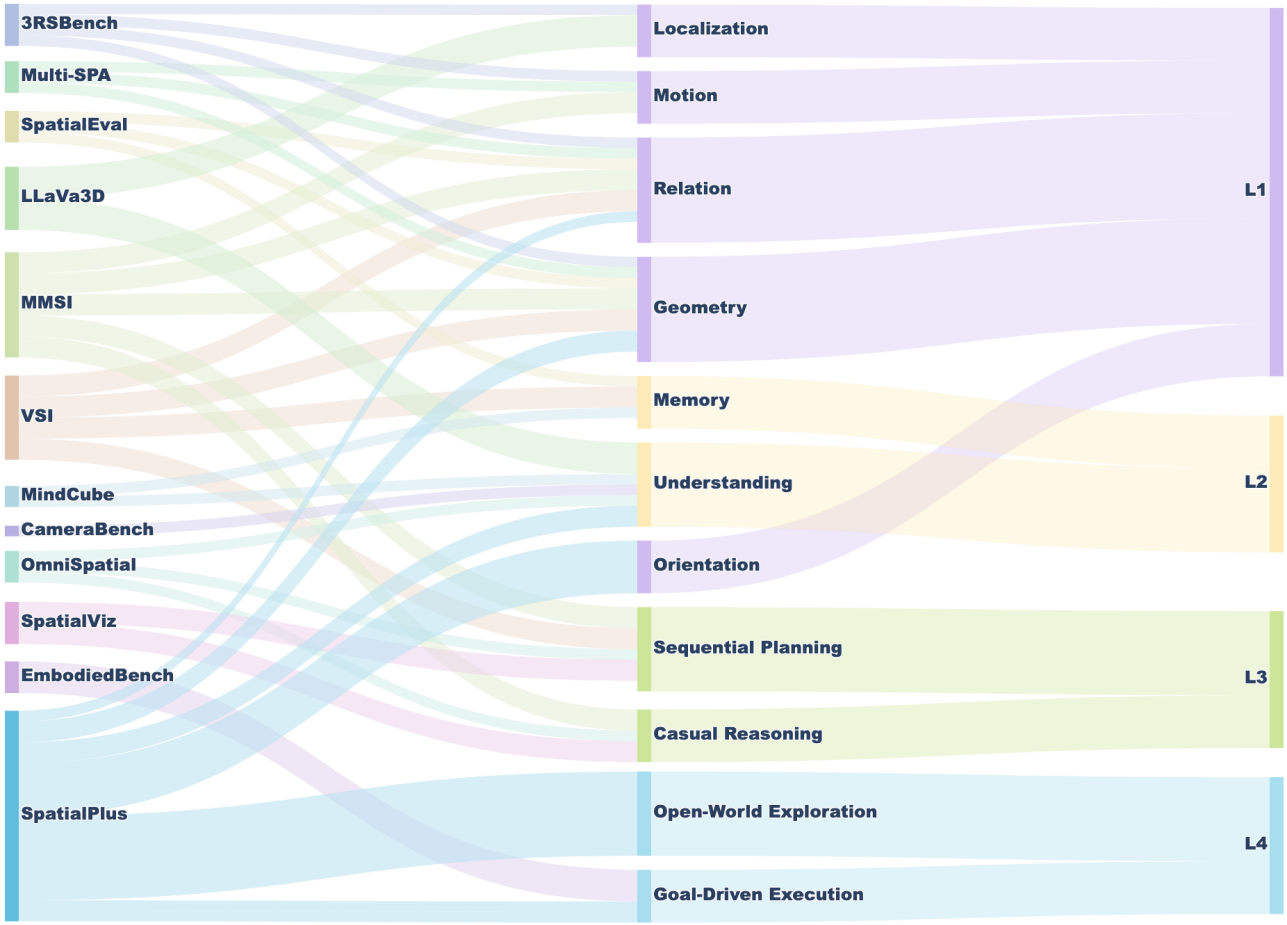


Figure A. **Construction of SpatialTree-Bench.** We build our benchmark by reorganizing various existing datasets and mapping them to our capability tree, where **SpatialPlus**, a complementary dataset are introduced to ensure the capability coverage.

A. Visualization of Data Sources

How different datasets contribute to our SpatialTree evaluation is shown in Fig. A. As illustrated in Fig. B, these metrics are primarily categorized into multiple-choice questions (70.7%), numeric accuracy (e.g., mean relative accuracy), LLM-based evaluation (LLM-as-a-Judge) and specific numeric metrics.

B. Evaluation Metrics Details

Multi-Option QAs. For multi-option question answering, each model is evaluated on its ability to select the correct option from a predefined set. We measure accuracy by comparing the predicted choice against the ground-truth answer. This paradigm captures a model's understanding of spatial relations, object properties, and causal dynamics within a scene, corresponding to the low- and mid-level capabilities in the SpatialTree (L1–L3).

Numeric QAs. Numeric QAs require models to predict continuous quantities such as distances, angles, or 3D coordinates. We evaluate performance using relative error metrics, for example:

$$\text{Relative Error} = \frac{|\hat{y} - y|}{|y|},$$

where $\hat{y}$ is the predicted value and y is the ground truth. This metric ensures that predictions are scaled appropriately across different magnitudes and emphasizes precision in spatial reasoning.

