A. Appendix

A.1. Simulator

Our research primarily uses simulators such as Habitat-Sim, AI2-Thor, and AirSim to collect data. The following is a brief introduction to these simulators.

Habitat-Sim. Habitat-Sim is a high-performance 3D simulator that supports physics engines. It's designed for 3D spatial scans of both indoor and outdoor environments and includes built-in support for datasets like HM3D and MP3D. The simulator can be configured with various sensors, including RGB-D cameras and semantic maps. Additionally, it features collision detection for robots as they move, making it a popular choice for embodied AI experiments such as navigation and instruction following.

AI2-Thor. AI2-THOR is a nearly photorealistic and interactive framework for embodied AI agents. In addition, ProcTHOR [10] can be run under the AI2-THOR framework, which includes 11k scenes. Meanwhile, it uses procedural generation to sample massively diverse, realistic, interactive, customizable, and performant 3D environments.

AirSim. AirSim is an open-source simulator for cars and drones built on Unreal Engine. It supports multiple outdoor environments, such as the Embodied City, and provides APIs that allow you to control a drone's movement as if you were playing a game. The simulator also supports RGB-D sensors.

A.2. Spatial Task Framework

There are image-related question types

- **object_size_estimation.** Given an object, identify the size of the object and output the specific value.
- **object_size_comparison.** Given multiple objects, compare their sizes.
- **object_volume_estimation.** Identify the volume of the object and output the corresponding value.
- **object_volume_comparison.** Compare the volume sizes of multiple objects in the video to find the largest or smallest one.
- **object_below.** Determine whether one object is positioned at a lower elevation relative to the other.
- **object_above.** Assess the relative vertical positions of two objects to establish whether one is positioned above the other.
- **object_direction_facing_complex.** Given three objects, after determining the perspective through two of them, determine the relative position of the other object in the video. Relative position includes front-left, front-right, back-left, or back-right.
- **object_direction_facing_simple.** Given three objects, after determining the perspective through two of them, determine the relative position of the other object in the video. Relative position includes left, right.
- **object_absolute_distance.** Calculate the absolute distance between two objects in the image.
- **object_relative_distance.** Determine which of the candidate objects is the closest to the target object.
- **object_nearby.** There are objects within a certain range near the object.
- **object_counting.** Count the total number of times a certain type of object appears in the image.
- **object_distance_camera_relative** From the perspective of the current picture, identify which of the two objects is farther or closer to the camera
- **object_distance_camera_absolute.** From the perspective of the current picture, identify the distance from an object to the camera's perspective.
- **object_direction_camera_complex.** From the current perspective of the picture, determine the positional relationship between two objects. Relative position includes front-left, front-right, back-left, or back-right.
- **object_direction_camera_simple.** From the current perspective of the picture, determine the positional relationship between two objects, whether one object is to the left or right of the other.
- **high_and_low_position.** Given two objects, assess their relative vertical positions to identify which one is at a higher elevation.

There are video-related question types:

- **object_appearance_order.** Identify the order in which objects appear in the video and list the object names in chronological order.
- **object_absolute_distance.** Calculate the absolute distance between two objects in the video.
- **object_counting.** Count the total number of times a certain type of object appears in the video and output this value.
- **object_relative_distance.** Determine which of the candidate objects is the closest to the target object.
- **object_size_estimation.** Determine the length, width, and height of the objects in the video and output the corresponding dimensions.
- **object_direction_facing_complex.** Given three objects, after determining the perspective through two of them, determine the relative position of the other object in the video. Relative position includes front-left, front-right, back-left, or back-right.
- **object_direction_facing_simple.** Given three objects, after determining the perspective through two of them, determine the relative position of the other object in the video. Relative position includes left and right.
- **object_volume_comparison.** Compare the volume sizes of multiple objects in the video to find the largest or smallest one.
- **object_in_frame** Identify all the objects that appear in the video at a certain moment and output them by category.

Table 4. Training configurations for different models.

