dog", "cat"], "indoor_or_outdoor": "outdoor"}}}} </tool>
  <observation> {{"results": [{{"object": "dog", "depth": 1.0, "error": null}}, {{"object":
"cat", "depth": 1.2, "error": null}}]}} </observation>
[END OF EXAMPLES]

Tool Plan: {tool_plan}
Tool Results: {tool_results}

***
**Notably, you should AVOID outputting terms like <final_thining>, <answer>, or <final_answer>
here.**
**Now, output your reasoning between <thinking> and </thinking>, the tool call in JSON format
between <tool> and </tool>, and the observation between <observation> and </observation>.**
***
```

Finally, the *summarizer* ($\Phi_{\text{sum}}$) consolidates the tool execution results and produces the final reasoning and answer, guided by the following instruction prompt:

```
[BEGIN OF GOAL]
Generate a final REASONING and ANSWER for spatial understanding questions about given images or
videos, based on tool results and prior Chain of Thought (CoT) steps.
```

```
[END OF GOAL]

[BEGIN OF TASK INSTRUCTIONS]
You are a helpful assistant tasked with solving spatial reasoning questions.
Given the question, tool execution results, and CoT steps, synthesize the information to provide
a final REASONING and ANSWER.
**The results of tool calls can sometimes be incomplete or incorrect, so please be critical and
decide how to make use of them.**
If tool results are unclear or contradictory, use your prior knowledge to think the problem
step-by-step.
For multi-choice questions, select the most appropriate answer from options based on reasoning.
Respond ONLY with the capital letter and its parentheses.
For judgment questions, answer with yes or no based on reasoning.  Respond ONLY with 'yes' or
'no'.
For open-ended measurement questions, answer the question by measuring the precise distance in
3D space through a 2D images or videos.  DO NOT use generic and unclear units like 'units' or
'pixels'
Respond ONLY with a numeric answer consisting of a scalar and a distance unit in the format of
**scalar distance_unit**.
For other questions, answer the question based on the given image or video.  Respond ONLY with a
concise and accurate scalar or a scalar with corresponding unit.
**CRITICAL: You MUST always provide a reasonable answer.  Never respond with 'cannot be
determined', 'none of the above', or similar phrases.**
[END OF TASK INSTRUCTIONS]

[BEGIN OF FORMAT INSTRUCTIONS]
***
Output:
  - <thinking> A complete analysis synthesizing all tool results and CoT steps to derive the
answer.  </thinking>
  - <answer> **The final answer** </answer>
***
[END OF FORMAT INSTRUCTIONS]

CoT Steps:  {cot_steps}

***
CRITICAL: You MUST always provide a reasonable answer.  Never respond with 'cannot be
determined', 'none of the above', or similar phrases.**
Now, output **your thinking** between <thinking> and </thinking>, and **your answer** between
<answer> and </answer>.
***
```

Moreover, in the *Plan-Execute* paradigm, scenarios may arise where either (i) the *planner* ($\Phi_{\text{plan}}$) fails to generate a correct plan, or (ii) the *executor* ($\Phi_{\text{exe}}$) encounters tool invocation failures. To address this, we set a maximum attempt threshold (default to 3). When the system exceeds this limit without completing the *Plan-Execute* reasoning process, SpatialAgent will bypass the workflow and directly answer the question using the following prompt:

```
[BEGIN OF GOAL]
Provide a direct ANSWER to a spatial understanding questions about given 2D images or videos
without external tools.
[END OF GOAL]

[BEGIN OF TASK INSTRUCTIONS]
You are a helpful assistant tasked with solving spatial reasoning questions.  Think step by
step.
Answer the spatial understanding question by reasoning about the provided images or videos.
Provide a direct answer by reasoning logically based on typical spatial relationships and visual
cues in the images or videos.
**CRITICAL: You MUST always provide a reasonable answer.  Never respond with 'cannot be
determined', 'none of the above', or similar phrases.**
***
For multi-choice questions, select the most appropriate answer from options based on reasoning.
```

```
Respond ONLY with the capital letter and its parentheses.
For judgment questions, answer with yes or no based on reasoning.  Respond ONLY with 'yes' or
'no'.
For open-ended measurement questions, answer the question by measuring the precise distance in
3D space through a 2D images or videos.  DO NOT use generic and unclear units like 'units' or
'pixels'
Respond ONLY with a numeric answer consisting of a scalar and a distance unit in the format of
**scalar distance_unit**.
For other questions, answer the question based on the given image or video.  Respond ONLY with a
concise and accurate scalar or a scalar with corresponding unit.
***
[END OF TASK INSTRUCTIONS]

[BEGIN OF FORMAT INSTRUCTIONS]
***
Output your response in the format:
  <thinking> [Your reasoning here] </thinking>
  <answer> [Your final answer] </answer>
***
[END OF FORMAT INSTRUCTIONS]

***
**CRITICAL: You MUST always provide a reasonable answer.  Never respond with 'cannot be
determined', 'none of the above', or similar phrases.**
Now, output **your thinking** between <thinking> and </thinking>, and **your answer** between
<answer> and </answer>.
***
```

**ReAct Paradigm.** For the *ReAct* paradigm, SpatialAgent comprises three components: *observer* ($\Phi_{\text{obs}}$), *executor* ($\Phi_{\text{exe}}$), and *summarizer* ($\Phi_{\text{sum}}$). By default, SpatialAgent can perform up to 10 rounds of dialogue iterations in this paradigm. In each iteration, we use the following prompt to guide the *observer* ($\Phi_{\text{obs}}$) to decide the next action based on the current context:

```
USER REQUEST: {USER REQUEST}
[BEGIN OF GOAL]
You are a helpful assistant, and your goal is to solve the # USER REQUEST #.
You can either rely on your own capabilities or perform actions with external tools to help you.
A list of all available actions is provided to you below.
[END OF GOAL]

[BEGIN OF ACTIONS]
{for each action in actions}
[END OF ACTIONS]

[BEGIN OF TASK INSTRUCTIONS]
1.  You must only select actions from # ACTIONS #.
2.  You can only call one action at a time.
3.  If no action is needed, please make actions an empty list (i.e.  \actions":  []).
4.  You must always call **Terminate** with your final answer at the end.
5.  Please note that the priority of the SelfThinking tool is relatively low.  Please give
priority to using other tools, and only consider using this tool if the problem cannot be solved
otherwise.
[END OF TASK INSTRUCTIONS]

[BEGIN OF TOOL USAGE INSTRUCTIONS]
1.  **Construct the correct image path** for the tool to use, ensuring the path can be accessed
and read properly.
2.  For object distance and object size(Length, width, height,tall, short, slim, or heavy)
problems, first observe the image.  If the scene is outdoors, **FIRST** use `LocalizeObjects`
to obtain the 2D bounding boxes, then determine the pair of points (one from each object) that
are closest to each other, and use these points as the 'point' inputs for `Get3DDistance` to get
the distance between the two objects.
Do **NOT** simply use the center points of the boxes as the closest points between two objects.
3.  For counting-related problems, **USE** `CountObjects`; the number of returned points equals
```