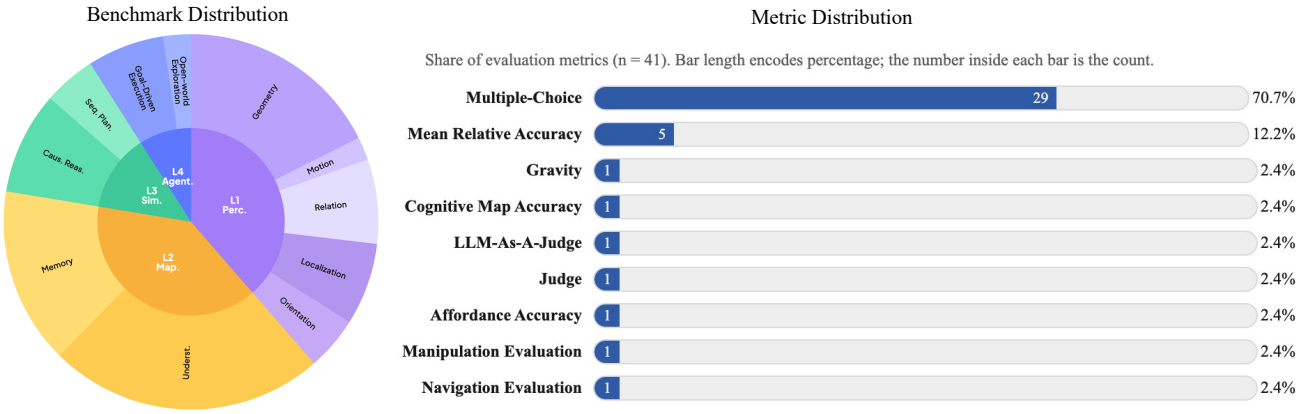


Figure B. **Distribution of Benchmark data and Evaluation Metrics.** We analyze the metric usage across 41 tasks in our benchmark. The evaluation relies primarily on multiple-choice questions (70.7%), complemented by task-specific numeric metrics (e.g., cognitive map accuracy) and LLM-as-a-Judge protocols.

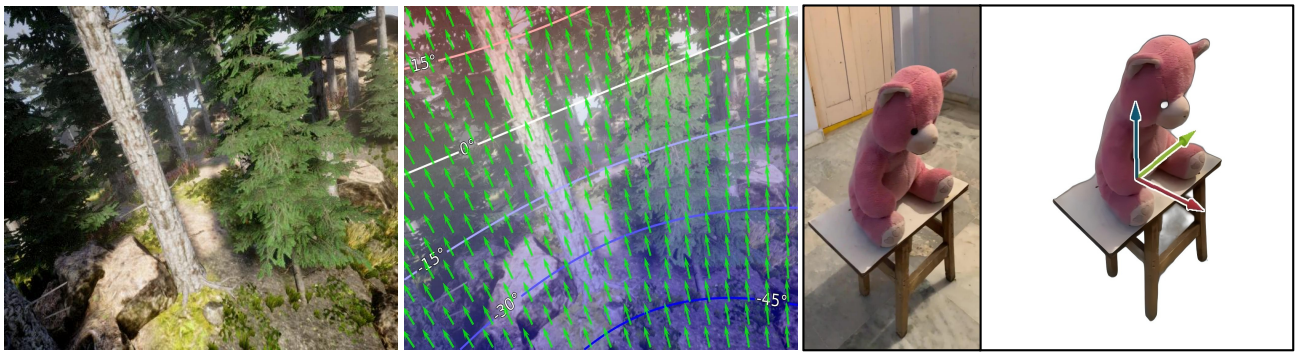


Figure C. **Orientation Annotations.** The left side is the gravity field estimated from GeoCalib [56], while the right side is from OrientAnything.

GPT Judge. For tasks that are open-ended or involve complex reasoning (e.g., trajectory description, action sequence explanation), we leverage a GPT-based judge to assess correctness. The judge evaluates whether the generated response satisfies the task requirements, optionally scoring partial correctness. This approach allows flexible evaluation beyond rigid numeric or multiple-choice formats, especially for mid- and high-level capabilities in L3–L4.

Agentic Evaluation. To assess agentic competence, models are deployed in interactive simulated environments, such as those provided by EmbodiedBench [70]. We evaluate navigation and manipulation tasks along multiple dimensions: success rate in completing the target goal, relative translation accuracy, and directional alignment. For each action step, a combined metric is computed using relative distance and cosine similarity of movement vectors, producing a normalized score in $[0, 1]$. Aggregating scores over all steps yields a comprehensive measure of an agent's ability to plan and execute actions in long-horizon, embodied tasks.

C. SpatialPlus: Complementary Data Annotations for SpatialTree

C.1. Orientations (L1)

The Orientation capability, a fundamental yet under-explored area, involves estimating both gravity direction and 3D object orientation. To generate annotations, we leveraged Geocalib [56] for gravity vector estimation and OrientAnything [61] for object poses. We applied these tools to datasets suited for each task: for gravity, we annotated 500 images sampled from the diverse, drone-captured TartanAir [59] dataset; for object orientation, we utilized the object-centric Co3dv2 [48] dataset (Seen in Fig. C). For gravity, the goal is to estimate the camera's orientation relative to the gravity vector, typically represented

by the *pitch* and *roll* angles. Formally, let the gravity vector in the world frame be:

$$\mathbf{g}_w = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}, \tag{1}$$

and let $\mathbf{R}_{cw} \in SO(3)$ denote the rotation from the world frame to the camera frame. The gravity direction in the camera frame is then:

$$\mathbf{g}_c = \mathbf{R}_{cw}\, \mathbf{g}_w. \tag{2}$$

From $\mathbf{g}_c = [g_x, g_y, g_z]^\top$, the pitch and roll angles can be computed as:

$$\text{pitch} = \arctan 2(-g_x, \sqrt{g_y^2 + g_z^2}), \tag{3}$$

$$\text{roll} = \arctan 2(g_y, g_z). \tag{4}$$

Here, *pitch* measures the forward–backward tilt of the camera, while *roll* measures the sideways tilt. To evaluate an MLLM's proficiency in this task, we require the model to analyze the input image and return these same three parameters in a structured JSON format. An example of our prompt template is shown in Fig. D.

```
{
  "role": "system",
  "content": "You are a vision model specialized in estimating camera orientation from images. Your task
is to infer the gravity direction from the input image by predicting the camera's pitch and roll angles,
as well as the vertical field of view (vFOV). Always output your prediction strictly in the following
JSON format:
  {
    \"pitch\": <float, camera pitch angle in degrees>,
    \"roll\": <float, camera roll angle in degrees>,
    \"vfov\": <float, vertical field of view in degrees>
  }
  "Do not include any additional text or explanation outside of the JSON object."
}
```

Figure D. **Prompt template** for Orientation Estimation.