Configuration	Qwen2.5-VL-7B	InternVL-8B
SFT Method	LoRA	LoRA
Rank	256	256
Alpha	512	512
Epochs	1	1
Batch Size	2	2
Grad. Accum. Steps	4	4
Learning Rate	5×10^{-6}	5×10^{-6}
LR Scheduler	Cosine	Cosine

- **object_volume_estimation.** Identify the volume of the object and output the corresponding value.
- **object_height_determination.** Identify the height differences of objects and determine which one is the highest.
- **temporal_appearance_sequence.** Analyze the sequence of objects based on the chronological order of their appearance.
- **object_nearby.** For problems like identifying which objects are within 5 meters in front of a certain object.
- **object_below.** Determine whether one object is positioned at a lower elevation relative to the other.
- **object_above.** Assess the relative vertical positions of two objects to establish whether one is positioned above the other.
- **high_and_low_position.** Given two objects, assess their relative vertical positions to identify which one is at a higher elevation.

A.3. Case Study

We use a video as an example to demonstrate the process of data construction. As shown in Figure 6, we first obtain the meta-information in the video and the scene from the simulator. Then, based on the obtained meta-information and the images from the continuous frame extraction of the video, we generate the corresponding questions. For each problem, generate a piece of code for metainfo processing through a code LLM. The result of code execution is the ground truth. For the problem that meets the requirements, select two objects involving the same object and, through the reasoning ability of the large model, infer a new problem and its answer.

A.4. Training Details

To validate the effectiveness of our data, we trained on qwenvl-2.5-7B and InternVL-8B. We employed a LoRA [11] fine-tuning approach for training.The training parameters are shown in Table 4. For video input, a uniform frame sampling strategy was employed, fixing the number of sampled frames at 32. We conducted the training on 8*A100 80G GPUs.

A.5. GRPO Training Details

In our study, we employed reinforcement learning using the GPRO framework. A dataset of 20,000 samples was constructed and subsequently transformed into multiple-choice questions via GPT-4o. The reward function was designed to incorporate both rule-based rewards, derived from answer correctness, and format-related rewards. The model was trained on 8*A100 80GB GPUs with a batch size of 32, and each sample was rolled out 8 times during training. After 200 training steps, the reward function had converged, leading to the selection of 6.4k samples for comparative analysis.

A.6. Benchmarks Description

- **VSI Bench.** VSI-bench is an evaluation benchmark focused on video spatial understanding intelligence. It contains 5k question-and-answer pairs derived from 288 videos. It assesses a model's ability to understand video spatial information from several perspectives, including object spatial layout, object spatial attributes, and the temporal information of objects within a video.
- **ViewSpatial-Bench.** Viewspatial-bench is a multi- viewpoint spatial understanding benchmark test containing over 5.7k pairs of questions and answers. It includes five types of test tasks. They are respectively scene simulation relative direction, relative direction, object view orientation, relative direction, object view orientation.
- **Q-Spatial-Scannet&Q-Spatial-Plus.** Q-Spatial Bench is a benchmark designed to measure the quantitative spatial reasoning. It involves issues such as the distance and size between objects in the recognition space. Among them, Q-Spatial-Scannet is obtained from the RGB-D scanning in the high-quality real-world environment annotation data ScanNet, and Q-Spatial-Plus is obtained from the actual measurement of real-world objects
- **ERQA.** ERQA is a benchmark for evaluating the embodied reasoning ability of multimodal models, including spatial reasoning and trajectory reasoning, action reasoning, state estimation, pointing, multi-view reasoning, and task reasoning.
- **CVBench.** CVBench is a vision-centric test benchmark that includes 2,638 manually annotated examples. It covers a range of tasks, including 2D spatial relationship judgment, 2D counting, 3D depth prediction, and 3D relative distance prediction. wo'zhe'li

A.7. Fine-grained Diagnostic Evaluation

To provide a more diagnostic understanding of how spatial reasoning improves, we conducted detailed statistics in all the benchmark (Table 5). Our method demonstrates performance gains in tasks requiring precise spatial awareness, particularly in queries related to absolute distance , orientation, and appearance order.