```
the number of objects.
4.  For camera-related problems, you may need to **USE** `GetCameraParametersVGGT` to obtain the
camera parameters.
====================== CRITICAL WARNING ======================
**DO NOT** invent or mention any tool that is **NOT explicitly defined** in #ACTIONS#.
**DO NOT** fabricate tool usage results if you have NOT actually called the tool.
You MUST only describe tool results that are actually obtained during execution.
Violation of this rule is considered a **SERIOUS ERROR**.
==============================================================
====================== RELIABILITY WARNING ======================
If a tool result contains **ambiguous references** - for example, `LocalizeObjects` returns
multiple bounding boxes for the same object - **the result is unreliable**.
In such cases, you SHOULD rely on **reasoning** instead of depending on the tool output.
Treat this as a high-risk situation and avoid making decisions solely based on such tool
results.
==============================================================
================== TOOL CHAIN LENGTH WARNING ==================
If the tool invocation chain becomes **too long**, you MUST **STOP** calling further tools to
avoid reaching the maximum number of allowed calls.
In such cases, immediately switch to using **SelfThinking** to answer, **INCLUDING all input
images** required for reasoning.
Failure to follow this rule may result in task termination without producing a valid answer.
==============================================================
[END OF TOOL USAGE INSTRUCTIONS]

[BEGIN OF FORMAT INSTRUCTIONS]
Your output should be in a strict JSON format as follows:
{"thought":  "the thought process, or an empty string", "actions":  [{"name":  "action1",
"arguments":  {"argument1":  "value1", "argument2":  "value2"}}]}
[END OF FORMAT INSTRUCTIONS]

[BEGIN OF EXAMPLES]
{for each demo in demo_examples}
[END OF EXAMPLES]
```

Next, we employ the following prompt to instruct the *executor* ($\Phi_{\text{exe}}$) to perform the selected action and produce new intermediate results:

```
OBSERVATION: {OBSERVATION}
The OBSERVATION can be incomplete or incorrect, so please be critical and decide how to make use
of it.
If you've gathered sufficient information to answer the question, call **Terminate** with the
final answer.
Now, please generate the response for the next step.
```

Finally, the *summarizer* ($\Phi_{\text{sum}}$) adopts a prompt similar to that of the *executor* ($\Phi_{\text{exe}}$) to summarize all intermediate results and provide the final conclusion:

```
ALL_OBSERVATION: {ALL_OBSERVATION}
The ALL_OBSERVATION can be incomplete or incorrect, so please be critical and decide how to make
use of it.
Call **Terminate** with the final answer.
Now, please generate the response for the next step.
```

As mentioned above, when using the *observer* ($\Phi_{\text{obs}}$) to select actions, we first provide a set of examples, as shown below. It is important to note that, given that MLLMs may hallucinate when handling long contexts, for instance, pretending to call tools and producing fabricated results, we restrict these in-context examples to be concise, focused, and unambiguous.

```
[
   {
      "user_request": """Between image-0 and image-1, what is the primary direction of the
camera's movement? Please answer with one of the following options: A. The camera moved to
the right B. The camera moved to the left C. The camera moved downward D. The camera moved
upward""",
      "steps": [
         {
            "id": 1,
            "thought": "To determine the camera's movement direction, I need to compute the
average optical flow between the two images using RAFT. The average optical flow indicates pixel
offsets, where positive mean_flow_x suggests camera movement to the right, negative to the left,
positive mean_flow_y downward, and negative upward.",
            "actions": [{"name": "EstimateOpticalFlow", "arguments": {"images": ["image-0",
"image-1"]}}],
            "observation": {"output": {"mean_flow_x": 2.5, "mean_flow_y": -0.3}}
         },
         {
            "id": 2,
            "thought": "The optical flow results show mean_flow_x = 2.5 (positive, indicating
camera moved right) and mean_flow_y = -0.3 (negative, indicating camera moved up). Since the
absolute value of mean_flow_x (2.5) is much larger than mean_flow_y (0.3), the primary camera
movement is to the right.",
            "actions": [{"name": "Terminate", "arguments": {"answer": "(A)"}}],
            "observation": {}
         }
      ]
   },
   {
      "user_request": """In image-0, which direction is the person facing? A. Facing the
viewer while slightly to the right B. Facing the viewer while slightly to the left C. Facing
away from the viewer while slightly to the right D. Facing away from the viewer while slightly
to the left""",
      "steps": [
         {
            "thought": "To determine the precise orientation of the person, I need to use
GetObjectOrientation to analyze their position.",
            "actions": [{"name": "GetObjectOrientation", "arguments": {"image": "image-0",
"objects": "person"}}],
            "observation": {"results": [{öbject": "person", "angle_data": {"azimuth":
315.0, "polar": 90.0, "rotation": 0.0, "confidence": 0.89}, "error": 'null'}]}
         },
         {
            "thought": "The person's azimuth angle is 315 degrees, which falls in the range of
292.5° < $\phi$ < 337.5°. According to the orientation guidelines, this means the person is facing
the viewer and to the right of the viewer.",
            "actions": [{"name": "Terminate", "arguments": {"answer": "(A)"}}],
            "observation": {}
         }
      ]
   },
   {
      "user_request": """In image-0, what is the distance between the mug and the laptop? A.
15 centimeters B. 29 centimeters C. 45 centimeters D. 60 centimeters""",
      "steps": [
         {
            "thought": "First, I need to determine whether the scene in image-0 is indoors
or outdoors. Observing the image, I see a desk, a mug, and a laptop in an office setting,
indicating an indoor scene. Since it is indoors, according to the instruction, I do not need
to call any tools and can directly estimate the distance.",
            "actions": [],
            "observation": {}
         },
         {
            "thought": "Based on the visual cues in the image, the distance between the mug
```

```
and the laptop is approximately 29 centimeters.",
                "actions": [{"name": "Terminate", "arguments": {"answer": "(B)"}}],
                "observation": {}
            }
        ]
    },
    {
        "user_request": """In image-0, which object on the desk is larger in size? A. The mug B.
It can not be decided given the image only C. The laptop D. They seem to be of almost the same
size""",
        "steps": [
            {
                "thought": "First, I need to determine whether the scene in image-0 is indoors
or outdoors. Observing the image, I see a desk with a mug and a laptop in an office setting,
indicating an indoor scene(For object size questions (including length, width, height, tall,
short, slim, or heavy), do NOT call any tools. You should answer directly based on the visual
information from the image.). Since it is indoors, according to the instruction, I do not need
to call any tools and can directly estimate the relative size of the objects.",
                "actions": [],
                "observation": {}
            },
            {
                "thought": "Based on the visual cues in the image, the laptop is clearly larger in
size compared to the mug.",
                "actions": [{"name": "Terminate", "arguments": {"answer": "(C)"}}],
                "observation": {}
            }
        ]
    }
]
```

**Exception Handling.** Despite our carefully crafted prompts for SpatialAgent, it may still occasionally produce malformed outputs or fall into infinite loops. To mitigate this, beyond the prompt-based error-correction mechanisms that inspect abnormal intermediate results, we also verify whether the final extracted answer is an empty string or `null`. If such failures occur, the agent system is downgraded to a single agent core to complete the spatial understanding task. Concretely, when evaluated on the 5,025 samples in our SpatialScore benchmark, Qwen3-VL-4B-SpatialAgent-PE and Qwen3-VL-8B-SpatialAgent-PE exhibit 113 and 414 reasoning failures, corresponding to failure rates of 2.25% and 8.24%, respectively. In contrast, Qwen3-VL-4B-SpatialAgent-ReAct and Qwen3-VL-8B-SpatialAgent-ReAct each fail on only one sample, yielding a failure rate of just 0.02%. This trend aligns with the characteristics of the *Plan-Execute* and *ReAct* paradigms: the former is more efficient and direct but lacks strong error-correction capabilities, whereas the latter, though requiring more complex iterative reasoning, ensures much higher stability and success rates.

## C.4. Toolbox Specifications

To facilitate our proposed multi-agent system, **SpatialAgent**, to effectively perform visual reasoning for spatial understanding questions via tool invocation and collaboration, we have designed detailed input-output descriptions for each tool, accompanied by concrete examples. These specifications serve as contextual information for the agent core to select and invoke proper expert tools, with the details elaborated as follows.

For general perception tools, we first implement the *LocalizeObjects* action using the Rex-Omni [8] model, which localizes objects based on given text prompts. The detailed tool specification is presented below:

```
description = """
    Localize specific objects in an image.
    Returns bounding boxes for target categories, optionally visualizing them.
"""
args_spec = {
    "image": "The image to analyze.",
    "objects": "A list of object categories to detect."
}
```

```
rets_spec = {"regions": "List of detected regions with label, bbox"}
examples = [{
    "name": "LocalizeObjects",
    "arguments": {"image": "image-0", "objects": ["dog", "cat"]}
}]
```

Next, we adopt Rex-Omni [8] to create *CountObjects* function, which can count specific objects based on given text prompts, with the following functionality explanation:

```
description = """
    Count target objects in an image. Returns the coordinates of each detected target as points.
"""
args_spec = {
    "image": "The image to analyze.",
    "objects": "List of object categories to count."
}
rets_spec = {"points": "Dictionary {category: [points...]}, points in normalized coordinates.}
examples = [{"name": "CountObjects", "arguments": {"image": "image-0", "objects": ["bed"]}}]
```

Then, we integrate Rex-Omni [8] for advanced object localization and SAM2 [17] for precise segmentation, creating *GetObjectMask* function, with the following tool description:

```
description = """
    Generate pixel-level segmentation masks for specified objects.
    Returns mask area ratios and bounding boxes for each detected object.
    Suitable for analyzing object shapes, sizes, and coverage.
"""
args_spec = {
    "image": "Image file to process.",
    "objects": "List of object descriptions to localize and segment."
}
rets_spec = {
    "results": "List of dicts with mask area ratio, bounding box, and optional error:
[{'object': str, 'mask_area': float, 'bbox': [left, top, right, bottom], 'error': str or
None}]"
}
examples = [
    {"name": "GetObjectMask", "arguments": {"image": "image-0", "objects": ["coffee mug",
"microwave"]}}
]
```

Additionally, we employ DetAny3D [31] as the tool for 3D object detection and build the *Detect3DObjects* module, with the detailed tool specifications provided below:

```
description = """
    Detect specific objects in an image and estimate their 3D bounding boxes.
    Returns 3D bounding box parameters in the following format:
    x, y, z -> object center in camera coordinates (meters);
    width, height, length -> physical size (width, height, length) in meters;
    yaw -> heading angle around vertical axis (radians).
"""
args_spec = {
    "image": "Path to the input image."
    "objects": "List of object categories to detect (or a single string)."
}
rets_spec = {
    "objects": "List of dicts with {label: str, bbox_3d: {x:float, y:float, z:float,
width:float, height:float, length:float, yaw:float}}"
}
```

```
examples = [
    {"name": "Detect3DObjects", "arguments": {"image": ["image-1"], "objects": ["dog",
"rabbit"]}
]
```

To further equip SpatialAgent with motion understanding and image transformation analysis, we have developed specialized expert tools such as *EstimateOpticalFlow* action implemented with RAFT [20]:

```
description = """
    Estimate optical flow between two images to measure motion in pixels.
    Returns average displacement in horizontal (x) and vertical (y) directions.
    First image is earlier in time; second is later.
    - mean_flow_x > 0: objects move left / camera moves right.
    - mean_flow_x < 0: objects move right / camera moves left.
    - mean_flow_y > 0: objects move up / camera moves down.
    - mean_flow_y < 0: objects move down / camera moves up.
    Useful for analyzing camera motion, object movement, and 3D spatial reasoning.
"""
args_spec = {
    "image": "A list of exactly two image paths to compute optical flow between. First image is
earlier in time."
}
rets_spec = {
    "output": "Dictionary containing 'mean_flow_x' (average horizontal pixel displacement) and
'mean_flow_y' (average vertical pixel displacement)."
}
examples = [{"name": "EstimateOpticalFlow", "arguments": {"image": ["image-1", "image-3"]}]
```

The *MatchImagesSIFT* functionality performs keypoint extraction and feature matching between images using SIFT [13] implemented via OpenCV. The detailed specifications are as follows:

```
description = """
    Match keypoints between two images using SIFT.
    Detects distinctive features and returns matched coordinate pairs for tasks like alignment or
recognition.
"""
args_spec = {
    "image": "List of two image paths.",
    "num_keypoints": "Max keypoints per image (default: 1200).",
    "ratio_th": "Ratio test threshold for matching (default: 0.75)."
}
rets_spec = {
    "matches": "List of matched coordinate pairs: [[x1, y1], [x2, y2]].",
    "num_matches": "Total number of matches found."
}
examples = [
    {"name": "MatchImagesSIFT", "arguments": {"image": ["image-0", "image-1"], "num_keypoints":
1200, "ratio_th": 0.75}}
]
```

The *EstimateHomographyMatrix* tool, also implemented via OpenCV, calculates the homography transformation matrix between two images based on extracted keypoints, with the following function description:

```
description = """
    Compute a 3*3 homography matrix between two images using SIFT features and RANSAC.
    Useful for alignment, perspective correction, and planar transformations.
"""
args_spec = {
    "image": "List of two image paths.",
```

```
    "num_keypoints": "Max keypoints per image (default: 1200).",
    "ratio_th": "Ratio test threshold (default: 0.75).",
    "ransac_reproj_threshold": "Max reprojection error in RANSAC (default: 5.0)."
}
rets_spec = {
    "homography_matrix": "3*3 matrix mapping points from first image to second.",
    "inliers_count": "Number of inlier matches used.",
    "total_matches": "Total matches found.",
    "status": "Success or failure."
}
examples = [
    {"name": "EstimateHomographyMatrix", "arguments": {"image": ["image-0", "image-1"],
"num_keypoints": 1200, "ratio_th": 0.75, "ransac_reproj_threshold": 5.0}}
]
```

Moreover, VGGT [22] can predict the camera intrinsics and extrinsics for each frame within a frame sequence. We implement the *GetCameraParametersVGGT* module using the following tool description:

```
description = """
    Extract camera extrinsic (3*4, relative to first image) and intrinsic (3*3) parameters from
images using VGGT.
    Useful for 3D reconstruction, novel view synthesis, and geometric analysis.
"""
args_spec = {"image": "List of image paths (at least one)."}
rets_spec = {
    "output": "List of dicts with image_index (int), extrinsic (3*4 matrix), and intrinsic (3*3
matrix)."
}
examples = [
    {"name": "GetCameraParametersVGGT", "arguments": {"image": ["image-0", "image-1"]}}
]
```

Additionally, to empower SpatialAgent with 3D spatial reasoning capabilities, we have integrated several specialized visual geometry models. The *EstimateObjectGeometryProperties* function is implemented via the integration of SAM2 [17], Depth-Anything-V2 [29], and VGGT [22] to obtain detailed spatial and geometry properties of objects in the given image, with the following tool specification:

```
description = """
    Analyze objects in an image to obtain bounding boxes, mask areas, depth (m), and camera
parameters.
    Camera parameters include intrinsic (3*3) and extrinsic (3*4) matrices for 3D geometry tasks.
"""
args_spec = {
    "image": "Image file path to analyze.",
    "object_descs": "List of object descriptions (e.g., ['dog', 'cat'])."
}
rets_spec = {
    "results": "List of dicts with object, bbox, mask_area, depth (m), and optional error.",
    "camera_parameters": "Dict with intrinsic (3*3) and extrinsic (3*4) matrices."
}
examples = [
    {"name": "EstimateObjectGeometryProperties", "arguments": {"image": "image-0",
"object_descs": ["coffee cup", "keyboard"]}}
]
```

We have also implemented *EstimateRegionDepth* action with Depth-Anything-V2 [29] for scene depth estimation and region-specific average depth calculation based on given bounding boxes, with the following tool description:

```
description = """
   Estimate metric depth (in meters) of specified regions in an image.
   Supports indoor (0-20m) and outdoor (0-80m) scenes.
   Works with single or multiple bounding boxes in pixel coordinates.
   Depth is distance from camera to object, not between objects or object size.
"""
args_spec = {
   "image": "Image to analyze.",
   "bboxes": "Bounding box or list of boxes in pixel coordinates: [left, top, right, bottom]
or [[...], ...].",
   "indoor_or_outdoor": "Scene type ('indoor' or 'outdoor').",
   "mode": "Depth calculation: 'mean' (average) or 'center' (center point). Default:
'mean'."
}
rets_spec = {
   "depths": "List of dicts with bbox, depth (m), and optional error: ['bbox': list, 'depth':
float, 'error': str or None]",
   "unit": "Always 'meters'."
}
examples = [
   {"name": "EstimateRegionDepth", "arguments": {"image": "image-0", "bboxes": [100, 50,
200, 150], [150, 100, 250, 200] "indoor_or_outdoor": "indoor"}}
]
```

By combining Rex-Omni [8] and Depth-Anything-V2 [29], we also facilitate *EstimateObjectDepth*, with the corresponding specification as follows:

```
description = """
   Estimate object depth (in meters) from an image.
   Supports indoor (0-20m) and outdoor (0-80m) scenes.
   Depth indicates distance from camera to object, not between objects or object size.
"""
args_spec = {
   "image": "Image to analyze.",
   "objects": "List of object descriptions to measure distance to (e.g., ['dog', 'cat']).",
   "indoor_or_outdoor": "Scene type ('indoor' or 'outdoor')."
}
rets_spec = {
   "results": "List of dicts with object description, depth (m), and optional error:
['object': str, 'depth': float, 'error': str or None]"
}
examples = [
   {"name": "EstimateObjectDepth", "arguments": {"image": "image-0", "objects": ["the red
car", "dog"], "indoor_or_outdoor": "outdoor"}}
]
```

OrientAnything [24] model is employed for the *GetObjectOrientation* functionality:

```
description = """
   Estimate 3D orientation of objects in an image using Orient-Anything.
   Measures:
   - Azimuth: Horizontal rotation (0-360° clockwise)
   - Polar: Vertical inclination (0-180°)
   - Rotation: In-plane rotation (-180° to +180°)
   - Confidence: Reliability score
   Useful for 3D understanding, pose estimation, and spatial reasoning.
"""
args_spec = {
   "image": "Image to analyze.",
   "objects": "Object description(s) to analyze; string or list."
}
```

```
rets_spec = {
    "results": "List of dicts with object orientation data: [{'object': str, 'angle_data':
{'azimuth': float, 'polar': float, 'rotation': float, 'confidence': float}, 'error': str or
None}]"
}
examples = [
    {"name": "GetObjectOrientation", "arguments": {"image": "image-0", "objects": "a red
car"}}
]
```

To estimate metric-based distances between points in 3D space, we employ MapAnything [9] to reconstruct the 3D scene and compute the real-world distances between points, following the specification below:

```
description = """
    Calculates the absolute 3D spatial distance (in meters) between two pixel points (x, y) in an
image.
    Note: this tool should be used in outdoor scenes.
    Returns the calculated distance (in meters).
"""
args_spec = {
    "image": "Path to the input image.",
    "point_1": "List of [x, y] pixel coordinates for the first point."
    "point_2": "List of [x, y] pixel coordinates for the second point."
}
rets_spec = {
    "distance_meters": "The calculated 3D distance (float, in meters)."
}
examples = [
    {"name": "Get3DDistance", "arguments": {"image": "image-0", "point_1": [100, 100],
"point_2": [1000, 1000]}}
]
```

Moreover, we have developed a suite of general-purpose tools to assist in tool invocation and reasoning processes. We design a dedicated *Terminate* action to formally conclude the reasoning process and give structured final answers.