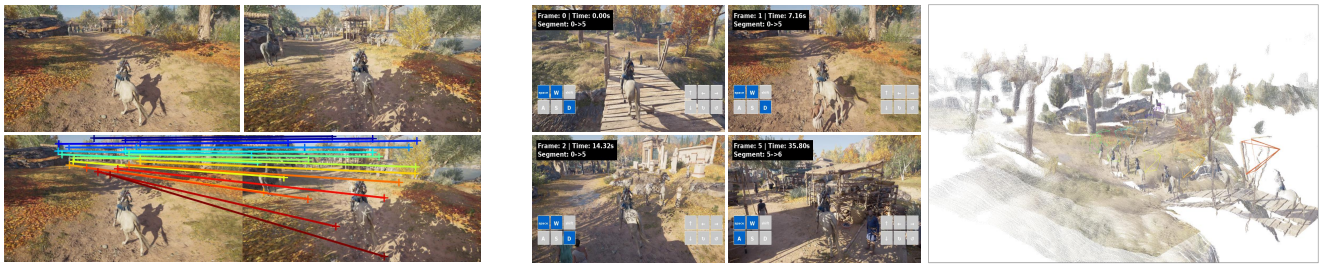
For evaluation, we move beyond a simple absolute error metric and adopt a probabilistic approach that accounts for the inherent uncertainty of the ground-truth annotations provided by *Geocalib*. For each predicted parameter (pitch, roll, and vFOV), *Geocalib* also outputs an uncertainty value, which we interpret as the standard deviation (σ_{gt}). We then calculate a normalized similarity score (S) for each parameter using a Gaussian kernel, defined as:

$$S(y_{\text{pred}}, y_{\text{gt}}, \sigma_{\text{gt}}) = \exp\left(-\frac{(y_{\text{pred}} - y_{\text{gt}})^2}{2\sigma_{\text{gt}}^2}\right) \tag{5}$$

where y_{pred} is the MLLM's prediction, y_{gt} is the ground-truth value from *Geocalib*, and σ_{gt} is its associated uncertainty. This scoring function gracefully penalizes deviations from the ground truth: the score is 1 for a perfect match and decays towards 0 as the error increases. Crucially, a larger uncertainty σ_{gt} in the ground truth leads to a slower decay, making the scoring more lenient when the ground truth itself is less certain. The final score for the task is the average of the individual scores for pitch, roll, and vFOV. For object orientation estimation, most of metrics are similar to gravity, and the evaluation are conducted on Azimuth, Polar and Rotation these three angles.

C.2. Agentic Competence (L4)

Spatial Action Mapping. In the context of spatial agents, navigation and manipulation represent the most common forms of interaction within 3D environments. We address each with a distinct action space design. For navigation, we conceptualize agent movement as a series of camera motion controls (referring to recent video world models [4, 40]). To enable precise and intuitive control, we decompose complex camera movements into a set of fundamental motion primitives inspired by



(a) (b)

Figure E. **Navigation Data Curation.** (a) shows paired images used for evaluation, where MLLMs are expected to move from left to right. (b) illustrates our curation process: reconstructing metric 3D models and camera trajectories, then converting them into actions.

Table A. **Spatial Action Mapping.** This table defines a standardized interface that maps continuous 6-DoF motions and discrete control signals into action primitives with unified parameterization, enabling MLLMs to plan and execute embodied behaviors for agentic competence evaluation.

Primitive	**Primitive Term**	**Category**	**Description**	**Action Mapping**	**Param.**	**Threshold**
P_{truck}	Truck	Translation	Move camera left/right (X-axis)	A/D	v_x	± 0.01 m/s
P_{dolly}	Dolly	Translation	Move camera forward/backward (Z-axis)	W/S	v_z	± 0.01 m/s
P_{pedestal}	Pedestal	Translation	Move camera up/down (Y-axis)	Q/E	v_y	± 0.01 m/s
P_{pan}	Pan	Rotation	Turn camera left/right (yaw)	$\leftarrow$ / $\rightarrow$	ω_y	$\pm 0.5^\circ$/s
P_{tilt}	Tilt	Rotation	Tilt camera up/down (pitch)	$\uparrow$ / $\downarrow$	ω_x	$\pm 0.5^\circ$/s
P_{roll}	Roll	Rotation	Roll camera CW/CCW (roll)	Z/X	ω_z	$\pm 0.5^\circ$/s
O_{gripper}	Gripper	Gripper Control	Open or close the gripper	G/H	State $\in \{0, 1\}$	N/A
$O_{\text{push/pull}}$	Push/Pull	Gesture	Push or pull object along forward axis	P/L	Dir. $\in \{-1, +1\}$	N/A
O_{grab}	Grab	Gesture	Grab or release object	None	State $\in \{0, 1\}$	G/H

established cinematography techniques. This approach allows us to translate high-level language instructions (e.g., "move to the left," "look up") into a structured, low-level action space. The six fundamental primitives, their corresponding cinematic terms, degrees of freedom (DoF), and parameterization are detailed in Tab. A.