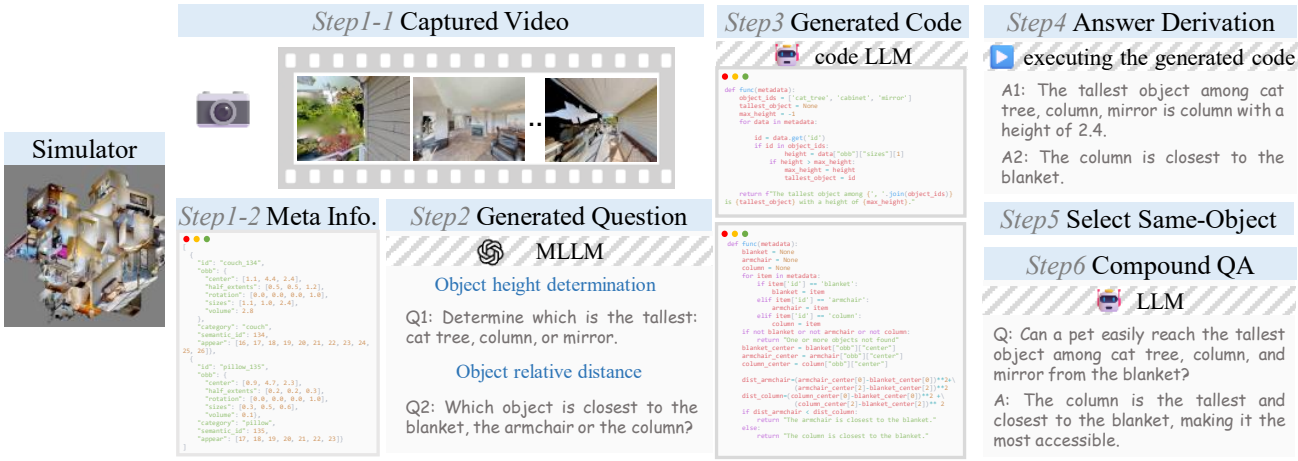

Figure 6. Case Study

Table 5. Qualitative and diagnostic analysis

Model	absolute distance	count	orient	location	order	relative distance	size
Qwen2.5-VL-7B	29.55	40.35	39.77	**45.66**	26.86	**34.93**	49.22
+**SPRITE** (ours)	**45.66**	**47.26**	**42.50**	44.42	**41.10**	33.66	**54.88**

A.8. Sim-to-Real Transferability

We provide experimental details by training models on simulator data (about 200k). Models trained on SPRITE demonstrate performance gains across various real-world benchmarks (Table 6). This demonstrates that the high-quality spatial data synthesized by our framework effectively bridges the reality gap, successfully transferring spatial understanding capabilities to physical-world scenarios.

Table 6. Performance of the simulator data training

Model	ViewSpatial	QSpatial-Plus	ERQA	CVbench
Qwen2.5-VL-7B	38.44	51.49	41.25	74.55
+Simulator data	**40.96**	**59.41**	**44.00**	**78.16**

A.9. Dataset Quality Assurance and Statistics

To ensure the high quality and reliability of our dataset, we implemented a two-stage verification process comprising automated filtering and manual auditing. During the automated phase, our filtering pipeline eliminated approximately 11% of the initially generated samples. Specifically, code execution failures accounted for roughly 2% of the rejections. Following this automated refinement, we conducted a manual audit to further validate the ground-truth accuracy. Three independent domain experts reviewed a random subset of 50 data points, yielding a high human-verified accuracy of 90.67% and demonstrating strong inter-annotator agreement with a Cohen's Kappa coefficient of 0.84.