```
description = """
    Use this function ONLY when you are completely confident in your final answer.
    For multiple-choice questions: Specify the letter of the correct option.
    For numerical answers: Include both the specific value and appropriate unit of measurement
(e.g., meter or centimeter).
    For yes/no questions: Clearly state 'Yes' or 'No'.
    DO NOT call this function if you are uncertain or need to perform additional analysis.
    Double-check your answer before terminating!
"""
args_spec = {
    "answer": "The final answer with proper formatting. For multiple choice: include letter
(e.g., 'A. explanation' or '(B)'). For numerical answers: include units (e.g., '3.25 meters')."
}
rets_spec = {
    "answer": "The final answer that will be submitted."
}
examples = [
    {"name": "Terminate", "arguments": {"answer": "A. Yes."}},
    {"name": "Terminate", "arguments": {"answer": "(B)."}},
    {"name": "Terminate", "arguments": {"answer": "B. 3.25 meters."}},
    {"name": "Terminate", "arguments": {"answer": "(A) 2 inches."}},
    {"name": "Terminate", "arguments": {"answer": "47.3 centimeters."}},
    {"name": "Terminate", "arguments": {"answer": "38.2 degrees."}}
]
```

A *SelfThinking* module is designed to guide MLLMs (*e.g.*, Qwen3-VL) in self-reflecting on questions through meticulous

Table 3. **Quantitative Comparisons on SpatialScore-OpenSource Subset.** Qwen3-VL is adopted in two ways: (i) supervised fine-tuned on our SpatialCorpus; and (ii) as the agent core to conduct reasoning using the Plan-Execute (PE) and ReAct paradigms in SpatialAgent.

| Methods | Overall | Mental. | Count. | Depth. | View-Rea. | Obj-Size. | Obj-Loc. | Obj-Dist. | Obj-Mo. | Camera. | Temp-Rea. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ***Baselines*** | | | | | | | | | | | |
| Chance-level (Random) | 26.12 | 23.71 | 22.80 | 20.19 | 31.84 | 24.76 | 34.94 | 21.83 | 25.30 | 26.60 | 28.68 |
| Human-level | **87.66** | **96.87** | **89.72** | **81.49** | **92.15** | **84.05** | **90.34** | **75.58** | **92.99** | **89.41** | **84.19** |
| ***Representative Models*** | | | | | | | | | | | |
| Qwen3-VL-30B-A3B [3] | 48.60 | 46.31 | 58.86 | 45.06 | 47.98 | 53.87 | 61.38 | 33.65 | 56.10 | 40.39 | 45.59 |
| Qwen3-VL-32B [3] | 51.86 | 43.40 | 61.16 | 51.18 | 50.22 | **57.81** | 67.13 | **46.47** | 61.28 | 34.73 | 47.79 |
| Qwen2.5-VL-72B [3] | 45.44 | 53.69 | 49.49 | 55.88 | 36.10 | 47.21 | 60.92 | 32.13 | 53.96 | 37.93 | 22.43 |
| InternVL3-78B [34] | 48.23 | 50.34 | 59.19 | 44.46 | 45.74 | 44.78 | 65.98 | 35.74 | 57.32 | 37.93 | 43.75 |
| Qwen3-VL-235B-A22B [3] | 54.82 | 57.27 | **65.19** | 52.84 | 52.47 | 56.85 | 66.90 | 44.28 | **67.38** | 38.42 | 49.63 |
| Claude-4.5-Sonnet [2] | 44.64 | 51.01 | 49.67 | 48.32 | 39.91 | 50.02 | 50.57 | 36.31 | 53.35 | 27.59 | 41.18 |
| Gemini-2.5-Pro [5] | 56.70 | 73.38 | 64.06 | 51.06 | 46.41 | 54.70 | 69.89 | 43.83 | 64.63 | 45.57 | 56.25 |
| GPT-5 [16] | **59.09** | **78.08** | 57.59 | **56.33** | **54.04** | 56.85 | **70.97** | 42.22 | 67.07 | **47.29** | **62.13** |
| ***Qwen3-VL-4B*** | | | | | | | | | | | |
| Blind (text only) | 24.96 | 27.29 | 11.20 | 25.63 | 25.34 | 37.29 | 27.82 | 22.74 | 23.17 | 22.66 | 20.22 |
| Zero-shot | 40.39 | 37.81 | 48.22 | 36.81 | 34.75 | 43.70 | 54.71 | 26.92 | 50.61 | 35.96 | 38.24 |
| w/ SpatialCorpus (Ours) | **46.74** | **65.55** | 52.24 | **53.87** | 33.86 | 34.52 | 52.18 | **37.27** | 55.49 | 40.89 | **44.85** |
| w/ SpatialAgent-PE (Ours) | 45.28 | 56.15 | **54.98** | 42.40 | 36.55 | **50.05** | **58.85** | 28.92 | **57.32** | 31.53 | 38.24 |
| w/ SpatialAgent-ReAct (Ours) | 46.49 | 46.53 | 53.46 | 47.36 | **39.01** | 45.15 | 54.48 | 31.86 | 53.05 | **54.93** | 42.28 |
| ***Qwen3-VL-8B*** | | | | | | | | | | | |
| Blind (text only) | 27.81 | 21.70 | 17.21 | 27.71 | 30.49 | 39.38 | 41.61 | 26.33 | 19.21 | 26.11 | 20.59 |
| Zero-shot | 42.97 | 38.26 | 50.35 | 43.58 | 37.67 | 48.61 | 55.86 | 31.45 | 51.52 | 34.48 | 41.54 |
| w/ SpatialCorpus (Ours) | 48.72 | **57.27** | 52.67 | **58.59** | 36.77 | 49.84 | 55.40 | **40.51** | 54.88 | 37.93 | 44.85 |
| w/ SpatialAgent-PE (Ours) | 49.58 | 50.11 | **54.48** | 50.99 | 43.50 | **54.84** | **62.30** | 38.43 | **58.23** | 40.64 | 43.38 |
| w/ SpatialAgent-ReAct (Ours) | **50.01** | 44.30 | 53.49 | 51.55 | **45.74** | 51.39 | 58.62 | 37.92 | 57.62 | **51.97** | **51.84** |

Table 4. **Quantitative Comparisons on SpatialScore-Repurpose Subset.** Qwen3-VL is adopted in two ways: (i) supervised fine-tuned on our SpatialCorpus; and (ii) as the agent core to conduct reasoning using the Plan-Execute (PE) and ReAct paradigms in SpatialAgent.

| Methods | Overall | Depth. | Obj-Size. | Obj-Loc. | Obj-Dist. | Obj-Mo. | Camera. |
|---|---|---|---|---|---|---|---|
| ***Baselines*** | | | | | | | |
| Chance-level (Random) | 36.13 | 31.86 | 50.33 | 44.66 | 31.78 | 24.14 | 31.18 |
| Human-level | **82.79** | **85.12** | **89.29** | **67.18** | **90.70** | **100.00** | **84.14** |
| ***Representative Models*** | | | | | | | |
| Qwen3-VL-30B-A3B [3] | 58.30 | 59.92 | 66.12 | 74.81 | 65.26 | 72.41 | 37.90 |
| Qwen3-VL-32B [3] | 62.22 | 53.85 | 65.75 | 73.66 | 66.41 | **87.36** | 48.39 |
| Qwen2.5-VL-72B [3] | 59.15 | 57.38 | 64.31 | 65.27 | 60.21 | 80.46 | 48.39 |
| InternVL3-78B [34] | 59.47 | **63.01** | 70.37 | 65.27 | 65.89 | 72.41 | 45.43 |
| Qwen3-VL-235B-A22B [3] | **63.14** | 58.01 | **70.92** | **75.19** | **71.61** | 77.01 | 47.58 |
| Claude-4.5-Sonnet [2] | 49.45 | 44.05 | 68.75 | 58.78 | 46.88 | 22.99 | 45.43 |
| Gemini-2.5-Pro [5] | 55.18 | 51.39 | 70.59 | 60.69 | 58.14 | 30.12 | 52.15 |
| GPT-5 [16] | 54.63 | 51.15 | 67.77 | 61.45 | 54.69 | 19.54 | **54.86** |
| ***Qwen3-VL-4B*** | | | | | | | |
| Blind (text only) | 39.45 | 33.65 | 63.64 | 40.84 | 36.43 | 16.09 | 38.98 |
| Zero-shot | 50.21 | 40.60 | 64.48 | 67.18 | 55.87 | 62.07 | 31.99 |
| w/ SpatialCorpus (Ours) | **75.53** | **73.89** | **77.93** | **70.23** | **81.47** | **96.55** | **72.04** |
| w/ SpatialAgent-PE (Ours) | 62.07 | 57.26 | 71.51 | 68.32 | 63.57 | 72.41 | 53.23 |
| w/ SpatialAgent-ReAct (Ours) | 64.04 | 66.38 | 69.94 | 64.12 | 64.64 | 54.02 | 63.44 |
| ***Qwen3-VL-8B*** | | | | | | | |
| Blind (text only) | 41.23 | 32.34 | 52.89 | 52.67 | 34.88 | 26.44 | 37.90 |
| Zero-shot | 54.53 | 58.94 | 64.86 | **69.85** | 57.18 | 72.41 | 33.87 |
| w/ SpatialCorpus (Ours) | **76.29** | **82.54** | **82.06** | 68.32 | **80.62** | **100.00** | **70.97** |
| w/ SpatialAgent-PE (Ours) | 64.19 | 65.91 | 72.77 | 67.18 | 61.35 | 73.56 | 57.53 |
| w/ SpatialAgent-ReAct (Ours) | 67.51 | 63.09 | 75.09 | **69.85** | 63.57 | 73.56 | 64.78 |

prompt engineering, thereby fully leveraging their inherent potential to better tackle spatial understanding tasks.