Formally, the camera trajectories are defined with a series of Camera-to-World (C2W) transformation matrices $\mathcal{T}_{\text{motion}} = \{\mathbf{T}|_0^i, i = 0, 1, \ldots, t\}$, while the camera transformation at each moment is $\mathcal{T}_{i \to i+1} = \mathbf{T}_{i+1}\mathbf{T}_i^{-1},\ i = 0, 1, \ldots, t-1$. Then the continuous camera transformation can be decomposed into different components corresponding to each motion primitive, and discretized into the navigation action A_{nav} using a speed threshold:

$$\begin{aligned} \mathbf{A}_i^{\text{nav}} = \mathbf{T}_{i \to i+1} = \{\Delta\mathbf{R}, \Delta\mathbf{t}\} \\ \approx \left\{ \begin{array}{l} t_i \cdot v_i,\ t_k \cdot \omega_k \mid i, k \in \{x, y, z\}, \\ t_i, t_k \in \mathbb{Z}_{\geq 0} \end{array} \right\} \end{aligned} \tag{6}$$

where $\Delta\mathbf{R} = (\Delta R_x, \Delta R_y, \Delta R_z)$ represents the rotation components obtained via Euler decomposition, $\Delta\mathbf{t} = (\Delta t_x, \Delta t_y, \Delta t_z)$ denotes the translation components along the x, y, and z axes, and t_i, t_k are discrete integers ranging from 0 up to the video frame rate (FPS). For manipulation, we focus on two representative scenarios to simplify the problem and enable controlled evaluation: human-hand manipulation and robotic gripper manipulation. For the gripper setting, we include gripper open/close actions along with wrist-level 6-DoF motion. For the human-hand setting, we define a small set of

intuitive gesture primitives (i.e., push, pull, grab) seen in Tab. A that capture essential interaction patterns. These manually defined mappings create a unified yet tractable action space for analyzing MLLMs' planning and manipulation competence.

Building on the proposed spatial action mapping, we curate annotated data from diverse sources, including human-hand manipulation videos, navigation video games, robotic arm manipulation datasets, and simulation environments. This unified dataset enables us to evaluate whether MLLMs can accurately plan and execute actions in the defined metric action space. Further implementation details are provided in Sec. 4 and in the experimental section.

Goal-driven Navigation. We leverage our SpatialEngine to get the action annotations as shown in Fig. E. We first extract the metric pose trajectories from the games videos, and convert them into discrete actions with our spatial action mapping, and then we randomly sample several image pairs from the video with the correspondence checking. For evaluation, the goal is a image, and the MLLMs are supposed to control the character to move to the target positions. We use the prompt template as below:

```
{
  "role": "system",
  "content": "Task Details:\n
  "Analyze Images: Compare the start image <Image 1> and the target image <Image 2> to understand the
required translation and rotation for the robot arm's end-effector.\n"
  "Define Motion: Decompose the movement into 6 steps, each containing one or more elementary
actions.\n"
  "Quantify Actions: For each action, specify an integer step_num that represents its intensity.\n\n"
  "Coordinate System:\n"
  "Right-hand frame attached to the end-effector: +Z forward, +X right, +Y downward.\n\n
  "Action Space:\n"
  "Translation: Dolly In (W), Dolly Out (S), Truck Left (A), Truck Right (D), Pedestal Up (space),
Pedestal Down (shift).\n"
  "Rotation: Pan Left (left arrow), Pan Right (right arrow), Tilt Up (up arrow), Tilt Down (down arrow),
Roll CW (clockwise), Roll CCW (counterclockwise).\n"
  "Special Action: Stay (STOP).\n\n"
  "Step Size:\n"
  "Translation: 0.019375 m/step. Rotation: 0.4509 rad/step.\n\n"
  "Output Format:\n"
  "Return a single JSON object with keys step_1-step_6. Each step contains:\n"
    "actions: list of action symbols\n"
    "step_nums: corresponding integers.\n\n"
  "Example:\n
  {
    \"step_1\": {
      \"actions\": [\"W\", \"A\"],
      \"step_nums\": [5, 2]
    },
    \"step_2\": {
      \"actions\": [\"W\", \"up_arrow\"],
      \"step_nums\": [3, 4]
    }
  }"
}
```

Figure F. **Prompt of navigation.**

In this prompt, translation and rotation steps are computed from the actual movement, while capping the number of steps at 10 to prevent overly long action sequences. To evaluate MLLMs, we compute a normalized metric in the range $[0, 1]$ by combining **relative distance** and **directional accuracy**. Specifically, for each step, let $\Delta\mathbf{p}_{\text{pred}}$ and $\Delta\mathbf{p}_{\text{gt}}$ denote the predicted and ground-truth translation vectors, respectively.

The **relative distance score** is defined as:

$$s_d = \max\left(0, 1 - \frac{\|\Delta\mathbf{p}_{\text{pred}} - \Delta\mathbf{p}_{\text{gt}}\|}{\|\Delta\mathbf{p}_{\text{gt}}\|}\right),$$

and the **directional score** is computed by the cosine similarity:

$$s_\theta = \frac{\Delta \mathbf{p}_{\text{pred}} \cdot \Delta \mathbf{p}_{\text{gt}}}{\|\Delta \mathbf{p}_{\text{pred}}\| \, \|\Delta \mathbf{p}_{\text{gt}}\|}.$$

The final step-wise accuracy is then: $s_{\text{step}} = s_d \cdot \max(0, s_\theta)$

which ensures a value in $[0, 1]$, where 1 indicates perfect alignment in both distance and direction. Aggregating s_{step} across all steps provides a comprehensive measure of the model's precision in executing end-effector motions.