A.10. Prompts

There are prompts to construct related questions and obtain ground truth.

generate image question prompt

```
You are a teacher for an embodied task
course, and your task is to create some
test questions for an image-based based on
the corresponding task type.
###Noted
1.The proposed test questions must only
use the candidate object information
provided by the user, especially the
object names, which must correspond
one-to-one.  The candidate objects include
the object names and their corresponding
category information
This is the name of the candidate
object and the corresponding type of
the candidate object, The content is a
dictionary, with the key being the object
name and the value being the object type.
###candidate object
```
{objects_info}
```
2.The proposed test questions require that
they can be solved through code based on
the meta-information of the object.
The following is an example of
meta-information.  The object information
in the example is not a candidate object.
###meta information
```
{meta_example}
```
The content and meaning are as follows:
id:  The unique name of an object
obb:obb is a three-dimensional oriented
bounding box.  In the coordinate system,
```

the Y-axis is perpendicular to the ground and upward
In obb information, "center" represents the centroid coordinate of an object in the world coordinate system
"half_extent" is the half-length of the OBB, indicating the distance from the center point to each face
"sizes" is the size of the OBB, indicating the length, width, and height of the OBB.
"volume" represents the size of the space occupied by OBB
"category" represents the type information of an object
In coordinate information, a movement of 0.1 unit is equivalent to 0.1 meter in reality
Additionally, the known extra information is the camera position:
camera_position = [-1.703, 0.985824, 0.922993]
The raised questions need to comply with the following task types and indicate the task types in the output results.
```
{question_type_all}
```
4.The proposed test questions need to explicitly include the name information of the object and the name of the task type.
5.The output needs to be in JSON format. The output example is as follows, and the question types can refer to the example:
```
{output_example}
```
instruction: Test question content
objects: The names of the objects involved in the question are selected from the names of the candidate objects
objcets_category: what should be written here is the name of the object category
category: Task type
6.For numerical problems, units need to be included in the problem
7.For questions about object dimensions and the like, just ask about one dimension of length, width, or height in the question.
8.For a picture, please select the appropriate question type and ask at least 20 questions, and the number of questions of different types should be as even as possible.
9.Please refer to the given picture and propose questions with practical significance.

generate video question prompt

There is currently a first-person video centered on the camera.
Please create some test questions for the tasks in the embodied scenarios based on this video.
###Noted
1.The proposed test questions can only use the candidate object information provided by the user, especially the object name information needs to correspond one-to-one This is the name of the candidate object and the corresponding type of the candidate object
candidate object
```
{objects_info}
```
2.The proposed test questions require that they can be solved through code based on the meta-information of the object.
###meta information
```
{meta_example}
```
The content and meaning are as follows
id: The unique name of an object
obb:obb is a three-dimensional oriented bounding box. In the coordinate system, the Y-axis is perpendicular to the ground and upward
In obb information, "center" represents the centroid coordinate of an object in the world coordinate system
"half_extent" is the half-length of the OBB, indicating the distance from the center point to each face
"rotation" is a quaternion representing the rotation of the OBB.
"sizes" is the size of the OBB, indicating the length, width, and height of the OBB.
"volume" represents the size of the space occupied by OBB
"category" represents the type information of an object
"appear" indicates in which pictures an object appears. For example, "appear = [0,1]" means it appears in the first and second pictures. The sequence of the pictures indicates the time information.
In coordinate information, a movement of 0.1 unit is equivalent to 0.1 meter in reality
3.The raised questions need to comply with the following task types and indicate the task types in the output results.
```
{question_type_all}
```
4.The proposed test questions need to explicitly include the name information of the object and the name of the task type.
5.The output needs to be in JSON format. The output example is as follows, and the question types can refer to the example:
```
{output_example}
```
instruction: Test question content

objcets: The names of the objects involved in the test questions, and the names of the objects involved must explicitly appear in the instruction.
objcets_category: what should be written here is the name of the object category
category: Task type
6.For a video, please select an appropriate type of question. At least 40 questions should be raised, and the number of questions of different types should be as even as possible.
7.For questions about object dimensions and the like, just ask about one dimension of length, width, or height in the question.
8.The perspective position information in the video is unknown, so the perspective information should not be assumed in the test questions. When designing test questions about perspective transformation, three objects need to be included to determine their relative positions.
9.Please refer to the given video and propose a video with practical significance.