```
description = """
    Modes:
    1.  Text-only:  Provide 'query' for pure language tasks.
    2.  Vision+Language:  Provide 'images' + 'query' for visual analysis.
    Suitable for:  Scene understanding, OCR, object/color recognition, classification, and
concept-level Q&A.
"""
args_spec = {
    "query":  "Text question or instruction (REQUIRED).",
    "image":  "List of image paths.  If omitted, the model performs text-only reasoning.",
}
rets_spec = {"response":  "Model's response string."}
examples = [
    {"name":  "SelfThinking", "arguments":  {"query":  "Summarize the image content.", "image":
"image-0"}}
]
```

## D. More Experiment Results

In this section, we present additional quantitative and qualitative results in Sec D.1 and Sec D.2, followed by comprehensive and in-depth analyses and discussions.

### D.1. Additional Quantitative Results

Here, we further conduct an in-depth analysis of the quantitative results on our proposed **SpatialScore** benchmark. Concretely, we divide the 5,025 samples in SpatialScore into two subsets: (i) the newly constructed subset repurposed from 3D annotations, denoted as **SpatialScore-Repurpose** (1,091 samples), and (ii) the remaining 3,934 samples collected from existing datasets and manually curated, which form the **SpatialScore-OpenSource** subset.

As presented in Tab. 3 and Tab. 4, we compare our data-driven and agent-based approaches with several representative baselines on these subsets, leading to the following key observations: (i) On both the SpatialScore-OpenSource and SpatialScore-Repurpose subsets, the Qwen3-VL models fine-tuned on SpatialCorpus, as well as our SpatialAgent, achieve substantial performance gains over their respective base models. This confirms the feasibility of both routes for enhancing spatial reasoning capabilities; (ii) Although supervised fine-tuning brings performance gains, it may introduce potential biases. For example, Qwen3-VL-8B fine-tuned on SpatialCorpus improves its overall accuracy on SpatialScore-OpenSource from 42.97 to 48.72, while on SpatialScore-Repurpose it increases from 54.53 to 76.29. This discrepancy largely arises because the latter subset is more closely aligned with the distribution of training data in SpatialCorpus; and (iii) In contrast, SpatialAgent shows a more moderate improvement on SpatialScore-Repurpose (from 54.53 to 67.51), but it also delivers consistent gains on the SpatialScore-OpenSource subset (from 48.72 to 50.01). As a training-free approach, it further preserves the base model's general capabilities and therefore avoids introducing distributional biases across different data sources.

### D.2. Additional Qualitative Results

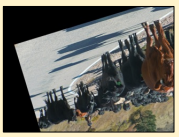
We further include more qualitative results on SpatialScore from the models fine-tuned on our SpatialCorpus, as well as from our constructed SpatialAgent, as presented in Fig. 5 and Fig. 6. These results demonstrate that both our data-driven and agent-based approaches achieve superior performance on spatial intelligence tasks, even surpassing larger-scale models and proprietary systems.

Which is further away from the camera?
A. kite; B. woman

**Ground Truth: B.**

**Qwen3-VL-4B**: A. ✗
**Qwen3-VL-8B**: A. ✗
**InternVL3-78B**: A. ✗
**Qwen3-VL-A22B**: A. ✗
**Gemini-2.5-Pro**: A. ✗
**GPT-5**: A. ✗

**Qwen3-VL-4B-SpatialCorpus**: A. ✗
**Qwen3-VL-4B-SpatialAgent-PE**: A. ✗
**Qwen3-VL-4B-SpatialAgent-ReAct**: **B.** ✓
**Qwen3-VL-8B-SpatialCorpus**: A. ✗
**Qwen3-VL-8B-SpatialAgent-PE**: A. ✗
**Qwen3-VL-8B-SpatialAgent-ReAct**: **B.** ✓



Compute the type and parameters of the spatial transformation between these two images.
A: {rot_angle: -163.08, trans_dx: 107.25, trans_dy: 44.19, scale: 1.033}
B: {rot_angle: 137.60, trans_dx: -27.01, trans_dy: -94.97, scale: 1.163}
C: {rot_angle: 137.30, trans_dx: 75.41, trans_dy: 55.66, scale: 1.134}
D: {rot_angle: -53.48, trans_dx: -52.11, trans_dy: -7.97, scale: 1.302}

**Ground Truth: A.**

**Qwen3-VL-4B**: C. ✗
**Qwen3-VL-8B**: D. ✗
**InternVL3-78B**: **A.** ✓
**Qwen3-VL-A22B**: D. ✗
**Gemini-2.5-Pro**: B. ✗
**GPT-5**: C. ✗

**Qwen3-VL-4B-SpatialCorpus**: **A.** ✓
**Qwen3-VL-4B-SpatialAgent-PE**: C. ✗
**Qwen3-VL-4B-SpatialAgent-ReAct**: **A.** ✓
**Qwen3-VL-8B-SpatialCorpus**: **A.** ✓
**Qwen3-VL-8B-SpatialAgent-PE**: **A.** ✓
**Qwen3-VL-8B-SpatialAgent-ReAct**: D. ✗

Please compute the homography matrix between these two images.
A. [[1.153, 0.013, 244.44], [0.415, 1.194, -155.59], [0.001, 6e-05, 1.009]];
B. [[0.547, 0.209, -108.35], [-0.082, 1.118, -236.48], [0.0, 0.0, 1.000]];
C. [[1.211, -0.032, 47.374], [0.210, 1.035, -107.36], [0.001, -6e-06, 1.000]];
D. [[1.776, -0.053, 263.17], [0.418, 1.599, -329.46], [0.001, 3e-05, 1.001]]

**Ground Truth: D.**

**Qwen3-VL-4B**: A. ✗
**Qwen3-VL-8B**: C. ✗
**InternVL3-78B**: C. ✗
**Qwen3-VL-A22B**: C. ✗
**Gemini-2.5-Pro**: C. ✗
**GPT-5**: C. ✗

**Qwen3-VL-4B-SpatialCorpus**: A. ✗
**Qwen3-VL-4B-SpatialAgent-PE**: A. ✗
**Qwen3-VL-4B-SpatialAgent-ReAct**: **D.** ✓
**Qwen3-VL-8B-SpatialCorpus**: A. ✗
**Qwen3-VL-8B-SpatialAgent-PE**: **D.** ✓
**Qwen3-VL-8B-SpatialAgent-ReAct**: **D.** ✓

Please compute the homography matrix between the two images.
A. [[0.916, -0.198, 70.502], [0.072, 0.684, -33.187], [6e-06, 0.0, 0.999]];
B. [[1.290, 0.435, -118.46], [-0.026, 1.423, 161.89], [-3e-05, 0.0, 1.001]];
C. [[4.372, 0.144, -818.24], [-0.252, 3.960, -549.15], [0.002, 0.0, 0.978]];