Goal-driven Manipulation For the **Goal-Driven Manipulation** capability, we utilize action annotations from the `Droid` [25] and `EgoDex` [19] datasets. This task requires the MLLM to generate a sequence of precise actions to move a robot end-effector or a human hand from a starting state to a target state, both specified by images. The action space for `Droid` encompasses 7-DoF control: 6-DoF for the end-effector's pose (translation and rotation) and a binary state for the gripper (open/close). A similar action space is adapted for `EgoDex`, controlling wrist pose and finger grip. The MLLM is prompted to generate a sequence of continuous action vectors, as shown in the template below:

```
{
  "role": "system",
  "content": "Task Details:\n
  "Compare the start image <Image 1> and target image <Image 2> to infer the translation and rotation
required for the robot arm's end-effector.\n"
  "Decompose the motion into up to 6 steps, each combining any number of elementary actions.\n\n"
  "Action Space:\n"
  "We define a 7D action vector per step:\n"
  "[dx, dy, dz, d_roll, d_pitch, d_yaw, gripper_state]\n"
  "- Translation (dx, dy, dz): Displacement in meters along +X, +Y, +Z.\n"
  "- Rotation (d_roll, d_pitch, d_yaw): Rotation in radians about +Z, +X, +Y respectively.\n"
  "- gripper_state: 0=open, 1=closed.\n\n"
  "Each dx, dy, dz, d_roll, d_pitch, d_yaw is computed from selected actions and their step_nums:\n"
  "delta_q = step_num * unit_step_size  (translation in meters or rotation in radians)\n\n"
  "Available Actions:\n"
  "W/S: Dolly In/Out (+/-Z)\n"
  "A/D: Truck Left/Right (-/+X)\n"
  "space/shift: Pedestal Up/Down (-/+Y)\n"
  "left_arrow/right_arrow: Pan Left/Right (+/- yaw)\n"
  "up_arrow/down_arrow: Tilt Up/Down (+/- pitch)\n"
  "clockwise/counterclockwise: Roll CW/CCW (+/- roll)\n"
  "STOP: No movement\n\n"
  "Output Format:\n"
  "Return a single JSON object where each step is a key (\"step_1\", \"step_2\", ...).\n"
  "Each step contains:\n"
  "- actions: a list of action symbols\n"
  "- step_nums: a list of integers specifying intensity (1-10)\n"
  "- gripper: 0 or 1 for gripper state\n\n"
  "Example:\n"
  {
    \"step_1\": {
      \"actions\": [\"W\", \"A\"],
      \"step_nums\": [5, 2],
      \"gripper\": 0
    },
    \"step_2\": {
      \"actions\": [\"clockwise\"],
      \"step_nums\": [3],
      \"gripper\": 1
    }
  }
}
```

Figure G. **Prompt for Goal-Driven Manipulation with 7D Action Representation.**

To evaluate the MLLM's performance, we assess the accuracy of the predicted action sequence against the ground truth. For the translational component of the motion, we reuse the step-wise accuracy metric s_{step} from the navigation task, which combines relative distance and directional scores. For the rotational component, we compute a normalized score based on the angular difference between the predicted orientation and the ground truth. Let R_{pred} and R_{gt} be the predicted and ground-truth rotation matrices for a step. The rotational error angle θ_{err} is calculated from the error rotation matrix $R_{\text{err}} = R_{\text{pred}} R_{\text{gt}}^T$:

$$\theta_{\text{err}} = \arccos\left(\frac{\text{Tr}(R_{\text{err}}) - 1}{2}\right).$$

The **rotation score** s_{rot} is then defined as:

$$s_{\text{rot}} = \max\left(0, 1 - \frac{\theta_{\text{err}}}{\pi}\right),$$

which normalizes the error to a $[0, 1]$ range, where 1 indicates a perfect rotational match. Finally, the **gripper score** s_{gripper} is a binary accuracy (1 if the predicted state matches the ground truth, 0 otherwise). The final score for each step is a weighted combination of these three metrics, providing a holistic evaluation of the model's ability to perform precise, multi-faceted manipulation tasks.