generate image code prompt

```
You are a senior engineer of Python code.
Your task is to write a function based
on user questions, and the requirement is
that the return result of the function is
a string.
###Known information
1.The function name is func, and the
parameters pass a metadata variable and
a camera_position variable. The function
is defined as follows
```
```
def func(metadata,camera_position):
```
```
2. The content of metadata is a list.
The following is part of the list. The
content format is as follows.
Please refer to the format. The specific
content is passed by the metadata
variable.
```
```
{meta_info}
```
```
The content and meaning are as follows:
id: The unique name of an object
obb: obb is a three-dimensional oriented
bounding box. In the coordinate system,
the Y-axis is perpendicular to the ground
and upward. The system follows the
right-handed coordinate rule.
In obb information, "center" represents
the centroid coordinate of an object in
the world coordinate system
"half_extent" is the half-length of the
OBB, indicating the distance from the
center point to each face
"sizes" is the size of the OBB, indicating
the length, width, and height of the OBB.
"volume" represents the size of the space
occupied by OBB
"category" represents the type information
of an object
In coordinate information, a movement of
0.1 unit is equivalent to 0.1 meter in
reality
3.The content of camera_position
is a list. The content format
is as follows. camera_position =
[-1.703,0.985824,0.922993]
###Noted
1.In the coordinate system, the Y-axis is
perpendicular to the ground and upward.
The system follows the right-handed
coordinate rule.
2.The code part in the answer only needs
to include the corresponding functions,
and there is no need to provide calling
examples.
3.Please use the object names involved in
the question when completing the code.
4.If it is an object-counting problem,
what is counted is the number of objects
of the same type, and the category of
objects needs to be used
5.The return result of the function must
be a string and conform to the description
in natural language
6.The output format can be referred to the
reference code.
###reference code.
```
```
{reference_code}
```
```
###question
```
```
{question}
```
```
###The names of the objects involved in
the question
```
```
{objects}
```
```
###The category of objects involved in the
problem
```
```
{categories}
```
```

## generate video code prompt

```
You are a senior engineer of Python code.
Your task is to write a function based
on user questions, and the requirement is
that the return result of the function is
a string.
###Known information
1.The function name is func, and the
```
```

parameter passes a metadata variable.
The function is defined as follows
```
def func(metadata):
```
2.The content of metadata is a list.
The following is part of the list. The content format is as follows.
Please refer to the format. The specific content is passed by the metadata variable.
```
{meta_info}
```
The content and meaning are as follows
id: The unique name of an object
obb:obb is a three-dimensional oriented bounding box. In the coordinate system, the Y-axis is perpendicular to the ground and upward
In obb information, "center" represents the centroid coordinate of an object in the world coordinate system
"half_extent" is the half-length of the OBB, indicating the distance from the center point to each face
"rotation" is a quaternion representing the rotation of the OBB.
"sizes" is the size of the OBB, indicating the length, width, and height of the OBB.
"volume" represents the size of the space occupied by OBB
"category" represents the type information of an object
"appear" indicates in which pictures an object appears. For example, "appear = [0,1]" means it appears in the first and second pictures. The sequence of the pictures indicates the time information.
In coordinate information, a movement of 0.1 unit is equivalent to 0.1 meter in reality
###Noted
1.The code part in the answer only needs to include the corresponding functions, and there is no need to provide calling examples.
2.Please use the object names involved in the question when completing the code.
3.If it is a object_counting problem, what is counted is the number of objects of the same type, and the category of objects needs to be used
4.The output format and code logic can be referred to the reference code.
###reference code.
```
{reference_code}
```
###question
```
{question}
```
###The names of the objects involved in the question
```
{objects}
```
###The category of objects involved in the problem
```
{categories}
```