D. [[0.402, -0.339, 102.29], [-0.213, 0.624, 216.78], [0.0, -2e-05, 1.0018]]

**Ground Truth: A.**

**Qwen3-VL-4B**: **A.** ✓
**Qwen3-VL-8B**: D. ✗
**InternVL3-78B**: **A.** ✓
**Qwen3-VL-A22B**: **A.** ✓
**Gemini-2.5-Pro**: B. ✗
**GPT-5**: B. ✗

**Qwen3-VL-4B-SpatialCorpus**: **A.** ✓
**Qwen3-VL-4B-SpatialAgent-PE**: **A.** ✓
**Qwen3-VL-4B-SpatialAgent-ReAct**: C. ✗
**Qwen3-VL-8B-SpatialCorpus**: **A.** ✓
**Qwen3-VL-8B-SpatialAgent-PE**: C. ✗
**Qwen3-VL-8B-SpatialAgent-ReAct**: C. ✗

What is the size of this room (in square meters)? If multiple rooms are shown, estimate the size of the combined space.

**Ground Truth: 10.1**

**Qwen3-VL-4B**: 17.0 ✗
**Qwen3-VL-8B**: 15.0 ✗
**InternVL3-78B**: 12.1 ✗
**Qwen3-VL-A22B**: 14.0 ✗
**Gemini-2.5-Pro**: **10-12** ✓
**GPT-5**: 9 ✗

**Qwen3-VL-4B-SpatialCorpus**: 13.0 ✗
**Qwen3-VL-4B-SpatialAgent-PE**: 12.0 ✗
**Qwen3-VL-4B-SpatialAgent-ReAct**: 12.0 ✗
**Qwen3-VL-8B-SpatialCorpus**: 14.0 ✗
**Qwen3-VL-8B-SpatialAgent-PE**: 12.0 ✗
**Qwen3-VL-8B-SpatialAgent-ReAct**: **10.5** ✓

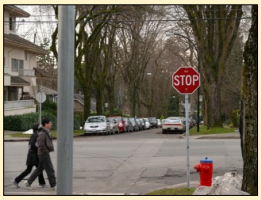
How many people on the subway platform are walking to the right?
A. 1; B. 8;
C. 4; D. 0

**Ground Truth: A.**

**Qwen3-VL-4B**: C. ✗
**Qwen3-VL-8B**: C. ✗
**InternVL3-78B**: C. ✗
**Qwen3-VL-A22B**: C. ✗
**Gemini-2.5-Pro**: D. ✗
**GPT-5**: D. ✗

**Qwen3-VL-4B-SpatialCorpus**: **A.** ✓
**Qwen3-VL-4B-SpatialAgent-PE**: **A.** ✓
**Qwen3-VL-4B-SpatialAgent-ReAct**: C. ✗
**Qwen3-VL-8B-SpatialCorpus**: **A.** ✓
**Qwen3-VL-8B-SpatialAgent-PE**: C. ✗
**Qwen3-VL-8B-SpatialAgent-ReAct**: C. ✗

Figure 5. **Additional Qualitative Results.**



Consider the real-world 3D. Is the stop sign directly above the fire hydrant?
A. Yes;  B. No

**Ground Truth: A.**

**Qwen3-VL-4B**: B. ✗
**Qwen3-VL-8B**: B. ✗
**InternVL3-78B**: B. ✗
**Qwen3-VL-A22B**: B. ✗
**Gemini-2.5-Pro**: B. ✗
**GPT-5**: B. ✗

**Qwen3-VL-4B-SpatialCorpus**: **A.** ✓
**Qwen3-VL-4B-SpatialAgent-PE**: B. ✗
**Qwen3-VL-4B-SpatialAgent-ReAct**: B. ✗
**Qwen3-VL-8B-SpatialCorpus**: **A.** ✓
**Qwen3-VL-8B-SpatialAgent-PE**: B. ✗
**Qwen3-VL-8B-SpatialAgent-ReAct**: B. ✗

How far is the center of decoration (red point) from the observer in the photograph? Ensure the answer is in meters.

**Ground Truth: 2.8.**

**Qwen3-VL-4B**: 1.5. ✗
**Qwen3-VL-8B**: 1.8. ✗
**InternVL3-78B**: 1.5. ✗
**Qwen3-VL-A22B**: 2.1. ✗
**Gemini-2.5-Pro**: 1.5. ✗
**GPT-5**: 1.6. ✗

**Qwen3-VL-4B-SpatialCorpus**: **3.0.** ✓
**Qwen3-VL-4B-SpatialAgent-PE**: 1.5. ✗
**Qwen3-VL-4B-SpatialAgent-ReAct**: **2.5.** ✓
**Qwen3-VL-8B-SpatialCorpus**: **3.0.** ✓
**Qwen3-VL-8B-SpatialAgent-PE**: 13.9. ✗
**Qwen3-VL-8B-SpatialAgent-ReAct**: 3.5. ✗

Please compute the homography matrix between these two images.
A. [[0.371, -0.024, 139.16], [-0.095, 0.627, 65.353], [-0.001, -2e-05, 0.999]];
B. [[0.753, -0.009, -71.273], [-0.176, 0.976, 6.311], [0.0, -2e-05, 1.0]];
C. [[0.983, -0.005, 22.209], [-0.012, 0.971, 45.658], [-8e-06, -2e-05, 1.0]];
D. [[1.050, 0.026, 108.77], [0.195, 1.005, -7.890], [0.001, -3e-05, 1.0]]

**Ground Truth: A.**

**Qwen3-VL-4B**: C. ✗
**Qwen3-VL-8B**: C. ✗
**InternVL3-78B**: C. ✗
**Qwen3-VL-A22B**: C. ✗
**Gemini-2.5-Pro**: D. ✗
**GPT-5**: D. ✗

**Qwen3-VL-4B-SpatialCorpus**: B. ✗
**Qwen3-VL-4B-SpatialAgent-PE**: C. ✗
**Qwen3-VL-4B-SpatialAgent-ReAct**: **A.** ✓
**Qwen3-VL-8B-SpatialCorpus**: C. ✗
**Qwen3-VL-8B-SpatialAgent-PE**: C. ✗
**Qwen3-VL-8B-SpatialAgent-ReAct**: **A.** ✓

Beginning at the open door and facing the nearest red wall. Navigate to the TV: 1. [fill in] 2. Go forward until the wall 3. [fill in] 4. Go forward.

A. Turn Left, Turn Left,
B. Turn Right, Turn Left,
C. Turn Right, Turn Right,
D. Turn Left, Turn Right

**Ground Truth: A.**

**Qwen3-VL-4B**: C. ✗
**Qwen3-VL-8B**: B. ✗
**InternVL3-78B**: **A.** ✓
**Qwen3-VL-A22B**: D. ✗
**Gemini-2.5-Pro**: B. ✗
**GPT-5**: C. ✗

**Qwen3-VL-4B-SpatialCorpus**: **A.** ✓
**Qwen3-VL-4B-SpatialAgent-PE**: B. ✗
**Qwen3-VL-4B-SpatialAgent-ReAct**: B. ✗
**Qwen3-VL-8B-SpatialCorpus**: **A.** ✓
**Qwen3-VL-8B-SpatialAgent-PE**: B. ✗
**Qwen3-VL-8B-SpatialAgent-ReAct**: B. ✗

Measuring from the closest point of each object, what is the distance between the door and the bed (in meters)?

**Ground Truth: 2.2**

**Qwen3-VL-4B**: 1.4. ✗
**Qwen3-VL-8B**: 1.4. ✗
**InternVL3-78B**: 1.5. ✗
**Qwen3-VL-A22B**: **1.9.** ✓
**Gemini-2.5-Pro**: 1.4. ✗
**GPT-5**: **2.0.** ✓

**Qwen3-VL-4B-SpatialCorpus**: **2.4.** ✓
**Qwen3-VL-4B-SpatialAgent-PE**: 1.4. ✗
**Qwen3-VL-4B-SpatialAgent-ReAct**: **2.5.** ✓
**Qwen3-VL-8B-SpatialCorpus**: **2.4.** ✓
**Qwen3-VL-8B-SpatialAgent-PE**: 3.0. ✗
**Qwen3-VL-8B-SpatialAgent-ReAct**: 3.0. ✗

What will be the first-time appearance order of: door, computer tower, whiteboard, kettle?

A. whiteboard, computer tower, door, kettle
B. whiteboard, door, computer tower, kettle
C. door, computer tower, whiteboard, kettle
D. whiteboard, door, kettle, computer tower

**Ground Truth: A.**

**Qwen3-VL-4B**: B. ✗
**Qwen3-VL-8B**: B. ✗
**InternVL3-78B**: **A.** ✓
**Qwen3-VL-A22B**: B. ✗
**Gemini-2.5-Pro**: B. ✗
**GPT-5**: D. ✗

**Qwen3-VL-4B-SpatialCorpus**: **A.** ✓
**Qwen3-VL-4B-SpatialAgent-PE**: B. ✗
**Qwen3-VL-4B-SpatialAgent-ReAct**: B. ✗
**Qwen3-VL-8B-SpatialCorpus**: **A.** ✓
**Qwen3-VL-8B-SpatialAgent-PE**: B. ✗
**Qwen3-VL-8B-SpatialAgent-ReAct**: **A.** ✓

Figure 6. **Additional Qualitative Results.**

## References


[1] Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025. 13