D. Ability Transfer via Prompting

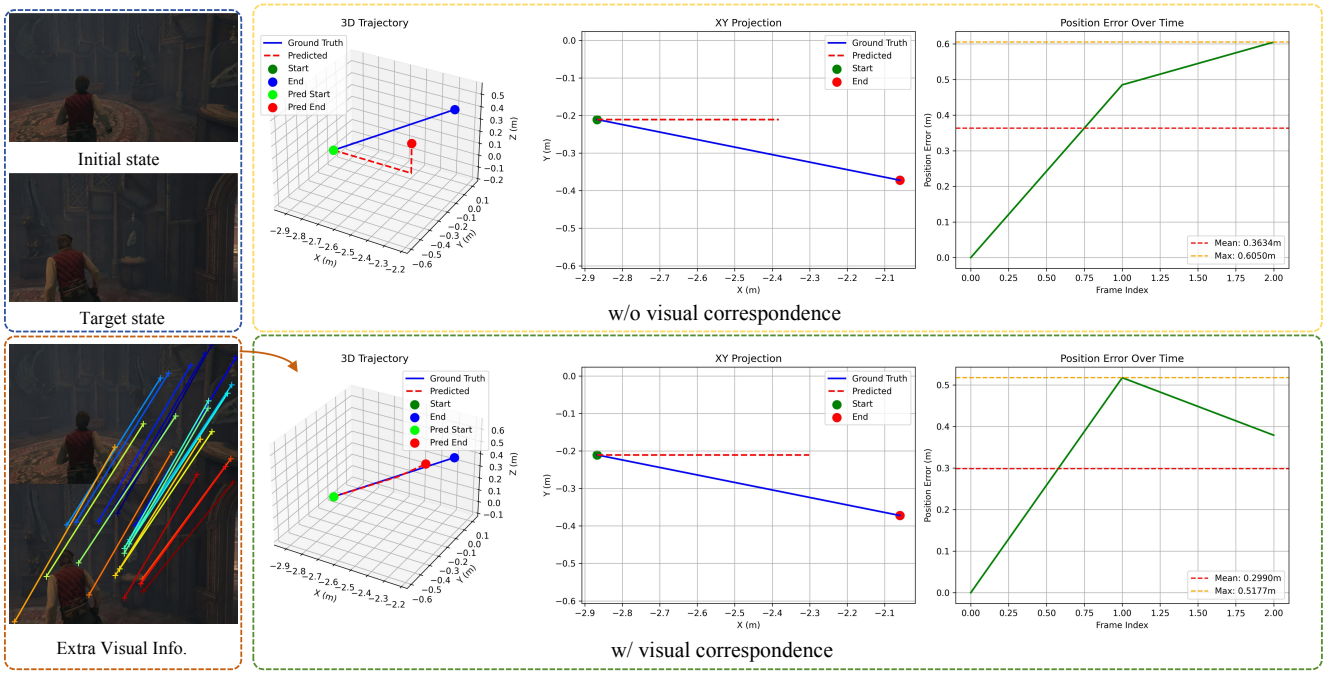


Figure H. **Correspondence Prompting for Navigation.** The correspondence prompt guides Gemini2.5-pro to navigate and move more accurately within 3D environments.

In addition to SFT, we investigate cross-level ability influence through explicit prompting. Specifically, we consider a representative task pair: low-level abilities (L1.Corr, L1.Dist, L1.Size) and a high-level task (L4.Imaged Goal Navigation). Intuitively, correspondence is a necessary component for navigation. Using Gemini2.5-pro, we provide models with explicit prompts derived from matching visualizations, depth, and size context. As shown in Fig. H, correspondence guidance improves target direction recognition, increasing accuracy by **7.1%**, while distance and size prompting yield gains of **5.5%** and **2.1%**, respectively. These results suggest that grounding MLLM reasoning with explicit low-level visual information can substantially enhance performance on complex spatial navigation tasks.

E. Benchmark Metric Aggregation

To derive a single, comprehensive score for a model's spatial intelligence, we employ a hierarchical aggregation methodology. This approach is designed to reflect the complex, multi-layered nature of spatial cognition, rather than treating all abilities as

equally important. The design is principally guided by established theories in cognitive psychology, which posit that spatial intelligence is constructed hierarchically, with fundamental perceptual skills forming the bedrock for more abstract reasoning and planning.

Our aggregation framework is built upon the *SpatialTree* structure. The assignment of weights within this tree is determined by a synthesis of theoretical principles and empirical, data-driven insights:

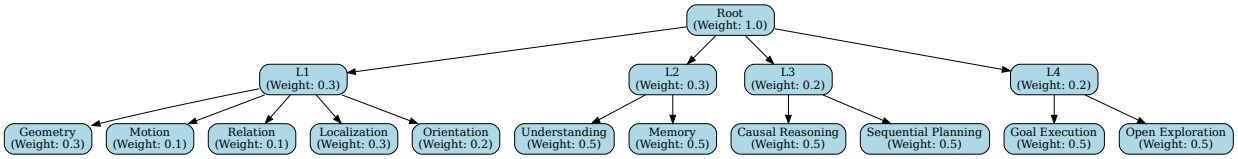


Figure I. An illustration of the hierarchical weighting scheme for metric aggregation with in the *SpatialTree*. Each node represents a capability layer, with the assigned weight used for the bottom-up calculation of the final score. The weighting prioritizes foundational perceptual abilities (L1) as they are prerequisites for higher-level cognitive tasks.

Cognitive Hierarchy. In line with cognitive science literature, our weighting scheme prioritizes foundational capabilities, as shown in Fig. I. The L1 layer, which represents low-level spatial perception, is assigned the largest weight, as these skills are prerequisites for almost all higher-level spatial tasks found in L2 (Mental Mapping) and L3 (Mental Simulation).

Empirical Dependency from Correlation Analysis. The theoretical hierarchy is further refined and validated by our empirical findings from the Pearson correlation heatmap (Fig. 4). The heatmap allows us to identify *atomic abilities* that exhibit strong, widespread correlations with a multitude of other skills. These influential abilities are considered more fundamental to the overall spatial intelligence network and are consequently assigned higher weights within their respective sub-trees. This ensures our metric is not just theoretically sound, but also reflects the actual dependencies observed in model performance.

The final score is calculated via a bottom-up, weighted summation. The performance score for any parent node in the tree is the weighted sum of the scores of its immediate children. This process is recursively applied until the root node is reached, yielding a single, principled score that holistically quantifies the spatial intelligence of a given MLLM.

F. More Visualizations for QAs in SpatialTree Bench

We declare that Large Language Models (LLMs) were used in a limited capacity during the preparation of this manuscript. Specifically, LLMs were employed for grammar checking, word choice refinement, and typo correction. All core technical contributions, experimental design, analysis, and conclusions are entirely our own. The use of LLMs did not influence the scientific methodology, result interpretation, or theoretical contributions of this research.

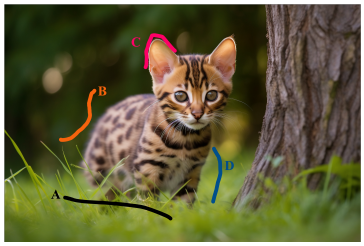
Geometry	
	
Prompt	Which is wider, the width of the painting on the wall or the width of the wooden table next to the sofa?
Answer	The wooden table
	
Prompt	The coordinates [x , y] are normalized to 0-1 and scaled by 1000, with [0 , 0] at the top-left. The x-axis represents the width, and the y-axis represents the height. What is the depth at the coordinates [389 , 180] in the image (in mm)?
Answer	`1915`
	
Prompt	Which line is closer to the edge?
Answer	C

Table B. Examples of *L1.Geometry*.

Relation	
	
Prompt	When you were taking the photo in Image 1, where was the exit of the room relative to you?
Answer	Rear left
	
Prompt	Track [150, 470] from Image 1 to its correspondence in Image 2.
Answer	C

Table C. Examples of *L1.Relation*.

Orientation	
	
Prompt	Analyze the image to determine the vertical field of view (vfov) and calculate the camera'against the vertical axis (gras roll and pitch angles vity).
Answer	["roll_unc": 0.61629718542099, "pitch_unc": 1.862020492553711, "vfov_unc": 20.077800750732422]
	
Prompt	If I stand at the cat's position facing where it is facing, is the knife in front of me or behind me?
Answer	In front of

Table D. Examples of *L1.Orientation*.

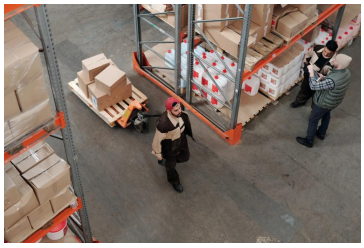
Motion	
	
Prompt	As shown in the video, which direction is the view moving towards ?
Answer	C
	
Prompt	From view 1 to view2, in which direction is the race car moving?
Answer	First to the front left, then to the front right. C

Table E. Examples of *L1.Motion*.

Localization	
	
Prompt	Please ground mirror in this image. The 3D bounding box is defined in the format: `"bbox_3d": [u_center, v_center, z_center, x_size, y_size, z_size, roll, pitch, yaw]`

Answer	
	`[`
	` "bbox_3d": [901, 558, 4.87, 0.541, 1.698, 0.243, 179.279, -29.539, 176.489]`
	`]`

Table F. Examples of *L1.Localization*.

Understanding	
	

Prompt	From the man wearing the red hat, how many people are on his left?
	
	`[`
	`A. zero,`
	`B. one,`
	`C. three,`
	`D. two`
	`]`

Answer	D
	
Prompt	To ride this bicycle along the seawall, where would you place your hands to steer, where would you sit, and where would you put your feet to pedal?
Answer	`[744, 439, 783, 489]`

Table G. Examples of *L2.Understanding*.

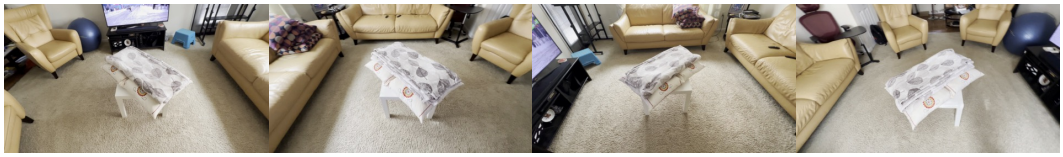
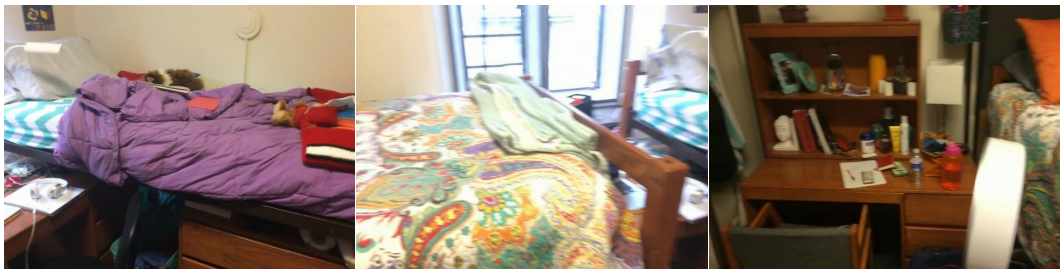
Memory	
	
Prompt	From the man wearing the red hat, how many people are on his left? [A. Leather loveseat with three seat cushions, B. TV, C. Two single sofas, D. Leather loveseat]
Answer	B
	
Prompt	How many bed(s) are in this room?
Answer	2

Table H. Examples of *L2.Memory*.

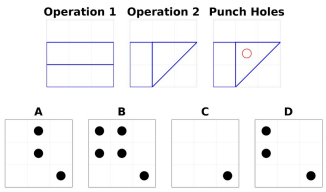
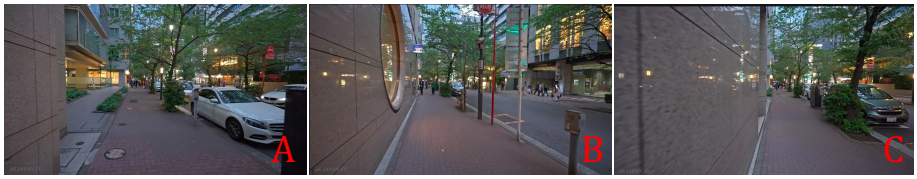
Causal Reasoning	
	
Prompt	If the dog on the right reaches the camera in 5 s, what is its speed? [A. 14.7m/s B. 1.9m/s C. 21.7m/s D. 11.9m/s]
Answer	B
	
Prompt	A 3x3 grid paper undergoes two folds as shown. A hole is punched in the folded state. Which option (A, B, C, or D) shows the unfolded paper?
Answer	A
	
Prompt	What object is located immediately to the right of point [710 991] in the second image, just outside of the frame ?
Answer	Wooden Table

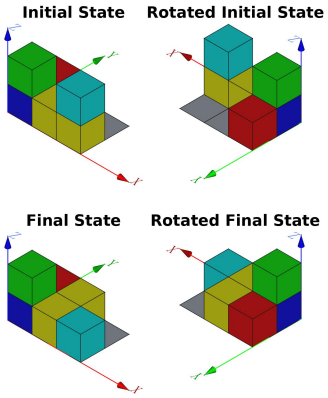
Table I. Examples of *L3.Causal Reasoning*.