[2] Anthropic. System card: Claude sonnet 4.5, 2025. 11, 26

[3] 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. 26

[4] Garrick Brazil, Abhinav Kumar, Julian Straub, Nikhila Ravi, Justin Johnson, and Georgia Gkioxari. Omni3d: A large benchmark and model for 3d object detection in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2023. 3, 4, 8

[5] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025. 11, 26

[6] 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 *Proceedings of the European Conference on Computer Vision*, 2024. 3

[7] Google. A new era of intelligence with gemini 3, 2025. 11

[8] Qing Jiang, Junan Huo, Xingyu Chen, Yuda Xiong, Zhaoyang Zeng, Yihao Chen, Tianhe Ren, Junzhi Yu, and Lei Zhang. Detect anything via next point prediction. *arXiv preprint arXiv:2510.12798*, 2025. 20, 21, 24

[9] Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, et al. Mapanything: Universal feed-forward metric 3d reconstruction. In *International Conference on 3D Vision*, 2026. 25

[10] Justin Lazarow, David Griffiths, Gefen Kohavi, Francisco Crespo, and Afshin Dehghan. Cubify anything: Scaling indoor 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2025. 3, 4, 5, 6, 8

[11] Yuan-Hong Liao, Rafid Mahmood, Sanja Fidler, and David Acuna. Reasoning paths with reference objects elicit quantitative spatial reasoning in large vision-language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2024. 3

[12] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024. 7

[13] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004. 22

[14] Wufei Ma, Haoyu Chen, Guofeng Zhang, Celso M de Melo, Alan Yuille, and Jieneng Chen. 3dsrbench: A comprehensive 3d spatial reasoning benchmark. In *Proceedings of the International Conference on Computer Vision*, 2025. 2

[15] Fanqing Meng, Chuanhao Li, Jin Wang, Quanfeng Lu, Hao Tian, Tianshuo Yang, Jiaqi Liao, Xizhou Zhu, Jifeng Dai, Yu Qiao, et al. Mmiu: Multimodal multi-image understanding for evaluating large vision-language models. In *Proceedings of the International Conference on Learning Representations*, 2025. 2, 3

[16] OpenAI. GPT-5 System Card. Technical report, OpenAI, 2025. Accessed: 2025-11-1. 11, 12, 26

[17] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. In *Proceedings of the International Conference on Learning Representations*, 2025. 21, 23

[18] Chan Hee Song, Valts Blukis, Jonathan Tremblay, Stephen Tyree, Yu Su, and Stan Birchfield. Robospatial: Teaching spatial understanding to 2d and 3d vision-language models for robotics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2025. 2

[19] 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. 11

[20] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Proceedings of the European Conference on Computer Vision*, 2020. 22

[21] 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 *Conference on Neural Information Processing Systems*, 2024. 2

[22] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2025. 23

[23] Wenqi Wang, Reuben Tan, Pengyue Zhu, Jianwei Yang, Zhengyuan Yang, Lijuan Wang, Andrey Kolobov, Jianfeng Gao, and Boqing Gong. Site: towards spatial intelligence thorough evaluation. In *Proceedings of the International Conference on Computer Vision*, 2025. 2

[24] Zehan Wang, Ziang Zhang, Tianyu Pang, Chao Du, Hengshuang Zhao, and Zhou Zhao. Orient anything: Learning robust object orientation estimation from rendering 3d models. In *Proceedings of the International Conference on Machine Learning*, 2025. 24

[25] Hongchi Xia, Yang Fu, Sifei Liu, and Xiaolong Wang. Rgbd objects in the wild: scaling real-world 3d object learning from rgb-d videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2024. 3, 6, 8

[26] Guowei Xu, Peng Jin, Li Hao, Yibing Song, Lichao Sun, and Li Yuan. Llava-o1: Let vision language models reason step-by-step. In *Proceedings of the International Conference on Computer Vision*, 2025. 11

[27] Jihan Yang, Shusheng Yang, Anjali W Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2025. 11, 13

[28] Kaiyu Yang, Olga Russakovsky, and Jia Deng. Spatialsense: An adversarially crowdsourced benchmark for spatial relation recognition. In *Proceedings of the International Conference on Computer Vision*, 2019. 2

[29] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. In *Conference on Neural Information Processing Systems*, 2024. 23, 24

[30] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the International Conference on Computer Vision*, 2023. 3, 5, 8

[31] Hanxue Zhang, Haoran Jiang, Qingsong Yao, Yanan Sun, Renrui Zhang, Hao Zhao, Hongyang Li, Hongzi Zhu, and Zetong Yang. Detect anything 3d in the wild. In *Proceedings of the International Conference on Computer Vision*, 2025. 21

[32] Jiahui Zhang, Yurui Chen, Yanpeng Zhou, Yueming Xu, Ze Huang, Jilin Mei, Junhui Chen, Yu-Jie Yuan, Xinyue Cai, Guowei Huang, et al. From flatland to space: Teaching vision-language models to perceive and reason in 3d. In *Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. 3

[33] Yang Zheng, Adam W Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J Guibas. Pointodyssey: A large-scale synthetic dataset for long-term point tracking. In *Proceedings of the International Conference on Computer Vision*, 2023. 3, 6, 8

[34] Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Yuchen Duan, Hao Tian, 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. 26