Sequential Planning



Prompt The camera is moving forward. Please arrange these three images in chronological order?

Answer B-C-A



Prompt The top row of images shows different views of the initial state of a cube stack, while the bottom row shows different views of the final state after transformation. During the transformation process, blocks can move one unit in any direction (forward, backward, left, right, up, down). If the target position is empty, the block can move there directly; if the target position already has a block, they swap places. Blocks cannot float in the air. If a block is moved away from a position, any block above it will fall down until reaching a supporting surface. The xyz axes are shown in the diagram, and each block's position can be precisely identified using coordinates (x1,y1,z1). Which of the following transformation sequences can change the cube stack from the initial state to the final state shown in the diagram? Please answer from options A, B, C, or D.

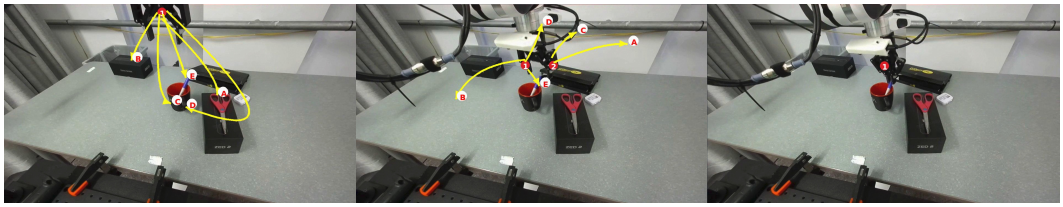
```
A. (1, 0, 0) y+ -- (0, 0, 1) z-
B. (1, 0, 0) x+ -- (1, 0, 0) y+
C. (2, 0, 0) x- -- (1, 0, 0) y+ -- (2, 0, 0) x-
D. (0, 0, 0) x+ -- (0, 1, 0) y- -- (0, 0, 1) y+
```

Answer

```
C
```

Table J. Examples of *L3.Sequential Planning*.

L4 Agentic Competence



Prompt

You are an intelligent agent observing a video sequence that depicts a task being performed which is ”please get the blue pan out of the bottle”. For each image in the sequence that provides candidate action options, select the single most appropriate action to perform in the current state. In the final frame, do not select any action. Instead, determine whether the overall task shown in the sequence has been successfully completed (1) or not completed (0). Please output the selected action for each intermediate frame and the completion flag for the final frame.

```
A. EB1
B. CC1
C. EA0
D. EE1
E. AD0
F. EE0
G. CD0
H. EC0
```

Answer

F

Prompt

Which image shows the robot making the most progress towards the task placing the fork to the right side of the orange kitchen wipe?

```
A. First.
B. Second.
C. Third.
D. Fourth
```

Answer

```
C
```

Table K. Examples of *L4 Agentic Competence*.

L4 Agentic Competence (Continued)

Prompt

Task: Visual Navigation Action Sequence Generation
You are an expert visual navigation agent. Your task is to generate a sequence of actions to navigate a robot from a starting visual state (Image 2) to a target visual state (Image 3) based on the provided visual information.

Context and Example
We provide three sequential images: Image 1, Image 2, and Image 3. To help you understand the task, we are providing the complete action sequence that navigates from **Image 1** to **Image 2** as an example.
Example Action Sequence from Image 1 to Image 2:

```
{
  "actions": ["Dolly In", "Truck Left", "Pedestal Up",
              "Pan Left", "Roll CW"],
  "step_nums": [0, 3, 4, 5, 0]
}
```

Your Core Task
Now, carefully observe **Image 2** (the starting state) and **Image 3** (the target state). Your mission is to generate the action sequence required to navigate from Image 2 to Image 3.

1. Action Space
You must choose from the following 12 elementary actions. In your output, you **must use the 'symbol' specified** to represent each action.

Category	Action	Sym	Description
Trans.	Dolly In	W	Move forward
	Dolly Out	S	Move backward
	Truck Left	A	Move left
	Truck Right	D	Move right
	Pedestal Up	space	Move up
	Pedestal Down	shift	Move down
Rot.	Pan Left	←	Neg. rot around +Y
	Pan Right	→	Pos. rot around +Y
	Tilt Up	↑	Pos. rot around +X
	Tilt Down	↓	Neg. rot around +X
	Roll CW	↻	Neg. rot around +Z
	Roll CCW	↺	Pos. rot around +Z
Special	Stay	STOP	No movement

2. Step Size Parameters The magnitude of each action is determined by `step_num` combined with a unit step size.
- **Translation**: 0.0626 meters per step.
- **Rotation**: 0.0725 radians per step.

E.g., `action_symbol: "W"` and `step_num: 10` means moving forward by 10×0.0626 meters.

3. Required Output Format Your final output **MUST** be a JSON object containing two keys: `"actions"` and `"step_nums"`. The lengths of both arrays must be identical.

Answer

```
{
  "actions": [
    "Truck Left",
    "Pedestal Up",
    "Pan Left"
  ],
  "step_nums": [
    1,
    4,
    1
  ]
}
```

Table K. Examples of *L4 Agentic Competence* (Continued).